Noise Suppression in Images by Median Filter

Dontabhaktuni Jayakumar, Neelapala Saisruthi, Laiphangbam Renita Devi

Assistant professor Department of ECE, Holy Mary Institute of Technology and Science, Hyderabad, India

Abstract— *A new and efficient algorithm for high-density salt and pepper noise removal in images and videos is proposed. In the transmission of images over channels, images are corrupted by salt and pepper noise, due to faulty communications. Salt and Pepper noise is also referred to as Impulse noise. The objective of filtering is to remove the impulses so that the noise free image is fully recovered with minimum signal distortion. Noise removal can be achieved, by using a number of existing linear filtering techniques. We will deal with the images corrupted by salt-and-pepper noise in which the noisy pixels can take only the maximum or minimum values (i.e. 0 or 255 for 8-bit grayscale images).*

Keywords*— *About five key words in alphabetical order, separated by comma.

I. INTRODUCTION

1.1 Introduction to project

In image processing it is usually necessary to perform high degree of noise reduction in an image before performing higher-level processing steps, such as edge detection. The median filter is a non-linear digital filtering technique, often used to remove noise from images or other signals. The idea is to examine a sample of the input and decide if it is representative of the signal. This is performed using a window consisting of an odd number of samples. The values in the window are sorted into numerical order; the median value, the sample in the center of the window, is selected as the output. The oldest sample is discarded, a new sample acquired, and the calculation repeats.

Median filtering is a common step in image processing. It is particularly useful to reduce speckle noise and salt and pepper noise. Its edge-preserving nature makes it useful in cases where edge blurring is undesirable Image synthesis is the process of creating new images from some form of image description. The kinds of images that are typically synthesized include as follows.

1.1.1 Noise

In common use, the word noise means unwanted sound or noise pollution. In electronics noise can refer to the electronic signal corresponding to acoustic noise (in an audio system) or the electronic signal corresponding to the (visual) noise commonly seen as 'snow' on a degraded television or video image. In signal processing or computing it can be considered data without meaning that is, data that is not being used to transmit a signal, but is simply produced as an unwanted by-product of other activities. In Information Theory, however, noise is still considered to be information. In a broader sense, film grain or even advertisements in web pages can be considered noise. Noise can block, distort, or change the meaning of a message in both human and electronic communication.

In many of these areas, the special case of thermal noise arises, which sets a fundamental lower limit to what can be measured or signaled and is related to basic physical processes at the molecular level described by well-known simple formulae.

1.1.2 Salt and Pepper Noise

Another common form of noise is data drop-out noise (commonly referred to as intensity spikes, speckle or salt and pepper noise). Here, the noise is caused by errors in the data transmission. The corrupted pixels are either set to the maximum value (which looks like snow in the image) or have single bits flipped over. In some cases, single pixels are set alternatively to zero or to the maximum value, giving the image a `salt and pepper' like appearance. Unaffected pixels always remain unchanged. The noise is usually quantified by the percentage of pixels which are corrupted.

In the following examples, images have been corrupted with various kinds and amounts of drop-out noise. In, pixels have been set to 0 or 255 with probability p=1%. In pixel bits were flipped with p=3%, and in 5% of the pixels (whose locations are chosen at random) are set to the maximum value, producing the snowy appearance.

For this kind of noise, conventional low pass filtering, *e.g.* mean filtering or Gaussian smoothing is relatively unsuccessful because the corrupted pixel value can vary significantly from the original and therefore the mean can be significantly different from the true value. Median filter removes drop-out noise more efficiently and at the same time preserves the edges and small details in the image better. Conservative smoothing can be used to obtain a result which preserves a great deal of high frequency detail, but is only effective at reducing low levels of noise.

1.1.3 Median Filter

The Common Names are Median filtering, Rank filtering. The median filter is normally used to reduce noise in an image, somewhat like the mean filter. However, it often does a better job than the mean filter of preserving useful detail in the image.

How Median filter works

Like the mean filter, the median filter considers each pixel in the image in turn and looks at its nearby neighbours to decide whether or not it is representative of its surroundings. Instead of simply replacing the pixel value with the mean of neighbouring pixel values, it replaces it with the median of those values. The median is calculated by first sorting all the pixel values from the surrounding neighbourhood into numerical order and then replacing the pixel being considered with the middle pixel value. (If the neighborhood under consideration contains an even number of pixels, the average of the two middle pixel values is used.) Figure 1.1 illustrates an example calculation.

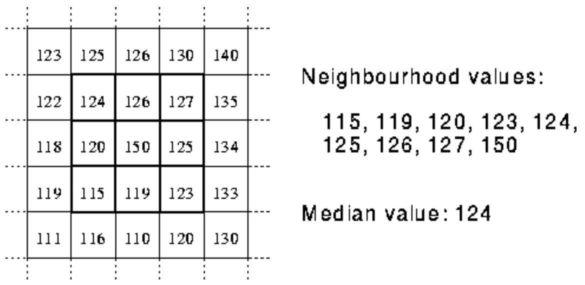


Fig.1.1: Calculating the median value of a pixel neighborhood.

As can be seen the central pixel value of 150 is rather unrepresentative of the surrounding pixels and is replaced with the median value 124. A 3×3 square neighborhood is used here and larger neighborhoods will produce more severe smoothing.

1.1.4 Mean Filter

The Common Names are Mean filtering, Smoothing, Averaging, Box filtering. Mean filtering is a simple, intuitive and easy to implement method of smoothing images, *i.e.* reducing the amount of intensity variation between one pixel and the next. It is often used to reduce noise in images.

How mean filter works

The idea of mean filtering is simply to replace each pixel value in an image with the mean (`average') value of its neighbours, including itself. This has the effect of eliminating pixel values which are unrepresentative of their surroundings. Mean filtering is usually thought of as a convolution filter. Like other convolutions it is based around a kernel, which represents the shape and size of the neighbourhood to be sampled when calculating the mean. Often a 3×3 square kernel is used, as shown in Figure 1.1, although larger kernels (*e.g.* 5×5 squares) can be used for more severe smoothing. (Note that a small kernel can be applied more than once in order to produce a similar - but not identical - effect as a single pass with a large kernel.)

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Fig.1.2: 3×3 averaging kernel often used in mean filtering

Computing the straightforward convolution of an image with this kernel carries out the mean filtering process.

1.1.5 Adaptive Median Filter

Comparing with Standard median filtering the Adaptive median filtering is an advanced method. Which pixels in an image have been affected by impulse noise can be determined by using spatial processing. AMF performs in the image by comparing each pixel with its surrounding neighbor pixels to classify pixels as noise. The neighborhood pixel of the size is adjustable, as well as for the comparison the threshold is adjustable. A pixel is not structurally aligned with those pixels to which it is similar, as well as pixel that is Different from a majority of its neighbors can be treated as impulse noise. The median pixel value of the pixels in the neighborhood can be replaced in the place of noise pixels that have passed the noise labeling test.

1.1.6 Unsymmetric Trimmed Median Filter

In this UTMF, the selected window elements are arranged in either increasing or decreasing order. Then the pixel values 0's and 255's in the image (i.e., the pixel values responsible for the salt and pepper noise) are removed from the image. Then the median value of the remaining pixels is taken. This median value is used to replace the noisy pixel. This filter is called trimmed median filter because the pixel values 0's and 255's are removed from the selected window.

1.1.7 Proposed Algorithm

The proposed Modified Decision Based Unsymmetrical Trimmed Median Filter (MDBUTMF) algorithm processes the Corrupted images by first detecting the impulse noise. The processing pixel is checked whether it is noisy or noisy free. That is, if the processing pixel lies between maximum and minimum gray level values then it is noise free pixel, it is left unchanged. If the processing pixel takes the maximum or minimum gray level then it is noisy pixel which is processed by MDBUTMF.

1.2 Introduction to VLSI

1.2.1 Digital Design

Prompted by the development of new types of sophisticated field-programmable devices (FPDs), the process of designing digital hardware has changed dramatically over the past few years. Unlike previous generations of technology, in which board-level designs included large numbers of SSI chips containing basic gates, virtually every digital design produced today consists mostly of high-density devices. This applies not only to custom devices like processors and memory, but also for logic circuits such as state machine controllers, counters, registers, and decoders. When such circuits are destined for high-volume systems they have been integrated into high-density gate arrays. However, gate array NRE costs often are too expensive and gate arrays take too long to manufacture to be viable for prototyping or other low-volume scenarios. For these reasons, most prototypes, and also many production designs are now built using FPD s. The most compelling advantages of FPDs are instant manufacturing turnaround, low start-up costs, low financial risk and (since programming is done by the end user) ease of design changes. The market for FPDs has grown dramatically over the past decade to the point where there is now a wide assortment of devices to choose from. A designer today faces a daunting task to research different types of chips, understand what they can best be used for, choose a particular manufacturer's product, learn the intricacies of vendor-specific software and then design the hardware. Not only the sheer number of FPDs available exacerbates confusion for designers, but also by the complexity of the more sophisticated devices. The purpose of this paper is to provide an overview of the architecture of the various types of FPDs. The emphasis is on devices with relatively high logic capacity; all of the most important commercial products are discussed.

1.2.2 Overview of Commercially Available FPDs

This section provides many examples of commercial FPD products. SPLDs are first discussed briefly, and then details are given for all of the most important CPLDs and FPGAs. The reader who is interested in more details on the commercial products is encouraged to contact the manufacturers, or their distributors, for the latest data sheets.

Commercially Available CPLDs

As stated earlier, CPLDs consist of multiple SPLD-like blocks on a single chip. However, CPLD products are much more sophisticated than SPLDs, even at the level of their basic SPLD-like blocks. In this section, CPLDs are discussed in detail, first by surveying the available commercial products and then by discussing the types of applications for which CPLDs are best suited. Sufficient details are presented to allow a comparison between the various competing products, with more attention being paid to devices that we believe are in more widespread use than others.

Commercially Available FPGAs

As one of the largest growing segments of the semiconductor industry, the FPGA market-place is volatile. As such, the pool of companies involved changes rapidly and it is somewhat difficult to say which products will be the most significant when the industry reaches a stable state. For this reason, and to provide a more focused discussion, we will not mention all of the FPGA manufacturers that currently exist, but will instead focus on those companies whose products are in widespread use at this time. In describing each device we will list its capacity, nominally in 2-input NAND gates as given by the vendor. Gate count is an especially contentious issue in the FPGA industry, and so the numbers given in this paper for all manufacturers should not be taken too seriously. Wags have taken to calling them "dog" gates, in reference to the traditional ratio between human and dog years. There are two basic categories of FPGAs on the market today 1. SRAM-based FPGAs and 2. antifuse-based FPGAs. In the first category, Xilinx and Altera are the leading manufacturers in terms of number of users, with the major competitor being AT&T. For antifuse-based products, Actel, Quick logic and Cypress, and Xilinx offer competing products.

1.2.3 Needs for FPGA

Because they offer high speeds and a range of capacities, FPGAs are useful for a very wide assortment of applications, from implementing random glue logic to prototyping small gate arrays. One of the most common uses in industry at this time, and a strong reason for the large growth of the FPGA market, is the conversion of designs that consist of multiple SPLDs into a smaller number of FPGAs. FPGAs can realize reasonably complex designs, such as graphics controller, LAN controllers, UARTs, cache control, and many others. As a general rule-of-thumb, circuits that can exploit wide AND/OR gates, and do not need a very large number of flip-flops are good candidates for implementation in FPGAs. A significant advantage of FPGAs is that they provide simple design changes through re-programming (all commercial FPGA products are re-programmable). Within system programmable FPGAs it is even possible to re-configure hardware (an example might be to change a protocol for a communications circuit) without power-down.

Designs often partition naturally into the SPLD-like blocks in a FPGA. The result is more predictable speed-performance than would be the case if a design were split into many small pieces and then those pieces were mapped into different areas of the chip. Predictability of circuit implementation is one of the strongest advantages of FPGA architectures.

II. LITERATURE SURVEY

In this section, we have gone through detail literature reviews of impulse noise removal on the reported recent articles and critically studied their performances through computer simulation. In traditional median filtering called standard median filter (SMF), the filtering operation is performed across to each pixel without considering whether it is uncorrupted. So, the image details, contributed by the uncorrupted pixels are also subjected to filtering and as a result the image details are lost in the restored version. To overcome this problem, an impulse noise detection mechanism is applied prior to the image filtering. A Dynamic Adaptive Median Filter (DAMF) was proposed for removing high density salt and pepper noise. The filter is dynamic in nature as it decides the window size for the test pixel locally before filtering during run time and is adaptive due to the selection of a proper window size. The progressive switching median filter (PSMF) was proposed which achieves the detection and removal of impulse noise in two separate stages. In first stage, it applies impulse detector and then the noise filter is applied progressively in iterative manners in second stage. In this method, impulse pixels located in the middle of large noise blotches can also be properly detected and filtered. The performance of this method is not good for very highly corrupted image.

Nonlinear filters such as adaptive median filter (AMF) can be used for discriminating corrupted and uncorrupted pixels and then apply the filtering technique. Noisy pixels will be replaced by the median value, and uncorrupted pixels will be left unchanged. An efficient decision-based algorithm (DBA) was proposed using a fixed window size of , where the corrupted pixels are replaced by either the median pixel or neighborhood pixels. It shows promising results, a smooth transition between the pixels is lost with lower processing time which degrades the visual quality of the image.

A novel improved median filtering (NIMF) algorithm is proposed for removal of highly corrupted with salt-and-pepper noise from images. Firstly all the pixels are classified into signal pixels and noisy pixels by using the Max-Min noise detector. The noisy pixels are then separated into three classes, which are low-density, moderate density, and high-density noises, based on the local statistic information. Finally, the weighted 8-neighborhoodsimilarity function filter, the median filter and the 4-neighborhood mean filter are adopted to remove the noises for the low, moderate and high level cases, respectively.

A Tolerance based Arithmetic Mean Filtering Technique (TSAMFT) is proposed to remove salt and pepper noise from corrupted images. Arithmetic Mean filtering technique is modified by the introduction of two additional features. In the first phase, to calculate the Arithmetic Mean, only the unaffected pixels are considered. In the second phase, a Tolerance value has been used for the replacement of the pixels. This proposed technique provides much better results than that of the existing mean and median filtering techniques.

A modified decision based unsymmetrical trimmed median filter (MDBUTMF) algorithm is proposed for the restoration of gray scale, and color images that are highly corrupted by salt and pepper noise. The proposed algorithm replaces the noisy pixel by trimmed median value when other pixel values, 0’s and 255’s are present in the selected window and when all the pixel values are 0’s and 255’s then the noise pixel is replaced by mean value of all the elements present in the selected window. When this algorithm tested against different gray scale and color images, it gives better Peak Signal-to-Noise Ratio (PSNR) and Image Enhancement Factor (IEF).

Existing System

Median filters are known for their capability to remove impulse noise without damaging the edges i.e., to preserve the edges. The main drawback of a standard median filter (SMF) is that it is effective only for low noise densities. At high noise densities, SMFs often exhibit blurring for large window sizes.

Proposed System

Adaptive Median is a “decision-based” filter known as (MDBUT median filter) that first identifies possible noisy pixels and then replaces them using the median filter or its variants, while leaving all other pixels unchanged. This filter is good at detecting noise even at a high noise level.

III. SOFTWARE AND HARDWARE DESIGN

3.1 Introduction

Digital images play a very important part both in applications such as television magnetic resonance imaging computer tomography as well as in field of science and technology such as geographical information system and astronomy. Sets of data collected by image sensors and other devices are generally contaminated by noise .Also noise can introduced due to transmission errors and compression. Hence denoising is often a necessary and first step to be performed before image data is analyzed and processed. An efficient denoising technique must be applied to compensate for such data corruption. Noise is generally modeled as Gaussian noise (Normal), Uniform noise and Impulse noise (salt and pepper noise). The impulse noise is of two types, Fixed valued and random valued. The fixed valued impulse noise is also known as salt and pepper noise which can have value either 0 or 255. Here 0 represent complete black and 255 represent complete white on gray scale image. The random valued impulse noise can have any value between 0 and 255; hence its removal is very important and difficult. Image de-noising is an important pre-processing step for image analysis. It refers to the task of recovering a good estimate of the true image from a degraded observation without altering and changing useful structure in the image such as discontinuities and edges. Image denoising still remains an important challenge for researchers because denoising process removes noise but introduces artifacts and also causes blurring.

Several nonlinear filters have been proposed for restoration of images contaminated by salt and pepper noise. Among these standard median filter has been established as reliable method to remove the salt and pepper noise without damaging the edge details. However, the major drawback of standard Median Filter (MF) is that the filter is effective only at low noise densities. When the noise level is over 50% the edge details of the original image will not be preserved by standard median filter. Adaptive Median Filter (AMF) performs well at low noise densities. But at high noise densities the window size has to be increased which may lead to blurring the image. In switching median filter, the decision is based on a pre-defined threshold value. The major drawback of this method is that defining a robust decision is difficult. Also these filters will not take into account the local features. As a result of which details and edges may not be recovered satisfactorily, especially when the noise level is high.

To overcome the above drawback, Decision Based Algorithm (DBA) is proposed. In this, image is de noised by using a window. If the processing pixel value is 0 or 255 it is processed or else it is left unchanged. At high noise density the median value will be 0 or 255 which is noisy. In such case, neighboring pixel is used for replacement. This repeated replacement of neighboring pixel produces streaking effect. In order to avoid this drawback, Decision Based Un symmetric Trimmed Median Filter (DBUTMF) is proposed. At high noise densities, if the selected window contains all 0's or 255's or both then, trimmed median value cannot be obtained. So this algorithm does not give better results at very high noise density that is at 80% to 90%. The proposed Modified Decision Based Unsymmetric Trimmed Median Filter (MDBUTMF) algorithm removes this drawback at high noise density and gives better Peak Signal-to-Noise Ratio (PSNR) and Image Enhancement Factor (IEF) values than the existing algorithm.

The proposed Modified Decision Based Unsymmetric Trimmed Median Filter (MDBUTMF) algorithm processes the corrupted images by first detecting the impulse noise. The processing pixel is checked whether it is noisy or noisy free. That is, if the processing pixel lies between maximum and minimum gray level values then it is noise free pixel, it is left unchanged. If the processing pixel takes the maximum or minimum gray level then it is noisy pixel which is processed by MDBUTMF. In many practical cases of image processing, only a noisy image is available. This circumstance is known as the blind condition. Many denoising methods usually require the exact value of the noise distribution as an essential filter parameter. So, the noise estimation methods in the spatial domain use the variance or standard deviation to estimate the actual added noise distribution. But it is found that the mean deviation provides better results than the variance or standard deviation to estimate the noise distribution. The advantage of this approach is that the mean deviation is actually more efficient than the standard deviation in practical situations. The standard deviation emphasizes a larger deviation; squaring the values makes each unit of distance from the mean exponentially (rather than additively) larger. The larger deviation will cause overestimation or underestimation of the noise. So, we assume that use of the mean deviation may contribute to more accurate noise estimation. Keeping these points in view, the authors have used the mean deviation parameter in deciding the noise pixel and replaced the central pixel by its mean deviation instead of its mean. The steps in the proposed MDBUTMF algorithm are given below.

3.2 Algorithm

Step 1 The MDBUTM Filter selects a 2D-window of size 3×3. The center pixel in the selected window is the processing pixel and it is denoted as Pij . It is given in Fig.3.1. The neighboring pixels of the processing pixel are present in the directions NW, N, NE, W, E, SW, S, and SE. The positions of these directions are (i-1,j-1), (i-1,j), (i-1,j+1), (i,j-1), (i,j+1), (i+1,j-1), (i+1,j) and (i+1,j+1) respectively. The directions are clearly mentioned in the following Fig.3.1. The X-axis is considered for 'i' and Y-axis is considered for 'j'.

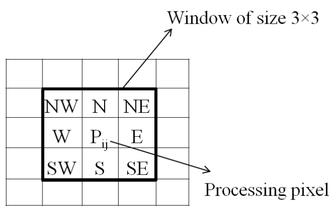


Fig.3.1: Pixel 2D-window of size 3×3

Step 2 If then is an uncorrupted pixel and its value is left unchanged.

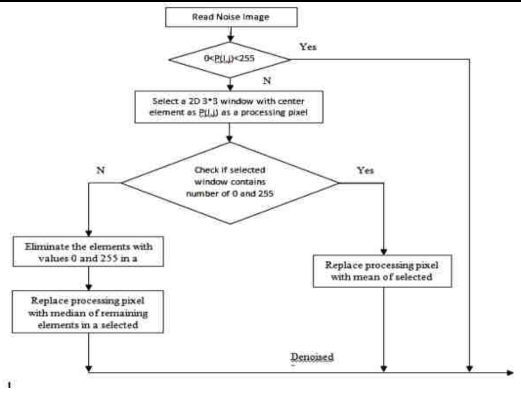
Step 3 If or then is a corrupted pixel then two cases are possible as given in Case i) and ii).

Case i) If the selected window contains all the elements as 0's and 255's. Then replacewith the mean of the element of window.

Case ii) If the selected window contains not all elements as 0's and 255's. Then eliminate255's and 0's and find

the median value of the remaining elements. Replace with the median value.
Step 4 Repeat steps 1 to 3 until all the pixels in the entire image is processed.
The pictorial representation of each case of the proposed algorithm is shown in below flow chart.Each and every pixel of the image is checked for the presence of salt and pepper noise. Different cases are illustrated below. If the processing pixel is noisy and all other pixel values are either 0's or 255's is illustrated in Case i). If the processing pixel is noisy pixel that is 0 or 255 is illustrated in Case ii). If the processing pixel is not noisy pixel and its value lies between 0 and 255 is illustrated in Case iii).
Case i) If the selected window contains salt/pepper noise as processing pixel (i.e., 255/0 pixel value) and neighboring pixel values contains all pixels that adds salt and pepper noise to the image
where "255" is processing pixel, i.e., .
Since all the elements surrounding are 0's and 255's.If one takes the median value it will be either 0 or 255 which is again noisy. To solve this problem, the mean of the selected window is found and the processing pixel is replaced by the mean value. Here the mean value is 170. Replace the processing pixel by 170.
Case ii) If the selected window contains salt or pepper noise as processing pixel (i.e., 255/0 pixel value) and neighboring pixel values contains some pixels that adds salt (i.e., 255 pixel value) and pepper noise to the image
where "0" is processing pixel, i.e., .
Now eliminate the salt and pepper noise from the selected window. That is, elimination of 0's and 255's. The 1-D array of the above matrix is [78 90 0 120 0 255 97 255 73]. After elimination of 0's and 255's the pixel values in the selected window will be [78 90 120 97 73]. Here the median value is 90. Hence, replace the processing pixel by 90.
Case iii) If the selected window contains a noise free pixel as a processing pixel, it does not require further processing. For example, if the processing pixel is 90 then it is noise free pixel
where "90" is processing pixel, i.e., .
Since "90" is a noise free pixel it does not require further processing.

Process Flow Chart of MDBUTMF
In this process we consider 0 to 15 values as 0 i.e papper noise and 230 to 255 values as 255 i.e salt noise.


Read Noise Image
0<P(i,j)<255
Yes
N
Select a 2D 3*3 window with center element as P(i,j) as a processing pixel
Check if selected window contains number of 0 and 255
N
Yes
Eliminate the elements with values 0 and 255 in a
Replace processing pixel with median of remaining elements in a selected
Replace processing pixel with mean of selected
Denoised
Fig.3.2: Flow chart of MDBUTMF

The above figure 3.2 indicates flow chart of MDBUTMF. It represents the process flow of a MDBUTMF and the process steps as shown below.
Step 1 Read a Noisy image.
Step 2 Check the each pixel value (P(i,j)) is in between 0 to 255 or not.
Step 3 If the pixel value is in between 0 to 255 then we consider it as a de noise value so we can take this as output.
Step 4 If the pixel value is 0 or 255 then select a 2D 3*3 window and keep that pixel value in center of the window and take the neighbor values also.
Step 5 we can check the number of 0's and number of 255's in that window.
Step 6 Check if the number of 0's or 255's are greater than 4 then we have to calculate Mean value by taking all pixel values from that window and keep that mean value in center and we consider it as a de noise value.
Step 7 Check if the number of 0's or 255's are less than 4 then we have to calculate Median value by taking all pixel values from that window and keep that median value in center and we consider it as a de noise value.
Step 8 Repeat the process from Step 2 to Step 7 for each pixel in a noisy image and take the output pixel values

3.3 Block Diagram

The below figure 3.3 represents the block diagram of Noise removal and its having main parameters Input Noisy Image, Xilinx Platform Studio, FPGA and Visual Basic.

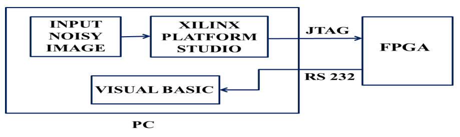


Fig.3.3: Block Diagram of Noise removal

- Input Noise Image block contains Noisy image and applied to Xilinx Platform Studio.
- We should upload header file of an image in Xilinx Platform Studio and we can write the

code for removing the noise in System C Language.

- Now we can dump these program to FPGA processor through JTAG cable.
- FPGA Processor do the preprocessing and gives the output i.e de noisy image to the Visual Basic through serial communication by using RS232 cable.
- In the Visual Basic we can see the input and output images along with the corresponding pixel values.

Internal Process of Noise removal

The above figure 3.4 represents internal process of noise removal and the process as follows.

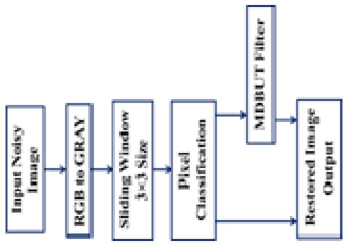


Fig.3.4: Internal Process of Noise removal

- Take the input noisy image. The image having three individual values for R,G and B.
- So we need to convert this RGB value to Gray value because its having only one value and we easily process by single value.
- After that we need to do pixel classification so we came to know noisy pixel values and de noisy pixel value.
- We can apply the noisy pixel value to MDBUT Filter, It process the noisy pixel value and give the de noisy pixel value.
- We can take de noisy pixel value and consider it as output. We consider these de noisy pixel values as output image.

3.4 Design

There are different ways to include processors inside Xilinx FPGA for System-on-a-Chip (SoC) PowerPC hard processor core, or Xilinx MicroBlaze soft processor core, or user-defined soft processor core in VHDL/Verilog. In this work, The 32-bit MicroBlaze processor is chosen because of the flexibility. The user can tailor the processor with or without advance features, based on the budget of hardware. The advance features include memory management unit, floating processing unit, hardware multiplier, hardware divider, instruction and data cache links etc. The architecture overview of the system is shown in Figure 2. It can be seen that there are two different buses (i.e., processor local bus (PLB) and fast simplex link (FSL bus) used in the system [5-6]. PLB follows IBM core connect bus architecture, which supports high bandwidth master and slave devices, provides up to 128- bit data bus, up to 64-bit address bus and centralized bus Arbitration. It is a type of shared bus. Besides the access overhead, PLB potentially has the risk of hardware/software incoherent due to bus arbitration. On the other hand, FSL supports point-to-point unidirectional communication. A pair of FSL buses (from processor to peripheral and from peripheral to processor) can form a dedicated high speed bus without arbitration mechanism. Xilinx provides C and assembly language support for easy access. Therefore, most of peripherals are connected to the processor through PLB; the DWT coprocessor is connected through FSL instead.

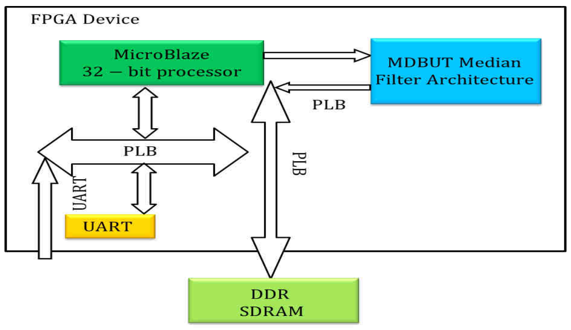

FPGA Device
MicroBlaze 32 — bit processor
MDBUT Median Filter Architecture
PLB
PLB
PLB
UART
UART
DDR SDRAM

Fig.3.5: System Overview

The current system offers several methods for distributing the data. These methods are a UART, and VGA, and Ethernet controllers. The UART is used for providing an interface to a host computer, allowing user interaction with the system and facilitating data transfer. The VGA core produces a standalone real-time display. The Ethernet connection allows a convenient way to export the data for use and analysis on other systems. In our work, to validate the DWT coprocessor, an image data stream is formed using VISUAL BASIC, then transmitted from the host computer to FPGA board through UART port.

In terms of its instruction-set architecture, MicroBlaze is very similar to the RISC-based DLX architecture described in a popular computer architecture book by Patterson and Hennessy. With few exceptions, the MicroBlaze can issue a new instruction every cycle, maintaining single-cycle throughput under most circumstances.

The MicroBlaze has a versatile interconnect system to support a variety of embedded applications. MicroBlaze's primary I/O bus, the Core Connect PLB bus, is a traditional system-memory mapped transaction bus with master/slave capability. A newer version of the MicroBlaze, supported in both Spartan-6 and Virtex-6 implementations, as well as the 7-Series, supports the

AXI specification. The majority of vendor-supplied and third-party IP interface to PLB directly (or through an PLB to OPB bus bridge.) For access to local-memory (FPGA BRAM), MicroBlaze uses a dedicated LMB bus, which reduces loading on the other buses. User-defined coprocessors are supported through a dedicated FIFO-style connection called FSL (Fast Simplex Link). The coprocessor(s) interface can accelerate computationally intensive algorithms by offloading parts or the entirety of the computation to a user-designed hardware module.

IV. SOFTWARE AND HARDWARE SPECIFICATIONS

4.1 Software Specifications

1. XILINX Platform Studio
2. MATLAB
3. Visual Basic

4.1.1 XILINX Platform Studio

The Xilinx Platform Studio (XPS) is the development environment or GUI used for designing the hardware portion of your embedded processor system. Xilinx Embedded Development Kit (EDK) is an integrated software tool suite for developing embedded systems with Xilinx MicroBlaze and PowerPC CPUs. EDK includes a variety of tools and applications to assist the designer to develop an embedded system right from the hardware creation to final implementation of the system on an FPGA. System design consists of the creation of the hardware and software components of the embedded processor system and the creation of a verification component is optional.

A Typical embedded system design project involves hardware platform creation, hardware platform verification (simulation), software platform creation, software application creation, and software verification. Base System Builder is the wizard that is used to automatically generate a hardware platform according to the user specifications that is defined by the MHS (Microprocessor Hardware Specification) file. The MHS file defines the system architecture, peripherals and embedded processors]. The Platform Generation tool creates the hardware platform using the MHS file as input. The software plat defined by MSS (Microprocessor Software Specification) file which defines driver and library customization parameters for peripherals, processor customization parameters, standard 110 devices, interrupt handler routines, and other software related routines. The MSS file is an input to the Library Generator tool for customization of drivers, libraries and interrupts handlers.

Algorithm Mapping

The FPGA implementation is divided into blocks, each block implementing a separate portion of the algorithm. This approach allowed for concurrent development and for testing of individual blocks. The inbuilt finite state machine (FSM) controls each block. In addition, a high-level FSM controls the interaction of the blocks. Each computational block is implemented in C and checked for proper functionality with simulators (ISE Simulator) The Algorithm primarily consists on mapping low-level operations like local filters. Conceptually, each pixel in the output image is produced by sliding an N×N window over the input image and computing an operation according to the input pixels under the window and the chosen window operator. The result is a pixel value that is assigned to the center of the window in the output image as shown below in Figure 4.4.

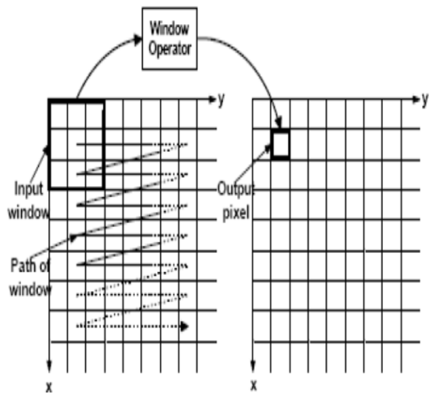


Fig.4.5: Mapping the window operation

For processing purposes, the straightforward approach is to store the entire input image into a frame buffer, accessing the neighborhood pixels and applying the function as needed to produce the output image. If processing of the video stream is required N×N pixel values are needed to perform the calculations each time the window is moved and each pixel in the image is read up to N×N times. Memory bandwidth constraints make obtaining all these pixels each clock cycle impossible. Input data from the previous N-1 rows can be cached using a shift register (or circular memory buffer) for when the window is scanned along subsequent lines.

Instead of sliding the window across the image, the above implementation now feeds the image through the window. Introducing the row buffer data structures adds additional complications. With the use of both caching and pipelining there needs to be a mechanism for adding to the row buffer and for flushing the pipeline. This is required when operating on video data, due to the horizontal blanking between lines and the vertical blanking between frames. If either the buffer or the pipeline operated during the blanking periods the results for following pixels would be incorrect due to invalid data being written to them. This requires us to stop entering data into the row buffers and to stall the pipeline while a blanking period occurs.

A better option is to replicate the edge pixels of the closest border. Such image padding can be considered as a special case of pipeline priming. When a new frame is received the first line is pre-loaded into the row buffer the required number of times for the given window size. Before processing a new row the first pixels are also pre-loaded the required number of times, as is the last pixel of the line and the last line with the implementation of the Row Buffers for Window Operations.

Memory Interfacing and C Compiler

Because the Spartan 3E FPGA that is used in the design does not have enough internal RAM for image storage, the processing blocks were interfaced with five on-board 256K×36-bit pipelined DDRAM devices. To reduce the hardware computation time, each sub-block can read and write within the same clock cycle; each sub-block was connected to two memory chips while active. Typically, a computational block reads its inputs from one memory and writes its outputs to another. It is also necessary to control/arbitrate the FPGA internal block RAM, which is used for storage of computed thresholds and other parameters. The memory interface provides the computational blocks with a common interface and hides some of the complex details.

C Compiler

Xilinx MicroBlaze Processor Supports Linux and C-to-FPGA Acceleration embedded systems can be developed to create hardware accelerated, single-chip applications that take advantage of the MicroBlaze processor features and C-to-hardware acceleration for complex, performance-critical applications. The addition of memory management to the MicroBlaze processor provides embedded systems designers with a powerful new alternative for hardware-accelerated embedded systems. By offloading critical C-language processes to dedicated hardware coprocessors, the system as a whole can operate at a slower clock speed, consume less power and yet provide vastly more processing performance than would be possible using a discrete processor.

Using the automated C-to-hardware compiler tools and interactive optimizers, performance gains well in excess of 100X over software-only approaches, in applications that include image processing, DSP and secure communications have been reported. The MicroBlaze configurable soft processor includes configurable coprocessor capabilities through its high-performance Fast Simplex Link (FSL) accelerator interface. The compiler automatically parallelizes and pipelines C-language algorithm and generates FSL interfaces, with little or no need for hardware design experience or hardware description language (HDL) coding. The automatic C-to-HDL capabilities of MicroBlaze dramatically accelerate system design.

Program Files

Input Files

1. MHS File

The Microprocessor Hardware Specification (MHS) file defines the hardware component. The MHS file serves as an input to the Platform Generator (Platgen) tool. An MHS file defines the configuration of the embedded processor system, and includes the following

- Bus architecture
- Peripherals
- Processor
- System Connectivity

2. MSS File

The Microprocessor Software Specification (MSS) is used as an input file to the Library Generator (Libgen). The MSS file contains directives for customizing OSs, libraries, and drivers.

3. UCF File

The User Constraints File (UCF) specifies timing and placement constraints for the FPGA Design.

Output Files

1. Block Memory Map

A BMM file is a text file that has syntactic descriptions of how individual Block RAMs constitute a contiguous logical data space. When updating the FPGA bitstream with memory initialization data, the Data2Mem utility uses the BMM file to direct the translation of data into the proper initialization form. This file is generated by the Platform Generator (Platgen) and updated with physical location information by the Bitstream Generator tool.

2. ELF File

The Executable and Linkable Format (ELF) is a common standard in computing. An executable or executable file, in computer science, is a file whose contents are meant to be interpreted as a program by a computer. Most often, they contain the binary representation of machine instructions of a specific processor, but can also contain an intermediate form that requires the services of an interpreter to be run.

V. SOFTWARE TESTING AND RESULTS

5.1 Image to Text Conversion

The below process indicates image to header file conversion and each step as shown in the below.

Step 1 The below figure 5.1 indicates the Creation of main page for image browsing and header file. Here we can create two rectangle boxes foe image browsing and header file creation and take one square box also for displaying the selected image.

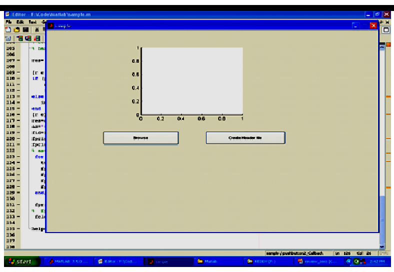
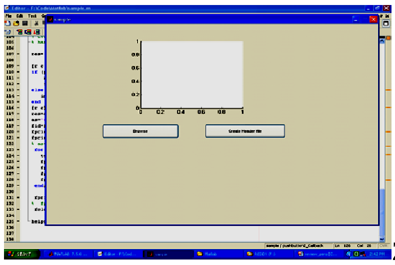


Fig.5.1 : Creation of main page for image browsing and header file

Step2 The below figure 5.2 indicates browse an image from the location where the image is existing. By clicking on browse button we can open an image.



Z

Fig.5.2: Browse an Image

Step3 Figure 5.3 indicates showing a selected image. After browsing an image its displayed on the square box in main page.

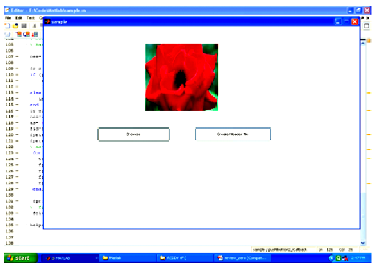


Fig.5.3: Showing a selected image

Step4 The below figure 5.3 indicates the Header file creation. After selecting the image we can click on the create header file button then it will create header file for that image and showing one dialog box saying that file created successfully after that we need to click on OK button which is showing in dialog box.

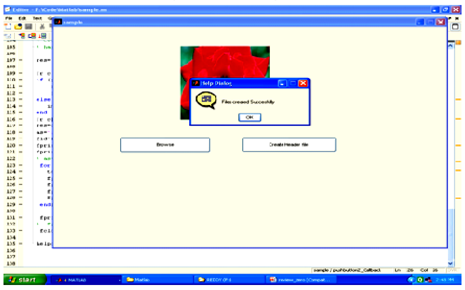


Fig.5.4: Header file creation

Step5 The below figure 5.5 indicates showing pixel values of a selected image. Its represents individual pixel value of an image.

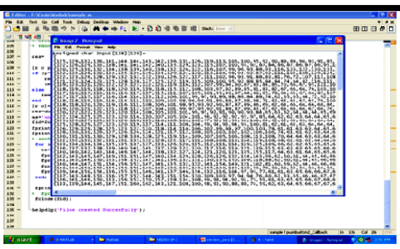


Fig.5.5: Showing pixel values of a selected image

5.2 Simulation Results

Experiments are performed on gray level images to verify the proposed method. These images are represented by 8 bits/pixel and size is 128 x 128. Image used for experiments are shown in below figure 5.6.

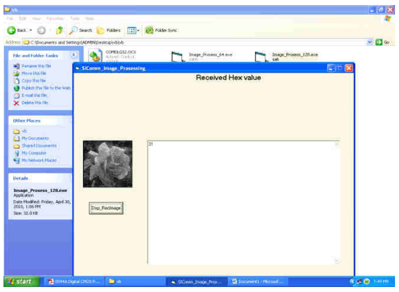


Fig.5.6: Input image

The above figure 5.6 represents input noisy along with pixel values and the below figure 5.7 indicates output image along with pixel values.The measurands used for proposed method are as follows

The entropy (E) is defined as where s is the set of processed coefficients and p (e) is the probability of processed coefficients. By using entropy, number of bits required for compressed image is calculated. An often used global objective quality measure is the mean square error (MSE) defined as Where, nxm is the number of total pixels. p(i,j) and p(i,j)' are the pixel values in the original and reconstructed image.

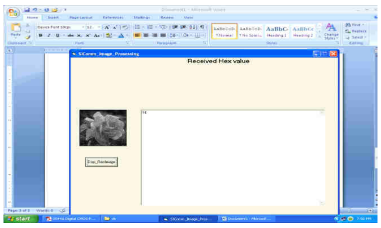


Fig.5.7: Output image

The synthesis report is below

The above figure 5.8 represents Synthesis report having the information about how many registers we are using and number of inputs etc.The Quantitative performance of the proposed algorithm is evaluated based on Peak signal to noise ratio (PSNR) ,Mean Square Error (MSE) and Image Enhancement Factor (IEF) which is given in equations 1 and 2 respectively.

$$MSE = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \left(X_{ij} - R_{ij} \right)^2 \quad \text{.... (1)}$$

Where

$$PSNR = 10 \log_{10} \left(\frac{255^2}{MSE} \right) \quad \text{... (2)}$$

Where x refers to Original image, R denotes restored image, M x N is the size of Processed image.

VI. CONCLUSION

In this paper, a new algorithm (MDBUTMF) is proposed which gives better performance in comparison with MF, AMF and other existing noise removal algorithms in terms of Peak signal to noise ratio (PSNR) and Image Enhancement Factor (IEF). The performance of the algorithm has been tested at low, medium and high noise densities on both gray-scale and color images. Even at high noise density levels the MDBUTMF gives better results in comparison with other existing algorithms. Both visual and quantitative results are demonstrated. The proposed algorithm is effective for salt and pepper noise removal in images at high noise densities.

In this paper, a We have presented an alternative implementation of median filtering for arbitrarily large windows. The architecture is immune to changes in window size, the area being determined solely by the bit width. This allows for a flexible window-size that can change from one calculation to another and we finally presented the results which are implemented on the Spartan-3 EDK evolution board.

6.1 Future Scope

In the Transmission of Videos over channel, Video frames are corrupted by salt and pepper noise (Impulse Noise), due to faulty communication systems. With this project we can implement a better filtering technique that makes the noisy video frames to noise free video frames. Median filters are the best known nonlinear digital filters based on order statistics to solve the present problem in videos. Median filters are known for their capability to remove salt and pepper noise and preserves the shape. The noise detection process to discriminate between uncorrupted pixels and the corrupted pixels prior to applying non-linear filtering is highly desirable to protect the signal details of uncorrupted pixels. We proposed A Modified Decision Based Unsymmetrical Trimmed Median filter (MDBUTM) algorithm for the restoration of gray scale, and color video frames that are highly corrupted by salt and pepper noise.

REFERENCES

[1] J. Astola and P. Kuosmaneen, Fundamentals of Nonlinear Digital Filtering. Boca Raton, FL CRC, 1997.
[2] H. Hwang and R. A. Hadded, “Adaptive median filter New algorithms and results,” IEEE Trans. Image Process., vol. 4, no. 4, pp. 499–502, Apr. 1995.
[3] S. Zhang and M. A. Karim, “A new impulse detector for switching median filters,” IEEE Signal Process. Lett., vol. 9, no. 11, pp. 360–363,Nov. 2002.
[4] P. E. Ng and K. K. Ma, “A switching median filter with boundary discriminative noise detection for extremely corrupted images,” IEEE Trans. Image Process., vol. 15, no. 6, pp. 1506–1516, Jun. 2006.
[5] K. S. Srinivasan and D. Ebenezer, “A new fast and efficient decision based algorithm for removal of high density impulse noise,” IEEE Signal Process. Lett., vol. 14, no. 3, pp. 189–192, Mar. 2007.
[6] V. Jayaraj and D. Ebenezer, “A new switching-based median filtering scheme and algorithm for removal of high-density salt and pepper noise in image,” EURASIP J. Adv. Signal Process, 2010.
[7] K. Aiswarya, V. Jayaraj, and D. Ebenezer, “A new and efficient algorithm for the removal of high density salt and pepper noise in images and videos,” in Second Int. Conf. Computer Modeling and Simulation,2010, pp. 409–413.
[8] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*
[9] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
[10] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
[11] K. Elissa, “Title of paper if known,” unpublished.
[12] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev. in press.