HARDWARE IMPLEMENTATION OF THE MEDIAN-RATIONAL HYBRID FILTERS FOR COLOUR IMAGES

Lazhar Khriji, Moncef Gabbouj

TICSP, T.U.T., P.O. Box 553 FIN-33
101 Tampere, Finland
e-mail: lazhar@cs.tut.fi

Giuseppe Bernacchia, Giovanni L. Sicuranza

D.E.E.I., Univ. of Trieste, via A. Valerio 10
I-34127 Trieste, Italy
e-mail: bernac@ipl.univ.trieste.it

ABSTRACT

A new class of nonlinear Filters called Vector Median Rational Hybrid Filter (VMRHF) [1][2] has been recently introduced and applied to colour image filtering problems. As shown in these papers, hybrid filters exhibit very desirable filtering features which often outperform those of the components of the hybrid filter itself. The VMRHF filter is a two-stage filter, which exploits the features of the vector-median filter and those of the rational operator. The performances of the proposed filter are compared with those of the Vector Median Filter (VMF), described in [3], and of the Directional-Distance Filter (DDF), reported in [4]. In this paper we also present a hardware implementation of the VMRHF which exploits in an effective way the features and the robustness of both median filters and rational filters. This architecture is suitable for the use in real-time applications due to its reduced hardware complexity.

1. MEDIAN-RATIONAL HYBRID FILTERS

The Vector Median Rational Filter is defined as follows:

Definition 1.1 *the output vector $\underline{y}(\underline{\mathbf{f}}_i)$ of the VMRHF, is the result of a vector rational function taking into account three input sub-functions which form an input function set $\{\underline{\mathbf{\Phi}}_1, \underline{\mathbf{\Phi}}_2, \underline{\mathbf{\Phi}}_3\}$, where the "central one" ($\underline{\mathbf{\Phi}}_2$) is fixed as a center weighted median sub-filter, and given by*

$$\underline{y}(\underline{\mathbf{f}}_i) = \underline{\mathbf{\Phi}}_2(\underline{\mathbf{f}}_i) + \frac{\sum_{j=1}^{3} \alpha_j \underline{\mathbf{\Phi}}_j(\underline{\mathbf{f}}_i)}{h + k \parallel \underline{\mathbf{\Phi}}_1(\underline{\mathbf{f}}_i) - \underline{\mathbf{\Phi}}_3(\underline{\mathbf{f}}_i) \parallel_2} \quad (1)$$

where $\alpha = [\alpha_1, \alpha_2, \alpha_3]$ characterizes the constant vector coefficient of the input sub-functions and $\| \cdot \|_2$ is the L_2-vector norm.

In this approach, we have chosen a very simple prototype filter coefficients which satisfy the condition: $\sum_{i=1}^{3} \alpha_i = 0$. In our study, $\alpha = [1, -2, 1]^T$. h and k are some positive constants. The parameter k is used to control the amount of the nonlinear effect.

The sub-filters Φ_1 and Φ_3 are chosen so that an acceptable compromise between noise reduction and edge and chromaticity preservation can be achieved. In this specific case both these sub-filters are bidirectional vector median filters. The former acts on a plus-shaped mask, the latter on a cross-shaped one.

The central term is a Center Weighted Vector Median Filter (CWVMF) acting on a plus-shaped mask.

Since the VMF as described in [3] is derived as a maximum likelihood estimates for the exponential distribution, it performs very well when the noise follows a long-tailed distribution. On the other side the rational operator exhibits good filtering capabilities in case of Gaussian noise. Therefore we expected the hybrid filter to perform well in both cases and the experimental results confirm this claim.

2. IMPLEMENTATION

The implementation of the VMRHF is rather complicated due to the amount of calculations required from both the vector median filters and the rational function.

The main driving constraints of this implementation are the chip size and the speed of the circuitry, in order to obtain a real time system which could be implemented on an FPGA for testing purposes.

In order to achieve a high throughput we used a pipelined structure, inspite of the number of memory units required by this kind of architecture. Therefore the design of the implemented system is a compromise between size and speed to fit in FPGA's.

Since implementations of the median filter are widely

reported in literature, in this work we mainly focus on the problems introduced by the rational function. In this Section we discuss our choices for the implementation of this operator.

As it can be seen from (1), the used function is very simple. The amount of hardware required for an accurate implementation is quite large though. This fact is due to the use of the L_2 norm, which requires the computation of the function

$$y = \sqrt{(\Phi_{11} - \Phi_{31})^2 + (\Phi_{12} - \Phi_{32})^2 + (\Phi_{13} - \Phi_{33})^2}$$

and to the division required by the rational function.

Previous works [5] on rational functions show that these operators are very robust with respect to errors in the calculation of both numerator and denominator. Thus we can exploit this feature in order to reduce the complexity of the system itself.

The first approximation we introduce is on the computation of $\| \underline{\Phi}_1 - \underline{\Phi}_3 \|_2$ by mean of a simple linear function:

$$f_a(\underline{\Phi}_1, \underline{\Phi}_3) = \frac{|\Phi_{11} - \Phi_{31}| + |\Phi_{12} - \Phi_{32}| + |\Phi_{13} - \Phi_{33}|}{c} \qquad (2)$$

where $c = 1, \frac{3}{4}, 2$ depending if respectively two, one or none of the addends are equal to 0. This is a coarse approximation which allows us to still have a good estimate for the L_2 value, but which is much simpler to implement.

The second approximation has been introduced in order to simplify the evaluation of the division. Before explaining the approximation itself, we need to consider the features of the specific rational function used. In (1) the parameters h and k are user-specified depending on the application. For our purposes we can rewrite the former equation for the i-th component in the following way:

$$y_i = k_1 \frac{\Phi_{1i} + \Phi_{3i} - 2\Phi_{2i}}{w + \| \underline{\Phi}_1 - \underline{\Phi}_3 \|_2} \qquad (3)$$

with

$$k_1 = 1/k \quad w = h/k.$$

Following the example in [6] the values of k_1 and w have been chosen as 100 and 625, respectively.

The values of the median operators are bounded to be in the interval [0,255], thus the numerator $\Phi_{1i}+\Phi_{3i}-2\Phi_{2i}$ can assume only values in the range [-510,510]. On the other side, the use of (2) leads us to bound the value of the denominator in the range [625,1008].

The key point of our approximation is the use of a scale-down-by-16 algorithm, which is equivalent to a simple binary shift and thus it doesn't require any hardware. If we want to round off the obtained value, then we need an extra adder, but this time the number of bits involved is smaller.

The previous observations allow us to say that for the denominator we need always two scale-down steps in order to bound its value in the range [0,16]. Moreover, the effect of the scaling by 16 is to further restrict this range to [2,4], and this will be very useful when performing the division.

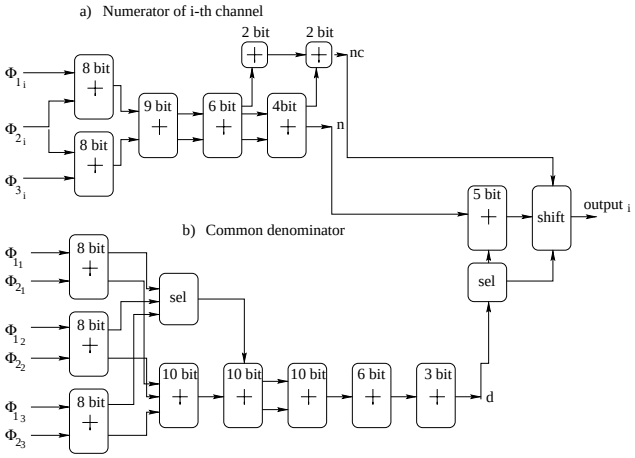

Figure 1: Structure of the rational function for the i-th component.

The number nc of scale steps required for the numerator is variable between 0 and 2, but again the obtained value n is bounded in the interval [0,16]. After the scalings we can rewrite (3) as follows:

$$y_i = k_1 \frac{n 16^{nc}}{d 16^2} = k_1 \frac{n}{d} 16^{nc-2} \qquad (4)$$

A further simplification can be done if we slightly modify the value of k_1 (and consequently that of w) so that we obtain a multiple of 16. In this specific case we choose $k_1 = 96 = 6 \times 16$ and $w = 600 = 6.25 \times 96$. We can exploit this particular choice for the parameters observing that d can be only 2,3 or 4 and therefore the ratio k_1/d can assume only the values 3×16 , 2×16 , $\frac{3}{2} \times 16$ and the expression (4) becomes

$$y_i = \frac{p \times n \times 16^{nc-1}}{q} \qquad (5)$$

with

$$p \in \{2,3\} \quad q \in \{1,2\} \quad (nc-1) \in \{-1,0,+1\}.$$

Using this approximation we can reduce the division to a sequence of binary shifts and sums for the round offs.

In Figure 1 we show a schematic representation of our implementation for the i-th component. The 5-bit adders include the shift by 16 and a demultiplexer in the case of the numerator.

3. RESULTS

The implemented system has been tested using colour images in 8-bit RGB format corrupted by adding simple impulsive, simple Gaussian or contaminated Gaussian noise.

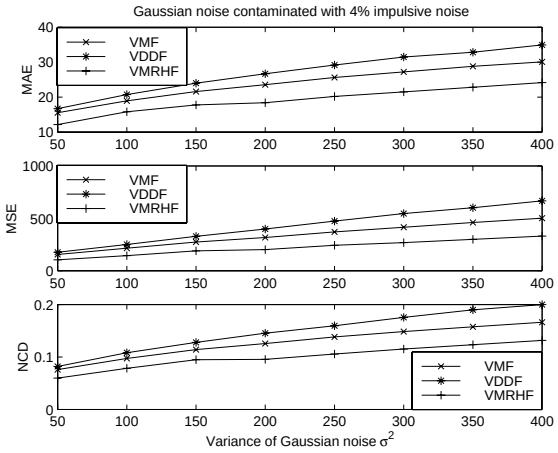

Figure 2: Comparative results between Vector Median Hybrid Filter (VMRHF), Vector Median Filter (VMF) and Directional-Distance Filter (DDF).

The test set for the comparisons with the mentioned VMF and DDF includes 16 test images with pure impulsive noise with percentage λ from 0.5 to 4, 16 images corrupted with Gaussian noise with σ^2 ranging from 50 to 400 and 16 images corrupted with contaminated Gaussian. In Figure 2 we report some comparative results for the proposed algorithm VMRHF with respect to the Vector Median Filter (VMF) and the Direction-Distance Filter (DDF) for the case $\lambda = 4\%$.

The common measures MSE and MAE used to compare the output image of a filter with the original one are not very appropriate to quantify the perceptual error between images. Experimental and comparative results in colour image filtering show that a very good performance measure can be obtained when the error is measured in the $L^*a^*b^*$ space. $L^*a^*b^*$ is known as a space where equal colour differences result in equal distances, and thus it is close to the human perception of colors. Therefore, in all our tests in addition to the MAE and MSE measures we used a normalized colour difference (NCD) [7], computed according to the following formula:

$$NCD = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} \|\triangle E_{Lab}\|}{\sum_{i=1}^{M}\sum_{j=1}^{N} \|E^*_{Lab}\|} \qquad (6)$$

where $\triangle E_{Lab}$ is the perceptual colour error between two colour vectors and E^*_{Lab} is the magnitude of the original image pixel vector in the $L^*a^*b^*$ space.

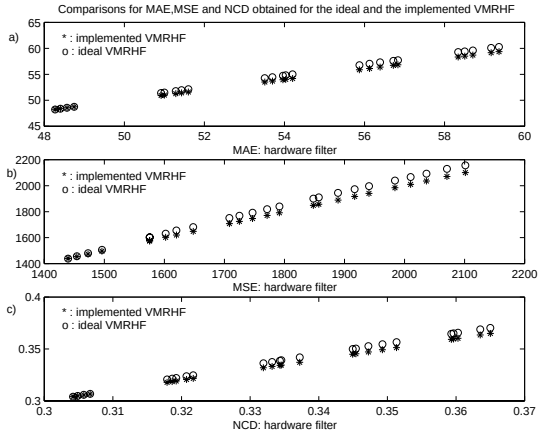

Figure 3: Comparative results between the theoretical VMRHF and the implementation.

As it can be seen from Figure 2, the proposed filter outperforms the other filters for all the performance measures.

We used the same test sets to compare the performances of the ideal VMRHF algorithm and those of the implemented system. The results are shown in Figure 3, and partially reported in Table 1, as performance measures with respect to the original images. In this figure we plot the results obtained for all the 16 images in the case of ideal VMRHF (marked with 'o') with respect to those obtained in the case of the implemented system (marked with '*'). We can see that in all the cases the implemented system outperforms the ideal one. This fact may be considered quite surprising according to the approximations we introduced.

An explanation of this behavior can be obtained looking at the comparison between the approximated and the ideal division, reported in Figure 4. The curves

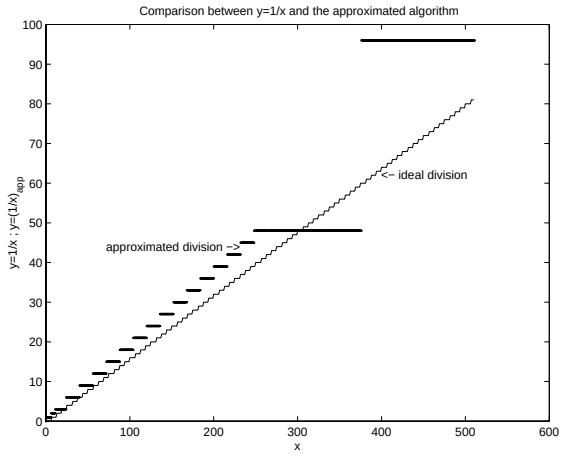

Figure 4: Comparative results between the approximated and the ideal division.

shown are relative to the division

$$\frac{k_1 \times n}{w + f_a}$$

for $n \ \epsilon\{0, 510\}$ and $f_a \ = 0$, but the results are the same for the other values of f_a.

We can see that the approximated algorithm outputs values higher than the ideal division, and that the differences increase as the numerator n increases. When the noise is high the values of the three median filters are very different and likely the numerator in (1) becomes large in absolute value. The output of the rational function modifies Φ_2, the output of the central weighted median filter, which is assumed as a good estimate for the actual value of the central pixel in the mask. Since the output of the approximated division is generally higher than that of the ideal division, Φ_2 is modified more than in the ideal case and this causes a better filtering. This fact is essentially demonstrated by simulations done with the ideal function. From the data shown in Figure 3 a ratio of about 1.2 has been found between the outputs of the approximated division and those of the ideal one. If we modify k_1 in (1) so that $k_1^* = 1.2 \times k_1 \ = 115$, the results of the ideal function become very close to those of our implementation. The slightly differences still existing between the two sets of data are probabily due to the saturation effect of the approximation, which prevents a too large modification of Φ_2 when the rational function outputs a large value.

Noise		MAE	MSE	NCD
$\lambda = 1$ $\sigma^2 = 100$	Noisy image	55.7	2142	0.31698
	Hardware	50.9	1576	0.31789
	Theoretical	51.4	1602	0.32071
$\lambda = 1$ $\sigma^2 = 200$	Noisy image	60.8	2433	0.34662
	Hardware	53.7	1725	0.33320
	Theoretical	54.4	1768	0.33752
$\lambda = 1$ $\sigma^2 = 300$	Noisy image	65.1	2689	0.37124
	Hardware	56.1	1857	0.34542
	Theoretical	57.0	1910	0.35049
$\lambda = 1$ $\sigma^2 = 400$	Noisy image	69.2	2969	0.39830
	Hardware	58.5	2010	0.35964
	Theoretical	59.4	2066	0.36504
$\lambda = 4$ $\sigma^2 = 100$	Noisy image	61.7	3231	0.35492
	Hardware	51.6	1648	0.32170
	Theoretical	52.1	1682	0.32476
$\lambda = 4$ $\sigma^2 = 200$	Noisy image	66.4	3478	0.38296
	Hardware	54.2	1792	0.33723
	Theoretical	55.0	1840	0.34192
$\lambda = 4$ $\sigma^2 = 300$	Noisy image	70.6	3739	0.40706
	Hardware	56.8	1941	0.35133
	Theoretical	57.7	1997	0.35664
$\lambda = 4$ $\sigma^2 = 400$	Noisy image	74.6	3986	0.43186
	Hardware	59.4	2101	0.36502
	Theoretical	60.3	2157	0.37040

Table 1: Quantitative comparison between theoretical algorithm and implemented system.

As it can be seen from Figure 1 the structure of the rational function is very simple: it requires only adders and a few extra circuitries for signal conditioning, like demultiplexers. Therefore it is very suitable for an implementation with FPGA. In fact the FPGA's usually support optimized macros for conventional binary adders, but they don't allow to handle very well general datapath structures like user-specified boolean functions. Therefore a design with only adders and multiplexers can be effectively implemented on FPGA and relatively high speed can be achieved even with low-performance devices. Moreover the small amount of blocks required by our proposed implementation makes a pipelined architecture easy to implement, allowing a high throughput. For instance, if we choose the 8-bit adder as a reference for the speed, a clock up 40 MHz can be obtained with a small-size FPGA, corresponding to 40 millions of output RGB vectors per second using the pipeline architecture.

4. CONCLUSIONS

In this paper we present a new class of nonlinear filters which are able to remove different kinds of additive i.i.d. noise. Moreover, we present a hardware implementation for such operators. Our tests have shown that the implemented system is very robust and that the relative small dimensions of its structure make it very suitable for direct implementations on FPGA's.

5. REFERENCES

[1] L. Khriji and M. Gabbouj, "Median-Rational Hybrid Filters for image restoration", *IEE Electronics Letters*, vol.34, no.10, pp.977-979, May 1998.

[2] L. Khriji and M. Gabbouj, "Median-Rational Hybrid Filters",*Intern. Conf. on Image Processing ICIP'98*, Chicago, Illinois, USA. October 4-7, 1998.

[3] J. Astola, P. Haavisto, Y. Neuvo, "Vector Median Filter",Proc. of the IEEE, vol.78, pp.678-689, April 1990.

[4] P.E. Trahanias, D. Karakos, A.N. Venetsanopoulos,"Directional Processing of colour Images:Theory and Experimental Results",*IEEE Trans. on Image Processing*, Vol.5, no.6, pp. 868-880, June 1996.

[5] G. Bernacchia, S. Marsi, "A VLSI implementation of a configurable rational filter", *IEEE Trans. on Consumer Electronics*,Vol. 44, no. 3, August 1998.

[6] G. Ramponi, "The Rational Filter for Image Smoothing", *IEEE Signal Proc. Letters*, vo. 3, no. 3, pp. 63-65, March 1996.

[7] K.N. Plataniotis, D. Androutsos and A.N. Venetsanopoulos,"Adaptive multichannel filters for Color image processing",*Signal Processing: Image Communication*, vol.11, pp.171-177,(1998).