IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 12, 2018, accepted May 16, 2018, date of publication May 21, 2018, date of current version July 6, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2838596

An Accelerated Method for Determining the Weights of Quadratic Image Filters

SÜLEYMAN UZUN[1,2] **AND DEVRIM AKGÜN**[3]

[1]Open and Distance Learning Application and Research Center, Bilecik Şeyh Edebali University, 11230 Bilecik, Turkey
[2]Institute of Natural Sciences, Sakarya University, 54050 Sakarya, Turkey
[3]Computer Engineering, Sakarya University, 54050 Sakarya, Turkey

Corresponding author: Süleyman Uzun (suleyman.uzun@bilecik.edu.tr)

ABSTRACT Quadratic filters are usually more successful than linear filters in dealing with nonlinear noise characteristics. However, determining the proper weights for the success of quadratic filters is not straightforward as in linear case. For this purpose, a search algorithm used to train weights of quadratic filters from sample images by formulating the problem into a single objective optimization function. In the presented study, comparative inspections for training quadratic image filters using genetic algorithm (GA) and particle swarm optimization (PSO) were presented. Because computation of fitness function involves consecutive image filtering operation using candidate solutions, this process usually results in long training durations due to the computationally expensive nature of image processing applications. In order to reduce the computation times, variable and variable random fitness methods were implemented, where the image size varied in the computation of fitness function. Experimental results show that proposed algorithm provides about 2.5 to 3.0 fold acceleration in computation durations using both GA and PSO.

INDEX TERMS Genetic algorithm, image processing, particle swarm optimization, quadratic image filters, Volterra filters.

I. INTRODUCTION

Convolution filters used in image processing applications such as filtering, extracting the details of the image, image enhancements, etc. [1], [2]. Quadratic filters present the effective solution to eliminate noises which usually have nonlinear characteristics due to its nonlinear structure [3]. In most of the practical applications, Volterra series are limited to second degree terms due to its computational complexity. In general, these types of filters are called quadratic filters. Quadratic image filters hold both linear and second degree nonlinear terms in their structure. In literature, quadratic image filters are preferred for their success in preserving image details. Jothilakshmi and Gopinathan [4] used adaptive Volterra filters to achieve contrast enhancement of mammogram images which are affected by Gaussian, Poison and White noises. Meenavathi and Rajesh [3] suggest a new class of filters based on Volterra for image enhancement and image restoration. Zhou *et al.* [5] introduce a new nonlinear filter called the alpha weighted quadratic filter for mammogram enhancement. Mitra's work suggests that linear filters do not achieve much when removing nonlinear noises from images [6]. He reports that linear filters are blurring the edges of the image and losing image details while clearing these noises and nonlinear filters should be used to remove such noises. Kanamadi *et al.* [7] offer a new nonlinear alpha weighted quadratic filter using image processing techniques. Filter is based on masking technique which can remove noise from corrupted grey scale images while preserving image details. Ramponi *et al.* [8] presented a matrix representation for the class of nonlinear 2-D filters based on the second order Volterra series. An optimization method developed for designing the filter according to the required input/output relationship. Pandey *et al.* [9] presented a new technique for enhancing digital mammogram images using Volterra filters. Thomas *et al.* [10] focused on quadratic Volterra filtering for removing impulsive noise from MRI images which happens at extremely noisy conditions. They saw that when compared with performance parameters, quadratic Volterra systems are superior to traditional systems (median, mean, Gaussian filters) for removing impulsive noises. Hari *et al.* [11] designed a quadratic edge detection filter for enhancing calcifications in mammograms. Feng *et al.* [12] proposed a novel face recognition algorithm based on nonlinear Volterra kernels mapping and direct discrimination analysis. Bhateja *et al.* [13] developed a framework for Non-linear Polynomial Filtering for sharpening and edge

enhancement of mammogram lesions. Hari *et al.* [14] presented a method based on Volterra filter for edge detection and better localization of microaneurysms in fundus retinal images. According to their study, quadratic filters are more successful in preserving edges than both Teager filters and conventional edge detecting approaches.

The focus of the present study is to evaluate the training of quadratic image filters using population based search algorithms. For comparison, Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) in the experimental results. Initially the fitness function for the training the weights is implemented directly and it is called as Direct Fitness Method (DFM). Computation of the fitness function involves filtering a Gaussian noise contaminating reference image and computing the mean absolute error (MAE) between reference and a filtered noisy image. Because intensive operations for image filtering, utilization of GA and PSO or similar population based heuristic approach is expected to computationally expensive. In the presented paper, an improvement to reduce computation time, initially a ratio of the image for computing the fitness function image is selected. In Variable Fitness Method (VFM) this ratio is increased as the iterations goes by and after a certain iteration, the whole image is used. The group of pixels is also be determined randomly in the Variable Random Fitness Method (VRFM) instead of selecting a certain region to improve the solution. In the presented paper above mentioned approaches are developed and compared experimentally. The remainder of the paper is organized as follows; in the following section background about the quadratic filters and population based search algorithms used in the experiments are given. In the third section, training approach and the implemented algorithms DFM, VFM and VRFM are explained. In the fourth section, numerical results from detailed analyses were given. Finally a brief conclusions about the results were given.

II. METHODS

A. QUADRATIC IMAGE FILTERS

Quadratic filters which are also known as polynomial or Volterra filters are most popular class of nonlinear filters [8], [15]. Volterra filters are very important model for the synthesis, analysis and display of nonlinear systems [7]. The class of nonlinear 2-D filter is based on truncated discrete Volterra filters with matrix notation [16]. Volterra series are expanded linear, quadratic, cubic and so on. The general Volterra system model is shown by Fig. 1 [3].

Equation 1 shows the output of the Quadratic filter which is Volterra filter up to second degree;

$$y(m,n) = y_0 + y_1(m,n) + y_2(m,n) \tag{1}$$

For the above equation, y_0 represents a constant, $y_1(m,n)$ represents the linear part, $y_2(m,n)$ represents the quadratic part of the image filter [17], [18]. The outputs $y_1(m,n)$ and

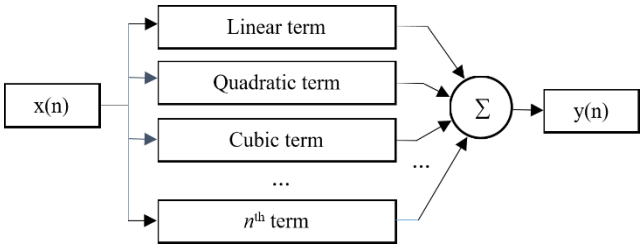

FIGURE 1. System model of volterra series.

$y_2(m,n)$ are written separately as;

$$\begin{aligned} y_1(m,n) &= \sum_{i=-(N-1)/2}^{(N-1)/2} \sum_{j=-N(N-1)/2}^{(N-1)/2} w^1_{i,j} x_{m+i,n+j} \\ y_2(m,n) &= \sum_{i=-(N-1)/2}^{(N-1)/2} \sum_{j=-N}^{(N-1)/2} \sum_{k=-(N-1)/2}^{(N-1)/2} \\ &\quad \times \sum_{l=-(N-1)/2}^{(N-1)/2} w^2_{i,j,k,l} x_{m+i,n+j} x_{k+i,l+i} \end{aligned} \tag{2}$$

Equation 2 involves a number of sum operators. These can be written in more concise forms;

$$y_1(m,n) = \sum_{i=0}^{N\times N} W^1_i X^1_{m,n}(i) \tag{3}$$

$$\begin{aligned} W_1 &= \left[w_1(-N,-N),\, w_1(-N,-N+1),\, \cdots,\, w_1(N,N)\right] \\ X^1_{m,n} &= \left[\, x(m-N,n-N),\, \cdots,\, x(m+N,n+N)\,\right]^T \end{aligned} \tag{4}$$

Therefore, linear part can be expressed as;

$$y_1(m,n) = W_1 {X^1_{m,n}}^T \tag{5}$$

Similarly, quadratic part can be expressed in two dimensional form;

$$y_2(m,n) = \sum_{i=0}^{N\times N} \sum_{j=0}^{N\times N} w_2(i,j) x_1(m+i),n+j) x_1(m+i,n+j) \tag{6}$$

$$y_2(m,n) = \sum_{i=0}^{N\times N} \sum_{j=0}^{N\times N} W^2_{i,j} X^1_{m,n}(i) X^1_{m,n}(j) \tag{7}$$

In the presented study the size of the quadratic filter is used as 3×3. For this purpose the above expansions can be simplified by setting N=3. In this case the weights and input vector given by Equation 8 can written as below;

$$\begin{aligned} W_1 &= \left[\, w_0\ w_1\ w_2\ w_3\ w_4\ w_5\ w_6\ w_7\ w_8\,\right] \\ X^1_{m,n} &= \left[\, x_0\ x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8\,\right]^T \end{aligned} \tag{8}$$

In Equation 8, the weights $w_1(-N,-N)$to $w_1(N,N)$ are replaced with w_0 to w_8 to simplify and in the same way the input vector is also simplified. Inspection of Equation 6 shows that nearly half of the multiplications of input signal is symmetric. This is shown by Equation 9 for the 3×3 case where

the quadratic input terms forms a matrix of 9×9 size. If the symmetric terms are reduced,

$$X^1_{m,n}{X^1_{m,n}}^T = \begin{bmatrix} x_0x_0 & x_0x_1 & \cdots & x_0x_8 \\ x_1x_0 & x_1x_1 & \cdots & x_1x_8 \\ \vdots & \vdots & \cdots & \vdots \\ x_8x_0 & x_8x_1 & \vdots & x_8x_8 \end{bmatrix} \quad (9)$$

If the Equation 6 is rewritten to eliminate symmetric terms, starting value of the index of second sum operator j, is set to $i$as below;

$$y_2(m,n) = \sum_{i=0}^{N\times N} \sum_{j=i}^{N\times N} W^2_{i,j} X^1_{m,n}(i) X^1_{m,n}(j) \quad (10)$$

According to Equation 10, lower triangular of the matrix given by Equation 9 is set to zero;

$$X^1_{m,n}X^1_{m,n} = \begin{bmatrix} x_0x_0 & x_0x_1 & \cdots & x_0x_8 \\ 0 & x_1x_1 & \cdots & x_1x_8 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \vdots & x_8x_8 \end{bmatrix} \quad (11)$$

The input product given by Equation 11 doesn't contain symmetric multiplications. By excluding the zeros, the rows of the matrix can be written in vector form;

$$X^2_{m,n} = \begin{bmatrix} x_0x_0 & x_0x_1 & \cdots & x_8x_8 \end{bmatrix}^T \quad (12)$$

Therefore quadratic part can also be expressed as,

$$y_2(m,n) = W_2 {X^2_{m,n}}^T \quad (13)$$

The weights vector in Equation 13 can be written in vector form as below,

$$W_2 = \begin{bmatrix} w^2_0 & w^2_1 & w^2_2 & \cdots & w^2_{45} \end{bmatrix} \quad (14)$$

The number of elements in W_2 and X^2 depends on the size of the filter kernel. For $N\times N$ size of kernel the number of terms in quadratic vector excluding the symmetric parts can be calculated by Equation 15;

$$N_{quadratic} = N \times (N+1)/2 \quad (15)$$

Summation of linear and quadratic part given by Equation 5 and Equation 13 gives the total expression for a filtered pixel as shown by Equation 14. Note that the constant term is excluded in this expression. The same expansions can be extended in the same way to express the quadratic image filters in greater dimensions.

$$y(m,n) = W_1 X^1_{m,n} + W_2 {X^2_{m,n}}^T \quad (16)$$

B. POPULATION BASED SEARCH ALGORITHMS

Optimization aims finding best solution to a given problem considering some parameters such as the problem type, problem constraints and solution approaches. Population based algorithms (PBS) which sometimes called heuristic, evolutionary or bio inspired algorithms provides a good alternative to classical mathematical approaches. Due to the advancements in evolutionary algorithms they have been used in many practical problems by researchers or engineers in various applications areas [19]–[21]. In general, evolutionary algorithms involves the basic steps described by Algorithm 1 [22]. In the first step, candidate solutions are produced randomly. The size of the population is determined according to the problem type and usually affects the convergence rate. Individuals form a solution which in the present case is a vector of image filter weights. Each individual is evaluated using fitness function to obtain fitness values. Then algorithm enters into a loop to repeat some search operations to converge the solution. Reproducing new individuals usually involves an operation on existing individuals to form a new population. The following operation is to update the fitness values for the new population. Some of the better fitness values can be transferred to next generations by replacing with worst solutions. The repeated steps continue till a termination condition is reached and then the individual that gives the best fitness is obtained as a solution to the problem. In the present work the number of iterations is used as the termination criterion.

In the present study, GA and PSO algorithms which are widely applied search algorithms in optimization problems were used in the experimental evaluations [23], [24]. These algorithms differ in the way that they reproduce their population in sequential iterations. GA calculates new generations based on the fitness values and new generations are obtained using genetic operators selection, crossover and mutation operators [25], [26]. These operators affects the success of the GA and may change according to problem types. For example, there are many kinds of crossover types such as single point crossover, two point crossover, uniform crossover, arithmetic crossover, etc. [27]. There are also many kinds of selection types used in GA such as, tournament selection, rank selection, state selection, etc. [28]. In PSO each solution is called as particles and there are velocities and positions for each particles in the swarm [29]. Before searching start velocity and position of a particle are initialized randomly. The new swarm in each iteration is reproduced by updating the positions and velocities of particles. Figure 2a and 2b shows the basic flowcharts for GA and PSO respectively.

III. PROPOSED IMPLEMENTATION AND DEFINATION OF VARIABLE FITNESS FUNCTION

Heuristic algorithms may demand intense computational power for the solution of a problem depending on the problem type. As described in the previous section, GA and PSO and similar search algorithms creates a random population

Algorithm 1 General Procedure for Population Based Search Algorithms

Input: Fitness function for given problem, Algorithm parameters
Output: Best individual for minimizing (or maximizing) the fitness function)

```
Initialization using random candidate solutions
Evaluate the fitness of each candidate
while Termination condition is not satisfied do
    Reproduce individuals to form a new population
    Evaluate the fitness of each solution
    Select solutions with better fitness values
    Update non-dominate solutions
return Best individual
```

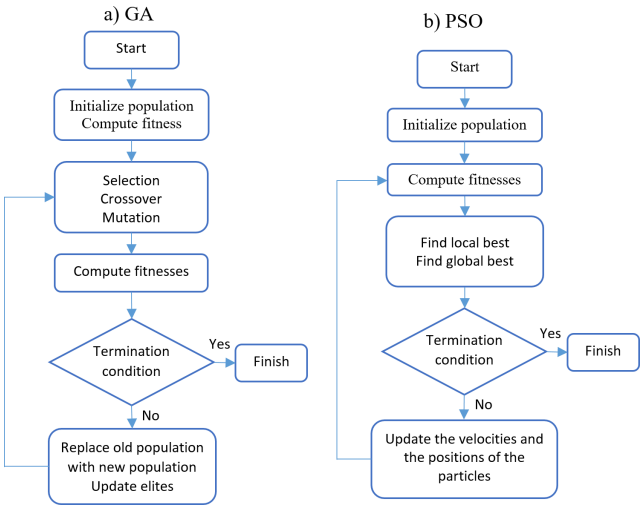

FIGURE 2. Outlines for GA and PSO algorithms.

initially and evolve it according to fitness values as iterations progress. Fitness values are computed using a fitness function where the objective function of the problem to be solved is formulated. In the present approach, a fitness value for an individual of the population is determined by computing the MAE using processed image and reference image. Determining the weights of the quadratic filter is formulated as a single objective function for optimization. Processed image is obtained by filtering the image using an individual as filter weights. This is repeated for each individual in the population for each iteration as the population evolves. Computing a fitness value for the whole population usually demands intense computational power. Therefore an algorithmic acceleration in the fitness function may reduce the computation time significantly. For this purpose, in the presented study, a variable approach and a variation with random selection are introduced to reduce training duration of the quadratic filter.

The VFM is integrated with the population based search algorithm and varies the number of pixels to be used in the fitness function. In these approach, the size of the image is determined according to the current iteration of PBS and then fitness function is computed using a predetermined number of pixels. The basic idea of the implemented method is the number of pixels varies according to the iteration number as shown by the block diagram of the method given by Fig. 3. PBS applies a candidate solution to quadratic filter as weights and then the noisy image is filtered. PBS also provides the width and height values to limit the number of pixels for computing fitness function. After MAE of filtered and reference images is computed, it is provided to PBS as fitness value. For the next iteration the number of pixels used in the computation of fitness function is increased by determining new width and height values as illustrated by Fig.4a. These operations are maintained until the last iteration and finally the entire image is used for the computation of fitness values. Because fitness computations are done on a subpart of the image rather than on the whole image, the training periods becomes shorter and GA can obtain the result faster.

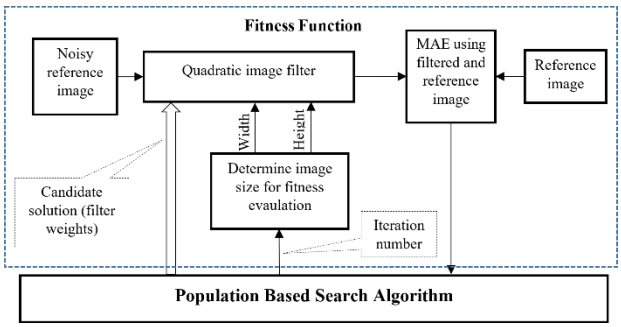

FIGURE 3. Block diagram of training process with population based search algorithms.

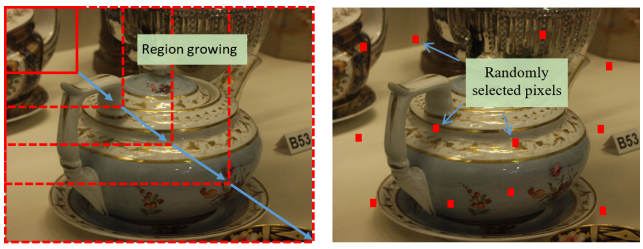

FIGURE 4. Methods for computing fitness function (a) Region growing (b) Randomly selected pixels.

As the filtering process is performed on smaller area rather than on the whole image, using a certain region for training may reduce overall performance. Instead of using a fixed region, random sampling of image as shown by Fig. 4b may improves the result of PBS by decreasing the probability of

staying in the local minimum. Also in this approach, the number of pixels is determined by the product of *width* and *height* of the image.

Algorithm 2 Variable Fitness Method

Input: *fitnessFunc*, *popSize*, *maxIter*, *initialRatio*, *WIDTH*, *HEIGTH*
Output: Best individual

```
Generate a specified number of candidate solutions to form a population
Pop = InitialPopulation(popSize)
Compute new width and height for initial fitnesses
width = WIDTH * initialRatio
heigth = HEIGHT * initialRatio
Compute fitness for all population members
fitnessVal = fitnessFunc(Pop, width, height)
Initial ratio for variable fitness
for k=1 to maxIter do
    Select parents using a selection method
    Selection(fitnessVal, Pop)
    Crossover selected parents
    Crossover(fitnessVal, Pop)
    Mutate randomly selected members
    Mutation(fitnessVal, Pop)
    Compute new width and height for the kth iteration
    width = WIDTH * initialRatio + (1 − initialRatio) * WIDTH * k/maxIter
    height = HEIGHT * initialRatio + (1 − initialRatio) * HEIGHT * k/maxIter
    Compute fitnesses for width×heigth part of image
    fitnessFunc(Pop, width, height)
    Keep elite candidates in the next generation
    Elitism((fitnessVal, Pop))
return best invidual
```

Algorithm 1 shows the basic steps of the training operation using PBS which, in the present case, are GA and PSO. Before entering for loop, the number of pixels that will be used for fitness computation for the first iteration is determined by according to specified ratio. The variables width and height are determined for every iteration in for loop and it grows gradually as the iterations progress. When iteration number is equal to maximum iteration limit, the size of the pixels covers the entire image. Algorithm 2 shows the fitness function that computes fitness values for all individuals for a given population. The algorithm selects an individual and then filters the noisy image using the individual as filter weights. Here the number of pixels to be filtered are restricted by the multiplication of *width* and *height*. As described above, *width* and *height* are determined according to iteration number in PBS. In this algorithm the pixels to be filtered are selected randomly as in Fig. 4b. Implementation that is represented by Fig. 4a can be realized if the variables k and l are set as x and y respectively instead of selecting randomly.

The number of filtering operations for both DFM and VFM can be compared according to described approaches. Fig. 5 illustrates the number of operations versus iteration number. Total number of pixel filtering operations using DFM (N_{DFM}) where full size of image is used for each iteration can be written as;

$$N_{DFM} = Width \times Height \times PopSize \times MaxIter \tag{17}$$

Total number of filtering operations using VFM (N_{VFM}) can be described in terms of N_{DFM} to simplify obtaining acceleration expression.

$$N_{VFM} = \frac{N_{DFM}}{R} + \frac{\left(N_{DFM} - \frac{N_{DFM}}{R}\right)}{2} \tag{18}$$

Above equation can be rearranged as below;

$$N_{VFM} = \frac{(R+1)\,N_{DFM}}{2R} \tag{19}$$

Therefore the algorithmic acceleration can be calculated as;

$$\text{Ratio of Areas} = \frac{N_{DFM}}{N_{VFM}} = \frac{N_{DFM}}{\frac{(R+1)N_{DFM}}{2R}} = \frac{2R}{R+1} \tag{20}$$

For example, if initial size of the image is 12.5% of the whole image, this means that R=8 and the ratio is calculated approximately as 1.78 times. Because the scaling is in both width and height, the expected acceleration can be the square of the ratio of areas, which is about 3.17. As shown by the results in the following section, computational times for VFM or VFRM close to the theoretical expectation.

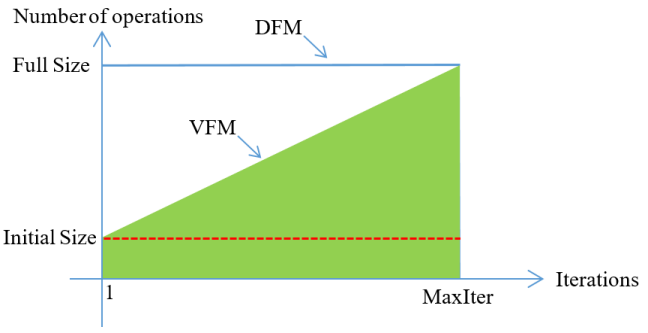

FIGURE 5. Number of operations for DFM and VFM.

IV. RESULT AND DISCUSSIONS

Experimental results were obtained on a server with Quad-Core AMD Opteron™2378 2.40GHz dual processor and 18GB Ram memory and Windows Server 2012 Essential ™64-bit operating system installed. The algorithms were written in VC++ Visual Studio 2012 ™platform. In the

FIGURE 6. Reference images used in experimental measurements.

experiments 12 different images with a size of 512×384 pixels are used [30]. The results obtained from the average of 10 times running of the algorithm for each image. MAE (Mean Absolute Error), MSE (Mean Square Error) and mean training times for various population sizes and iteration sizes were measured in experimental studies. In the experiments, mutation rate was 0.1, the crossover rate was 0.5. The upper and lower limits for the linear part were selected as 1.0 and −1.0 respectively and the upper and lower limits for the quadratic were selected as −0.5 and 0.5 respectively.

The reference images used for obtaining the experimental results are shown in Fig. 6 [30]. Noisy images for filtering were obtained by adding a Gaussian noise to the reference images.

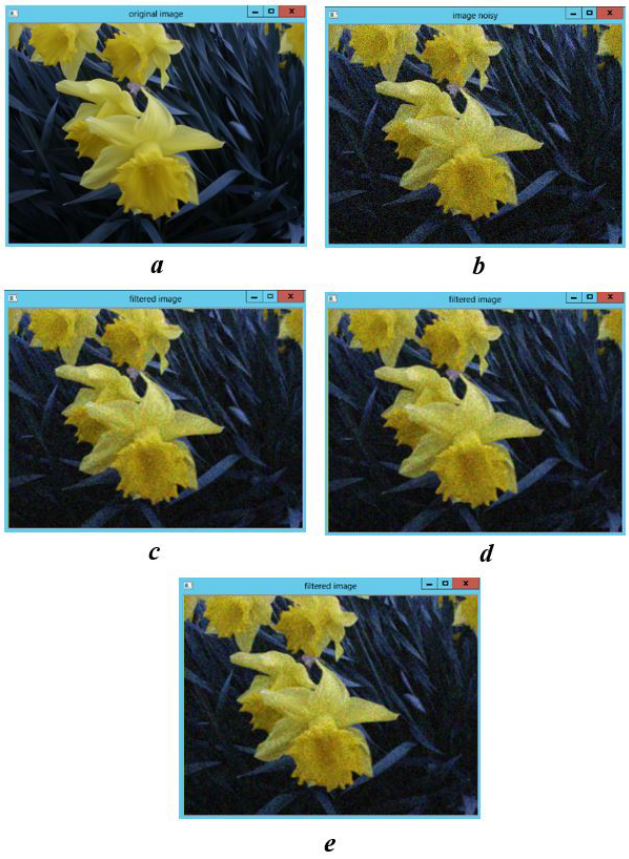

FIGURE 7. Example filtered image by using the weights from DFM, VFM and VRFM (Population size: 200, number of iteration: 1000) a) Reference image, b) Noisy reference image (MAE: 23.02), c) DFM filtered image (MAE: 8.61), d) VFM filtered image (MAE: 8.69), e) VRFM filtered image (MAE: 8.72).

Example filtering results using the filters obtained using the VFM, VRFM and DFM are show in Fig. 7. The reference image is shown in Fig. 7a. A noisy image obtained by adding a Gaussian noise to the reference image is shown in Fig. 7b. The result of filtering performed reference images obtained by using the proposed DFM, VFM and VRFM are shown in Fig. 7c, Fig. 7d and Fig. 7e respectively. The quality measurements are expected to be close to each other while computing durations of VFM and VFRM are expected to be shorter than DFM.

Figure 8 shows the MAE convergence graphs versus the number of iterations. The test runs were obtained using the number of iterations equal to 200 and the population size equal to 200 for both GA and PSO. Convergence of the MAE values for tested methods seems close to each other because of the scaling of the graph.

Table 1 shows the numerical values for the various test runs to check the quality variations of consecutive runs more

Algorithm 3 Fitness Function

Input: *Pop*, *width*, *height*, *refImage*
Output: MAE vector as fitness values for the population

```
Iterate over all individuals in the population
for k=1 to length(Pop) do
    Fiter image using kth individual as weights
    for k=1 to height do
        for l=1 to width do
            Generate random indices
            x = randomVal()%HEIGHT;
            y = randomVal()%WIDTH;
            Computed filtered pixel
            filtImage(x, y) = filterPixel(x, y, Pop(k));
    Compute fitness of kth individual
    fitnessArray(k) = MAE(filtImage, refImage);
return fitnessArray
```

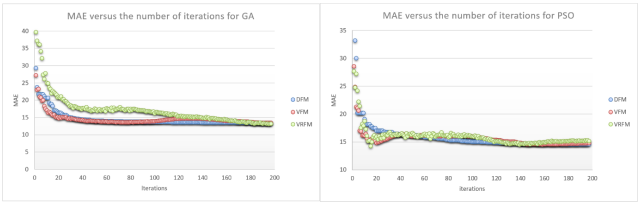

FIGURE 8. MAE versus iteration number for GA using DFM, VFM and VRFM.

closely for DFM, VFM and VRFM. The average quality obtained with GA shows very consistent results for different runs. Also the average MAE values using DFM, VFM and VRFM are very close to each other. The average deviation for GA is at most 1.31% which is a proper values for the present case. But this behavior may change according to the test image the number of iterations, the population size and parameters that are specific to algorithm. The behavior of PSO for consecutive runs is consistent when the test runs with DFM compared to the test runs with VFM and VFRM. In general PSO using VFM and VFRM don't produce close results to DFM when the gap between the minimum and maximum values are investigated. Table 2, Table 3 and Table 4 shows quality measurements that are computed using DFM, VFM and VRFM respectively. Also the comparisons of DFM, VFM and VRFM for PSO and GA is given in each table. The results were repeated by setting the number of iterations to 100, 200 and 500. The number of populations is fixed to 500 for each of the measurements. When the MAE for the population size of 200 are compared, DFM slightly outperforms VFM results. The same behavior is also observed for the population sizes of 200 and 500 respectively. This is due to the computation approach of VFM which selects smaller regions for fitness function computation.

TABLE 1. MAE variations obtained from the test runs from GA and PSO.

	GA			PSO		
	DFM	VFM	VRFM	DFM	VFM	VRFM
Run 1	8.54	8.57	8.90	8.87	9.38	9.94
Run 2	8.52	8.53	8.73	8.67	9.30	10.31
Run 3	8.55	8.52	8.59	8.62	10.03	9.43
Run 4	8.51	8.53	8.63	9.00	9.15	10.47
Run 5	8.54	8.56	8.57	8.70	9.09	9.10
Run 6	8.52	8.52	8.61	8.76	9.37	9.15
Run 7	8.54	8.54	8.92	8.64	9.25	9.16
Run 8	8.53	8.55	8.80	8.73	9.48	9.17
Run 9	8.53	8.60	8.58	8.81	9.09	9.56
Run 10	8.51	8.53	8.62	8.81	9.65	9.73
Average	8.53	8.55	8.70	8.76	9.38	9.60
Avg. Dev.	0.01	0.02	0.11	0.09	0.20	0.41
Avg. Dev. %	1.13	0.23	1.31	1.01	2.18	4.5
Min	8.51	8.52	8.57	8.62	9.09	9.10
Max	8.54	8.60	8.90	9.00	10.03	10.47

TABLE 2. Experimental measurements for DFM (image size: 512×384, population: 500).

Test Image	Number of Iteration=100				Number of Iteration =200				Number of Iteration =500			
	MAE GA	MAE PSO	MSE GA	MSE PSO	MAE GA	MAE PSO	MSE GA	MSE PSO	MAE GA	MAE PSO	MSE GA	MSE PSO
1	**13.05**	13.24	**311.73**	321.73	**12.97**	13.04	**306.91**	310.35	**12.88**	12.95	**302.04**	305.23
2	**12.44**	12.68	**280.14**	294.54	**12.37**	12.50	**276.88**	283.90	**12.34**	12.43	**274.69**	280.96
3	**14.77**	14.88	**378.42**	384.57	**14.64**	14.70	**370.36**	374.54	**14.59**	14.62	**368.59**	370.35
4	**8.87**	9.11	**125.95**	132.83	**8.85**	8.96	**125.29**	128.41	**8.78**	8.88	**123.25**	126.24
5	**8.65**	8.83	**122.48**	128.17	**8.65**	8.76	**122.59**	125.61	**8.62**	8.66	**121.62**	122.76
6	**13.73**	14.14	**313.72**	334.94	**13.70**	13.86	**312.30**	321.03	**13.62**	13.75	**308.95**	314.84
7	**12.30**	12.54	**260.25**	273.63	**12.27**	12.40	**258.54**	265.77	**12.23**	12.30	**256.79**	261.13
8	**10.10**	10.62	**167.27**	184.05	**10.12**	10.35	**168.00**	175.49	**10.05**	10.22	**165.58**	171.16
9	**11.56**	11.72	**242.53**	250.75	**11.49**	11.60	**239.52**	244.66	**11.46**	11.51	**237.97**	241.01
10	**14.25**	14.58	**348.38**	367.56	**14.17**	14.31	**344.35**	352.60	**14.13**	14.22	**342.16**	347.08
11	**10.27**	10.53	**170.61**	179.65	**10.25**	10.41	**169.72**	175.67	**10.21**	10.29	**168.76**	171.47
12	**8.53**	8.96	**118.69**	130.00	**8.48**	8.67	**116.98**	122.15	**8.49**	8.57	**117.20**	119.58

TABLE 3. Experimental measurements for VFM (image size: 512×384, population: 500).

Test Image	Number of Iteration=100				Number of Iteration =200				Number of Iteration =500			
	MAE GA	MAE PSO	MSE GA	MSE PSO	MAE GA	MAE PSO	MSE GA	MSE PSO	MAE GA	MAE PSO	MSE GA	MSE PSO
1	**13.12**	13.15	**310.50**	314.84	**13.01**	12.95	**307.12**	304.98	**12.94**	12.88	303.92	**302.33**
2	**12.51**	13.52	**283.04**	332.45	**12.44**	14.73	**280.64**	395.40	**12.35**	14.65	**277.08**	393.65
3	**14.80**	14.72	**374.35**	374.83	**14.70**	15.00	**369.64**	391.07	**14.60**	14.76	**367.36**	375.79
4	**8.87**	9.10	**126.05**	132.31	**8.84**	9.16	**124.92**	133.96	**8.79**	9.00	**123.43**	129.69
5	**8.71**	8.76	**123.22**	126.07	**8.67**	8.71	**122.29**	124.17	**8.63**	8.72	**121.68**	124.33
6	**13.72**	14.13	**313.30**	335.15	**13.67**	14.59	**311.75**	363.54	**13.66**	14.22	**310.40**	346.98
7	**12.28**	12.67	**260.24**	277.88	**12.28**	12.47	**259.35**	268.99	**12.25**	12.38	**257.54**	264.16
8	**10.16**	11.12	**169.25**	190.60	**10.15**	10.78	**168.73**	189.24	**10.09**	11.95	**167.05**	232.26
9	**11.57**	11.83	**243.34**	256.00	**11.53**	11.75	**240.83**	251.12	**11.53**	11.89	**240.53**	254.10
10	**14.29**	14.70	**351.36**	370.63	**14.22**	14.44	**346.87**	359.44	**14.19**	15.16	**345.55**	395.33
11	**10.27**	10.56	**170.57**	180.54	**10.24**	10.68	**169.81**	184.86	**10.21**	10.67	**168.79**	184.51
12	**8.59**	9.74	**120.77**	157.27	**8.61**	9.91	**121.02**	167.33	**8.52**	9.90	**118.35**	167.41

TABLE 4. Experimental measurements for VRFM (image size: 512×384, population: 500).

Test Image	Number of Iteration=100				Number of Iteration =200				Number of Iteration =500			
	MAE GA	MAE PSO	MSE GA	MSE PSO	MAE GA	MAE PSO	MSE GA	MSE PSO	MAE GA	MAE PSO	MSE GA	MSE PSO
1	**13.07**	15.30	**311.55**	416.11	**13.02**	14.98	**308.95**	394.33	**12.96**	15.37	**306.43**	401.46
2	**12.51**	13.82	**285.34**	342.90	**12.46**	14.62	**285.08**	389.50	**12.45**	14.70	**284.97**	396.33
3	**14.75**	15.34	**374.62**	398.73	**14.71**	15.71	**374.35**	418.28	**14.63**	16.48	**371.96**	467.08
4	**8.89**	9.19	**126.48**	134.84	**8.86**	9.20	**125.52**	135.09	**8.82**	9.05	**124.69**	130.70
5	**8.71**	9.47	**124.50**	144.97	**8.68**	9.00	**124.01**	131.23	**8.66**	8.82	**124.16**	125.87
6	**13.70**	14.46	**311.75**	351.69	**13.67**	14.50	**308.87**	360.61	**13.63**	14.54	**306.59**	364.74
7	**12.34**	14.00	**264.30**	335.19	**12.31**	13.11	**262.91**	296.72	**12.26**	13.64	**260.61**	325.30
8	**10.15**	10.89	**168.12**	193.35	**10.10**	10.94	**167.27**	195.30	**10.07**	11.63	**166.64**	219.76
9	**11.58**	12.12	**244.55**	268.90	**11.51**	11.90	**240.53**	257.47	**11.52**	11.98	**240.79**	258.20
10	**14.34**	15.49	**355.10**	409.60	**14.27**	15.17	**349.03**	393.59	**14.17**	15.39	**343.05**	403.34
11	**10.29**	10.91	**171.48**	192.66	**10.24**	10.91	**169.86**	192.04	**10.23**	11.36	**169.55**	210.15
12	**8.70**	9.57	**122.81**	150.87	**8.66**	10.14	**121.86**	175.81	**8.63**	10.05	**120.89**	171.99

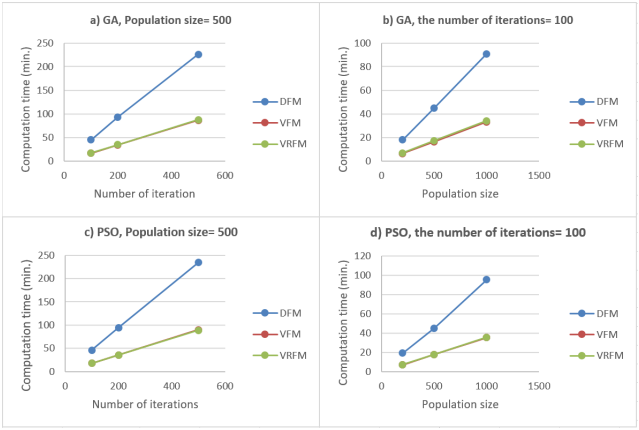

FIGURE 9. Computation times versus the number of iterations for GA and PSO.

According to numerical results, increasing the number of iterations improves the results for all tested approaches. When the number of iterations are increased to 500 the MAE results are get closer to each other. It should be noted that the results are very sensitive to selected algorithm parameters and the success of the applied algorithms may change according to the number of populations and the number of iterations significantly. The computational time of the implemented methods are shown in Fig 9. It is seen that the proposed methods provides a considerable time gains compared to DFM. For example, increasing the number of iteration from 100 to 200 doubles the computation times for each method. If the number of population is increased from 200 to 500 the computation times for each method increases roughly about 2.5 times. As expected increase in the number of population, computation time shows linear behavior. On the other hand, when the computation times compared, VFM outperforms DFM by reducing the computation times significantly reaching about 2.5 to 3.0 times acceleration over DFM.

V. CONCLUSIONS

In the presented study, determining the weights of quadratic filters were examined in detail using GA and PSO which are population based search algorithms. For this purpose, the problem was formulated as a single objective optimization problem. As a straightforward approach, DFM was used directly to obtain optimized weights using both PSO and GA. The experimental results showed that both algorithms produced MAE and MSE results close to each other. Due to the cost of image filtering operations, repeated evaluations of fitness function takes quite a long time. Hence, the

execution durations for both GA and PSO implementations were obtained close to each other. In order to reduce the computation durations, VFM and its randomized version VRFM were implemented using both search algorithms. Both approach initially start searching using a small part of the image, and as the iterations progress this region grows and converges to the whole image. Random selection of the group of pixels instead of a certain region improves the quality of filtering. According to the experimental results obtained, both DFM and VRFM produces similar results for GA and for some results for PSO. The results also show that VFM or VRFM provides about 2.5 to 3.0 fold algorithmic acceleration when compared to DFM. The implemented methods can be combined with most of the other population based search methods.

REFERENCES

[1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed. 2007.
[2] D. Akgün and P. Erdoğmuş, "GPU accelerated training of image convolution filter weights using genetic algorithms," *Appl. Soft Comput.*, vol. 30, pp. 585–594, May 2015.
[3] M. B. Meenavathi and K. Rajesh, "Volterra filtering techniques for removal of Gaussian and mixed Gaussian-impulse noise," *Int. J. Appl. Math. Comput. Sci.*, vol. 4, no. 9, pp. 51–56, 2007.
[4] G. Jothilakshmi and E. Gopinathan, "Mammogram enhancement using quadratic adaptive Volterra filter—A comparative analysis in spatial and frequency domain," *ARPN J. Eng. Appl. Sci.*, vol. 10, no. 13, pp. 5512–5517, 2006.
[5] Y. Zhou, K. Panetta, and S. Agaian, "Mammogram enhancement using alpha weighted quadratic filter," in *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, Sep. 2009, pp. 3681–3684.
[6] S. K. Mitra, "Image processing using quadratic Volterra filters," in *Proc. 5th Int. Conf. Comput. Devices Commun. (CODEC)*, Dec. 2012, pp. 1–2.
[7] M. Kanamadi, V. Waghamode, and S. Bandekar, "Alpha weighted quadratic filter based enhancement for mammogram," in *Proc. Int. Conf. 'Emerg. Res. Comput., Inf., Commun. Appl.' (ERCICA)*, 2013, pp. 68–74.
[8] G. F. Ramponi, G. L. Sicuranza, and W. Ukovich, "A computational method for the design of 2-D nonlinear Volterra filters," *IEEE Trans. Circuits Syst.*, vol. 35, no. 9, pp. 1095–1102, Sep. 1988.
[9] A. Pandey, A. Yadav, and V. Bhateja, "Design of new Volterra filter for mammogram enhancement," in *Advances in Intelligent Systems and Computing*, vol. 199. 2013, pp. 143–151.
[10] L. Thomas, G. Krishnan, R. A. Mol, and A. Roy, "Removal of impulsive noise from MRI images using quadratic filter," *Int. J. Eng. Res. Technol.*, vol. 3, no. 4, pp. 2220–2223, Apr. 2014.
[11] V. S. Hari, R. V. P. Jagathy, and R. Gopikakumari, "Enhancement of calcifications in mammograms using Volterra series based quadratic filter," in *Proc. Int. Conf. Data Sci. Eng. (ICDSE)*, Jul. 2012, pp. 85–89.
[12] G. Feng, H. Li, J. Dong, and J. Zhang, "Direct discriminant analysis using Volterra kernels for face recognition," in *Pattern Recognition* (Communications in Computer and Information Science), vol. 662. 2016, pp. 404–412.
[13] V. Bhateja, M. Misra, and S. Urooj, "Non-linear polynomial filters for edge enhancement of mammogram lesions," *Comput. Methods Programs Biomed.*, vol. 129, pp. 125–134, Jun. 2016.
[14] V. S. Hari, V. P. J. Raj, and R. Gopikakumari, "Quadratic filter for the enhancement of edges in retinal images for the efficient detection and localization of diabetic retinopathy," *Pattern Anal. Appl.*, vol. 20, no. 1, pp. 145–165, Feb. 2017.
[15] G. Ramponi, "Edge extraction by a class of second-order nonlinear filters," *Electron. Lett.*, vol. 22, no. 9, pp. 482–484, Apr. 1986.
[16] J. Tang, E. Peli, and S. Acton, "Image enhancement using a contrast measure in the compressed domain," *IEEE Signal Process. Lett.*, vol. 10, no. 10, pp. 289–292, Oct. 2003.
[17] S. Y. Fakhouri, "Identification of the Volterra kernels of nonlinear systems," *IEE Proc. D-Control Theory Appl.*, vol. 127, no. 6, pp. 296–304, Nov. 1980.
[18] R. D. Nowak and B. D. Van Veen, "Random and pseudorandom inputs for Volterra filter identification," *IEEE Trans. Signal Process.*, vol. 42, no. 8, pp. 2124–2135, Aug. 1994.
[19] A. Ouarda and M. Bouamar, "A comparison of evolutionary algorithms: PSO, DE and GA for fuzzy c-partition," *Int. J. Comput. Appl.*, vol. 91, no. 10, pp. 32–38, 2014.
[20] E. Assareh, M. A. Behrang, M. R. Assari, and A. Ghanbarzadeh, "Application of PSO (particle swarm optimization) and GA (genetic algorithm) techniques on demand estimation of oil in Iran," *Energy*, vol. 35, no. 12, pp. 5223–5229, Dec. 2010.
[21] S. Das, A. Abraham, and A. Konar, "Particle swarm optimization and differential evolution algorithms: Technical analysis, applications and hybridization perspectives," in *Advances of Computational Intelligence in Industrial Systems*. Springer, 2008, pp. 1–38.
[22] S. Cheng, B. Liu, T. O. Ting, Q. Qin, Y. Shi, and K. Huang, "Survey on data science with population-based algorithms," *Big Data Anal.*, vol. 1, no. 1, p. 3, 2016.
[23] V. Kachitvichyanukul, "Comparison of three evolutionary algorithms: GA, PSO, and DE," *Ind. Eng. Manage. Syst.*, vol. 11, no. 3, pp. 215–223, 2012.
[24] X. Yu and M. Gen, *Introduction to Evolutionary Algorithms*. Springer, 2010.
[25] B. Bolat, K. O. Erol, and C. E. İrmak, "Genetic algorithms in engineering applications an the function of operators," *J. Eng. Nat. Sci.*, vol. 4, pp. 264–271, 2004.
[26] H. M. Abbas and M. M. Bayoumi, "Volterra-system identification using adaptive real-coded genetic algorithm," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 36, no. 4, pp. 671–684, Jul. 2006.
[27] A. J. Umbarkar and P. D. Sheth, "Crossover operators in genetic algorithms: A review," *ICTACT J. Soft Comput.*, vol. 6, no. 1, pp. 1083–1092, 2015.
[28] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," *Found. Genet. Algorithms*, vol. 1, no. 1, pp. 69–93, 1991.
[29] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intell.*, vol. 1, no. 1, pp. 33–57, Jun. 2007.
[30] G. Schaefer and M. Stich, "UCID: An uncompressed color image database," *Proc. SPIE*, vol. 5307, pp. 472–480, Dec. 2003.

SÜLEYMAN UZUN graduated from the Faculty of Informatics, Gazi University, in 2011. He is currently pursuing the Ph.D. degree with Sakarya University. Since 2013, he has been a Teaching Assistant with the Open and Distance Learning Application and Research Center, Bilecik Şeyh Edebali University. His research interests include parallel programming, optimization algorithms, quadratic image filter, image processing, GPU programming, and FPGA applications.

DEVRIM AKGÜN received the Ph.D. degree from Sakarya University in 2008. He has been with the Department of Computer Engineering, Sakarya University, since 2013. His current research interests include parallel computing, signal and image processing, and machine learning.