AN ITERATIVE METHOD FOR VECTOR MEDIAN FILTERING

*Clay Spence and Craig Fancourt**

Sarnoff Corporation
CN5300
Princeton, NJ 08543-5300
USA

ABSTRACT

We present a new iterative approach to computing a median of a set of vectors. Like the standard approach, the proposed method minimizes the sum of the distances from the data points to the median. However, unlike the standard approach, the resulting median vector is not restricted to be a member of the data set. The proposed approach has several advantages over the standard definition, such as fast convergence, computational complexity that scales linearly with the number of exemplars, and a result that is closer to the center in an intuitive sense. As an example, we apply the method to color image filtering.

Index Terms— Median filters, Vector median, Image color analysis

1. INTRODUCTION

Due to their simplicity, robustness to outliers, and preservation of sharp transitions, median filters are often used for noise suppression, in spite of the existence of algorithms with superior performance [1, 2, 3]. For vector data such as color images, a vector version of the median can be defined by minimizing the sum of the (L_2) distances from an unknown vector, $\mathbf{v}_{\rm med}$, to a given vector field, $\mathbf{v}_i$:

$$J = \sum_i \|\mathbf{v}_i - \mathbf{v}_{\rm med}\| \tag{1}$$

Each term can be visualized as a cone with its vertex at a data point. This is non-differentiable, so that conventional optimization algorithms like Newton's method will fail.

To avoid this problem, standard vector median algorithms [4] choose a member of the data set that minimizes (1):

$$\begin{aligned} j^* &= \arg\min_j \sum_i \|\mathbf{v}_i - \mathbf{v}_j\|, \\ \mathbf{v}_{\rm med} &= \mathbf{v}_{j^*}. \end{aligned} \tag{2}$$

Unfortunately, this result is not necessarily a minimum of J. Moreover, none of the data vectors may lie near an intuitive center of the data. Consider three points at the corners of an equilateral triangle. None is more central, but the standard algorithm would choose one anyway. Furthermore, a direct implementation of (2) can be expensive, especially for large data sets.

1.1. Iterative Vector Median (IVM)

We propose an iterative approach to finding a vector median that is the true minimum of J, by observing that the optimal vector median satisfies

$$\frac{\partial J}{\partial \mathbf{v}_{\rm med}} = -\sum_i \frac{\mathbf{v}_i - \mathbf{v}_{\rm med}}{\|\mathbf{v}_i - \mathbf{v}_{\rm med}\|} = 0. \tag{3}$$

We now pretend that $\mathbf{v}_{\rm med}$ in the numerator is different from that in the denominator. Label the former $\mathbf{v}_{\rm med}^{t+1}$ and the latter $\mathbf{v}_{\rm med}^{t}$, then solve for $\mathbf{v}_{\rm med}^{t+1}$ to get

$$\mathbf{v}_{\rm med}^{t+1} = \frac{\sum_i \frac{\mathbf{v}_i}{\|\mathbf{v}_i - \mathbf{v}_{\rm med}^t\|}}{\sum_i \frac{1}{\|\mathbf{v}_i - \mathbf{v}_{\rm med}^t\|}} \tag{4}$$

where t is the iteration count. The algorithm begins with a choice for $\mathbf{v}_{\rm med}^0$, such as the mean of the vector field, and then continually reapplies (4) until $\mathbf{v}_{\rm med}$ converges, according to some criterion such as sufficiently small change. We refer to this as the iterative vector median or *IVM* algorithm, and the same algorithm used for filtering as the iterative vector median filter or *IVMF* algorithm.

The IVM algorithm is a generalization to vectors of a known algorithm for finding the median of a set of scalar values [5]. There, the equivalent of (3) requires that the number of data points less than the median should be equal to the number that are greater. With vectors, (3) is the requirement that the sum of the unit vectors from the median to each of the data points is zero. In the scalar case, if there are an even number of data points, there is an entire interval of values that satisfy the criterion. By convention the center of the interval is chosen as the median. In the vector case, the same degeneracy can only occur if the data points are colinear, essentially reducing to the scalar case. The formula (4) for the recursion

*Now at Merck Research Laboratories, Rahway, New Jersey.

is the same in both cases, except that the vectors are replaced with scalars, and the norm is replaced with the absolute value.

We will show that (1) only has exactly one global minimum unless there are an even number of colinear data points. In this case the set of global minima is a line segment between two data points.

Furthermore, no matter where an iterate lies, one step of (4) moves the iterate into the convex hull of the data points. If an iterate comes sufficiently close to a minimum, the algorithm (4) converges to it, almost always exponentially. Also, all limit points are global minima. We have not yet proven that (4) must converge, but have never observed otherwise.

Note that the IVM algorithm can return a different solution than (2). In fact, the value found iteratively can be a data point, but often it is not. Consider again three data points at the vertices of an equilateral triangle. Unlike algorithm (2), the proposed algorithm will choose the center of the triangle. For some applications this is an advantage, giving a solution that is more nearly in the center of the data points.

Suppose now that the triangle is nearly flat, with one point near the middle of the line joining the other two points. The IVM algorithm will converge to the nearly-central point. The gradient (3) will not be well-defined, but the nearly-central data point is still a minimum of (1).

Furthermore convergence is fast. In a normals smoothing application, convergence to within $1/10^\circ$ is typically obtained in less than five iterations for a three-dimensional vector field with nine members. Existing methods require nine passes through the vector field, where each pass is comparable to an iteration, in order to calculate the distance from each vector to the rest of the field.

In fact, a naive implementation of (2) takes $O(DN^2)$ operations, where D is the number of dimensions of the vectors and N is the number of vectors. By contrast the IVM method takes $O(DNM)$ operations, where M is the number of iterations. Given that convergence is usually exponential, M is much less than N for large N.

2. ALGORITHM BEHAVIOR

2.1. Comparison with Newton's method

Consider first the objective function (1) and its derivatives. The individual terms of J are convex, so J itself is convex. The gradient is given in (3). Clearly it is not defined at the data points. If we define $r_i = \|\mathbf{v}_i - \mathbf{v}_{\text{med}}\|$ and the unit vectors $\mathbf{n}_i = (\mathbf{v}_i - \mathbf{v}_{\text{med}})/r_i$, the gradient can be rewritten as

$$\frac{\partial J}{\partial \mathbf{v}_{\text{med}}} = -\sum_i \mathbf{n}_i. \tag{5}$$

Thus the condition that the gradient be zero at the median, if it is not a data point, is that the unit vectors sum to zero.

The Hessian can be computed from (3), giving

$$\mathbf{H}(\mathbf{v}_{\text{med}}) = \sum_i \frac{1}{r_i}\Big(\mathbf{I} - \mathbf{n}_i\mathbf{n}_i^T\Big). \tag{6}$$

Note that each term of J can be imagined as a cone, so that the curvature of that term is zero in the direction of its gradient and positive in any other direction. The curvature at any non-data point $\mathbf{x}$ is therefore positive in all directions *unless* all data points and $\mathbf{x}$ are colinear.

Furthermore, the iteration (4) is a form of gradient descent. If we define a modified Hessian, $\tilde{\mathbf{H}} = \mathbf{I}\sum_i 1/r_i$, related to the Hessian by dropping the second term in (6), we can be rewrite (4) as

$$\mathbf{v}_{\text{med}}^{t+1} - \mathbf{v}_{\text{med}}^{t} = -\tilde{\mathbf{H}}^{-1}\frac{\partial J}{\partial \mathbf{v}_{\text{med}}}. \tag{7}$$

In particular each step is parallel to the gradient and downhill.

It is interesting that using the full Hessian, i.e., Newton's method, can cause divergence. For example, suppose that an iterate is far outside the convex hull around the data. The full Hessian is ill-conditioned, since J increases linearly as we move away from the data. A step of Newton's method would move the iterate far beyond the data points.

By contrast, the IVM algorithm behaves well. To see this, note that another way to represent a step is

$$\mathbf{v}_{\text{med}}^{t+1} - \mathbf{v}_{\text{med}}^{t} = H(r)\langle\mathbf{n}\rangle, \tag{8}$$

where $H(.)$ is the harmonic mean, and $\langle.\rangle$ is the arithmetic mean. With $\mathbf{v}_{\text{med}}^{t}$ far from the data points, $\langle\mathbf{n}\rangle$ points toward the cloud and its magnitude is nearly one, while $H(r)$ is between the smallest and largest distance to a data point. Thus one step moves the estimate close to the cloud of data points.

In fact the IVM algorithm moves any point into the convex hull of the data. Write the sums as integrals over space, with point density $\rho(\mathbf{x})$. Choose coordinates with origin at the current iterate and arbitrary axes. To integrate out the coordinates besides x_1, let $\sigma(x_1) = \int d^{D-1}x\, \rho(\mathbf{x})/\|\mathbf{x}\|$. We get

$$\begin{aligned} H(r) &= \Big[\int d^D x \frac{\rho(\mathbf{x})}{\|\mathbf{x}\|}\Big]^{-1} \\ &= \Big[\int dx_1 \sigma(x_1)\Big]^{-1} \qquad \text{and} \end{aligned} \tag{9}$$

$$\langle\mathbf{n}\rangle_1 = \int d^D x \frac{\rho(\mathbf{x})}{\|\mathbf{x}\|}\mathbf{x} = \int dx_1\, x_1 \sigma(x_1). \tag{10}$$

The IVM algorithm steps along the chosen direction by

$$(\mathbf{v}_{\text{med}}^{t+1} - \mathbf{v}_{\text{med}}^{t})_1 = \frac{\int dx_1\, x_1 \sigma(x_1)}{\int dx_1 \sigma(x_1)}. \tag{11}$$

Since $\sigma(x_1)$ is non-negative and zero outside the data, this has a value within the extremes of x_1 in the data set, so the new iterate lies within a slab that tightly bounds the data in the

chosen direction. Since this is true for any choice of direction, the new iterate lies within the intersection of all such slabs, which is the convex hull of the data.

It is more difficult to analyze the behavior inside the convex hull, except for some special cases. In the following section we discuss the behavior near data points, and fixed points that are not data points.

2.2. Convergence

Here we discuss the convergence properties of the IVM algorithm. The most basic property is that J is convex, which implies that all minima are global, and the set of minima is convex. We now determine the number of minima, and show that the IVM algorithm converges to one of them.

If the data points are not colinear, the Hessian (6) is positive definite everywhere except at data points, where it is not defined. This is because the individual terms are positive-semidefinite, and with non-colinear points the null directions of different terms are not the same. Since J is convex, it is not possible to have more than one minimum in the non-colinear case. Even in the colinear case, it is easy to see that there is only one minimum if the number of data points is odd. With an even number of data points, the set of global minima is a closed line segent with two data points as endpoints.

In the following, we assume that the data points are not colinear. Likewise, for subsets of data points, we assume that the iterate is not colinear with the points in the subset. Under these assumptions the Hessian is always positive-definite.

Now consider an iterate near a data point. Let this be $\mathbf{v}_{\text{med}}^t = \mathbf{v}_j + \boldsymbol{\epsilon}$, for some j, where $\boldsymbol{\epsilon}$ is a small vector. We can show that the iteration gives

$$\mathbf{v}_{\text{med}}^{t+1} = \mathbf{v}_j + \epsilon \sum_{i \neq j} \mathbf{n}_{ij} + O(\epsilon^2), \tag{12}$$

where $\mathbf{n}_{ij} = (\mathbf{v}_i - \mathbf{v}_j)/\|\mathbf{v}_i - \mathbf{v}_j\|$ and $\epsilon = \|\boldsymbol{\epsilon}\|$. Thus the new distance from $\mathbf{v}_j$ is

$$\|\mathbf{v}_{\text{med}}^{t+1} - \mathbf{v}_j\| = \epsilon \left\| \sum_{i \neq j} \mathbf{n}_{ij} \right\| + O(\epsilon^2). \tag{13}$$

If $\| \sum_{i \neq j} \mathbf{n}_{ij} \|$ is less than one, the distance from $\mathbf{v}_j$ decreases, so $\mathbf{v}_j$ is an attractor, and convergence to it is exponential. If $\| \sum_{i \neq j} \mathbf{n}_{ij} \|$ is greater than one, $\mathbf{v}_j$ is a repellor.

In the rare case that $\| \sum_{i \neq j} \mathbf{n}_{ij} \|$ is exactly one, we can still determine the behavior. If we assume that $\mathbf{v}_j$ is not colinear with the other data points, the Hessian near $\mathbf{v}_j$ is positive-definite. The gradient at $\mathbf{v}_j$ due to the other points ($-\sum_{i \neq j} \mathbf{n}_{ij}$) has magnitude one, and the gradient due to $\mathbf{v}_j$ has magnitude one, so the total gradient of J approaches zero as we approach $\mathbf{v}_j$ from the direction of $\sum_{i \neq j} \mathbf{n}_{ij}$. However, because the Hessian is positive-definite the total gradient points toward $\mathbf{v}_j$ and is *not* zero. Because each step is proportional to the negative gradient, the iteration moves toward $\mathbf{v}_j$. We can show that this converges as $1/t$.

Fig. 1. Filtering a noisy image: (top) original corrupted with speckle (multiplicative) noise, level = 0.20; (middle) result of the IVMF; (bottom) result of the standard vector median filter. Both median filter results used 5x5 windows.

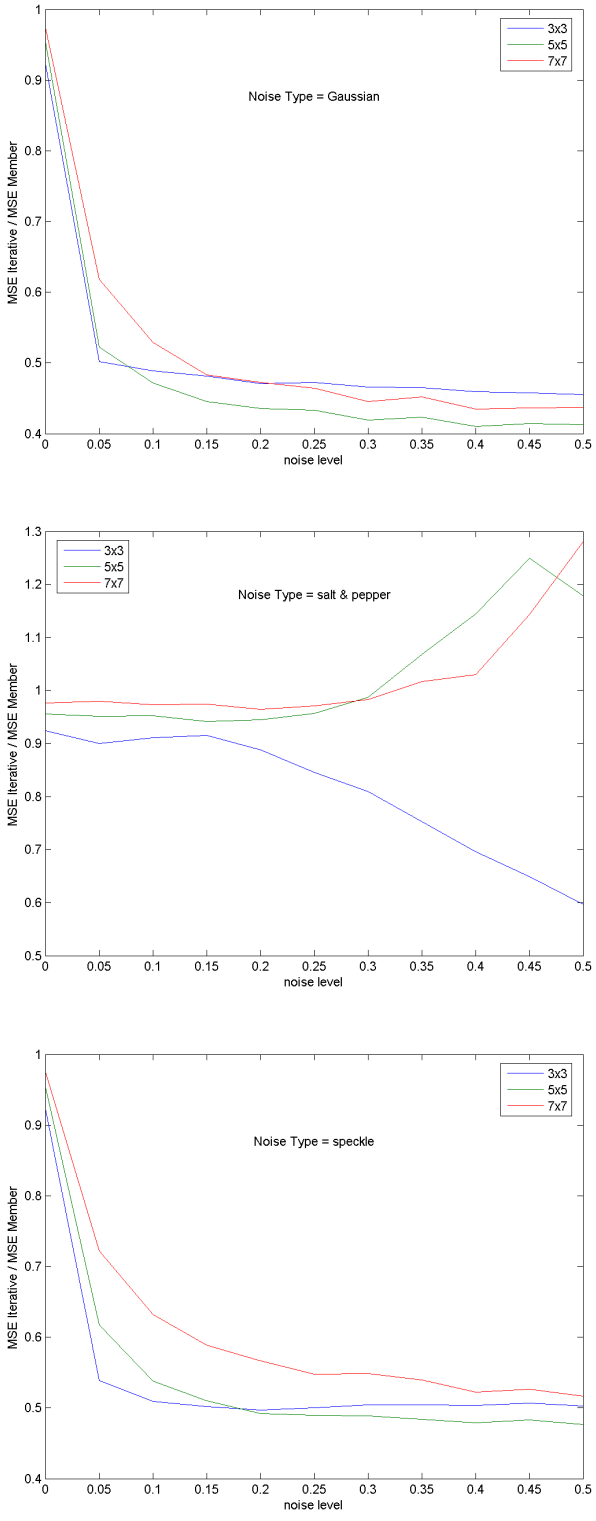

Fig. 2. IVMF MSE divided by standard vector median MSE versus noise level for three noise types: (top) Gaussian; (middle) salt-and-pepper; (bottom) speckle. The three curves in each graph are for window sizes of 3x3, 5x5, and 7x7.

Consider now a non-data fixed point $\mathbf{v}^*_{\text{med}}$. Again, let $\mathbf{v}^t_{\text{med}} = \mathbf{v}^*_{\text{med}} + \boldsymbol{\epsilon}$, and expand the formula for the iteration in powers of ϵ. We can show from this that

$$\|\mathbf{v}^{t+1}_{\text{med}} - \mathbf{v}^*_{\text{med}}\| \leq \epsilon + O(\epsilon^2). \tag{14}$$

Equality holds only if the points are colinear with $\boldsymbol{\epsilon}$. Thus any non-data fixed point is an attractor, as long as the data points are not colinear. Furthermore, convergence is exponential

3. EXAMPLE

The IVMF can be used to smooth any vector field. We have applied it to smoothing 3D LIDAR point clouds and surface normals. Here, we apply it to color images. The objective function is $J = \sum_{x,y \in R} \| c(x,y) - c_{\text{med}}(x,y) \|$, where $c(x,y)$ is a vector representing the color of the pixel at image coordinates (x,y) within the neighborhood R. Again, the IVMF is applied over local regions, in this case in the image plane.

We compare the IVMF with standard vector median filtering on a 256x256 color Lena image corrupted with three types of noise: Gaussian, salt-and-pepper (shot), and multiplicative (speckle), at a range of noise levels. Example results are shown in Figure 1, and plots of the ratio of the mean-squared error (MSE) of the IVMF to that of the standard vector median filter are shown in Figure 2. The noise types and values of the noise levels correspond to arguments to the Matlab® function `imnoise`. In most cases the IVMF is significantly better, except for salt-and-pepper noise and large filter windows. These results suggest that the standard vector median can be better for impulse noise, whereas the IVMF is better for other types. Like the standard approach, the IVMF is robust to outliers and tends to preserve edges.

4. REFERENCES

[1] J.T. Astola and E.R. Dougherty, *Nonlinear Filters for Image Processing*, SPIE Optical Engineering Press, 1999.

[2] J.K. Romberg, Hyeokho Choi, and R.G. Baraniuk, "Bayesian tree-structured image modeling using wavelet-domain hidden Markov models," *IEEE Trans. Image Proc.*, vol. 10, no. 7, pp. 1056–1068, July 2001.

[3] J. Portilla, V. Strela, M.J. Wainwright, and E.P. Simoncelli, "Image denoising using scale mixtures of Gaussians in the wavelet domain," *IEEE Trans. Image Proc.*, vol. 12, no. 11, pp. 1338–1351, Nov. 2003.

[4] J. Astola, P. Haavisto, and Y. Neuvo, "Vector median filters," *Proceedings of the IEEE*, vol. 78, no. 4, pp. 678–689, April 1990.

[5] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes in C*, Cambridge University Press, 1988.