

Hardware Implementation of the Median-Rational Hybrid Filters

G. Bernacchia[†], L. Khriji[‡], M. Gabbouj[‡], G. Sicuranza[†]

[†]DEEI, University of Trieste, via A. Valerio 10, I-34127 Trieste, Italy

[‡]TICSP, Tampere University of Technology, P.O. Box 553 FIN-33101, Tampere, Finland

e-mail: [bernac,sicuranza}@ipl.univ.trieste.it, [lazhar,moncef}@cs.tut.fi

Abstract

Median-Rational Hybrid Filter (MRHF) was introduced recently as a new class of nonlinear filters and applied to image filtering problems. It has been shown that the MRHF have the inherent property that on smooth areas they provide good noise attenuation whereas on changing areas the noise attenuation is traded for a good response to the change. Moreover, they act in small window and few number of operations, resulting in simple and fast filter structures. We present in this paper a hardware implementation of the MRHF 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 due to its reduced hardware complexity in applications where size and cost are of critical significance.

1 Introduction

Image restoration in a noisy environment is a fundamental problem in image processing. Various filtering techniques have been developed to suppress noise in order to improve the quality of images [3]. Among these, nonlinear digital techniques have recently received an increasing interest, due to their superior capabilities with respect to linear approaches.

This paper focuses on the hardware implementation of the new nonlinear rational type hybrid filters MRHF, recently introduced in [1] [2].

The MRHFs are based on rational functions. There are several advantages in the use of this function. Similarly to a polynomial function, a rational function is a universal approximator (it can approximate any continuous function arbitrarily well); however, it can achieve a desired level of accuracy with a lower complexity, and possesses better extrapolation capabilities.

The MRHF filter uses a two-step approach to remove both impulsive and Gaussian noise from an image. This filter is formed by a rational operator, whose inputs are the results of three sub-functions, i.e. two median filters (MF) and one center weighted median filter (CWMF).

It has been shown that the median filter effectively removes impulsive noise while the rational filter [4] per-

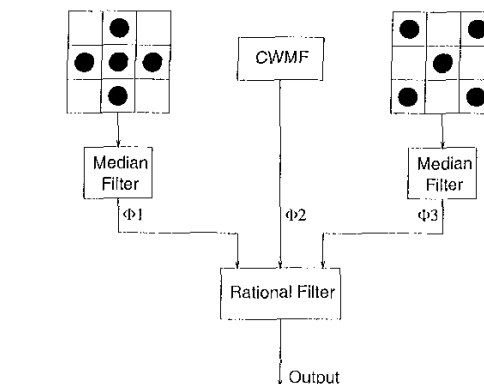


Figure 1: Structure of MRHF using two bidirectional median sub-filters .

forms well for relatively high *SNR* Gaussian contaminated environments. When both impulsive and Gaussian noise appear the MRHF outperforms the previous filters because it exploits the feature of both at the same time.

2 Median-Rational Hybrid Filters

Definition 2.1 *The output of the MRHF is the result of a rational function taking into account as inputs three median filters which form an input function set $\{\Phi_1, \Phi_2, \Phi_3\}$. The function Φ_2 is a center weighted median filter whereas the masks of the sub-filters Φ_1 and Φ_3 are chosen so that an acceptable compromise between noise reduction and edge preservation can be achieved.*

The MHRF function can be written as follows:

$$y(n) = \Phi_2(n) + \frac{\sum_{i=1}^3 \alpha_i \Phi_i(n)}{h + k(\Phi_1(n) - \Phi_3(n))^2} \quad (1)$$

In this work, we have chosen a very simple prototype filter with coefficients $\alpha = [1, -2, 1]^T$, satisfying the condition: $\sum_{i=1}^3 \alpha_i = 0$. h and k are some positive constants used to control the nonlinear effect. The structure used for the MHRF is shown in Fig.1. In this case the sub-functions Φ_1 and Φ_3 are two bidirectional vector median filters acting respectively on a plus-shaped and on a cross-shaped mask. The term Φ_2 is obtained from a center

weighted median filter acting on a plus mask as shown in the following:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

3 Hardware Implementation

The implementation of a filter can be carried out in several ways depending on the applications and the constraints to be applied. In our work we decided to implement the MRHF on an FPGA, because they are easily programmable and the structures can be tested and modified very quickly. Obviously this approach presents some limitations, mainly due to the constraints implied by the inner structure of the FPGA's.

As it can be seen from its structure, the MRHF is built from a block with the three median filters, which involve sorting of the incoming samples from the image, and a block which realizes the rational function and therefore requires some arithmetic functions.

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 the literature, our attention focuses on the implementation of the rational function.

3.1 Rational Function

The rational function used in the MRHF is quite simple, nevertheless the design of this block presents some problems if we want to achieve compactness in size and high speed in computation when implemented on FPGA's.

As a first approach we decided to use a residue number system (RNS) in order to perform the calculation of both numerator and denominator of the rational operator, because it allows a high parallelism in the operations. A residue number system though usually implies redundancy and therefore it needs more hardware. Moreover adders and multipliers in RNS are non-standard structures which require an *ad hoc* hardware to be implemented. On the other hand FPGA's supply built-in operators optimized for the standard binary system, like adders/subtractors and comparators; exploiting these blocks led us to obtain a faster and smaller circuitry.

The features of a system depend very much on the architecture chosen for the implementation. In image processing the amount of calculations to be performed in the time unit is very high even if the image size is small. The system should be able to read and to output data at high rate. To this purpose we decided to implement our rational operator with a pipelined structure. The main drawback of this choice is the large amount of memory elements required at each step. Moreover the FPGA has a relatively small number of built-in memory elements. Therefore the system has to be designed carefully in order to

minimize the latency time, or better the number of stages in the pipeline.

As it can be seen from (1) the more demanding arithmetic operations are the square function and the division.

The main problem introduced by the square function is the increase in the word width, i.e. in the number of bits in the result, whereas in the RNS all the operations are evaluated with a constant number of bits. In our work the operator is fixed; thus we can exploit our knowledge about the parameters appearing in the function in order to reduce its complexity. In the denominator in (1) two parameters are used. In particular we can rewrite (1) as follows:

$$k(\Phi_1(n) - \Phi_3(n))^2 = (\sqrt{k}(\Phi_1(n) - \Phi_3(n)))^2$$

Following the example in [4] these two parameters have been chosen as follows: $h = 6.25$, $k = 0.01$. If we approximate k with a simple sum of powers of 2 the number of bits required by this operation can be reduced. For instance if $k = 2^4 + 2^5$ the range of the term to be squared is limited to 5 bits instead of 8. Moreover the square function can be implemented as a simple shift-and-add multiplier with a precision of 11 bits in the result.

Simulation runs over several images showed that the loss of precision introduced with this approximation is very small and the evaluation of the denominator has been simplified to 6 additions.

In this design the role of the division is the most critical, since this operation is highly time and size consuming.

The choice of the actual implementation of this operation strongly depends on the constraints on the design and on the precision required by the rational operator and the application. In fact not all the rational operators allow a coarse approximation in the results if we want them to work effectively, like, for example, the interpolator.

We developed two different algorithms for the division. The first one is an iterative algorithm derived from [5] which can be implemented both in RNS and in standard binary. This algorithm allows us to obtain a precision in the results of about 1% after 4 iterations. Further tests on images showed that 3 iterations are enough to obtain a good result and this allows a good save in terms of size. None the less this algorithm requires few combinatorial blocks or lookup tables for the selection of the quotient and variable shifts as the new dividend is obtained using the standard formula $y_n = y_{n-1} - q_{n-1} \times x$, where q_{n-1} is the partial quotient at iteration $n - 1$. Moreover the latency time would be greatly increased due to the need of several iterations.

The second algorithm is very simple and is based on successive scalings of both numerator and denominator. Given two numbers n and d , the division $y = n/d$ can be represented as follows:

$$y = \frac{n}{d} = \frac{s^{e_n} n_s}{s^{e_d} d_s} = s^{(e_n - e_d)} \frac{n_s}{d_s} = s^{(e_n - e_d)} n_s d_s^{-1}$$

where s is a scaling factor, n_s and d_s are respectively the scaled numerator and the scaled denominator and e_x is a

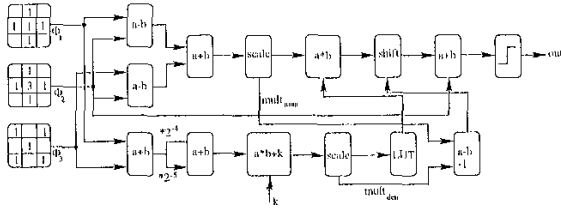


Figure 2: Block diagram of the implementation of the rational function.

number such that $s^{e^*} \times x_s$ is the largest number less than x .

In this case we can choose $s = 16$: in this way each scaling can be performed just rejecting the 4 less significant bits and the rounding can be easily accomplished summing 1 or 0 depending on the 4 bits rejected.

Since the numerator and the denominator are represented with 10 and 11 bits respectively, after at most 2 scalings by 16 the range is reduced to $[0,16]$, taking into account the rounding effect. A small lookup table supplies the correspondent inverse of the denominator d_s^{-1} . Since the maximum number of scaling can be 2, the difference $e_n - e_d$ can assume values in the interval $[-2,2]$ only, reducing the complexity of the shift.

As the results in the following section show the first algorithm outperforms the second one and allows us to obtain even better results than the theoretical algorithm. The second algorithm, though, is very compact and introduces a latency in the system which is less than 1/4 of the latency of the other algorithm. Moreover the required hardware can be easily and effectively implemented exploiting the features of the FPGA. In Fig.2 the block diagram of the resulting circuitry for the rational function is shown. We implemented the system on a medium size FPGA containing 400 Configurable Logic Blocks (CLB).

As it can be seen from (1) the numerator is a very simple linear function; it requires 2 8-bit adder/subtractors and one 9 bit adder. As we aforementioned the design tools for FPGA's allow to exploit some of the features of these chips. As a comparison with the RNS system, an 8 bit subtractor in standard binary requires 6 CLB's and can be operated at about 58 MHz; on the other hand a 5 bit adder for the modulo 17 addition requires 7 CLB's and it's operating frequency is about 56 MHz. This simple example well justifies the choice of the binary system due to the simplicity of the rational operator.

The denominator is slightly more demanding in terms of hardware because of the scaling and multiplication. The number of CLB's required for the denominator is 2.5 times that for the numerator and comprises the flip-flop's for the pipeline.

The division itself requires as much hardware as that used for both numerator and denominator, even if the implemented algorithm is relatively simple. The main timing constraints derive from this block, mainly due to the arbitration logic used to choose the appropriate results instead of the traditional shift register. This is done because

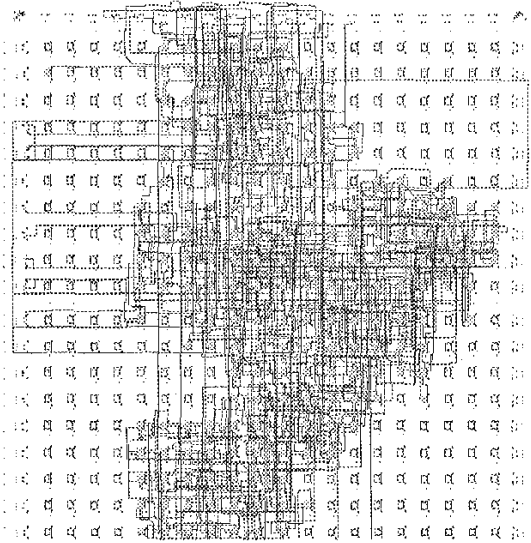


Figure 3: Layout on FPGA of the MRHF.

we don't know *a priori* how many shifts we need at each step; since the number of shifts is relatively small a simple logic can directly output the result in one clock cycle.

Actually the latency time of the system is 8 clock cycles: 4 are introduced in the evaluation of the denominator, 3 in the evaluation of the division and 1 more to obtain the final sum.

The total number of CLB's used for the rational function is 191, about 47% of the total number at our disposal; the number of flip-flop is 182 and almost 1/3 is used to store Φ_2 during the evaluation of the rational function. The maximum achievable frequency is 41 MHz, therefore this system can be effectively used for real time application with images of 768x625 pixels at a frame rate of 50Hz. In Fig. 3 the layout on the FPGA is reported.

4 Experimental results

The implemented system has been tested using images corrupted with i.i.d. noise having the following probability distribution:

$$\nu = (1 - \lambda)\mathcal{N}(0, \sigma_n) + \lambda\mathcal{N}(0, \frac{\sigma_n}{\lambda}) \quad (2)$$

Three values of λ and several different values for the SNR have been chosen: $\lambda = 0.1$ (very impulsive noise), $\lambda = 0.2$ (mixed Gaussian-impulsive noise) and $\lambda = 1$ (purely Gaussian noise), with SNR= 3dB, 6dB, 9dB and 15 dB. As an example, Figs. 4(a)-(e) are respectively the original clean image, the noisy image with $\lambda = 0.1$, $SNR = 3dB$, the image filtered by theoretical MRHF, the output of the implemented MRHF system with the high precision algorithm for division and the output of the implemented MRHF with the low precision algorithm for division. It can be clearly seen that the images in Figs.

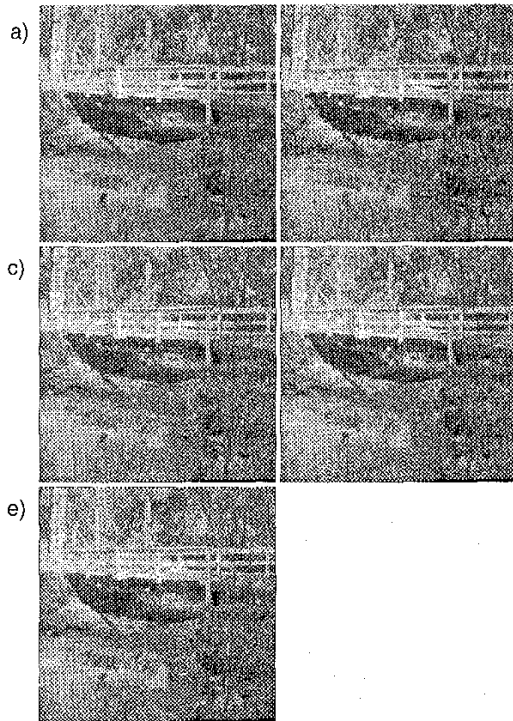


Figure 4: Qualitative comparison between implemented and theoretical filter (See text).

4(c) and 4(d) are very close to each other, and both show a good noise attenuation and present sharper edges and smoother flat areas, while Fig.4(e) presents a worse quality.

In Fig.5 we present the comparisons of results obtained for some images. For completeness sake we reported both the results obtained with the iterative (*Hardware (1)*) and with the simplified (*Hardware (2)*) algorithm for the division. All the data have been normalized to the maximum valued obtained. In all the cases the implemented structure with the iterative algorithm gives even better results than the theoretical one, with a mean of the absolute relative error of about 3.5% for MSE and 2.1% for MAE. This can be explained in two ways basically. The first concerns the coefficients h and k in the denominator. Their values have been chosen following the referenced paper [4], but they are not a result of an optimization process involving the MRHF. The introduced approximation modifies these values, therefore it is possible that they are closer to optimum values than the chosen ones. If we modify h and k with the values obtained with the approximation the results obtained in the theoretical case are better indeed, but they are still worse than those obtained with the implemented system.

The second explanation concerns the rounding errors introduced in the evaluation of the denominator. These rounding errors have the effect to reduce the actual value of the denominator. Therefore it doesn't affect the numer-

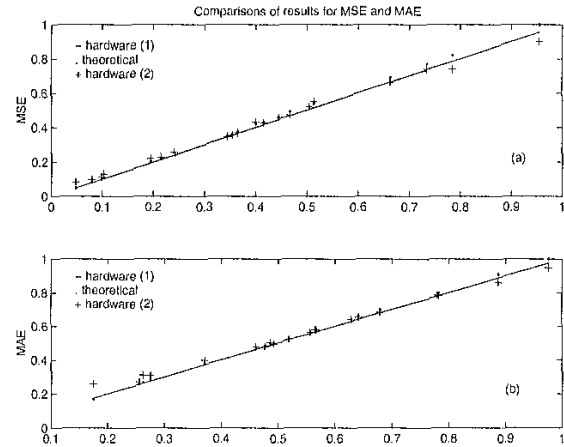


Figure 5: Comparisons between implemented and theoretical filter (See text).

ator so strongly as in the theoretical algorithm. The overall effect is a low-pass filtering which slightly improves the results.

The results obtained with mixed noise show that in this case the simplified algorithm usually outperforms the theoretical operator.

5 Conclusions

In this paper we present a hardware implementation of the new median-rational hybrid filter, which is able to remove different kinds of additive i.i.d. noise. Our tests have shown that the implemented system is very robust and can be easily implemented on a small-sized FPGA. The maximum achievable frequency is about 40MHz, which makes this filter very suitable for real-time applications.

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] I. Pitas and A.N. Venetsanopoulos, "Nonlinear Digital Filters: Principles and Applications", *Kluwer Academic*, Dordrecht, Holland, 1991.
- [4] G. Ramponi, "The Rational Filter for Image Smoothing", *IEEE Signal Processing Letters*, vol.3, no. 3, pp. 63-65, March 1996.
- [5] Mi Lu, Jen-Shinu Chiang, "A Novel Division Algorithm for the Residue Number System", *IEEE Transactions on Computers*, vol.41, no.8, Aug 92.