Performance Analysis of Algorithms over FPGA for Removing of Impulse Noise

Mohammad Ismaeil, Kh. Jilen Kumari Devi, Keisham Pritamdas, and Samarth Goyal
Department of Electronics and Communication Engineering
National Institute of Technology Manipur, Imphal, INDIA

ABSTRACT
This paper presents a novel approach in detection of impulse noise based on the entropy of the pixels. Then the detected noisy pixels are replaced with the output of the vector median filter (VMF). Performance is carried out using filters described widely in the literature such as vector median filter (VMF), Basic vector directional filter (BVDF), and Distance directional filter (DDF) implemented in Matlab. Entropy vector median filter (EVMF) shows better result as compared to others vector filters on removing impulse noise from color images and after verification in Matlab the filter is implemented on a hardware simulation tool with Artix-7(XC7A100T-1CSG324). The results are verified through the filtered images, resource consumption, and PSNR, MAE, MSE, & NCD values. The implementation methodology involves model based engineering involving MATLAB/Simulink and system generator software using the xIsgRoot library to perform co-simulation. This result shows that FPGA implementation of EVMF has a good tradeoff between resource consumption and noise removal efficiency.

Index Terms— Entropy, impulsive noise, vector, and system generator.

I. INTRODUCTION
IMAGES are usually corrupted by impulse noise due to bit errors in transmission or circuitry of a digital camera or signal acquisition or problem in camera sensor. Generally impulse noise can be classified into two categories namely salt & pepper noise (SVN) and Random valued impulse noise (RVIN). Noise removal techniques depends on the type of noise degrading the image and largely on the percentage of noise corrupting the image. Impulse noise which may be salt & pepper noise or Random valued impulse noise is one of the most naturally occurring noises on digital images. Non-linear filters which actually works on spatial domain suits well for impulse noise removal from color images [1]. Rank order filters which are also order statistics filters works efficiently for the removing of impulse noise. It uses sliding window approach of normally odd size, where on each sliding-iteration, only the center pixel in the window is replaced by the value of the pixel, which is calculated based on the ordered intensity value of the pixels contained in the area defined by the filtering window. The Vector median filter (VMF) [2], the basic vector directional filter (BVDF) [3], the distance directional filter (DDF) [4] are the popular example where the filtering is done uniformly across the pixels without using an actual noise detection algorithm. These filtering tends to modify the uncorrupted pixels which leads to loss of fine details of the image. To overcome the issue of noise detection, blurring of images, loss of fine details of the images an entropy based filters is been introduced. Entropy vector filter algorithms distinguish the corrupted pixels from the uncorrupted pixels. Pixels which have higher entropy value will be replaced by the output of the vector median filter (VMF) and the pixels which have lowest entropy value will be kept as it is.

VMF and EVMF are difficult to implement in real-time because of their high computational complexity. Therefore in this paper, we propose several techniques to reduce computational complexity by using data reuse methodology. We use 3x3 window size in this proposed technique which can be scalable to any window size.

Data reuse technique stores the sum of the distances between the vectors in a filtering window and uses them for next filtering window instead of computing them again.

The performance of the filter investigated in two phases. In first phase the image quality, Peak signal to noise ratio (PSNR), MSE, MAE, and NCD values for different amount of noise calculated using MATLAB application, in next phase the filter performance such as area, number of slice register, number of LUT, Flip-flops pairs, number of bonded IOBs, number of buffers calculated using Xilinx system Generator supported by xIsgRoot library to perform hardware co-simulation. Hardware performance can also be checked with VHDL/Verilog coding, because of the complex algorithm we prefer model based design in place of VHDL/Verilog coding.

In this work, we identify the noisy pixels with the help of a noise detection algorithm based on the fact...
that noisy pixels possess more entropy as compared to an uncorrupted pixel. Then the detected noisy pixels are replaced with the output of vector median filter \([1]\) whereas the uncorrupted pixels are kept untouched. Section I gives a brief introduction of evolution and importance of noise detection and noise removal mechanisms in digital image processing. Section II describes the proposed method with data reuse technique and hardware implementation method. Noise model performance measuring parameters are described briefly in the next section III and finally the details are summarized and concluded in section IV & V.

II. PROPOSED METHOD

Depending on the fact that the entropy of noisy pixel is normally higher, proposed algorithm based on entropy is used for noise detection. Once the noisy pixels are detected, they will be replaced by the output of vector median filter (VMF). Let a color image \(X\) of size \(M\times N\) be represented by a 2-D array of 3 component vectors represented as

\[
x(m,n) = [x^R(m,n), x^G(m,n), x^B(m,n)]
\]

Where \(m=1,2,\ldots,M\) and \(n=1,2,\ldots,N\) represents the row and column indices \(R\), \(G\) and \(B\) represents the Red, Green and Blue components of the vector pixels \(x(m,n)\). As usual in basic filtering approach a window \(W\) of odd size \(r \times c\) centered at \(x(m,n)\) is considered (see Fig.1). Then the local contrast probability for each pixel \(x_i\), is found out using

\[
P_i = \frac{||x_i - x_{\text{mean}}||}{||x_j - x_{\text{mean}}||} \text{ for } i = 1,2,\ldots,r \times c
\]

Where,

\[
||x_i - x_{\text{mean}}|| = ||x_i^R - x_{\text{mean}}^R|| + ||x_i^G - x_{\text{mean}}^G|| + ||x_i^B - x_{\text{mean}}^B||
\]

Taking a 3x3 window of color vectors with their coordinates.

<table>
<thead>
<tr>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(x_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x(m-1,n-1))</td>
<td>(x(m-1,n))</td>
<td>(x(m-1,n+1))</td>
</tr>
<tr>
<td>(x_4)</td>
<td>(x_5)</td>
<td>(x_6)</td>
</tr>
<tr>
<td>(x(m,n-1))</td>
<td>(x(m,n))</td>
<td>(x(m,n+1))</td>
</tr>
<tr>
<td>(x_7)</td>
<td>(x_8)</td>
<td>(x_9)</td>
</tr>
<tr>
<td>(x(m+1,n-1))</td>
<td>(x(m+1,n))</td>
<td>(x(m+1,n+1))</td>
</tr>
</tbody>
</table>

The parameter \(p\) is used to define the distance between the pixels. Normally the value takes as 1. The flow chart of EVMF is as shown in figure

![Flow chart of EVMF](image)

The contrast probability is compared with corresponding normalized entropic threshold given by

\[
TH_i = \frac{p_i^\frac{1}{p} \log_2 \frac{1}{P_i}}{\sum_{j=1}^{n} p_j^\frac{1}{p} \log_2 P_j}
\]

For checking the pixels to be noisy or not, the local contrast probability of the considered pixel \(x(m,n)\), say given by \(P_{x(m,n)}\), is compared with the threshold value given by \(TH_{x(m+1,n+1)}\), multiplied by a weighted factor \(\partial\), to give \(\partial \times TH_x(m+1,n+1)\) and if \(P_{x(m,n)} > \partial \times TH_x(m+1,n+1)\), then the centered pixel \(x(m,n)\) is replaced with the output of the vector median filter (VMF) is given as

\[
l_p(i) = \sum_{j=1}^{n} L_p(X_i, X_j) \text{ for } i = 1,2,3 \ldots r \times c
\]

Where

\[
L_p(X_i, X_j) = \frac{||X_i - X_j||_p}{(\sum_{k=1}^{3} ||X_i^k - X_j^k||_p)^{1/p}} \text{ for } i = 1,2,3 \ldots r \times c
\]

And

\[
(X_i, X_j) = X_i^1 X_j^1 + X_i^2 X_j^2 + X_i^3 X_j^3
\]
A. Proposed data reuse technique
In proposed entropy vector median filter (EVMF) we have used fixed 3×3 filtering window. All the noisy pixel after detection passing through the vector median filter algorithm, in VMF algorithm it calculate the Euclidean distance for each pixel from the neighborhood pixel and then the center pixel of a 3×3 window is replaced the pixel value which has the minimum distance from the neighborhood pixel. Calculation of the three pixels in filtering window as shown in figure 4.

<table>
<thead>
<tr>
<th>4-1</th>
<th>4-2</th>
<th>4-3</th>
<th>5-1</th>
<th>5-2</th>
<th>5-3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-7</td>
<td>4-8</td>
<td>4-9</td>
<td>5-7</td>
<td>5-8</td>
<td>5-9</td>
</tr>
</tbody>
</table>

(a) (b) (c)

In the above figure ‘R’ indicate that the value is used for next calculation as a data reuse method, as the subtraction value of |4−5|=|5−4|=C and |5−6|=|6−5|=C, in the same way we can use subtracted value for calculating the distance of all the vectors from their neighborhoods.

<table>
<thead>
<tr>
<th>Arithmetic operation</th>
<th>Without data reuse</th>
<th>With data reuse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subtraction</td>
<td>64</td>
<td>36</td>
</tr>
<tr>
<td>Absolute value</td>
<td>64</td>
<td>36</td>
</tr>
</tbody>
</table>

Table 1. Arithmetic operations required for a 3×3 filtering window

Calculating the sum of the distance of a vector from the neighbor 8 vector requires 64 subtraction without data reuse technique. So, the number of these arithmetic operations can be significantly reduced by data reuse technique.

Data reuse technique is applicable to 8 vectors out of 9 vector in a 3×3 filtering window. When we slides from 1st vector to 2nd vector then next vector the number of subtraction decreases as data reuse technique stores all the distances between other vectors in the register.

B. Proposed hardware implementation method
Programming logic is emerging as an attractive solution for many digital image processing applications. Image smoothing algorithms are particularly suitable for implementation on FPGA, due to the parallelisms that may be exploited. Research has been conducted to improve speed by designing block by block. The proposed algorithm of entropy vector median filtering calculation consists of square-root, divider, logarithmic functions, adders, subtraction, and absolute functions. Hardware divider, square-root and logarithmic functions are quite large and slow. We can implement those functions in two ways, either by using ROM block of system generator or by using cordic algorithmic block. In ROM block, for respective functions vector values stored for the operations and the values multiplied with the respective pixel values. Cordic algorithm works with the iteration which is less complex but it takes more time as compared to ROM, due to higher iteration, delay is also more and consumes more time.

In VHDL/Verilog coding, we can use the bit shifting method of division. A divide by 9 is implemented by dividing value 1 by 9, then result is shifted by two integer in left side of the decimal point (up sampling) and use the value on required calculation and in the final results the pixel value is down sampled for required number in fixed point window.

III. FILTER EVALUATION
The filter is evaluated in this section on a set of commonly used color images. Fig 2 (a), (b), (c) shows the parrot, Lena and the pepper image on which the performance of the new filter is analyzed by considering a 3×3 window size.

![Fig. 3 (a) Parrot image.](image1)

![Fig. 3 (b) Lena image.](image2)

![Fig. 3 (c) Pepper image.](image3)
A. Noise Model

The most commonly used impulse noise model is used in this work [11]. Let $a$ be the probability of the impulse noise corruption of the color image. Since a color image has three vector components, each component is being corrupted with a respective corruption probability. Let $a_R$, $a_G$, and $a_B$ be the probabilities of impulse noise corruption of the three components respectively.

$$
x \text{ with probability } 1 - a
$$

$$
y = \begin{cases} 
(n_R, x_G, x_R) \text{ with probability } a_R & a_R \\
(n_B, x_G, x_B) \text{ with probability } a_G & a_G \\
(n_B, n_G, n_B) \text{ with probability } [1 - (a_R, a_G, a_B)] & a_R
\end{cases}
$$

$x \square \square x_R, x_G, x_B \square$ and $y \square \square y_R, y_G, y_B \square$ represents the original and the corrupted vector pixels respectively. And the impulsive noise is represented by the random vector.

$n \square \square n_R, n_G, n_B \square$ which can be a vector of 0 or 255 or both.

B. Filter Performance Measurements

Filtering performance is evaluated using certain parameters like Execution time, mean absolute error (MAE) [10], mean squared error (MSE) [11], normalized color difference (NCD) [12] and peak signal to noise ratio (PSNR) [13]. NCD measures the color chromaticity preservation of a color image. If a filtered image is free from shadowy effects then the image is said to preserve chromaticity. MSE and MAE represent the noise suppression and the signal-detail preservation respectively. MAE and MSE are mathematically expressed as

$$
\text{MAE} = \frac{1}{3 \times M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} \left[ |R(i, j) - \hat{R}(i, j)| + |G(i, j) - \hat{G}(i, j)| + |B(i, j) - \hat{B}(i, j)| \right]
$$

[10]

$$
\text{MSE} = \frac{1}{3 \times M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} \left[ (R(i, j) - \hat{R}(i, j))^2 + (G(i, j) - \hat{G}(i, j))^2 + (B(i, j) - \hat{B}(i, j))^2 \right]
$$

[11]

Where $M$ and $N$ represent the image dimensions, $(R(i, j), G(i, j), B(i, j))$ and $(\hat{R}(i, j), \hat{G}(i, j), \hat{B}(i, j))$ are the coordinates of the pixel $(i, j)$ in the original and the filtered images, respectively. It can be seen that the above two equation differs only with a square. In case of MSE, the error is magnified with the square because of slight difference between the original and filtered image can be highlighted properly for a better comparison in the performance of the filters. The NCD is given as

$$
\text{NCD} = \sum_{i=1}^{M} \sum_{j=1}^{N} \sqrt{(L_{ab}(i, j) - \hat{L}_{ab}(i, j))^2 + (a(i, j) - \hat{a}(i, j))^2 + (b(i, j) - \hat{b}(i, j))^2}
$$

[12]

$L$, $a$, $b$ represents the light and the two chrominance components of a color image. $L_{ab}(i, j)$, $a(i, j)$ and $b(i, j)$ represents the original pixel values depicting the light and two chrominance components at the coordinate$(i, j)$, whereas$L_{ab}(i, j), \hat{a}(i, j)$ and $\hat{b}(i, j)$ are the filtered components.

IV. IMPLEMENTATION AND RESULTS

The filter algorithm is implemented on MATLAB 2012 and on system generator 2016 by using XSGLibLib library for co-simulation. In hardware, the design has been checked by converting image file into .coe file and load it to block RAM.
V. CONCLUSIONS

In this paper, an entropy filter, for impulsive noise removal is introduced that works well both in lower and higher noise ratio and the performance of the filter is analyzed on hardware. Experiments on three images show that the filter outperforms the vector median filter in all the noise ratio in preserving the signal content. The filter not only preserves the image’s details efficiently but also maintains the chromaticity. Future works will be implementing a modified EVM filter with better signal to noise ratio with less hardware resources.
VI. REFERENCES


