FAST IMPLEMENTATION OF VECTOR DIRECTIONAL FILTERS

M. Emre Celebi

Department of Computer Science
Louisiana State University, Shreveport, LA, USA
ecelebi@lsus.edu

ABSTRACT

Vector filters based on order-statistics have proved successful in removing impulsive noise from color images while preserving edges and fine image details. Among these filters, the ones that involve the cosine distance function (directional filters) have particularly high computational requirements, which limits their use in time critical applications. In this paper, we introduce two methods to speed up these filters. Experiments on a diverse set of color images show that the proposed methods provide substantial computational gains without significant loss of accuracy.

Index Terms— Noise removal, impulsive noise, order-statistics, vector directional filter, minimax

1. INTRODUCTION

Color images are often contaminated with noise, which is introduced during acquisition or transmission. In particular, the introduction of impulsive noise into an image not only lowers its perceptual quality, but also makes subsequent tasks such as edge detection and segmentation more difficult. Therefore, the removal of such noise is often an essential preprocessing step in many color image processing applications.

Numerous filters have been proposed for the removal of impulsive noise from color images [1, 2, 3, 4]. Among these, nonlinear filters have proved successful in removing the noise while preserving the edges and fine details. The early approaches to nonlinear filtering of color images often involved the application of a scalar filter to each color channel independently. However, since separate processing ignores the inherent correlation between the color channels, these methods often introduce color artifacts to which the human visual system is very sensitive. Therefore, vector filtering techniques that treat the color image as a vector field and process color pixels as vectors are more appropriate. An important class of nonlinear vector filters is the one based on order-statistics with the vector median filter (VMF) [5], the basic vector directional filter (BVDF) [6], and the directional-distance filter (DDF) [7] being the most widely known examples. These filters involve reduced ordering [8] of a set of input vectors within a window to determine the output vector. The ordering is often achieved using a combination of two functions: Minkowski distance and cosine distance. VMF and its derivatives use the former, whereas the VDF family (BVDF and its derivatives) use the latter. The DDF family uses a combination of the two functions. In this paper, we refer to the filters that use the cosine distance function, i.e. the members of VDF and DDF families, as directional filters.

Several researchers have noted the high computational requirements of order-statistics based vector filters; however, relatively few studies have focused on alleviating this problem. Furthermore, the scope of these studies was limited to VMF [9, 10]. In a recent study [4], we compared 48 order-statistics based vector filters and concluded that some of the most effective filters are based on the cosine distance function. In fact, one of these filters, namely the adaptive center weighted directional distance filter [11], was shown to be the most effective filter. It was also shown that the directional filters are significantly slower than those based on the Minkowski distance.

In this paper, we introduce two techniques to speed up the order-statistics based directional filters. The rest of the paper is organized as follows. Section 2 introduces the notation and describes the techniques to speed up the cosine distance function. Section 3 presents the experimental results. Finally, Section 4 gives the conclusions.

2. PROPOSED IMPLEMENTATIONS

Consider an $M \times N$ red-green-blue (RGB) input image $\mathbf{X}$ that represents a two-dimensional array of three-component vectors $\mathbf{x}(r,c) = [x_1(r,c), x_2(r,c), x_3(r,c)]$ occupying the spatial location (r,c), with the row and column indices $r = \{1, 2, \ldots, M\}$ and $c = \{1, 2, \ldots, N\}$, respectively. In the pixel $\mathbf{x}(r,c)$, the $x_k(r,c)$ values denote the red ($k = 1$), green ($k = 2$), and blue ($k = 3$) component. In order to isolate small image regions, each of which can be treated as stationary, and reduce processing errors by operating in such a localized area of the input image, an $\sqrt{n} \times \sqrt{n}$ supporting window $W(r,c)$ centered on pixel $\mathbf{x}(r,c)$ is used. The window slides over the entire image $\mathbf{X}$ and the procedure replaces the input

This publication was made possible by a grant from The Louisiana Board of Regents (LEQSF2008-11-RD-A-12).

vector $\mathbf{x}(r,c)$ with the output vector $\mathbf{y}(r,c) = F(W(r,c))$ of a filter function $F(\cdot)$ that operates over the samples inside $W(r,c)$. Repeating the procedure for each pair (r,c), with $r = \{1,2,\ldots,M\}$ and $c = \{1,2,\ldots,N\}$, produces the output vectors $\mathbf{y}(r,c)$ of the $M \times N$ filtered image $\mathbf{Y}$. Note that for the sake of simplicity, the input vectors inside $W(r,c)$ are re-indexed as $W(r,c) = \{\mathbf{x}_i : i = 1,2,\ldots,n\}$ (see Fig. 1), as commonly seen in the related literature [1, 2, 3, 4]. In this notation, the center pixel in W is given by $\mathbf{x}_C = \mathbf{x}_{(n+1)/2}$ and in the vector $\mathbf{x}_i = [x_{i1}, x_{i2}, x_{i3}]$ with components x_{ik}, the i and k indices denote the block location and color channel, respectively. The VMF family orders the input vectors in a

$$\begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 \\ \mathbf{x}_4 & \mathbf{x}_5 & \mathbf{x}_6 \\ \mathbf{x}_7 & \mathbf{x}_8 & \mathbf{x}_9 \end{bmatrix}$$

Fig. 1. Indexing convention inside a 3×3 window

window according to their relative magnitude differences using the Minkowski distance function. For example, the output of VMF, the most well-known member of its class, is given by the lowest ranked input vector:

$$\begin{aligned} \mathbf{y}(r,c) &= \underset{\mathbf{x}_i \in W(r,c)}{\operatorname{argmin}} \left(\sum_{j=1}^{n} L_p(\mathbf{x}_i, \mathbf{x}_j) \right) \\ L_p(\mathbf{x}_i, \mathbf{x}_j) &= \left(\sum_{k=1}^{3} |x_{ik} - x_{jk}|^p \right)^{1/p} \end{aligned} \tag{1}$$

where L_p denotes the Minkowski distance.

The VDF family operates on the direction of the color vectors with the aim of eliminating vectors with atypical directions. The input vectors in a window are ordered according to their angular differences using the cosine distance function. For example, the output of BVDF, the most well-known member of its class, is the input vector in the window whose direction is the maximum likelihood estimate of the input vector directions [12]:

$$\begin{aligned} \mathbf{y}(r,c) &= \underset{\mathbf{x}_i \in W(r,c)}{\operatorname{argmin}} \left(\sum_{j=1}^{n} A(\mathbf{x}_i, \mathbf{x}_j) \right) \\ A(\mathbf{x}_i, \mathbf{x}_j) &= \arccos\left(\frac{x_{i1}x_{j1} + x_{i2}x_{j2} + x_{i3}x_{j3}}{\|\mathbf{x}_i\|_2 \|\mathbf{x}_j\|_2} \right) \end{aligned} \tag{2}$$

where $A(\mathbf{x}_i, \mathbf{x}_j)$ denotes the angle between the two input vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ and $\| \, . \, \|_2$ is the L_2 (Euclidean) norm. Note that in addition to BVDF, the angular function $A(.,.)$ was used in the design of a number of other filters [1, 3, 4].

The DDF family combines the VMF and VDF families by simultaneously minimizing their ordering functions. The output of DDF, the most well-known member of its class, is given by:

$$\mathbf{y}(r,c) = \underset{\mathbf{x}_i \in W(r,c)}{\operatorname{argmin}} \left(\left(\sum_{j=1}^{n} A(\mathbf{x}_i, \mathbf{x}_j) \right) \left(\sum_{j=1}^{n} L_p(\mathbf{x}_i, \mathbf{x}_j) \right) \right) \tag{3}$$

As mentioned in §1, the VDF and DDF family members have much higher computational requirements than the VMF family members. This is due to the computationally expensive cosine distance function $A(.,.)$ used in (2) and (3). For example, on a typical 1024×1024 image, VMF takes about 1.39 seconds, while BVDF and DDF take approximately 27.9 and 28.5 seconds[1], respectively. In the following subsections, we introduce techniques to speed up the directional filters, i.e. the members of the VDF and DDF families.

2.1. Method 1

This method involves approximating the inverse cosine (ARCCOS) function in $A(.,.)$ using a minimax polynomial [13] of degree q:

$$P_q([a,b]) = \left\{ \begin{array}{l} a_0 + a_1 z + \, \ldots \, + a_q z^q : \\ z \in [a,b], a_i \in \mathbb{R}, i = 0,1,\ldots q \end{array} \right\} \tag{4}$$

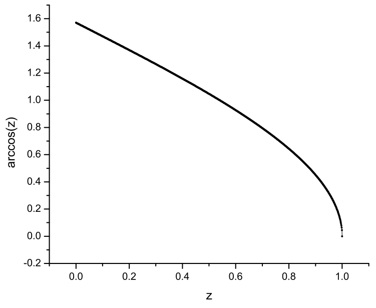


Fig. 2. Inverse cosine function in the interval $[0,1]$

The ARCCOS function takes arguments from the interval $[0,1]$ (see Fig. 2). Unfortunately, approximating ARCCOS over this interval is not easy because of its behavior near 1. This can be circumvented using the following numerically more stable identity for $z \geq 0.5$:

$$\arccos(z) = 2 \arcsin\left(\sqrt{0.5(1-z)} \right) \tag{5}$$

where the inverse sine function (ARCSIN) receives its arguments from the interval $[0, 0.5]$ (see Fig. 3). Instead of plugging the value of $\sqrt{0.5(1-z)}$ into a minimax approximation for the ARCSIN function and then multiplying the result by 2,

[1]Programming language: C, Compiler: gcc 3.4.4, CPU: Intel Pentium D 2.66Ghz

two multiplication operations can be avoided if the following function is approximated:

$$\begin{aligned} &\tau = \sqrt{1 - z} \\ &\arccos(z) = 2 \arcsin\left(\tau / \sqrt{2}\right) \end{aligned} \tag{6}$$

where the argument τ falls into the interval $\left[0,\ 1/\sqrt{2}\right]$.

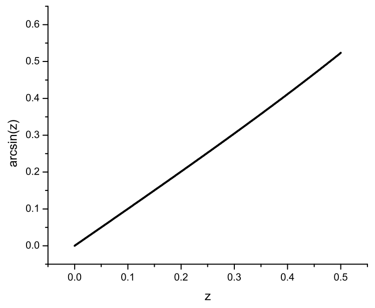


Fig. 3. Inverse sine function in the interval $[0, 0.5]$

The ARCSIN and ARCCOS functions exhibit strong linearity in their respective intervals and therefore can be accurately approximated by polynomials. Fourth degree minimax polynomials for these functions are given by $\arcsin(z) \approx$ 2.097797e-5+1.412840z+1.429881e-2z^2+6.704361e-2z^3+ 6.909677e-2z^4 ($\varepsilon = $ 2.097814e-5) and $\arccos(z) \approx 1.570786-$ 9.990285e-1$z-$1.429899e-2z^2-9.481335e-2z^3-1.381942e-1z^4 ($\varepsilon = $ 1.048949e-5).

2.2. Method 2

This method involves the substitution of the function $A(.,.)$ with a computationally cheaper function $B(.,.)$:

$$\begin{aligned} &B(\mathbf{x}_i, \mathbf{x}_j) = \left(|r_i - r_j|^p + |g_i - g_j|^p + |b_i - b_j|^p\right)^{1/p} \\ &[r_i, g_i, b_i] = \left[\frac{x_{i1}}{\sum_{k=1}^{3} x_{ik}}, \frac{x_{i2}}{\sum_{k=1}^{3} x_{ik}}, \frac{x_{i3}}{\sum_{k=1}^{3} x_{ik}}\right] \end{aligned} \tag{7}$$

Here, $B(.,.)$ is a Minkowski distance function in the chromaticity coordinate space (rgb) [14].

3. EXPERIMENTAL RESULTS

In this section, we evaluate the performance of the proposed methods on a set of test images commonly used in the color image filtering literature. In the experiments, the filtering window was set to 3×3 and whenever the Minkowski distance is involved the L_2 norm was used as commonly seen in the related literature [1, 2, 3, 4].

Table 1. BVDF filtering results at 10% noise level

	Airplane (512 × 512 pixels)			Lenna (512 × 512 pixels)		
Filter	MAE	PSNR	TIME	MAE	PSNR	TIME
$NONE$	6.400	17.865	0.000	6.369	18.195	0.000
$BVDF$	3.273	31.506	10.544	3.929	31.747	10.583
$BVDF_{mmx}$	3.278	31.508	0.694	3.929	31.747	0.697
$BVDF_{rgb}$	3.276	31.496	0.306	3.933	31.732	0.302
	Parrots (1536 × 768 pixels)			Peppers (512 × 480 pixels)		
$NONE$	6.348	18.123	0.000	6.371	17.964	0.000
$BVDF$	0.916	38.993	50.495	2.263	32.538	9.498
$BVDF_{mmx}$	0.969	39.054	4.147	2.277	32.391	0.658
$BVDF_{rgb}$	0.921	38.925	1.786	2.271	32.571	0.294

The corruption in the test images was simulated by the widely used correlated impulsive noise model [15]:

$$\mathbf{x} = \begin{cases} \mathbf{o} & \text{with prob. } 1 - \varphi, \\ \{r_1, o_2, o_3\} & \text{with prob. } \varphi_1 \cdot \varphi, \\ \{o_1, r_2, o_3\} & \text{with prob. } \varphi_2 \cdot \varphi, \\ \{o_1, o_2, r_3\} & \text{with prob. } \varphi_3 \cdot \varphi, \\ \{r_1, r_2, r_3\} & \text{with prob. } (1 - (\varphi_1 + \varphi_2 + \varphi_3)) \cdot \varphi \end{cases} \tag{8}$$

where $\mathbf{o} = \{o_1, o_2, o_3\}$ and $\mathbf{x} = \{x_1, x_2, x_3\}$ represent the original and noisy color vectors, respectively, $\mathbf{r} = \{r_1, r_2, r_3\}$ is a random vector that represents the impulsive noise, φ is the sample corruption probability, and φ_1, φ_2, and φ_3 are the corruption probabilities for the red, green, and blue channels, respectively. In the experiments, the channel corruption probabilities were set to 0.25.

Filtering performance was evaluated using two effectiveness and one efficiency criteria. The effectiveness criteria were the mean absolute error (MAE) [14] and peak signal-to-noise ratio (PSNR) [14], which measure the signal detail preservation and noise suppression capability of a filter, respectively. Note that for the MAE measure lower values are better, whereas for the PSNR higher values are better. The efficiency of a filter was measured by the execution time in seconds.

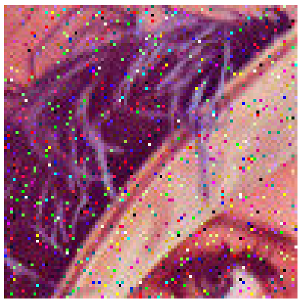
Tables 1-2 show the filtering results for the original BVDF (denoted by $BVDF$), method 1 (denoted by $BVDF_{mmx}$), and method 2 (denoted by $BVDF_{rgb}$). It can be seen that, in terms of filtering effectiveness, $BVDF_{mmx}$ and $BVDF_{rgb}$ consistently performed similar to the $BVDF$. In fact, in some cases, they performed slightly better than $BVDF$. With respect to the execution speed, $BVDF_{mmx}$ and $BVDF_{rgb}$ were up to 15 and 35 times faster than $BVDF$, respectively.

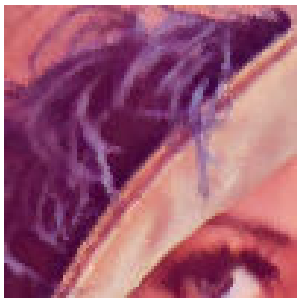
Fig. 4 shows the filtering results for a close-up of the Lenna image. As expected, $BVDF_{mmx}$ and $BVDF_{rgb}$ gave similar results as $BVDF$.

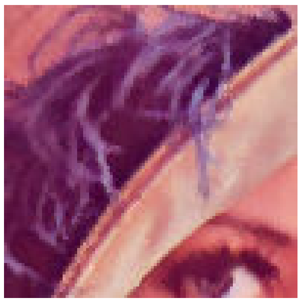
It should be noted that the presented techniques can benefit a multitude of directional filters. For more a comprehensive survey of modern directional filters see [4].

Table 2. BVDF filtering results at 15% noise level

	Airplane (512×512 pixels)			Lenna (512×512 pixels)		
Filter	MAE	PSNR	TIME	MAE	PSNR	TIME
$NONE$	9.553	16.131	0.000	9.560	16.429	0.000
$BVDF$	3.439	30.878	10.481	4.090	31.264	10.491
$BVDF_{mmx}$	3.445	30.876	0.694	4.090	31.264	0.703
$BVDF_{rgb}$	3.443	30.862	0.309	4.093	31.248	0.309
	Parrots (1536×768 pixels)			Peppers (512×480 pixels)		
$NONE$	9.514	16.363	0.000	9.576	16.199	0.000
$BVDF$	1.003	37.935	51.074	2.510	30.976	9.398
$BVDF_{mmx}$	1.058	37.975	4.159	2.531	30.732	0.658
$BVDF_{rgb}$	1.010	37.715	1.789	2.511	31.113	0.288


(a) 10% noisy (MAE 6.369, PSNR 18.195)


(b) $BVDF$ (MAE 3.929, PSNR 31.747)


(c) $BVDF_{mmx}$ (MAE 3.929, PSNR 31.747)

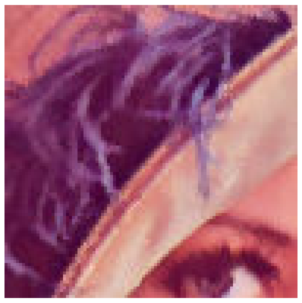

(d) $BVDF_{rgb}$ (MAE 3.933, PSNR 31.732)

Fig. 4. Filtering results for Lenna corrupted with 10% noise

4. CONCLUSIONS

In this paper, we presented two methods to speed up order-statistics based directional filters. Experiments on a diverse set of color images showed that these methods can provide excellent accuracy and high computational gains. The presented approximation methods have applications that go beyond color image filtering including computer graphics and computational geometry.

5. REFERENCES

[1] B. Smolka, K.N. Plataniotis, and A.N. Venetsanopoulos, *Nonlinear Signal and Image Processing: Theory, Methods, and Applications*, chapter Nonlinear Techniques for Color Image Processing, pp. 445–505, CRC Press, 2004.

[2] R. Lukac *et al.*, "Vector Filtering for Color Imaging," *IEEE Signal Process Mag*, vol. 22, no. 1, pp. 74–86, 2005.

[3] R. Lukac and K.N. Plataniotis, *Advances in Imaging & Electron Physics*, chapter A Taxonomy of Color Image Filtering and Enhancement Solutions, pp. 187–264, Academic Press, 2006.

[4] M.E. Celebi, H.A. Kingravi, and Y.A. Aslandogan, "Nonlinear Vector Filtering for Impulsive Noise Removal from Color Images," *J Electron Imaging*, vol. 16, no. 3, pp. 033008, 2007.

[5] J. Astola, P. Haavisto, and Y. Neuvo, "Vector Median Filters," *Proc IEEE*, vol. 78, no. 4, pp. 678–689, 1990.

[6] P.E. Trahanias and A.N. Venetsanopoulos, "Vector Directional Filters: A New Class of Multichannel Image Processing Filters," *IEEE Trans Image Process*, vol. 2, no. 4, pp. 528–534, 1993.

[7] D. Karakos and P.E. Trahanias, "Generalized Multichannel Image Filtering Structures," *IEEE Trans Image Process*, vol. 6, no. 7, pp. 1038–1045, 1997.

[8] V. Barnett, "The Ordering of Multivariate Data," *J R Stat Soc Ser A*, vol. 139, no. 3, pp. 318–355, 1976.

[9] M. Barni, "A Fast Algorithm for 1-Norm Vector Median Filtering," *IEEE Trans Image Process*, vol. 6, no. 10, pp. 1452–1455, 1997.

[10] M. Barni *et al.*, "A Quasi-Euclidean Norm to Speed up Vector Median Filtering," *IEEE Trans Image Process*, vol. 9, no. 10, pp. 1704–1709, 2000.

[11] R. Lukac, "Adaptive Color Image Filtering Based on Center-Weighted Vector Directional Filters," *Multidimens Syst Signal Process*, vol. 15, no. 2, pp. 169–196, 2004.

[12] N. Nikolaidis and I. Pitas, "Nonlinear Processing and Analysis of Angular Signals," *IEEE Trans Signal Process*, vol. 46, no. 12, pp. 3181–3194, 1998.

[13] E.W. Cheney, *Introduction to Approximation Theory*, AMS, Second edition, 2000.

[14] K.N. Plataniotis and A.N. Venetsanopoulos, *Color Image Processing and Applications*, Springer, 2000.

[15] T. Viero, K. Oistamo, and Y. Neuvo, "Three-Dimensional Median-Related Filters for Color Image Sequence Filtering," *IEEE Trans Circuits Syst Video Technol*, vol. 4, no. 2, pp. 129–142, 1994.