

A DEDICATED HARDWARE SYSTEM FOR A CLASS OF NONLINEAR ORDER STATISTICS RATIONAL HYBRID FILTERS WITH APPLICATIONS TO IMAGE PROCESSING

Lazhar Khriji †, Giuseppe Bernacchia ‡, Moncef Gabbouj † and Giovanni Sicuranza ‡

†TICSP, T.U.T., P.O. Box 553 FIN-33 101 Tampere, Finland. e-mail: (lazhar,moncef)@cs.tut.fi
‡D.E.E.I., Univ. Trieste, via Valerio, 10, 34100 Trieste, Italy. e-mail: bernac@ipl.univ.trieste.it

ABSTRACT

A dedicated hardware system is developed for a recent class of nonlinear hybrid filters called Order Statistics-Rational Hybrid Filters (OSRHF). The performance of these filters is also studied and compared against three effective nonlinear filters from the literature. The application at hand is noise filtering in grey level images. The proposed hardware system uses a residue number system (RNS) to compute the numerator and the denominator of the rational filter. The resulting structure is suitable for direct implementation on FPGAs.

1. INTRODUCTION

Nonlinear filters have been found effective in removing different types of noises from images [1] [8]. This is especially the case if the noise corrupting the image is not Gaussian and/or image details are not to be blurred. For this purpose, a number of order statistics based techniques have been proposed in the literature, we cite among them median, rank order, weighted median, weighted order statistics and stack filters, see [12] for more details.

This paper extends our previous work presented at ICIP'98 [6] where we first proposed the hybrid filter structure. Here, we continue to study the performance of these filters by comparing them against several quite effective nonlinear filters from the literature, stack filters [10], rank-order morphological filters [11], and rational filters [9]. Furthermore, we propose a dedicated hardware system to implement these new nonlinear rational type hybrid filters [5].

Three order statistics subfilters are first computed in the first layer of the hybrid structure. The idea here is to remove impulsive type with minimal distortion of the image details. Order statistics have been found to be very effective in such applications. The final layer of OSRH filters is a rational function. For more details concerning rational functions and their use in filtering, see [9]. The aim of the final layer, in addition to its detail preserving capability, is to attenuate Gaussian type noise. The hybrid approach is therefore suitable

for applications where both impulsive and Gaussian noise are expected.

2. ORDER STATISTICS-RATIONAL HYBRID FILTERS

Definition 2.1 *The output of an Order Statistics-Hybrid Rational Filter (OSRHF) is the result of a rational function operating on three order statistics subfilters $\{\Phi_1, \Phi_2, \Phi_3\}$ and given by*

$$y(n) = \Phi_2(n) + \frac{\sum_{i=1}^3 \alpha_i \Phi_i(n)}{h + k(\Phi_1(n) - \Phi_3(n))^p} \quad (1)$$

The set of coefficient $\alpha = [\alpha_1, \alpha_2, \alpha_3]$ are used as weights for the subfilters' outputs; while, parameters p , h and k are some positive constants. Parameter k plays an important role in rational filtering as it is used to gauge the effect of the nonlinear term in the denominator.

In this paper, we focus on the special case where the center subfilter (Φ_2) is a center weighted median; while, the other two are bidirectional median subfilters. The set of coefficients are selected to satisfy the condition: $\sum_{i=1}^3 \alpha_i = 0$. Here we used $\alpha = [1, -2, 1]^T$. Parameter p is set to 2. The resulting filter is thus called Median-Rational Hybrid Filter (MRHF).

The sub-filters Φ_1 and Φ_3 are chosen so that an acceptable compromise between noise reduction and edge preservation is obtained. It is easy to observe that this MRHF differs from a linear low-pass filter mainly in the scaling, which is introduced on the Φ_1 and Φ_3 terms. Indeed, such terms are divided by a factor proportional to the output of an edge-sensing term characterized by the squared difference between the two sub-filters Φ_1 and Φ_3 . The weight of the median-operation output term is accordingly modified, in order to keep the gain constant. The behavior of the proposed MRHF structure for different positive values of parameter k is as follows:

- $k \simeq 0$, the form of the filter is given as a linear low-pass combination of the three nonlinear sub-functions:

$$\underline{y}(n) = c_1 \Phi_1(n) + c_2 \Phi_2(n) + c_3 \Phi_3(n). \quad (2)$$

where, the coefficients c_1 , c_2 , and c_3 are some constants.

- $k \rightarrow \infty$, the output of the filter is identical to the central sub-filter output and the rational function has no effect:

$$y(n) = \Phi_2(n). \quad (3)$$

- For intermediate values of k , the $(\Phi_1(n) - \Phi_3(n))^2$ term perceives the presence of a detail and accordingly reduces the smoothing effect of the operator.

As a result, a MRHF operates as a linear lowpass filter between three nonlinear sub-operators, the coefficients of which are modulated by the edge-sensitive component.

The proposed structure of the MRHF is shown by Fig. 1 using two bidirectional vector median sub-filters. Only the points indicated in black in each window mask are retained to be used in the corresponding operation. The center weighted median filter Φ_2 has the following filter weights corresponding to the plus-shaped mask.

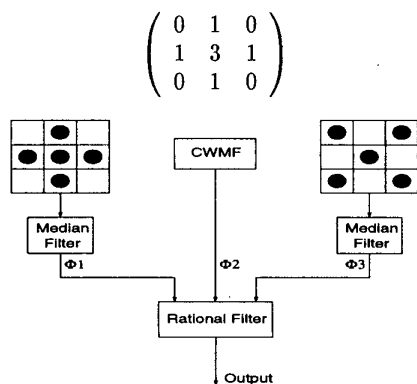


Figure 1: Structure of an instance of the OSRHF using two bidirectional median sub-filters: the Median-Rational Hybrid Filter.

3. HARDWARE IMPLEMENTATION

As it can be seen from its structure, the MRHF could be split in two main sections:

- block with the three median filters, which involve sorting of the incoming samples from the image. The results of these filters are exact
- block which realizes the rational function and therefore requires some arithmetic function.

Each of these sections presents some problem when it has to be implemented. Since median filters and weighted median filters have been widely studied and many references can be found in literature, our attention focuses on the implementation of 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 an high throughput we used a pipelined structure. The drawback of this choice is the need of many memory units in the circuitry and this could be rather difficult to obtain using an FPGA.

3.1. Rational Function

The rational function used in the MRHF is a quite simple one, requiring very few arithmetic operations. Even if the structure is simple, the design of this block presents some problems if we want to achieve compactness in size and high speed.

In this work we decided to use a residue number system (RNS) in order to perform the calculation of both numerator and denominator of the rational operator. For the division we developed a new algorithm derived from [7], which allows us to obtain a good precision in the result with a quite small sized circuitry.

An RNS system is defined by a set of *moduli* m_1, m_2, \dots, m_n . An integer a can be represented in this system by mean of the *residues* modulo m_i : $a = a_1, a_2, \dots, a_n$ where $a_i = a \bmod m_i$.

The number of positive representable integers $M = \prod_{i=1, \dots, n} m_i$. If we want to represent both negative and positive values, then we have:

$$a_i = \begin{cases} a \bmod m_i & \text{if } a \geq 0 \\ (M + a) \bmod m_i & \text{if } a < 0 \end{cases}$$

and the range of representable integers is $[-\frac{M}{2}, \frac{M}{2}]$, if M is even and $[-\frac{M-1}{2}, \frac{M-1}{2}]$, if M is odd.

For this specific application we used three modules: $m_1 = 15, m_2 = 31$ and $m_3 = 32$, therefore we can represent integers in the range $[0, 14880]$ or $[-7440, 7440]$. As we can see from 1, if p is even the denominator is always positive and the results has the same sign of the numerator. Therefore we can simply consider the results as the ratio of two positive numbers and propagate the sign to the last addition. The advantage of this approach is that the range at our disposal to represent the numbers is twice as much as the one for signed integers. In any case the presence of the square function implies that a scaling algorithm is necessary in order to calculate numerator and denominator.

In Fig. 2 the block diagram of the circuitry for the rational function is shown. Let us now briefly describe the implemented functions.

3.1.1. Addition/Subtraction

In our system we use both standard binary and RNS adders. The first ones are used to compute the two expressions $\Phi_1 + \Phi_3 - 2\Phi_2$ and $\Phi_1 - \Phi_3$ and allow us to compare these quantities with the references for

the scaling using normal binary comparators instead of RNS ones. The delay of these adders is the time reference for the design of the other blocks. The RNS adders are used for the binary to RNS converter, for the final sum at the denominator, for the scaling algorithm and for the multipliers.

With the given modules each residue can be represented at last with 5 bits; therefore the adders can be implemented in a very effective way and can be merged with other blocks in order to have a more compact circuit and use less memory elements for the pipeline.

In fact the gain in speed achievable using the FPGA is not so high as it can be expected from the reduced number of bits in the operations. This is because the FPGA's have a fixed inner structure which obviously is not very flexible if we want to implement non-standard structures like in the RNS systems. As an example, a 8 bit binary subtractor requires 6 Configurable Logic Blocks (CLB) and can be operated at about 58 MHz. On the other hand the implemented modulo 31 adder/subtractor requires 8 CLB's and can be clocked at slightly more than 66 MHz.

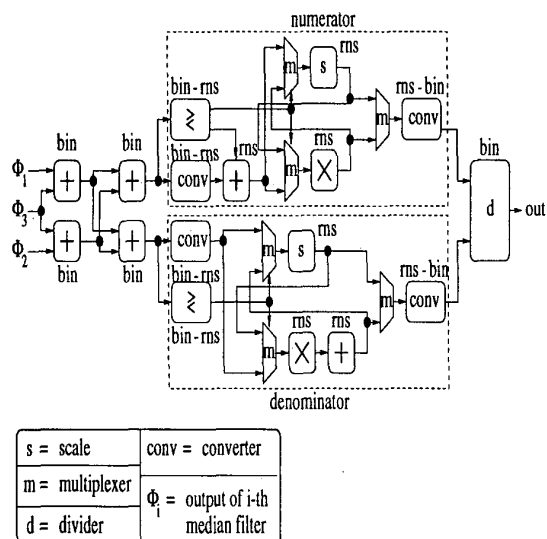


Figure 2: Block diagram of the implementation of the rational function.

3.1.2. Multiplication

Let us remember that for an RNS system operations like addition/subtraction and multiplication are split in many blocks (one per module) and evaluated in parallel. Since the RNS adders are very small in size, we decided to implement the multiplier with the shift-and-add algorithm. For each module just three adders are required and the result is exact. Moreover, since the product of two numbers in RNS presents a high regularity, the multiplication can be avoided in the implementation of square function or in general of power

functions. As an example the three blocks performing the square function in the chosen modules set require 1 CLB each and can be clocked at almost 100 MHz.

3.1.3. Scaling

Given the set of modules $\{m_i\}_{i=1,\dots,n}$, an integer a can be represented in the so called Mixed Radix System (MRS) in the following way:

$$a = \alpha_1 + \alpha_2 m_1 + \alpha_3 m_1 m_2 = (\alpha_1, \alpha_2, \alpha_3) \quad (4)$$

where α_i can be directly obtained from the residues m_i .

Let us suppose now to scale the number a by the module m_1 . Then we can write:

$$a' = \frac{a}{m_1} = \frac{\alpha_1}{m_1} + \alpha_2 + \alpha_3 m_2$$

Since $\alpha_1 < m_1$ the ratio $\frac{\alpha_1}{m_1} < 1$. The residues of the scaled integer a' can be calculated as follows:

$$\begin{aligned} a'_1 &= (r + \alpha_2 + \alpha_3) \bmod m_1 \\ a'_2 &= (r + \alpha_2) \bmod m_2 \\ a'_3 &= (r + \alpha_2 - \alpha_3) \bmod m_3 \end{aligned}$$

where $r = 0, 1$ depending if $\alpha_1 < \frac{m_1}{2}$ or $\alpha_1 \geq \frac{m_1}{2}$ and is used to lower the truncation error introduced by the scaling. Exploiting the features of the chose set of modules the scaling process requires three clock cycle and the hardware consists just on adders.

3.1.4. Conversion

One of the main problems of non standard arithmetic systems is the conversion from binary to the required system and vice-versa.

Many algorithms have been found ([3]) which can be easily implemented in order to perform a fast conversion from binary to RNS. In our implementation the conversion reduces to a simple RNS addition exploiting the features of module's systems of type $(2^k - 1, 2^k, 2^k + 1)$.

The conversion from RNS to binary is slightly more difficult and usually a conversion to MRS is exploited. This is the algorithm we choose for our system, since we use the block for the conversion from RNS to MRS for the scaling. The final conversion is obtained as sum of product like in eq. (4) and the output is represented with 13 bits.

3.1.5. Division

In this design the role of the division is the most critical, since this operation is highly time and size consuming. Therefore we developed a new division algorithm which can be implemented both in RNS and in standard binary.

Since it involves a few comparisons, we choose to implement the algorithm in standard binary arithmetic, even if it allows to skip the RNS to binary conversion when implemented in RNS.

The basic idea is the following. Given a range M we can find $k = \lceil \log_2 M \rceil$ powers of 2 belonging to the interval $[0, M]$.

If x and y are divisor and dividend, we compare both with the k powers of two, thus finding the numbers q and p such that $2^q \leq x$ and $2^p \leq y$. The first approximation of the quotient is $Q_1 = 2^{p-q}$.

If $2^{p-q} < 1$ then we scale x or y depending on their magnitude. If $x \geq 1.5 * 2^q$ or $y \geq 1.5 * 2^p$ the value of Q_1 is modified.

Once we obtain Q_1 we calculate $y_1 = y - Q_1x$ and restart the cycle using the new value y_1 , until the result with the required precision is achieved.

We found that after 4 iterations the result obtained differs less than 1% from the theoretical value. In reality the final tests on images showed that 3 iterations are enough to obtain a good result and this allows a good save in terms of size. In the final table results from 3-iterations implementation are reported.

4. EXPERIMENTAL RESULTS

The proposed hardware system has been tested using different images from the TUT image database [4] to assess its performance. Six corrupted test images have been obtained from the Bridge image, in [4], by adding to the original image i.i.d. noise with the following contaminated Gaussian probability distribution,

$$\nu \sim (1 - \lambda)\mathcal{N}(0, \sigma_n) + \lambda\mathcal{N}(0, \frac{\sigma_n}{\lambda}). \quad (5)$$

Three values of the parameter λ have been selected corresponding to different types of noise. $\lambda = 0.1$ (very impulsive noise), $\lambda = 0.2$ (mixed Gaussian-impulsive noise) and $\lambda = 1$ (purely Gaussian noise). Two sets of images have been formed according to their SNR values, 3dB and 9dB. For subjective evaluation of the proposed filter performance, Figs. 3(a)-3(d) displays respectively the clean Bridge image, the noisy image with $\lambda = 0.2$ and $SNR = 9dB$, the output of the implemented MRHF system and the filtered image by "theoretical" MRHF.

For objective comparison, MAE and MSE criteria are used to quantitatively assess the capabilities of our implemented system in terms of noise attenuation. These are compared with the "theoretical" (i.e., the software implementation of) MRHF and several well known techniques from the literature: stack filters (Stack) [10], rank-order morphological filters (ROMF) [11] and rational filters [9]. Table 1 summarises the simulation results.

It can be clearly seen that the images in Figs. 3(c) and 3(d) are very close, and both show a good noise attenuation especially in flat areas and preserved sharp edges. Moreover, from Table 1, one can conclude that in most cases the implemented structure gives even better results than the "theoretical" one. The difference is due to the effects of rounding errors in the approximation of the rational function.

The implemented system appears to be relatively small, considering the use of RNS system and the needs of the division algorithm. The resources required by the system usign three steps in the recursive division are slightly less than 400 CLB's. This means that it can be fit into a medium sized FPGA. As we aforementioned the operationing speed is dictated by the binary adders/subtractors, which is about 50 MHz. Therefore this system is very suitable for real-time application, being able to easily handle television format images of 720 by 576 pixels at frame rates of 50 Hz.

5. CONCLUSIONS

Order statistics-rational hybrid filters are introduced in this paper. The performance of these filters is studied for the case of mixed impulsive and Gaussian noise filtering, and a dedicated hardware implementation system is proposed and evaluated.

Simulations show that the implemented system is robust and that the relative small dimensions of its structure make it very suitable for direct implementations on FPGAs.

6. REFERENCES

- [1] J. Astola and P. Kuosmanen, *Fundamentals of Nonlinear Digital Filtering*, CRC Press, Boca Raton, New York, 1997.
- [2] M. Dugdale, "VLSI Implementation of Residue Adders Based on Binary Adders," *IEEE Transactions on Circuit and Systems II*, vol.39, no.5, May 1992.
- [3] K.M. Elleithy, M.D. Bayoumi, "Fast and Flexible Architectures for RNS Arithmetic Decoding," *IEEE Transactions on Circuit and Systems II*, vol.39, no.4, April 1992.
- [4] M. Gabbouj and I. Tabus, "TUT noisy image database," *Technical Report, no. 13, Tampere University of Technology*, December 1994.
- [5] L. Khriji and M. Gabbouj, "Median-Rational Hybrid Filters for image restoration," *IEE Electronics Letters*, vol.34, no.10, pp. 977-979, May 1998.
- [6] L. Khriji and M. Gabbouj, "Median-Rational Hybrid Filters," *IEEE International Conference on*

Image Processing, ICIP'98, Chicago, Illinois, October 4-7, 1998.

- [7] M. Lu, J.-S. Chiang, "A Novel Division Algorithm for the Residue Number System," *IEEE Transactions on Computers*, vol.41, no.8, August 1992.
- [8] I. Pitas and A.N. Venetsanopoulos, *Nonlinear Digital Filters: Principles and Applications*, Kluwer Academic, Dordrecht, Holland, 1991.
- [9] G. Ramponi, "The Rational Filter for Image Smoothing," *IEEE Signal Processing Letters*, vol.3, no. 3, pp. 63-65, March 1996.
- [10] I. Tabus, D. Petrescu and M. Gabbouj, "A Training Framework for Stack and Boolean Filtering - Fast Optimal Design Procedures and Robustness Case Study," *IEEE Trans. on Image Processing*, vol.5, no. 6, pp. 809-826, June 1996.
- [11] P. Kraft, S. Marshall, J.J. Soroghan, and N.R. Harvey, "Parallel Genetic Algorithms for Optimizing Morphological Filters," *Proc. Fifth International Conference on Image Processing and Its Applications*, Edinburgh, UK, 1995.
- [12] L. Yin, R. Yang, M. Gabbouj and Y. Neuvo, "Weighted Median Filters: A Tutorial," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 43, no. 3, pp. 157-192, March 1996.

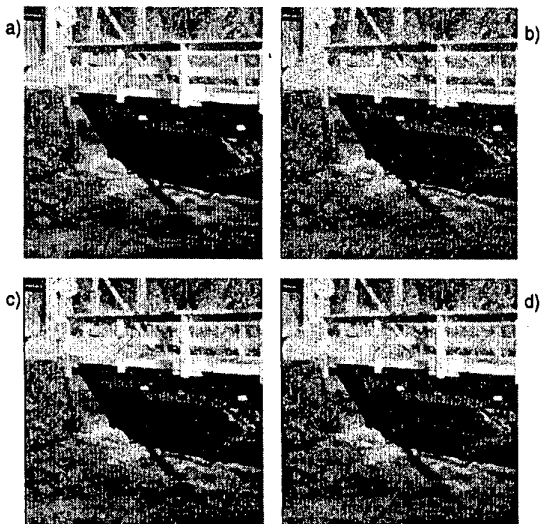


Figure 3: Qualitative comparison between implemented and theoretical filter (see text).

Image	System	MSE	MAE
$SNR = 3dB$ $\lambda = 0.1$	Noisy image	1503.50	23.445
	ROMF	240.25	11.400
	Stack	227.01	11.402
	Rational	249.20	11.640
	Theoretical	196.92	10.465
	Hardware	186.01	10.154
$SNR = 3dB$ $\lambda = 0.2$	Noisy image	1508.10	25.172
	ROMF	240.25	11.400
	Stack	227.01	11.400
	Rational	249.20	12.300
	Theoretical	213.81	10.920
	Hardware	202.29	10.600
$SNR = 3dB$ $\lambda = 1$	Noisy image	1498.80	31.002
	ROMF	320.50	13.540
	Stack	429.21	16.238
	Rational	290.00	13.100
	Theoretical	300.10	13.380
	Hardware	286.28	13.030
$SNR = 9dB$ $\lambda = 0.1$	Noisy image	377.90	9.390
	ROMF	134.44	7.500
	Stack	85.80	6.160
	Rational	117.95	7.382
	Theoretical	88.10	6.065
	Hardware	85.94	6.023
$SNR = 9dB$ $\lambda = 0.2$	Noisy image	374.53	11.743
	ROMF	136.15	8.125
	Stack	117.43	7.660
	Rational	125.91	9.077
	Theoretical	124.25	7.737
	Hardware	119.31	7.586
$SNR = 9dB$ $\lambda = 1$	Noisy image	375.97	15.467
	ROMF	178.02	10.098
	Stack	178.05	10.243
	Rational	142.01	9.160
	Theoretical	151.81	9.427
	Hardware	144.19	9.152

Table 1: Objective comparison.