A Method for Design of Impulse Bursts Noise Filters Optimized for FPGA Implementations

Zdenek Vasicek, Lukas Sekanina and Michal Bidlo
Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic
Email: vasicek@fit.vutbr.cz, sekanina@fit.vutbr.cz, bidlom@fit.vutbr.cz

***Abstract*—This paper deals with the evolutionary design of area-efficient filters for impulse bursts noise which is often present in remote sensing images such as satellite images. Evolved filters require much smaller area in the FPGA than conventional filters. Simultaneously, they exhibit at least comparable filtering capabilities with respect to conventional filters. Low-cost embedded systems equipped with low-end FPGAs represent a target application for presented filters.**

I. Introduction

Image preprocessing in general, and image filtering in particular, is a task which is performed by many FPGA-based embedded systems. The quality of image filtering significantly influences the quality of subsequent operations – image recognition, classification, presentation etc. However, advanced well-performing filters which could be useful in the embedded systems require significantly more area on the FPGA than standard filters. For example, the adaptive median filter with the 5×5-pixel filtering window (kernel) requires 5.6 times more slices than a standard median filter with the 3×3-pixel filtering window and 1.3 times more slices if the standard median filter uses the 5×5-pixel filtering window (input FIFOs are not considered in the calculations) [1]. All these filters are based on exact mathematical algorithms that have been proven to work sufficiently for a particular noise removal.

Recent studies on the salt-and-pepper noise filtering have demonstrated that evolutionary algorithms (EA) can generate image filters that show the same filtering quality as the adaptive median filters; however, for half cost in the FPGA [2], [3], [4]. The basic principle of the evolutionary circuit design is that electronic circuits that are encoded as bit strings (chromosomes) are constructed and optimized by the evolutionary algorithm in order to obtain the implementation satisfying the specification given by designer [5], [6]. In order to evaluate a candidate circuit, the new configuration of a reconfigurable device (or a circuit simulator) is created on the basis of the chromosome content. This configuration is evaluated for a chosen set of input stimuli. The fitness function, which reflects the problem specification, can include behavioral as well as non-behavioral requirements. For example, the correct functionality is a typical behavioral requirement. As a non-behavioral requirement, we can mention the requirement for minimum power consumption or minimum area occupied on the chip. Once the evaluation of the population of candidate circuits is complete, a new population can be produced. That is typically performed by applying the genetic operators (such as mutation and crossover) on existing circuit configurations. High-scored candidate circuits have got a higher probability that their genetic material (parts of configuration bitstreams) will be selected for next generations. The process of evolution is terminated when a perfect solution is obtained or when a certain number of generations is evaluated. As the EA is a stochastic algorithm, the quality of resultant circuits is not guaranteed at the end of evolution. However, the method has two important advantages: (1) The artificial evolution can in principle produce intrinsic designs for electronic circuits which lie outside the scope of circuits achievable by conventional design methods. (2) The challenge of conventional design is replaced by that of designing an evolutionary algorithm that automatically performs the design in a target place (e.g., in space). This may be harder than doing the design directly, but makes autonomy possible.

This paper deals with the evolutionary design of area-efficient filters for impulse bursts noise which is often present in remote sensing images such as satellite images [7]. Low cost embedded systems equipped with low-end FPGAs represent a target application for presented filters. The filter structure is designed using Cartesian Genetic Programming (CGP) working at the functional level [8]. In contrast to previous work [9], the main goal of the paper is to demonstrate that evolved filters require much smaller area in the FPGA than conventional filters. Simultaneously, it will be demonstrated that evolved filters exhibit at least comparable filtering capabilities with respect to conventional filters. In our case, evolved filters are trained using a training image containing a particular noise and tested on a database of images corrupted by the same type of noise.

The rest of the paper is organized as follows. Section II surveys the approaches proposed to image filtering in FPGAs. In Section III, extended Cartesian Genetic Programming (CGP) is introduced for evolution of impulse bursts noise filters. Section IV deals with the experimental evaluation of proposed method: evolved filters are compared with conventional filters in terms of the quality of filtering and the area occupied in the FPGA. Finally, conclusions are given in Section V.

978-3-9810801-6-2/DATE10 © 2010 EDAA

II. Image filters in FPGAs

For purposes of this paper, the impulse bursts noise is characterized using two parameters: p and q. Let p denote a probability that a certain pixel belongs to the impulse burst. In fact, this parameter determines the maximum amount of pixels that are corrupted in the input image. Let q be a parameter which determines the maximum length of burst (i.e. the maximum number of consecutive pixels which are affected by the impulse). The number of burst fragments in the image depends on both parameters. In this paper, we will consider that the images are transferred as one-dimensional arrays in which the rows of the image pixels are stored in sequence.

Impulse bursts noise is a specific kind of noise which is difficult to filter even if a nonlinear filter is used. This is caused as both the central pixel and the neighboring pixels are corrupted. It has been shown that median filters are capable of removing impulse bursts but at the same time they usually destroy the image details too heavily. Other filters (e.g. weighted median) are not robust enough and tend to leave a lot of impulse bursts unfiltered in the images [7]. Apart from the median-based filters, training-based optimized soft morphological filters were developed to suppress this type of noise [10], [7].

Impulse bursts noise is typically filtered using larger filtering windows ($k \times k$ pixels, where $k = 5$ and higher) in order to repair a corrupted pixel using the pixels from several rows of the image. The utilization of the filtering window of $k \times k$ pixels implies that a memory structure has to be instantiated to store k neighboring rows. Hence the area efficient filters try to minimize k as much as possible. Figure 1 shows a typical implementation of the FIFO devoted to reading the pixel values from the image memory. The FIFO is typically implemented using several BRAMs that serve as row buffers. Since each BRAM is able to store 2048 bytes, a row buffer consisting of one BRAM is suitable for images that constain up to $2048+k$ pixels per row. Table I gives the implementation cost of typical FIFOs; the circuits were synthesized using Precision Synthesis to Xilinx Virtex II Pro XC2VP50. The implementation cost includes the cost of the circuit which generates necessary control signals (e.g. DONE signal).

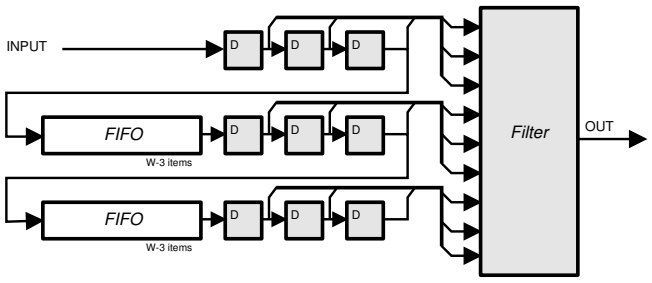


Fig. 1. The row buffers for the 3×3-pixel filter window (W is the image width)

Table III summarizes the implementation cost of selected filters. The median filter is the most popular nonlinear filter for removing the impulse noise [11]. Among more sophisticated approaches we can find switching median filters [12], weighted median filters [13], weighted order statistic filters [14] and adaptive median filters [15]. The adaptive median filters produce significantly better resulting images than conventional medians [16]; however, their implementation cost is relatively high in FPGA [1].

TABLE I
The implementation costs of row buffers for various sizes of the filter window

window size	# CLBs	# BRAMs	max. frequency
3×3	106	3	264 MHz
5×5	252	5	264 MHz
7×7	462	7	264 MHz

For comparison, Table III also shows the implementation cost of two filters evolved using CGP to suppress the salt and pepper noise ('evolved s&p filter' and 'bank s&p filter') [2], [3]. The evolved filters exhibit better filtering quality and lower implementation cost in comparison to existing solutions (in particular, median and adaptive median filters).

III. Proposed design method

Every image filter is considered as a digital circuit of several 8-bit inputs and a single 8-bit output, which processes grayscale (8-bit/pixel) images. As Fig. 2 shows, every pixel value of the filtered image is calculated using a corresponding pixel and some of its neighbors in the processed image.

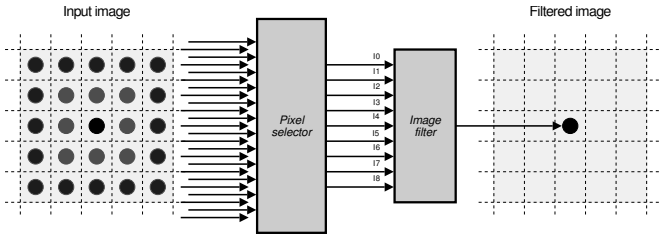


Fig. 2. The concept of filtering using a 5×5 filter kernel followed by a selector

In order to evolve the image filter capable of removing a given type of noise, we need (a) a set of suitable elementary functions (building blocks of the filter circuit), (b) rules for interconnecting those functions and (c) the original (training) image to measure the fitness values of the candidate filters (i.e., to evaluate the quality of a candidate filter). The goal of the evolutionary algorithm is to minimize the difference between the original image and filtered image. The generality of evolved filters (i.e., the ability to operate sufficiently also for other images containing the same type of noise) is tested by means of a test set.

A. Original CGP for filter evolution

The evolutionary design of filter circuits is based on Cartesian Genetic Programming [17]. In the original usage of CGP for filter design (according to [18]), a candidate filter is represented using a graph which contains n_c (columns) $\times$ n_r (rows) nodes. The role of the EA is to find the interconnection

TABLE II
THE LIST OF FUNCTIONS THAT CAN BE IMPLEMENTED IN EACH PROGRAMMABLE NODE

code	function	description
0	255	constant
1	x	identity
2	$255 - x$	inversion
3	$max(x, y)$	maximum
4	$min(x, y)$	minimum
5	$x \gg 1$	right shift by 1
6	$x \gg 2$	right shift by 2
7	$x + y$	+ (addition)
8	$x +^S y$	+ with saturation
9	$(x + y) \gg 1$	average
10	y if $(x > 127)$ else x	condition
11	$\|x - y\|$	absolute difference
12	$x \vee y$	bitwise OR
13	$x \wedge y$	bitwise AND
14	$x \oplus y$	bitwise XOR
15	$\overline{x \wedge y}$	bitwise NAND

of the programmable nodes and the functions performed by the nodes. Each node represents a two-input function that receives two 8-bit values and produces an 8-bit output. Examples of functions are given in Table II. A node input may be connected either to the output of another node, which is placed anywhere in the preceding columns or to a primary input of the filter. The filter circuits are encoded as strings of integers of the size $3 \times n_r \times n_r + 1$. For each node, three integers are utilized which encode the connection of the node inputs and operation. The last integer of the array encodes the primary output of the candidate filter.

CGP was originally used to design filters with 9 inputs, i.e. with the 3×3-pixel filter window. Unfortunately, the evolutionary search method is not scalable if larger filtering windows are considered (e.g. the number of inputs is increased to 25).

B. Extended CGP for filter evolution

In order to simultaneously support larger filtering windows and leave the problem reasonably difficult for evolution, CGP was extended by a selector that determines the pixels of filtering window which will be used in evolved 9-input filters [9]. The selector is encoded as a string consisting of S bits that are joined to the chromosome (see Fig. 3). If the bit of the selector string corresponding to the given pixel of the filter window possesses logic 1, then the pixel will be selected; otherwise, the pixel will not be considered as the input to the filtering logic.

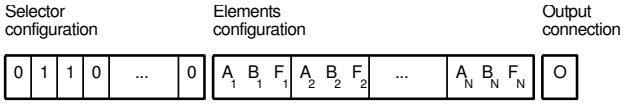


Fig. 3. Proposed encoding of candidate filters

The CGP utilizes a single genetic operation – the mutation – which may modify up to 5% of genes (integers) of the chromosome. If the selector is going to be mutated, a special procedure is utilized for its effective modification. This procedure takes two different indexes from the range 0 to $S - 1$ according to which the appropriate bits of the selector string are swapped. If a bit possessing logic 0 is swapped with a bit possessing logic 1, then the selector is altered and a new combination of pixels will be selected from the filter window on the basis of the 1's arrangement in the selector string.

C. Fitness function

The design objective is to minimize the difference between the filtered image and the original image. Usually, the *mean difference per pixel* also known as the *mean absolute error* (MAE) is minimized. Let u denote a corrupted image, v is the filtered image and w is the original (uncorrupted) version of u. The image size is $K \times K$ (K=256) pixels but only the area of $(256 - k + 1) \times (256 - k + 1)$ pixels is considered because the pixel values at the borders are ignored and thus remain unfiltered. The fitness value of a candidate filter is obtained by calculating the error function:

$$f = \frac{1}{(K-2)^2} \sum_{i=1}^{K-2} \sum_{j=1}^{K-2} |v(i,j) - w(i,j)|.$$

The objective is to minimze f, i.e. the lower value of f the better filter.

It is evident that the robustness of evolved filter depends on the selection of the training data. In the previous research, it has been determined that the image containing 128×128 pixels provides a sufficient amount of training data for evolution of robust 3×3 filters [19]. Because we utilize a larger filter window in this work, we will choose the training image consisting of 256×256 pixels.

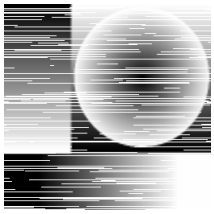


Fig. 4. The training image consisting of 256×256 pixels

IV. RESULTS

An extended version of CGP was utilized for evolution of impulse bursts noise filters with the 5×5-pixel filtering window. In fact, CGP has evolved 9-input filters whose inputs were identified using the selector. In order to compare the results with the standard CGP approach, the impulse bursts noise filters with the 3×3-pixel filtering window were also evolved.

A. CGP parameters

The CGP parameters were initialized as follows: $n_c \times n_r = 6 \times 6$; population size = 8; mutation rate = 5%; the number of generations = 50,000; the number of independent runs = 150; the function set of CGP was taken according to Table II. Figure

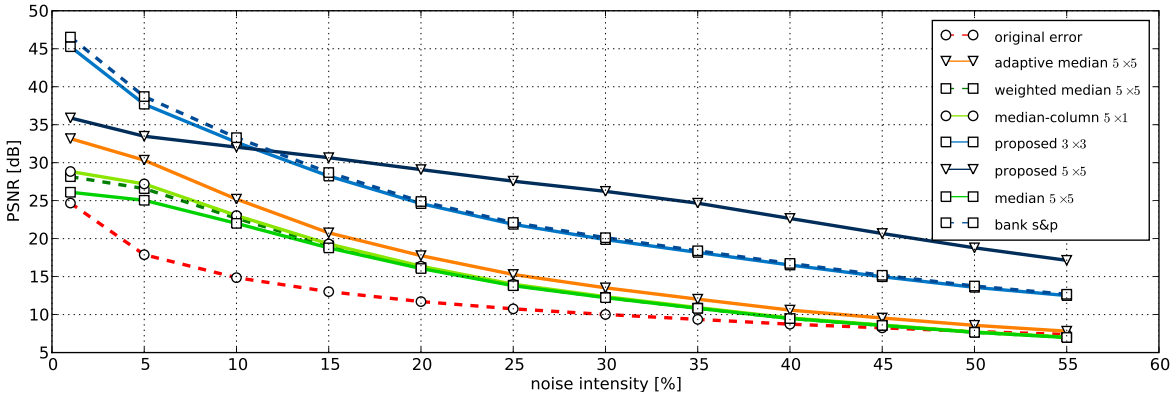


Fig. 5. The average PSNR calculated using 15 images for different levels of noise intensity

4 shows the training image used in the fitness function. The initial population was generated randomly. Previous studies showed for filter design problem that the setting of CGP does not influence the results significantly.

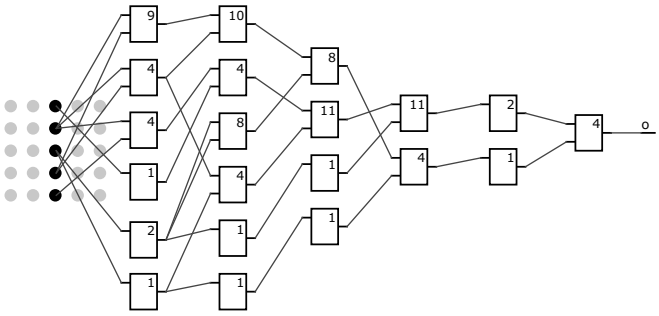


Fig. 6. The structure of the best evolved filter for the impulse burst noise (the functions are numbered as in Table 2)

B. Implementation cost

Figure 6 shows one of filters evolved with the 5×5 filtering window. All components of the filter are equipped with registers in order to support pipeline filtering. The filter's chromosome was converted to VHDL description and synthesized using Precision Synthesis to Xilinx Virtex II Pro XC2VP50. Table III compares the implementation cost of evolved filters with various existing approaches. The filter denoted as 'proposed' with the 5×5 filter window occupies 108 slices and can operate at 242 MHz. We can observe that evolved filters are relatively small.

C. Quality of filtering

The filtering quality of evolved (proposed) filters 3×3 and 5×5 is evaluated using 15 test images (256×256 pixels, randomly selected from [20]) that have been corrupted by aforementioned impulse bursts noise with the intensity of 5-55%. This corresponds to the following parameters of the noise model: $p = 0.05 \div 0.55$; $q = 128$.

In order to evaluate the quality of filtering of selected approaches, the average values of the peak signal to noise ratio (PSNR) and MAE were calculated over all test images for selected noise levels (Figures 5 and 7). We included evolved filters ('proposed 3×3' and 'proposed 5×5'), standard filters ('median 5×5', 'weighted median 5×5', 'adaptive median 5×5') and a bank of filters trained for salt and pepper noise ('bank s&p 3×3' – according to [2]). Moreover, an experimental median filter utilizing five pixels of the central column as its inputs (inspired by the evolved 5×5-pixel window filter from Fig. 6) is included for the comparison purposes; this filter is denoted as 'median–column' in Figures 5 and 7. Since the conventional methods provide low-quality of filtering if the 3×3 filter window is used, only the conventional filters that utilize the 5×5 filter window are considered for comparison with evolved (proposed) filters. It can be seen that both measures (MAE and PSNR) provide very similar results.

TABLE III
RESULTS OF SYNTHESIS FOR VARIOUS FILTERS

filter	window size	# CLB	max. freq.	latency
median filter	3×3	268	305 MHz	9
median filter	5×5	1 506	305 MHz	24
median-column	5×1	69	310 MHz	5
adaptive median	5×5	2 024	303 MHz	15
adaptive median	7×7	6 567	298 MHz	21
proposed	3×3	72	281 MHz	5
proposed	5×5	108	242 MHz	6
evolved s&p	3×3	199	318 MHz	8
bank s&p	3×3	843	305 MHz	13

Figure 8 shows filtering capabilities of various filters for the image corrupted by the 5% noise and Figure 9 shows the results for the image corrupted by 40% noise. The results demonstrate that the evolved (proposed) filter with 5×5-pixel window provides the best results in most cases, especially for higher noise intensity. It is evident that the filtering quality of the experimental 'median–column' filter is very low in comparison with evolved 5×5-pixel window filter (see Figures 8d,h and Figures 9c,f).

D. Computational requirements

The experiments were conducted on a cluster consisting of 100 PCs (Pentium IV, 2.4GHz, 1GB RAM) using the Sun

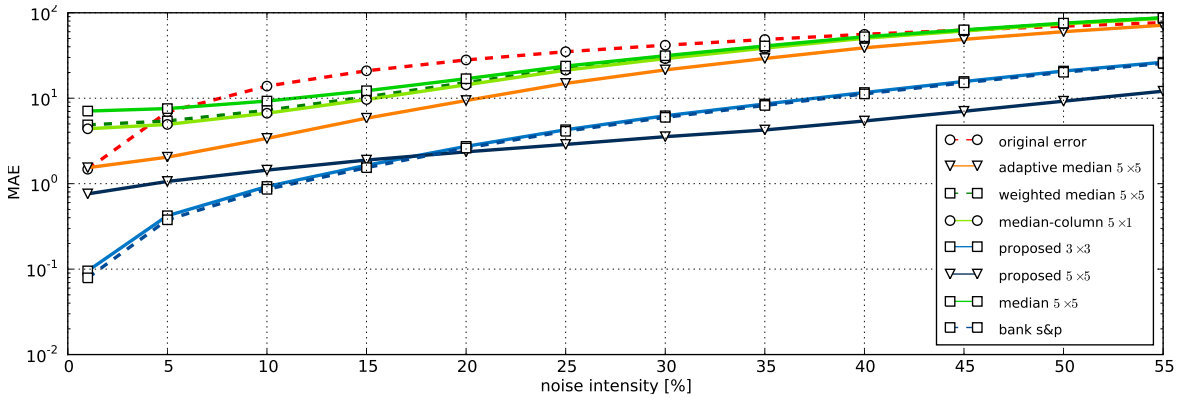


Fig. 7. The average Mean Absolute Error calculated using 15 images for different levels of noise intensity

Grid Engine (SGE) that enables to run up to 100 independent experiments in parallel. The evolution time of a single run is approximately 6 hours until the CGP algorithm reaches 50,000 generations.

V. Conclusions

The 'average' results from Figures 5 and 7 show that evolved filters exhibit a very good quality of filtering in comparison with other filters. Simultaneously, the implementation cost given in Table III is quite favorable. It is interesting that the best filter discovered by CGP (see Figure 6) utilizes only the pixels of the central column of the filter window. These pixels are clearly the most important ones to suppress this type of noise.

Similarly to the results presented for the salt and pepper noise [2], [3], [4], we can claim that CGP is able to provide very good implementations of impulse bursts noise filters. In our future work, we aim to apply the CGP for evolution of area-efficient filters for other noise types.

Acknowledgments

This work was partially supported by the Grant Agency of the Czech Republic under contract No. P103/10/1517 Natural Computing on Unconventional Platforms, No. GD102/09/H042 Mathematical and Engineering Approaches to Developing Reliable and Secure Concurrent and Distributed Computer Systems and the Research Plan No. MSM 0021630528 Security-Oriented Research in Information Technology.

References

[1] Z. Vasicek and L. Sekanina, "Novel hardware implementation of adaptive median filters," in *Proc. of 2008 IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop*. IEEE Computer Society, 2008, pp. 110–115.

[2] ——, "An area-efficient alternative to adaptive median filtering in FPGAs," in *Proc. of the 17th Conf. on Field Programmable Logic and Applications*. IEEE Computer Society, 2007, pp. 1–6.

[3] ——, "An evolvable hardware system in Xilinx Virtex II Pro FPGA," *International Journal of Innovative Computing and Applications*, vol. 1, no. 1, pp. 63–73, 2007.

[4] ——, "Reducing the area on a chip using a bank of evolved filters," in *Evolvable Systems: From Biology to Hardware*, ser. LNCS, vol. 4684. Springer Verlag, 2007, pp. 222–232.

[5] T. Higuchi, M. Iwata, D. Keymeulen, H. Sakanashi, M. Murakawa, I. Kajitani, E. Takahashi, K. Toda, M. Salami, N. Kajihara, and N. Otsu., "Real-World Applications of Analog and Digital Evolvable Hardware," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 3, pp. 220–235, 1999.

[6] T. Higuchi, Y. Liu, and X. Yao, *Evolvable hardware*. Berlin: Springer, 2006.

[7] P. Koivisto, J. Astola, V. Lukin, V. Melnik, and O. Tsymbal, "Removing Impulse Bursts from Images by Training-Based Filtering," *EURASIP Journal on Applied Signal Processing*, vol. 2003, no. 3, pp. 223–237, 2003.

[8] L. Sekanina, *Evolvable components: From Theory to Hardware Implementations*, ser. Natural Computing. Springer-Verlag Berlin, 2004.

[9] Z. Vasicek, M. Bidlo, L. Sekanina, J. Torresen, K. Glette, and M. Furuholmen, "Evolution of impulse bursts noise filters," in *Proc. of the 2009 NASA/ESA Conference on Adaptive Hardware and Systems*. IEEE Computer Society, 2009, pp. 27–34.

[10] P. Koivisto, H. Huttunen, and P. Kuosmanen, "Training-based optimization of soft morphological filters," *Journal of Electronic Imaging*, vol. 5, no. 3, pp. 300–322, 1996.

[11] M. O. Ahmad and D. Sundararajan, "A fast algorithm for two-dimensional median filtering," *IEEE Transactions on Circuits and Systems*, vol. 34, pp. 1364–1374, 1987.

[12] W. Zhou and Z. David, "Progressive switching median filter for the removal of impulse noise from highly corrupted images," *IEEE Trans On Circuits and Systems: Analog and Digital Signal Processing*, vol. 46, no. 1, pp. 78–80, 1999.

[13] D. R. K. Brownrigg, "The weighted median filter," *Commun. ACM*, vol. 27, no. 8, pp. 807–818, 1984.

[14] S. Marshall, "New direct design method for weighted order statistic filters," *VISP*, vol. 151, no. 1, pp. 1–8, February 2004.

[15] H. Hwang and R. Haddad, "Adaptive median filters: new algorithms and results," *IP*, vol. 4, no. 4, pp. 499–502, April 1995.

[16] H. Hwang and R. A. Haddad, "New algorithms for adaptive median filters," in *Proc. SPIE Vol. 1606, p. 400-407, Visual Communications and Image Processing '91: Image Processing, Kou-Hu Tzou; Toshio Koga; Eds.*, K.-H. Tzou and T. Koga, Eds., Nov. 1991, pp. 400–407.

[17] J. F. Miller and D. Job, "Principles in the evolutionary design of digital circuits – part I," *Genetic Programming and Evolvable Machines*, vol. 1, no. 1, pp. 8–35, April 2000.

[18] L. Sekanina, "Image filter design with evolvable hardware," in *Applications of Evolutionary Computing*, vol. 2002, no. 2279. Springer Verlag, 2002, pp. 255–266.

[19] T. Martinek and L. Sekanina, "An evolvable image filter: Experimental evaluation of a complete hardware implementation in FPGA," in *Evolvable Systems: From Biology to Hardware*, ser. LNCS, vol. 3637. Springer Verlag, 2005, pp. 76–85.

[20] "Berkeley Segmentation Dataset: Images," 2003, `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/BSDS300/html/dataset/`.

(a) original image

(b) corrupted image

(c) proposed filter 3×3

(d) proposed filter 5×5

(e) adaptive median filter 5×5

(f) bank

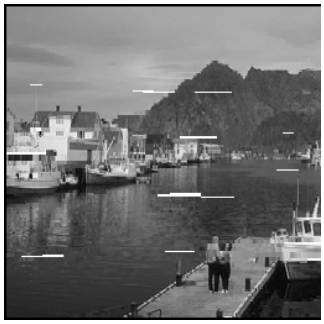
(g) median filter 5×5

(h) median-column filter 5×1

Fig. 8. Image (219090) corrupted by 5% impulse bursts noise filtered by various filters

(a) corrupted image

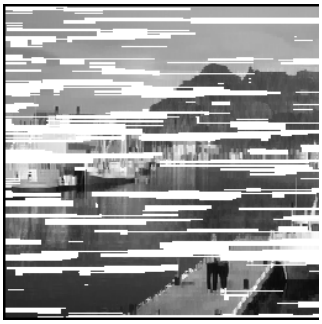
(b) proposed filter 3×3



(c) proposed filter 5×5

(d) adaptive median filter 5×5

(e) bank



(f) median-column filter 5×1

Fig. 9. Image (219090) corrupted by 40% impulse bursts noise filtered by various filters