JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 30, 605-618 (2014)

DOI:10.1688/JISE.2014.30.3.4

Evolutionary Design of Evolvable Hardware Image Filters Using Fuzzy Noise Models*

CHIH-HUNG WU[1,+], CHIEN-JUNG CHEN[2] AND CHIN-YUAN CHIANG[1]
[1]*Department of Electrical Engineering*
National University of Kaohsiung
Kaohsiung, 811 Taiwan
[2]*Intelligent Systems Department*
Infochamp Systems Corporation
Kaohsiung, 806 Taiwan

Image filtering, which removes or reduces noise from contaminated images, is an important task in image processing. This study deals with evolutionary design of image filters that can be implemented on evolvable hardware platforms using fuzzy noise models. Two fuzzy sets, similarity and divergence, are defined for classifying noise. Three filtering modules for pixels with various degrees of noise contamination are trained supervisedly by Cartesian genetic programming. The recovery of a noisy pixel is the fuzzy weighted sum of the output from the three filtering modules. Because each image filter is dedicated to a specific type of noise, it can produce a more accurate value for pixel recovery. With the proposed method, better accuracy of image filtering can be obtained. This paper evaluates and compares the performance of our proposed method with other ones.

Keywords: evolvable hardware, Cartesian genetic programming, image filter, fuzzy sets, salt and pepper noise

1. INTRODUCTION

For the past decades, various image filtering techniques have been designed and developed. The efficiency and accuracy of image filters are keys to success in industrial applications. Adaptive image filters that can learn how to handle unknown types of noise have been receiving a lot of attention. Similar to other machine learning methods, most adaptive image filters are implemented as software. Usually, a large amount of, yet complicated, calculation steps are performed sequentially by software-implemented image filters for computing recovery pixels. On the contrary, hardware-implemented image filters deal with the recovery of image pixels in parallel and usually perform efficiently. However, it is difficult to fully implement in hardware the functions of adaptability and flexibility due to their implementation complexity [1, 2].

Recently, developing hardware of adaptive ability is an emerging research topic. Evolvable hardware (EHW) [3] is a combination of evolutionary algorithm (EA) and reconfigurable hardware devices. An EHW-based solution is a hardware configuration that is encoded as a chromosome describing the structure of reconfigurable hardware devices, *e.g.*, Field Programmable Gate Array (FPGA) or Complex Programmable Logic Device

Received February 28, 2013; accepted June 15, 2013.
Communicated by Hung-Yu Kao, Tzung-Pei Hong, Takahira Yamaguchi, Yau-Hwang Kuo, and Vincent Shin-Mu Tseng.
* This work was partially supported by National Science Council, Taiwan, under Grant No. NSC 100-2221-E-390-029.
+ Corresponding author.

(CPLD). With the evolving of EA, EHW can change adaptively the structures of chromosomes (hardware configurations) and produce flexible results for un-modeled problems. EHW-based image filtering [4, 5] that takes the advantages of EA's adaptivity and hardware parallelism is an emerging role in industrial applications. Recent studies have demonstrated that EHW-based image filters can outperform several traditional implementations of hard-ware filters. Most studies assume that a single filtering function is applicable for all pixels and the degree of noise contamination is assumed to be uniformly contained in an image. Real cases reveal that various degrees of noise contamination exist and a single filtering function results in low accuracy even when the filtering function is obtained by a long training time. Besides, the reconfigurable devices only provide fundamental circuits that perform simple operations. Exhaustive search is inevitable for synthesizing a complicated function from a huge number of combinations of simple circuits.

This study deals with the design of EHW-based image filters using fuzzy noise models. Two fuzzy sets, similarity and divergence, are defined for classifying noise. Three filtering modules for pixels with various degrees of noise contamination are trained supervisedly by Cartesian genetic programming. Because each image filter is dedicated to a specific degree of noise contamination, it can recover pixels that are of various degrees of noise contamination more accurately. The recovery of a noisy pixel is the fuzzy weighted sum of output from the three filtering modules. This paper presents the main idea and demonstrates the performance of our proposed method on filtering salt-and-pepper noises. Empirical studies show that the proposed approach outperforms conventional designs in terms of recovery accuracy and training efficiency.

The remaining of the paper is organized as follows. Section 2 surveys the techniques of EHW-based image filtering. Section 3 describes main concept of the evolutionary design of EHW-based image filters. In Section 4, the analysis of noisy pixels by fuzzy sets for evolution of image filters is presented. Section 5 presents the experimental results. Finally, conclusions are given in Section 6.

2. RELATED WORK

Many filtering techniques have been developed. For example, the median filter [6] is the most popular nonlinear method for image filtering. The median filter replaces a pixel by the medium value of the neighboring pixels included in a sliding window. The median filter is simple and straightforward to be implemented in either software or hardware manners. However, most pixels are modified and a limited filtration quality is obtained. Among the variant versions of the median method [7-9], the adaptive median filter [10] employs sliding windows of variable sizes and detects the size of the noises. It is shown that the adaptive median filter produces significantly better results than conventional image filters. Several EHW-based image filters have been developed in the recent years. Lukáš Sekanina briefs the issues of design and implementation of EHW-based image filters in [4, 5]. A single EHW filter is developed in the early work. The concept of 3-bank EHW-based image filters is presented in [11, 12]. Instead of evolving a single EHW image filter, the 3-bank system evolves three filters simultaneously in the training phase. When recovering a noisy pixel, the recovering value is the average of outputs from three filters. Li and Huang [13] propose a two-stage EHW image filter. The first

stage detects noises and the second stage performs noise cancellation. Function level evolution is performed for training EHW image filters for noise that are detected in the first stage. An EHW-based image enhancement technique is presented in [14]. EHW circuits for image enhancement are generated by evolved histogram stretching transformation. Zdeněk *et al* develop EHW-based image filters for impulse burst noises [1]. An enhanced version is presented in [15] that incorporates a switch as a noise detector. In [16], the evolutionary design of area-efficient filters for impulse burst noises with respect to physical FPGA constraints is studied. Implementing EHW image filters in FPGAs should considers physical hardware constraints, such as circuit areas, complexity, power consumption, *etc*. The readers may refer to [2, 17-19] for more information about the implementation issues of EHW.

3. EHW-BASED IMAGE FILTERING

This section presents the typical design process of EHW-based image filters. Let I and I' be clean and noisy images with $k \times k$ pixels in a color domain D, respectively. Let $I(x, y)$ be a pixel in I at the location (x, y), $1 \le x, y \le k - 1$. Typically, an image filter F works with a sliding window M_m of $m \times m$ in size, where m is usually an odd integer. Let $M_m(x, y)$ denote the set of $m \times m$ pixels covered by M_m centered at (x, y). Filter F can be considered as a function $f: D^{m^2} \to D$ mapping $M_m(x, y)$ to a recovering value. The purpose of image filtering is to obtain an image I'' by scanning all pixels in I' and determines their corresponding outputs according to f. In this study, D is the 8-bit/pixel gray scale domain. The filtering effectiveness of F is evaluated by the peak signal to noise ratio (PSNR).

$$PSNR = 10\log_{10}\frac{255^2}{\frac{1}{k^2}\sum_x\sum_y [I''(x,y) - (x,y)^2]} \tag{1}$$

In EHW-based image filtering, F is a combination of digital circuit components with $m \times m$ input and a single output. The structure of F is determined by EA according to the hardware specifications and the training set of pixels. The EA employed in this study is Cartesian Genetic Programming (CGP) [20]. CGP is a supervised genetic algorithm for symbolic regression. CGP employs 2-D graphical representation consisting of $c \times r$ nodes interconnected to each other. An input to a node may be from a primary input of the filter or from an output of another node. Each node performs a two-input function and produces an output to other nodes. The functions can be organized by the fundamental circuits that are available in the hardware devices, such as the ones listed in Table 1. These functions are commonly used in other EHW-based methods [4, 11, 12].

Like in most genetic-based methods, CGP employs chromosome-like encoding. Each node in CGP is described by three integers, the first two denote the two input sources and the last one denotes the function adopted by the node. A circuit is encoded as a list of $3 \times c \times r + 1$ integers, describing the 2-D layout of $c \times r$ nodes. The final integer in the list denotes the output from a node in the last columns of nodes. The connectivity of nodes is restricted by *level-back*, which denotes the number of backward levels of nodes that can serve as input to a node. Fig. 1 illustrates a 2-D graphical representa-

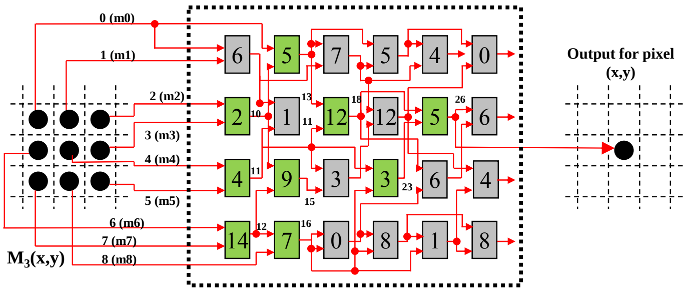
tion of a circuit in CGP and the corresponding chromosome. The genetic operators like crossover and mutation are also applicable in CGP. Various circuit configurations are produced by applying crossover or mutation on chromosomes. To prevent the generation of meaningless circuits, node functions are restricted to mutate to the ones in the given list and the input sources of a node are restricted to mutate to the primary inputs or outputs from neighboring nodes. The objective of designing EHW-based image filters is to minimize the difference between the recovered and original (clean) images. Therefore, the mean difference per pixel a.k.a. the mean absolute error (MAE) is employed as the fitness of a chromosome c_i. That is,

$$fitness(c_i) = \frac{1}{(k-2)^2} \sum_{x=1}^{k-2} \sum_{y=1}^{k-2} | I^{c_i}(x, y) - I(x, y) |, \tag{2}$$

where I^{c_i} is the image recovered by the filter corresponding to the chromosome c_i. The processing flow of CGP is similar to other genetic-based methods. It starts with a set of random chromosomes, evaluates the fitness of all chromosomes, performs re-production, crossover, and mutation, and repeats the whole process if necessary. For more details about CGP, please refer to [20].

Table 1. Common functions used in EHW-based methods: $f(i_1, i_2)$.

ID	Function	ID	Description	ID	Function	Description
0	255	0	Constant	8	$i_1 \gg 1$	Right shift by 1
1	i_1	1	Identity	9	$i_1 \gg 2$	Right shift by 2
2	$255 - i_1$	2	Inversion	10	Swap(i_1, i_2)	Swap nibbles
3	$i_1 \vee i_2$	3	OR	11	$i_1 + i_2$	+ (Addition)
4	$\neg i_1 \vee i_2$	4	($\neg i_1$) or i_2	12	$i_1 + S i_2$	+ with Saturation
5	$i_1 \wedge i_2$	5	AND	13	$(i_1 + i_2) \gg 1$	Average
6	$\neg(i_1 \wedge i_2)$	6	NAND	14	max(i_1, i_2)	Maximum
7	$i_1 \oplus i_2$	7	XOR	15	min(i_1, i_2)	Minimum



(a) A 2-D layout of 4 × 6 nodes with level-back = 2 and 3 × 3 mask.

0	1	6	2	3	2	4	5	4	6	7	14	0	10	5	9	11	1	10	12	9	12	8	7	13	9	7	13	11	12	11	15	3	16	16	0	13	17	5	17	19	12

11	16	3	20	16	8	21	17	4	18	23	5	18	20	6	24	16	1	21	22	0	26	27	6	22	28	4	24	28	8	1

(b) The chromosome (3 × 4 × 6 + 1 integers) corresponding to the layout of (a).

$output = \{[m0 \wedge (255 - m2)] + S(\neg m4 \vee m5)\} \wedge \{(\neg m4 \vee m5) \vee [(max(m6, m7)) \oplus m8]\}$

(c) The function corresponding to the chromosome of (b).

Fig. 1. Circuit representation in CGP.

4. THE PROPOSED METHOD

4.1 Modeling Image Noises

Previous studies claim that EHW-based image filters outperform traditional filtering methods. However, it assumes that pixels can be recovered by a single filtering function. A single noise model is adopted in these methods and a single filtering function is learned. Basically, training patterns are pixels extracted from the sliding windows, $M_m(x, y)$. If all $M_m(x, y)$ present a common pattern, learning a single EHW function is efficient and effective. For complicated image filtering problems, there may exist large variance between various $M_m(x, y)$; it makes the training process hard to converge even running after a long training time. This study defines noise models using fuzzy sets and generates multiple image filters each of which deals with a specific model of noise patterns. Below is the main idea.

Let $p_{x,y}$ denote the value of a pixel $I(x, y)$ at the location (x, y) of an image I. Two indicators, *similarity* and *divergence*, are defined for describing noise models.

$$similarity(x, y) = \frac{1}{m^2 - 1} \sum_{I(i,j)} sim(p_{i,j}, p_{x,y}) \tag{3}$$

$$sim(p_{i,j}, p_{x,y}) = \frac{\min\{p_{i,j}, p_{x,y}\}}{\max\{p_{i,j}, p_{x,y}\}} \tag{4}$$

where $I(i, j)$ are elements of $M_m(x, y)$ surrounding $I(x, y)$, $I(i, j) \neq I(x, y)$.

$$divergence(x, y) = \frac{1}{m^2 - 1} \sum_{I(i,j)} \left| p_{i,j} - p_{x,y} \right| \tag{5}$$

The indicator $sim(p_{i,j}, p_{x,y})$ describes the similarity between two pixels. The indicator $similarity(x, y)$ evaluates the degree of similarity of pixel $I(x, y)$ with respect to its neighboring pixels contained in $M_m(x, y)$. The indicator $divergence(x, y)$ represents the average differences of $I(x, y)$ against to its neighboring pixels. For example, Fig. 2 presents the similarity and divergence of pixels in two sliding windows.

4.2 Fuzzy Modeling of Noises

Based on the features of image filtering, this study defines fuzzy noise models. With the two indicators, similarity and divergence, the following fuzzy sets are defined.

1. Fuzzy variables: similarity ($similarity(x, y)$) and divergence ($divergence(x, y)$); noise type ($noise(x, y)$).
2. Fuzzy terms: Terms associated with $similarity(x, y)$: LOW and HIGH; terms associated with $divergence(x, y)$: SMALL and LARGE; terms associated with $noise(x, y)$: LOW, MEDIUM, HIGH.
3. Fuzzy membership functions associated with $similarity(x, y)$ and $divergence(x, y)$ are defined as continuous functions, as shown in Fig. 3, respectively.

130	137	149
131	141	149
127	145	150

(a) Mask-1.

0.922	0.972	0.946
0.941	0.941	0.946
0.972	0.972	0.940

(b) Similarity of (a).

11	4	8
10	8.5	8
14	4	9

(c) Divergence of (a).

130	0	149
131	255	149
0	145	150

(a) Mask-2.

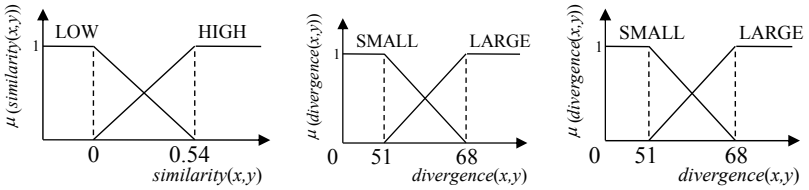
0.510	0	0.584
0.514	0.419	0.584
0	0.569	0.588

(e) Similarity of (d).

125	255	106
124	148.25	106
255	110	105

(f) Divergence of (d).

Fig. 2. Similarity and divergence in two masks.



(a) Similarity. (b) Divergence(x, y). (c) Noise possibility.

Fig. 3. Fuzzy membership functions for describing noise.

This study considers *noise*(x, y) as a fuzzy set derived from similarity and divergence using the following fuzzy rules.

- **IF** *similarity*(x, y) is LOW **AND** *divergence*(x, y) is SMALL **THEN** *noise*(x, y) is MEDIUM.
- **IF** *similarity*(x, y) is LOW **AND** *divergence*(x, y) is LARGE **THEN** *noise*(x, y) is HIGH.
- **IF** *similarity*(x, y) is HIGH **AND** *divergence*(x, y) is SMALL **THEN** *noise*(x, y) is LOW.
- **IF** *similarity*(x, y) is HIGH **AND** *divergence*(x, y) is LARGE **THEN** *noise*(x, y) is MEDIUM.

With these simple fuzzy rules, the noise type of pixel $I(x, y)$ is determined by standard fuzzy rule inference. That is, *similarity*(x, y) ∘ *divergence*(x, y) is first evaluated, where ∘ denotes the ∨-∧ composition, *i.e.*,

$$noise(x, y) = \arg\max_{type_{ij}} \{\mu_{noise(x,y)=type_{ij}}\}. \tag{6}$$

For each type of *noise*(x, y),

$$\mu_{noise(x,y)=type_{ij}} = \min_{i,j} \{\mu_{similarity(x,y)=S_i}, \mu_{divergence(x,y)=D_i}\}, \tag{7}$$

where $type_{i,j} \in$ {LOW(L), HIGH(H), MEDIUM(M)} and S_i, D_j, $0 \le i, j \le 1$, are the fuzzy terms associated with *similarity*(x, y) and *divergence*(x, y), respectively.

4.3 Training EHW-based Image Filters with Fuzzy Noise Models

Our proposed method is to evolve three EHW filtering modules, $F_\pi = \{L|H|M\}$,

each of which works for a specific noise model, *i.e.*, F_L (LOW), F_H (HIGH), F_M (MEDIUM). Pixels for training F_π is first filtered by the fuzzy rules for determining their noise types (degrees of contamination). A pixel $I(x, y)$ belonging to noise model $noise(x, y)$ is used for training its corresponding filter. CGP evolves image filter F_π by pixels with the same noise model.

Suppose that a clean image I_0 ($k \times k$) and a noisy image I (corrupted I_0) are given. A mask of size m is defined in the training phase, three CGP workspaces are created. In each CGP workspace, typical genetic operations apply as usual, except that only the pixels with the specific noise type $type_{ij}$ are used for training the corresponding EHW filter $Ftype_{ij}$. The main processing steps of our method are listed below.

Step 1: Initialize three CGP workspaces. Assign CGP workspace CGP_π for training filer $F_\pi = \{L|H|M\}$.
Step 2: Initialize s chromosomes in each CGP_π.
Step 3: For each chromosome ξ_i, $1 \le i \le s$, in each CGP, do fitness calculation.
Step 4: For each pixel $I(x, y)$ in I, compute $similarity(x, y)$ and $divergence(x, y)$.
Step 5: Apply the fuzzy rules for deciding $I(x, y)$'s noise type ($noise(x, y)$).
Step 6: Calculate the fitness of ξ_i for F_π on $I(x, y)$ with $noise(x, y) = \pi$(Steps 7-9).
Step 7: Compute the recovery output $I'(x, y)$ of $I(x, y)$ from F_π using ξ_i and compare the difference between $I'(x, y)$ and $I_0(x, y)$; then obtain $|I_0(x, y) - I'(x, y)|$.
Step 8: Accumulate $|I_0(x, y) - I'(x, y)|$ to the fitness (by Eq. (2)) of ξ_i for filter F_π.
Step 9: Repeat Steps 4-9 until all pixels have been done.
Step 10: Repeat Steps 3-10 until all chromosomes ξ_i for all F_π have been evaluated.
Step 11: If the termination conditions of a CGP workspace CGP_π are met, pick up the best chromosome ξ_π from CGP_π and output the corresponding image filter F_π.
Step 12: Apply genetic operators on the chromosomes in each CGP_π and goto Step 3.
Step 13: Output all image filters F_L, F_H, and F_M.

The training process is illustrated in Fig. 4.

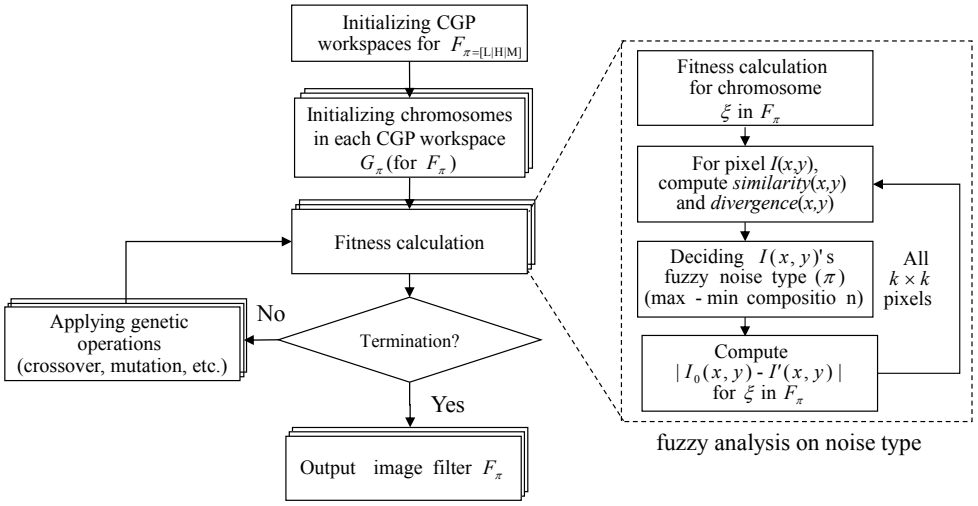


Fig. 4. The main processing flow of CGP.

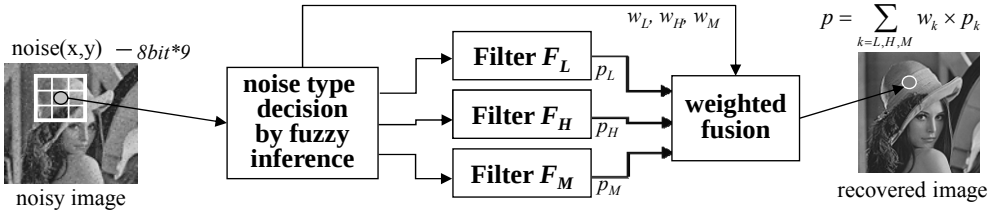


Fig. 5. The proposed fuzzy weighted recovery process.

4.4 Fuzzy Weighted Recovery

For recovering a noisy image I'' in the recovery phase, the noise type of each pixel $I''(x, y)$ is evaluated against the three image filters obtained from CGP. Applying the fuzzy inference rules mentioned in Section 4.2, the fuzzy membership degrees of $I''(x, y)$ are calculated. Let w_L, w_H, w_M be the degrees of $noise(x, y)$ = LOW, $noise(x, y)$ = HIGH, $noise(x, y)$ = MEDIUM, respectively. Assume that p_L, p_H, p_M denotes the output from image filters F_L, F_H, F_M, respectively. The value for recovering $I''(x, y)$ is the fuzzy weighted sum of p_L, p_H, and p_M.

$$I''(x, y) = w_L \times p_L + w_H \times p_H + w_M \times p_M. \tag{8}$$

The concept of fuzzy weighted recovery process is depicted in Fig. 5.

5. EXPERIMENTS

The effectiveness of our proposed method has been verified on the recovery of some real-world images. This section presents the experimental results. In the experiments, the proposed method was implemented in C. The CGP parameters listed in Table 2 and the functions given in Table 1 were used. The experiments were executed on an Intel Xeon E5620, 2.4GHz CPU with 6 GB RAM. All experiments were repeated at least for three times with the same parameter settings. The results were averaged.

Table 2. The parameter setting on CGP.

Parameters:	Value	Parameters:	Value
Population size:	100	Generations:	5000
Prob. of crossover:	0.2	Mask size :	3×3
Prob. of mutation:	0.1	CGP grid size :	4×8

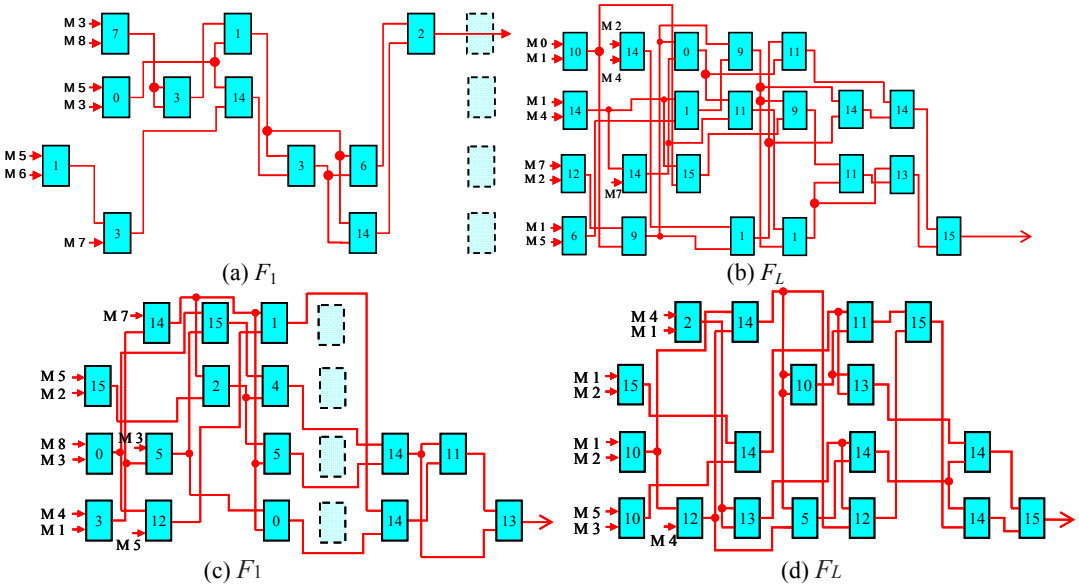
Table 3. Training results: MAE.

Image	F_1	F_L	F_H	F_M	MAE*
Peppers	3.58	0.00	5.44	4.10	2.91
Lena	6.32	0.00	6.00	6.31	4.69
Cameraman	8.13	0.00	7.84	5.78	4.46
Airplane	5.94	0.00	6.28	10.13	5.26

5.1 Training Effectiveness

Four images, Peppers, Lena, Cameraman, Airplane, were used to test the proposed method. These images were 256×265 in size and in 8-bit/pixel gray scale. Training images are the original (clean) images corrupted by 40% salt-and-pepper noises. Notice that the degree of noise contamination of the training patterns do affects the effectiveness of a filter learned by EHW. In the literature of EHW-based filter design, a 40% of noise contamination is widely accepted. This study also follows this rule. The training performance is mainly evaluated by MAE, whereas the recovery performance is evaluated by PSNR. For comparison, the original version of EHW-based image filter (F_1), *i.e.*, only one CGP module, was also implemented and trained with the same parameter settings. Table 3 presents the training results, where MAE* denotes the weighted average of MAE of F_L, F_H, and F_M. The experimental results show that the averaged training MAE of the proposed method is better than F_1. It somehow indicates that training a filter for a specific type of pixels is easier than that for all pixels. However, training filters F_L, F_H, and F_M has various difficulty. In all cases, F_L can be well-trained with MAE = 0. In some cases, F_H and F_M have larger MAE than F_1, *e.g.*, F_H and F_M for Pepper, F_H for Cameraman, and F_H and F_M for Airplane.

Fig. 6 presents the best image filters F_1, F_L, F_H, F_M evolved from CGP for filtering Lena. Interestingly, the most complex circuit for filtering Lena and Cameraman is F_1; whereas for Peppers and Goldhill, the most complex one is F_H. This may be caused by the imperfection on defining similarity and divergence. In the experiments, the fuzzy sets associated with similarity and divergence have the same definitions. It is also learned from the experiments that the fuzzy rules classify pixels; however, it is also found that the sizes of training patterns are imbalanced. Some pixels may be mis-classified by the fuzzy rules. Generating EHW circuits for homogeneous pixels may be easier than that for heterogeneous ones.



(a) F_1 (b) F_L

(c) F_1 (d) F_L

Fig. 6. The best circuits for Lena.

(a) Noisy image (b) Median filter (c) F_1 filter (d) Ours

Fig. 7. The recovery results of Lena (40% noise).

5.2 Testing the EHW Image Filters

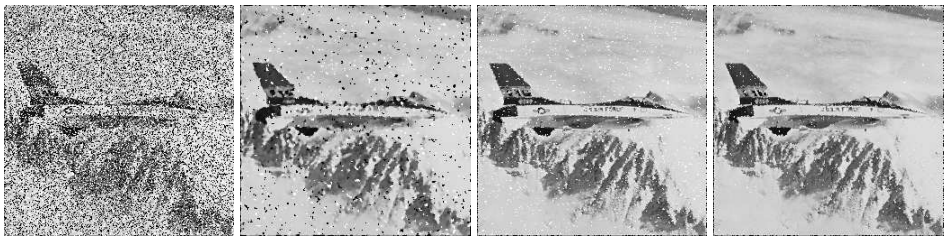
Next, the EHW image filters learned in Section 5.1 are applied for recovering images with various degree of noise density. For comparisons, a median filter was also implemented to filter the same corrupted images. Figs. 7-10 provide visual comparisons on the corrupted and recovered images. The results show that our proposed method is effective to recover noisy images than the median filter and filter F_1. Referring to Table 3, F_1 has simpler circuit complexity for Goldhill. But F_1 does not have a better recovery ability in the testing phase. The corresponding circuits of F_H and F_M are more complex but work well in the testing phase. This may be explained by the stochastic nature of CGP that overfits the training data. The same reason may also be applicable to F_L, F_H, and F_M. However, in the proposed method, each pixel is recovered by one of the three filters that are trained according to the characteristics of pixels. The proposed method can have a better recovery ability on filtering pixels.

(a) Noisy image (b) Median filter (c) F_1 filter (d) Ours

Fig. 8. The recovery results of Cameraman (40% noise).

(a) Noisy image (b) Median filter (c) F_1 filter (d) Ours

Fig. 9. The recovery results of Peppers (40% noise).



(a) Noisy image (b) Median filter (c) F_1 filter (d) Ours

Fig. 10. The recovery results of airplane (40% noise).

Table 4. Performance comparisons: PSNR v.s. noise density.

Image	Filter	Noise Density						Average
		5%	10%	20%	40%	50%	70%	
Lena	Median	28.44	27.83	25.86	18.32	14.89	9.88	20.87
	F_1	30.26	29.77	28.52	25.07	23.06	19.20	25.98
	Ours	33.06	31.49	29.82	26.06	23.45	16.55	26.74
Cameraman	Median	26.72	26.15	24.42	17.94	14.38	9.52	19.86
	F_1	32.16	30.02	27.08	23.35	20.88	14.32	24.64
	Ours	33.96	31.29	28.40	24.72	22.22	15.31	25.98
Pepper	Median	31.78	30.48	27.17	18.53	15.02	9.91	22.15
	F_1	30.59	30.11	29.35	26.47	22.78	15.21	25.75
	Ours	36.95	34.88	32.25	28.39	25.41	16.94	29.14
Airplane	Median	26.00	25.49	23.78	17.21	14.40	9.19	19.35
	F_1	31.12	29.58	26.90	21.99	19.27	14.64	23.92
	Ours	31.72	30.23	27.27	23.14	20.71	15.96	24.84

Table 4 presents the PSNR of the images recovered from various noise densities (5%-70%) using the EHW image filters. It is shown that the proposed method outperforms the median filter and F_1 in most cases except for recovering Lena when noise density is 70%. However, most filters also fail with low PSNR when noise density is high. The total performance of the proposed method is competitive.

6. CONCLUSIONS

This paper presents a new method that constructs fuzzy image filters for EHW platforms. By analyzing the patterns of pixels contained in a sliding window, pixels are classified by fuzzy sets of similarity and divergence. In the training phase, three modules of EHW image filters are built simultaneously. In the recovery phase, the recovery value of a pixel is the fuzzy weighted sum of outputs from the three EHW-based image filters. With the proposed method, the accuracy of image filtering can be improved. Although our proposed method presents a better solution to recover salt-and-pepper noises and outperforms other methods, there are several points to be improved. Currently, the fuzzy sets associated with similarity and divergence have the same definitions. Advanced analysis on the degrees of noise contamination using sophisticated fuzzy membership functions may further improve the accuracy of noise degree discrimination. Moreover,

pixels of specific noise types are used for training specific EHW modules exclusively, there may exist pixels belongs to two noise types with equal/similar degree of memberships of noise types. The training process may consider the weights of pixels of noise types and evaluate the fitness of chromosomes by considering the contributions of pixels in different noise types. One of the disadvantages of EHW-based methods is the training performance. This is due to long calculation time in evaluating chromosome fitness. Because all pixels (256 × 256) have to be investigated for a chromosome, a number of CGP iterations consume a considerable amount of computing time. Additionally, the training the fuzzy EHW filters is triple than that of F_1 because there are three filters to be trained. Improving the training performance via parallel processing may be a workable direction. These will be included in the future work.

REFERENCES

1. Z. Vasĭćek, M. Bidlo, L. Sekanina, J. Torresen, K. Glette, and M. Furuholmen, "Evolution of impulse bursts noise filters," in *Proceedings of IEEE NASA/ESA Conference on Adaptive Hardware and Systems*, 2009, pp. 27-34.
2. K. S. R. Krishna, A. G. Reddy, M. G. Prasad, K. C. Rao, and M. Madhavi, "Genetic algorithm processor for image noise filtering using evolvable hardware," *International Journal of Image Processing*, Vol. 4, 2010, pp. 240-250.
3. G. W. Greenwood and A. M. Tyrrell, *Introduction to Evolvable Hardware: A Practical Guide for Designing Self-Adaptive System*s, IEEE Press Series on Computational Intelligence, Wiley-IEEE Press, NJ, 2006.
4. L. Sekanina, "Virtual reconfigurable circuits for real-world applications of evolvable hardware," in *Proceedings of the 5th International Conference on Evolvable Systems: from Biology to Hardware*, 2003, pp. 186-197.
5. L. Sekanina, "Evolutionary hardware design," in *Proceedings of SPIE*, Vol. 8067, 2011, pp. 1-11.
6. M. Ahmad and D. Sundararajan, "A fast algorithm for two dimensional median filtering," *IEEE Transactions on Circuits and Systems*, Vol. 34, 1987, pp. 1364-1374.
7. D. R. K. Brownrigg, "The weighted median filter," *Communications of the ACM*, Vol. 27, 1984, pp. 807-818.
8. Z. Wang and D. Zhang, "Progressive switching median filter for the removal of impulse noise from highly corrupted images," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 46, 1999, pp. 78-80.
9. S. Marshall, "New direct design method for weighted order statistic filters," *IEE Proceedings on Vision, Image and Signal Processing*, Vol. 151, 2004, pp. 1-8.
10. H. Hwang and R. Haddad, "Adaptive median filters: new algorithms and results," *IEEE Transactions on Image Processing*, Vol. 4, 1995, pp. 499-502.
11. Z. Vasĭćek and L. Sekanina, "Reducing the area on a chip using a bank of evolved filters," in *Proceedings of International Conference on Evolvable Systems: From Biology to Hardware*, LNCS 4684, 2007, pp. 222-232.
12. Z. Vasĭćek and L. Sekanina, "An area-efficient alternative to adaptive median filtering in FPGAs," in *Proceedings of IEEE International Conference on Field Programmable Logic and Applications*, 2007, pp. 216-221.

13. J. Li and S. Huang, "Adaptive salt-&-pepper noise removal: A function level evolution based approach," in *Proceedings of NASA/ESA Conference on Adaptive Hardware and Systems*, 2008, pp. 391-397.
14. J. Li, "Evolvable hardware based gray-level image enhancement," in *Proceedings of NASA/ESA Conference on Adaptive Hardware and Systems*, 2009, pp. 67-74.
15. Z. Vasǐćek, M. Bidlo, L. Sekanina, and K. Glette, "Evolutionary design of efficient and robust switching image filters," in *Proceedings of IEEE NASA/ESA Conference on Adaptive Hardware and Systems*, 2011, pp. 192-199.
16. Z. Vasǐćek, L. Sekanina, and M. Bidlo, "A method for design of impulse bursts noise filters optimized for FPGA implementations," in *Proceedings of Design, Automation and Test in Europe*, 2010, pp. 1731-1736.
17. J. Wang and C. Lee, "Complete FPGA implemented evolvable image filters," in A. Gelbukh, C. Reyes-Garcia, eds., *Advances in Artificial Intelligence*, LNCS, Vol. 4293, Springer, Heidelberg, 2006, pp. 767-777.
18. P. Fernando, S. Katkoori, D. Keymeulen, R. Zebulum, and A. Stoica, "Customizable FPGA IP core implementation of a general-purpose genetic algorithm engine," *IEEE Transactions on Evolutionary Computation*, Vol. 14, 2010, pp. 133-149.
19. R. Salvador, A. Otero, J. Mora, E. de la Torre, T. Riesgo, and L. Sekanina, "Implementation techniques for evolvable hardware systems: virtual vs. dynamic reconfiguration," in *Proceedings of the 22nd International Conference on Field Programmable Logic and Applications*, 2012, pp. 547-550.
20. J. F. Miller, *Cartesian Genetic Programming*, 1st ed., Natural Computing Series, Vol. XXII, Springer, Heidelberg, 2011.

Chih-Hung Wu (吳志宏) received the B.S. degree in Engineering Science from National Cheng-Kung University in 1990, and the M.S. and Ph.D. degrees in Electronic Engineering from National Sun Yat-sen University, Taiwan, in 1992 and 1996, respectively. He is currently an Associate Professor with the Department of Electrical Engineering, National University of Kaohsiung, Taiwan. Dr. Wu is the Director of ICAL (Intelligent Computation and Applications Lab.), National University of Kaohsiung, Taiwan. His research interests include AI, soft computing, robotics, GPS, and cloud-computing. He is a member of the Taiwanese Association for Artificial Intelligence, Taiwan Fuzzy Systems Association, and Taiwan Automation Intelligence and Robotics Association. He is also a member of the IEEE (CI, I&M, and SMC Societies).

Chien-Jung Chen (陳建榮) was born in 1988. He received his B.S. degree from the Department of Information Technology and Management, Shih Chien University, Taiwan in 2010 and M.S. degree from the Department of Electrical Engineering, National University of Kaohsiung, Taiwan in 2012, respectively. He is an Engineer with Intelligent Systems Department, Infochamp

Systems Corporation, Kaohsiung, Taiwan. His research interests include fuzzy system, evolvable hardware, and evolutionary computing.

Chin-Yuan Chiang (江錦源) was born in 1989. He received his B.S. and M.S. degrees from the Department of Electrical Engineering, National University of Kaohsiung, Taiwan in 2011 and 2013, respectively. His research interests include parallel processing and fuzzy systems.