Correspondence

Edge Detection in Medical Images Using a Genetic Algorithm

Markús Guđmundsson, Essam A. El-Kwae, and Mansur R. Kabuka*

***Abstract*— An algorithm is developed that detects well-localized, unfragmented, thin edges in medical images based on optimization of edge configurations using a genetic algorithm (GA). Several enhancements were added to improve the perfomance of the algorithm over a traditional GA. The edge map is split into connected subregions to reduce the solution space and simplify the problem. The edge-map is then optimized in parallel using incorporated genetic operators that perform transforms on edge structures. Adaptation is used to control operator probabilities based on their participation.**

The GA was compared to the simulated annealing (SA) approach using ideal and actual medical images from different modalities including magnetic resonance imaging (MRI), computed tomography (CT), and ultrasound. Quantitative comparisons were provided based on the Pratt figure of merit and on the cost-function minimization. The detected edges were thin, continuous, and well localized. Most of the basic edge features were detected. Results for different medical image modalities are promising and encourage further investigation to improve the accuracy and experiment with different cost functions and genetic operators.

***Index Terms*— Edge detection, genetic algorithms, medical images, optimization.**

I. Introduction

Machine vision is the process of acquiring an image from a scenery, and processing it in order to interpret the contents of the image. Before an object can be extracted from the scenery, a way of assessing its shape and size is needed. This can be accomplished using edge detection or segmentation, both of which are low-level image processing tasks, or their combination. The success of the higher-level recognition is, therefore, highly dependent on these lower-level processes.

An edge can be defined as the boundary between two regions separated by two relatively distinct gray-level properties [1]. Edge detection is the process of locating these edges. While simple kernel-based edge detectors, such as Canny's edge detector [2], have shown good performance, their lack of edge notion, i.e., built-in knowledge of what constitutes good edges, limits their ability to exploit local edge continuity information to reduce fragmented edges in noisy environments [3]. Furthermore, because of their inherent low-pass filtering, they have a tendency to dislocate edges or produce spurious edge points at corners, or where two edges come close together, within the width of the filter. Attempts to correct for these defects using scale space have been made [4]–[7]. Scale space refers to using kernels of different sizes and combining their edge maps. A knowledge-base system in scale space is used [8] to correct for dislocations, and to reduce noise and false edges. Another approach to eliminate noise is by fitting models of edges [9] to the surface in the image in the least-square sense. Refinements addressing problems with instability and dislocation of edges at angles, using greedy algorithms [10], dynamic programming [11], and the finite element method [12] have been proposed.

In spite of the mathematical sophistication of these techniques, the problem of finding true edges that correspond to physical boundaries of an object in an image is still a very difficult one [13]. One of the reasons is that most techniques are based on a very restricted definition of what constitutes an edge. Moreover, most of the aforementioned approaches consider the edge-detection problem as one that is based upon the response of the edge detector at a single pixel location. So, in spite of the fact that the performance of the edge detector is optimized at each individual pixel location, the edge image as a whole could still be unsatisfactory.

Tan *et al.* [3], [14] formulated the problem of edge detection as one of cost minimization. Their work attempts to overcome the shortcomings of existing techniques by using an edge definition that is general enough to include most edge types. They also explicitly consider the local edge structure in the neighborhood of the hypothesized edge pixels. They then suggested using a simulated annealing (SA) approach [15] to minimize the cost.

The largest body of research in the area of edge detection in medical images uses specially tailored algorithms [16]–[19]. While these algorithms perform efficiently for detecting edges in a certain modality or a certain anatomical structure, they are brittle in the sense that they cannot be generalized to other modalities or other anatomic structures. A robust algorithm is thus required to overcome this brittleness. GA's [20]–[22] are robust in that they are not affected by spurious local optima in the solution space. This robustness is backed up by a strong mathematical foundation [20]–[22]. Many advanced GA operators have been designed for performing effective search of the solution space and speeding the convergence of solutions. GA's have been previously used in different areas of image processing such as edge detection [13], image interpretation [23], image segmentation [24], and contour matching [25].

In this paper, an edge detector based on a cost minimization approach was implemented using GA's to accurately locate thin and continuous edges in medical images. To fulfill this objective, the minimization of cost functions successfully used by Tan *et al.* [3], [14] was adopted. The cost function takes into account valid edge structures, only allowing thin, sharp enough edges, connected edge points, and a high contrast perpendicular to the contour direction.

The rest of this paper is organized as follows: In Section II, modeling edge detection as a cost minimization problem is explained, together with a definition of each of the cost factors used in the model. In Section III, a quick overview of GA's is given. The 2-D GA is explained in Section IV. Section V presents the results of using the algorithm for the edge detection of different medical image modalities and compares the performance of the algorithm to SA. Conclusions and future directions are given in Section VI.

II. Edge Detection as a Cost Minimization Problem

The cost functions used in the optimization have been used by Tan *et al.* [3], [14]. An edge structure is defined within a 3×3

Manuscript received March 4, 1997; revised January 28, 1998. The Associate Editor responsible for coordinating the review of this paper and recommending its publication was M. Sonka. *Asterisk indicates corresponding author*.

M. Guđmundsson and E. A. El-Kwae are with the Center for Medical Imaging and Medical Informatics (CMIAMI), Department of Radiology, University of Miami, Miami, FL 33136 USA.

*M. R. Kabuka is with the Department of Electrical and Computer Engineering and the Department of Radiology, University of Miami, 1251 Memorial Drive, MCA Room 406, Coral Gables, FL 33146 USA (e-mail: kabuka@clinnet.med.miami.edu).

Publisher Item Identifier S 0278-0062(98)06453-2.

neighborhood $W_{ij}(S)$ around a single center pixel $l = s(i,j)$ in $S \in \mathrm{S}$, where $\boldsymbol{S}$ is the set of all possible edge configurations in an image I. An edge structure is valid, if (1) point $s(i,j)$ is an edge point and its 8-neighborhood $V_{ij}(S)$ forms a valid edge structure within the window $W_{ij}(S)$, (2) it consists of at least 3, connected points, and (3) it has only thin edges. The total cost for an edge configuration $S \in \boldsymbol{S}$ is the sum of the point costs at every pixel in the image

$$F(S) = \sum_{l \in I} F(S,l) = \sum_{l \in I} \sum_{j} w_j c_j(S,l). \tag{1}$$

The c_j represent the five cost factors: the dissimilarity cost C_d, the curvature cost C_c, the edge pixel cost C_e, the fragmentation cost C_f and the cost for thick edges C_t. The w_j are the corresponding cost weights w_d, w_c, w_e, w_f, w_t used to control the shape of the optimized edges. If the equation

$$w_t > 2w_f - w_c + w_d - w_e \tag{2}$$

is satisfied, thin edges are guaranteed in an optimal configuration [3].

To measure the dissimilarity, two regions of interest $R1$ and $R2$ are defined on either side of a candidate edge. The dissimilarity function is

$$f(R1,R2) = m(d(i,j)) = \left(\left| \frac{1}{|R1|} \sum_{(i,j) \in R1} g(i,j) - \frac{1}{|R2|} \sum_{(i,j) \in R2} g(i,j) \right| \right) \tag{3}$$

where $g(i,j)$ is the gray value of the pixel at location $(i,j), m(d(i,j))$ is a strictly monotonic mapping function as defined in [14] used to map the dissimilarity measure, so that $0 \leq f(R1,R2) \leq 1$.

The dissimilarity map is created by first selecting the edge structures that give the maximum dissimilarity, at each pixel location. A dissimilarity cost C_d is then defined, based on the dissimilarity map

$$C_d(S,l) = \begin{cases} 0, & \text{if } s(l) \text{ is an edge pixel} \\ d(l), & \text{if } s(l) \text{ is not an edge pixel.} \end{cases} \tag{4}$$

The curvature cost C_c tends to favor smooth edges. Let $\phi_i(l)$ be the larger of the two angles between the ith pair of edge segments connected by $s(l)$ and let $\Theta_i(l) = \phi_i(l) - 180$. Curvature is then defined as

$$\Theta(l) = \begin{cases} \max\limits_{1 \leq i \leq n} \Theta_i(l), & \text{if } s(l) \text{ is an edge pixel} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where n is the maximum number of distinct pairs of straight edge segments connected by $s(l)$ and the cost for curvature is defined by

$$C_c(S,l) = \begin{cases} 0, & \text{if } \Theta(l) = 0^\circ \\ 0.5, & \text{if } \Theta(l) = 45^\circ \\ 1.0, & \text{if } \Theta(l) = 90^\circ. \end{cases} \tag{6}$$

The edge pixel cost factor C_e assigns a cost to each edge point to limit the number of edge points detected. It is defined as

$$C_e(S,l) = \begin{cases} 1, & \text{if } s(l) \text{ is an edge pixel} \\ 0, & \text{if } s(l) \text{ is not an edge pixel.} \end{cases} \tag{7}$$

The fragmentation cost factor C_f tends to link or remove fragmented edges locally. It is defined as

$$C_f(s,l) = \begin{cases} 1.0, & \text{if } s(l) \text{ is an isolated endpoint edge pixel} \\ 0.5, & \text{if } s(l) \text{ is a nonisolated endpoint edge pixel} \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

The thick edge cost factor C_t favors thin edges and is defined as in [3]

$$C_t(S,l) = \begin{cases} 1, & \text{if } s(l) \text{ is a thick edge pixel} \\ 0, & \text{if } s(l) \text{ is not a thick edge pixel.} \end{cases} \tag{9}$$

III. Genetic Algorithm Overview

A GA is a heuristic search/optimization technique for obtaining the best possible solution in a vast solution space. The classical GA is based on the original work of Holland [21]. To apply a GA, an initial population is generated and the fitness of each member of the population is evaluated. The algorithm then iterates the following: members from the population are selected for reproduction in accordance to their fitness evaluations. The reproduction operators are then applied, which generally include a crossover operator that models the exchange of genetic material between the parent chromosomes and a mutation operator to maintain diversity and introduce new alleles into the generation, or a combination of both, to generate the offspring of the next generation. The fitness of the offspring is then evaluated, and the algorithm starts a new iteration. The algorithm stops when either a sufficiently good solution is found, or after a predetermined number of iterations.

The parameters that control the GA can significantly affect its performance, and there is no guidance in theory as to how to properly select them. The most important parameters are the population size, the crossover rate and the mutation probability. Researchers have commonly used these settings as either fixed or dynamic. In general, very good performance can be obtained with a range of parameter settings [26]. Davis [20], [27] introduced dynamic adaptive settings which are advantageous to use over fixed settings, especially when the number of operators used is high. In this model, operators are given credit each time a better solution is found. A proportion of this credit is passed back to the parents of the created individual to assure operators are given credits for participating in creating the better solution.

IV. The Proposed 2-D Genetic Algorithm

Edge detectors normally represent edges in a binary image, where each pixel takes on either the value zero (off) for a nonedge pixel or one (on) for an edge pixel. Each pixel in the binary map corresponds to an underlying pixel in the original image. This edge representation is simple, allows direct illustration of results, location of edge points maps directly onto the original image and adjacency and orientation are preserved. By using the edge map as a solution space for the GA, no special mappings are required, small neighborhood windows can be overlaid, and edge structures and pixels can be modified on a local, intuitive basis. Furthermore, this representation allows for easy transition into an extended type of chromosome for the GA, the bit array. We define the bit array as an array of bits instead of the conventional bit vector, or string, used in traditional GA's. A bit array is highly compact, leading to a compact memory usage, which is essential when running a simulation with a large population, or a large image.

The solution space for the edge-detection problem is huge. For example, for a 256×256 image, the sample space has $2^{65\,536}$ combinations. To reduce the sample space, and hence simplify the problem, the original image is split into linked sub-images, or regions as shown in Fig. 1. Cost is then calculated for smaller subregions. We also suggest that the links between adjacent pixels are maintained, by relating a boundary pixel on an individual's bit array with the corresponding boundary pixel of the best-fit individual's bit array from the adjacent subregion. Consequently, there is a single independent GA for each region, which seeks to optimize the edge

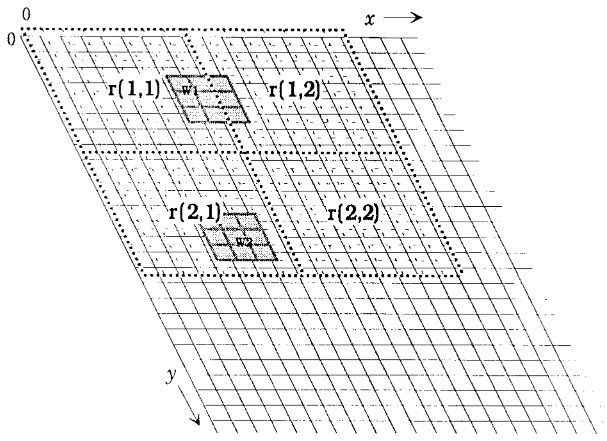

Fig. 1. The bit array representation. Each subregion $r(i, j)$, has its own GA. Also shown are windows W1 and W2, where pixel cost is calculated and can be also regarded as transformation windows. However, the transformation window at site W1 is illegal, since it overlaps subregion boundaries.

TABLE I
APPROXIMATION OF THE SIZE OF SEARCH SPACE ASSUMING INDEPENDENT SUBREGIONS

Size	No of Regions	No of Combinations	Search Space
256x256	1	2^{65536}	$>10^{19728}$
128x128	4	2^{16364}	$>4.10^{4932}$
64x64	16	2^{4096}	$>16.10^{1233}>10^{1234}$
32x32	64	2^{1024}	$>64.10^{308}>10^{310}$
16x16	256	2^{256}	$256.10^{77}>10^{79}$
8x8	1024	2^{64}	$>1024.10^{19}>10^{22}$
4x4	40960	2^{16}	$>4096.10^{4}>10^{8}$

configuration within its subregion, relative to the external boundaries, defined by the best-fit individuals of the adjacent regions. These external boundaries can be constantly changing until stable solutions are found. The effect passes on from one region to another. A few examples of search space reductions are demonstrated in Table I. Table I is an approximation based on the fact that each sub-image can be built independently. It shows how fast search space diminishes with smaller sub-images. The fact that the subregions are dependent is not taken into account. Thus, Table I represents the lower bound of the search space. Interestingly, memory requirements do not change as we shift to smaller or larger sub-images.

According to [28], even for large search spaces (e.g., 10^{30} points), acceptable combinations are found after only ten simulated generations. This means we should confine ourselves to subregions not larger than 16×16. In our experiments, we confined the algorithm to region sizes of 4×4 and larger.

Initializing the population refers to setting the initial values of the chromosomes within the population. One technique is to create a totally random population by assigning a value of zero or one with an equal probability to each bit in the chromosome's bit array. Another technique uses heuristics; the possible states are selected by restricting edge points to be in the vicinity of a high dissimilarity. This can be accomplished by creating a thresholded edge map based on the dissimilarity map, which is then dilated to expand the edge points to their vicinity. The bits corresponding to ones in the vicinity map then take random values with equal probability, others will be set as nonedge points and will not participate in mutations and crossovers.

Rank selection is used for parent selection since it maintains population diversity and is independent of fitness/cost magnitudes.

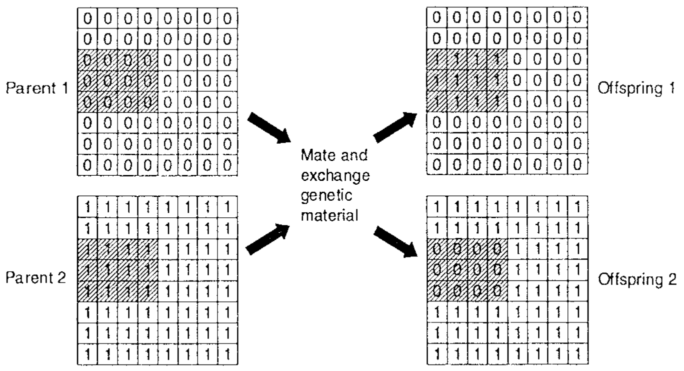

Fig. 2. Two point 2-D crossover.

We ensure that all cost is calculated after all the sub-images have been updated. In this way, the effects of changing the adjacent subregions will be accounted for in the cost calculations, after the whole image has been processed once.

We want our two-dimensional (2-D) crossover to be consistent with the traditional crossover operator in which two points are selected at random, and the genetic material between the two points is exchanged. The only difference is that each point selected is represented by a vector of length two, instead of a single number, and the genetic material exchanged is an array, not a vector (Fig. 2). It is essential to implement reduced surrogate crossover to keep the crossover effective, even when parts of each chromosome do not participate as edges. If the rectangular regions defined by the crossover points of the two parents are identical, another two crossover points are selected. After five unsuccessful trials, the parents are replaced with two new mates and the process is repeated.

Our mutation operator simply behaves like the one in the traditional GA, where each gene of the chromosome has a low, equal opportunity to undergo a mutation. When mutation occurs on a bit string representation, the bit corresponding to the gene to be mutated is flipped, i.e., its value is changed from 0 to 1 or vice versa. In the bit array chromosome, genes are selected for mutation according to the mutation probability in a linear, raster scan fashion, starting with the first row vector. This is called the single point mutation operator.

Greffenstette [29] showed that using problem-based operators could improve fine local tuning in the convergence process, reducing the need for post-processing. In order to guide the optimization search, Tan *et al.* [3] introduced various transformations or state changes, where a site 1 is selected at random, and the state is changed according to one of five strategies, selected randomly. These strategies can be intuitively implemented in the hybrid GA. The relative probability of these strategies occurring during state change was kept fixed by Tan *et al.* [3], but will be changed adaptively in our GA. Four additional mappings are created. These mappings are based on a 3×3 window $W_l(S)$. Similar to the simple mutation operator, a site l is selected, or a gene in the chromosomes, according to the mutation probability. The site l is limited, so that the window $W_l(S)$ will be fully within the chromosome, which means genes corresponding to the first and the last rows and columns of the bit array are excluded. This puts no limits on the reversibility [3] of the transformations. The only limitation it asserts is that a single transformation cannot transform pixels on both side of the subregion boundaries at the same time, so that when these pixels need to be transformed, it cannot be done in a single step.

In the second mutation strategy, another site $1'$ within the neighborhood Vl(S) is selected, and the edge representation at both sides is complemented. This is called the double point mutation operator.

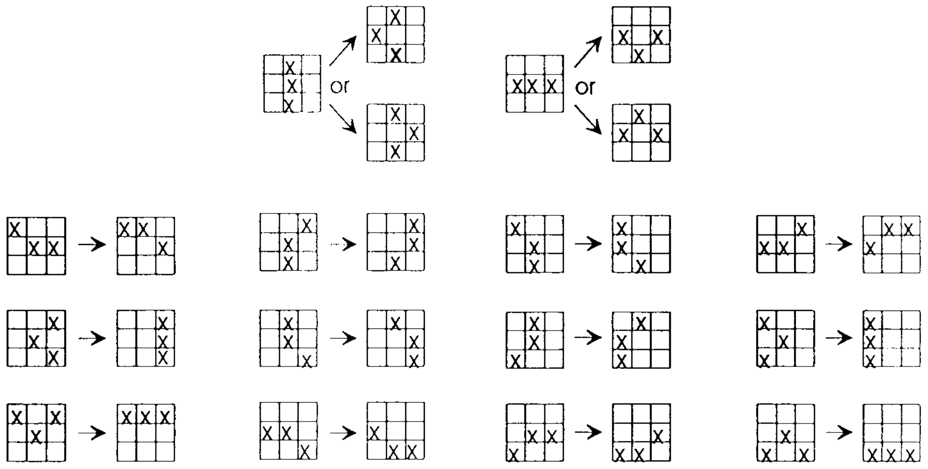

Fig. 3. Single-point transformation mutation operator.

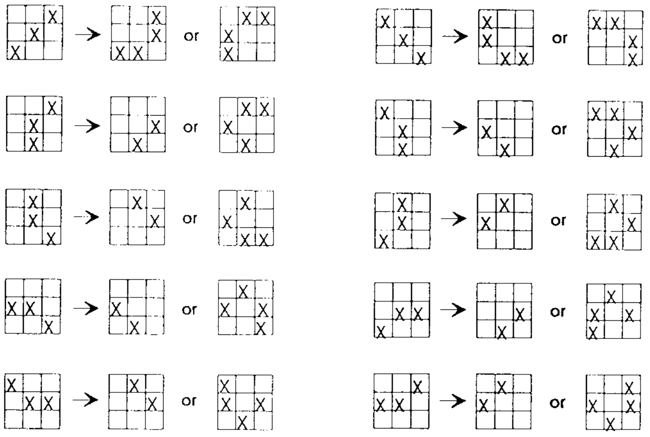

Fig. 4. Multiple-point transformation mutation operator.

In the third mutation strategy, if an edge configuration that matches one of the 14 configurations in Fig. 3 is found, the corresponding transformation is performed. If the edge configuration at this site does not correspond to any of those, no change is made. We call this the single-point transformation mutation operator. The multiple-point transformation mutation operator behaves in the same way, except that it uses the ten transformations in Fig. 4. Finally, we have the random edge-configuration transformation operator which creates a totally random configuration around the selected site.

Based on the guidelines in [20], it is advantageous to introduce stochastic evaluation into the algorithm, as used in the SA process. This can be described as follows: whenever an independent mutation occurs at a pixel site l or a window $W_l(S)$, the cost before and after the change is evaluated, and with a certain probability, the original configuration is retained if cost was found to be higher. Experiments with no stochastic evaluation, with 50% and 85% probability of reverting a higher cost configuration to the original, showed that the 85% revert probability increases convergence speed. A 0% revert probability means that the transformations are totally random, while a 100% revert probability means they are only accepted if they lower the cost. An 85% revert probability was used in most of our simulations.

V. Experimental Results and Analysis

Several experiments were used to test the GA performance. Due to space limitations, a subset of the results is included here while extensive results can be found in [30]. Throughout this section, the relative cost weights are set to: $we = l, wd = 2, wc = 0.5, wf = 3$, and $wt = 6.51$

In one experiment, a synthesized image consisting of concentric circles as described in [31] was generated. Different amounts of noise ranging from signal-to-noise ration (SNR) $= 50$ to SNR $= 1$ were applied. A 2-D Gaussian filter was used to reduce the noise in the images. A good test for a GA is running it from a totally random initial population. Fig. 5 demonstrates how the GA converges to lower cost solutions, from a fully random initial population.

Then, a quantitative comparison of the algorithm to the SA approach is performed. The SA method uses a single edge map representation and a complete stochastic evaluation where total incremental cost in state changes are considered. The GA uses a population of interconnected region-based edge maps and a stochastic evaluation where only the incremental point costs of modified edge points are considered. Another structural difference is that while the SA-based method uses fixed probabilities for the transformation strategies, the GA takes advantage of dynamic operator probabilities. This has the advantage that the participation of the transformation strategies is adaptive; each operator or strategy participates relatively according to its effectiveness.

Comparison was first based on the Pratt's figure of merit. The Pratt's figure of merit can be used to quantitatively compare the results of different edge detectors. It measures the deviation of the output edge from a known ideal edge in the following manner [32]:

$$P = \frac{1}{\max(I_A, I_I)} \sum_{i=1}^{I_A} \frac{1}{1 + \alpha d^2(i)} \tag{10}$$

where I_A is the number of edge points detected, I_I is the number of edge points in the ideal edge image, $\propto$ is a scaling factor, and $d(i)$ is the distance of the detected edge pixel from the nearest ideal edge position. Pratt's figure of merit is a rough indicator of edge quality. A higher value for the Pratt figure of merit denotes a better edge image. This figure does not take into account certain desired edge characteristics such as edge thinness or continuity. We used an ideal step image of size 256×256, with a gray value of 115 in the right part of the image, a gray value of 140 in the left part of the image, and a single pixel column of gray value 128 in the middle. A zero-mean Gaussian noise with variance $= 100$ was then added. The Pratt's figure of merit was calculated for the ideal edge image using a scaling factor of 0.01. Both edge-detection approaches were applied to five images, and the average of their results over the five

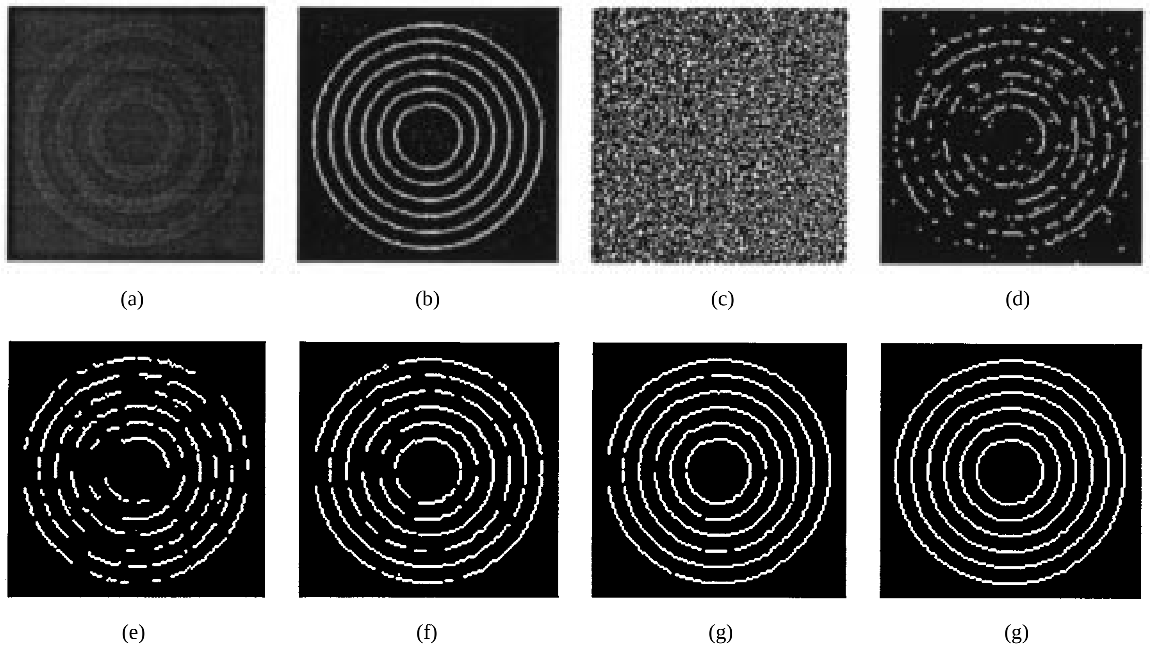

Fig. 5. (a) The rings image with SNR $= 50$. (b) The dissimilarity map, (c) after the first generation from the random initial population, (d) after 100 generations, (e) after 200 generations, (f) after 500 generations, (g) after 1000 generations, and (h) after 2300 generation the algorithm has converged to a low cost state, completing the edges.

runs was obtained. The average value for the SA approach was 0.77. The GA, on the other hand, increased this value to 0.85.

The edge-detection algorithm was also applied to different medical image modalities including computed tomography (CT) and ultrasound. Only the magnetic resonance (MR) results are shown here. Edge detection was performed on two T1-weighted magnetic resonance imaging (MRI) brain images, one is a 256×256 sagittal image, the other is a 256×256 axial image. The parameters used to acquire images were as follows; TE $= 7.0$ ms, TR $= 43$ ms, 3–4-mm slice thickness and 256×256 image matrix. The images and the results of the edge detection were processed using a GA with subregion size 8×8, bias 1.5, population size 50 and mutation probability 0.08. The algorithm was run for 300 generations. The results of the edge detection are shown in Fig. 6. The detected edges for the MRI images were thin, continuous, well localized, and most of the basic edge features were detected.

In order to quantitavely compare the MR results to SA, the value of the cost function was used. The cost function quantifies the edge quality favoring thin, continuous and well-localized edge. The smaller the cost function, the better the edge quality. Fig. 7 shows a comparison between the cost function values throughout the 300 generations. The GA achieved an objective function that is 18% lower than that obtained by SA. Similar results were obtained for the CT and ultrasound images.

The GA can be implemented in parallel, both by making each subregion be processed separately and by processing each individual of the population independently. It was shown in [30] that the fully parallel GA had considerable speed advantage over the parallel SA algorithm, while the serial implementation was slower.

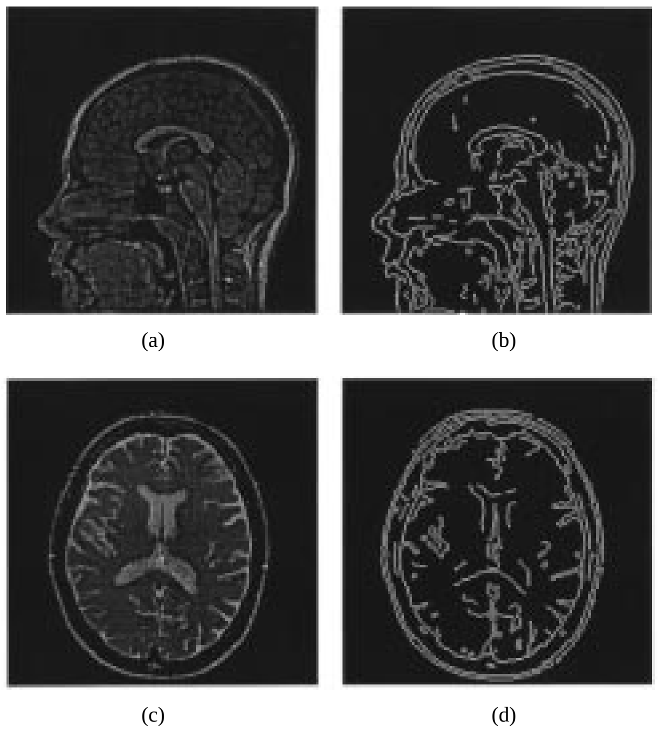

Fig. 6. Edge detection of MRI brain images. (a) A sagittal T1-weighted MRI brain image and (b) its edge image, after 300 generations. (c) An axial T1-weighted MRI brain image and (d) its edge image after 300 generations.

VI. Conclusion

In this paper, we have investigated the application of GA's to edge detection of medical images using cost minimization to accurately localize thin, continuous edges. In our effort, we based the optimization on cost evaluations and transformations defined by Tan *et al.* [3], [14], where SA was used for the optimization. We extended the bit string chromosome of the traditional GA to a bit-array chromosome, which conforms closely with a logical edge representation. We introduced problem space reduction with dependent regions. To increase the performance of the traditional GA, we added reduced surrogate crossover, ranking selection, dynamic operator rates, and stochastic evaluation on the operators.

The GA was compared to the SA approach using ideal and actual medical images from different modalities including MRI, CT, and

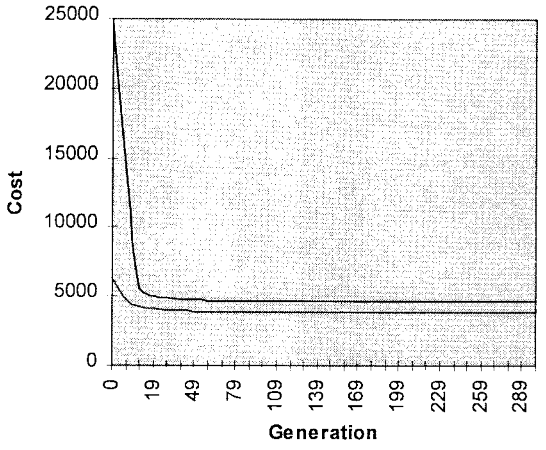

Fig. 7. Quantitative comparison between the GA and SA using the cost function.

ultrasound. The GA improved the Pratt figure of merit from 0.77–0.85 for ideal images. For actual images, the value of the cost function was used for quantitative comparison. For MR images, the GA improved the cost function value by 18%. Similar results were obtained for other modalities. The detected edges were thin, continuous, and well localized. Most of the basic edge features were detected.

Further study can be made in this field of research on the basis of the results gathered from this paper. The GA could be adapted to different 2-D image processing problems such as segmentation and feature extraction. Other representations and cost evaluation functions could be studied. For example, a representation based on graph theory as proposed in a graph theoretic edge finding approach [33] can be used to improve the accuracy of the algorithm by avoiding subregion dividing.

References

[1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Reading, MA: Addison Wesley, reprint, 1992.

[2] J. F. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, pp. 679–698, Aug. 1986.

[3] H. L. Tan, S. B. Gelfand, and E. J. Delp, "A cost minimization approach to edge detection using simulated annealing," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, pp. 3–18, Jan. 1991.

[4] F. Bergholm, "Edge focusing," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, pp. 726–741, June 1987.

[5] A. F. Korn, "Toward a symbolic representation of intensity changes in images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10., pp. 610–625, May 1988.

[6] D. Williams and M. Shah, "Edge contours using multiple scales," *Comput. Vision, Graphics, Image Processing*, vol. 51, pp. 256–274, 1990.

[7] S. Mallat and S. Zhong, "Characterization of signals from multiscale edges," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, pp. 710–732, July 1992.

[8] Y. Lu and R. C. Jain, "Reasoning about edges in scale space," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, pp. 450–468, Apr. 1992.

[9] V. S. Nalwa and T. O. Binford, "On detecting edges," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, pp. 699–714, June 1986.

[10] D. Williams, and M. Shah, "A fast algorithm for active contours and curvature estimation," *CVGIP: Image Understanding*, vol. 55, no. 1, pp. l4–26, 1992.

[11] A. A. Amini, S. Tehrani, and T. E. Weymouth, "Using dynamic programming for minimizing the energy of active contours in the presence of hard constraints," in *Proc. 2nd Int. Conf: on Computer Vision*, 1988, pp. 95–99.

[12] L. D. Cohen and I. Cohen, "Deformable models for 3-D medical images using finite elements and balloons," in *Proc. 12th Int. Conf on Pattern Recogn.*, 1992, pp. 592–597.

[13] S. M. Bhandarkar, Y. Zhang, and W. D. Potter, "An edge detection technique using genetic algorithm-based optimization," *Pattern Recogn.*, vol. 27, no. 9, pp. 1159–1180, 1994.

[14] H. L. Tan, Edge detection by cost minimization, Ph.D. dissertation, Purdue Univ., West Lafayette, IN, 1989.

[15] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated annealing," *Sci.*, vol. 220, pp. 671–680, 1983.

[16] K. P. Philip, E. L. Dove, D. D. McPherson, N. L. Gotteiner, M. J. Vonesh, W. Stanford, J. E. Reed, J. A. Rumberger, and K. B. Chandran, "Automatic detection of myocardial contours in cine-computed tomographic images," *IEEE Trans. Med. Imag.*, vol. 13, June 1994.

[17] M. A. Guttman, J. L. Prince, and E. McVeigh, "Tag and contour detection in tagged MR images of the left ventricle," *IEEE Trans. Med. Imag.*, vol. 13, Mar. 1994.

[18] Z. Yue, A. Goshtasby, and L. V. Ackerman, "Automatic detection of rib borders in chest radiographs," *IEEE Trans. Med. Imag.*, vol. 14, Sept. 1995.

[19] M. Sonka, M. D. Winniford, and S. M. Collins, "Robust simultaneous detection of coronary borders in complex images," *IEEE Trans. Med. Imag.*, vol. 14, Mar. 1995.

[20] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Learning*. Reading, MA: Addison-Wesley, 1989.

[21] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. Michigan Press, 1975.

[22] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolutionary Programs*. Berlin, Germany: Springer-Verlag, 1992.

[23] A. Hill and C. J. Taylor, "Model-based image interpretation using genetic algorithms," *Imag., Vision Computing*, vol. 10, no. 5, 1992.

[24] D. N. Chun and H. S. Yang, "Robust image segmentation using genetic algorithm with a fuzzy measure," *Pattern Recogn.*, 29, no. 7, pp. 1195–1211, 1996.

[25] A. Toet and W. P. Hajema, "Genetic contour matching," *Pattern Recogn. Lett.*, vol. 16, pp. 849–856, 1995.

[26] J. J. Greffenstette, "Optimization of control parameters for genetic algorithms," *IEEE Trans. Syst., Man. Cybern.*, vol. SMC-16, pp. 122–128, Jan. 1986.

[27] L. Davis, "Adapting operator probabilities in genetic algorithms," J. D. Schaffer, Ed., in *Proc. 3rd Int. Conf. Genetic Algorithms*, 1989, pp. 61–69.

[28] K. A. De Jong, "Learning with genetic algorithms: An overview," *Mach. Learning 3*, 1988, pp. 121–138.

[29] J. J. Greffenstatte, "Incorporating problem specific knowledge into genetic algorithms," in *Genetic algorithms and simulated annealing*, L. Davis, Ed. New York: Morgan Kaufman, 1987.

[30] M. Gudmundsson, "Edge detection using genetic algorithms," M.S. thesis, Univ. Miami, Coral Gables, FL, Aug. 1994.

[31] L. Kitchen and A. Rosenfeld, "Edge evaluation using local edge coherence," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 11, no. 9, pp. 597–605, 1981.

[32] W. K. Pratt, *Digital Image Processing*. New York: Wiley, 1978.

[33] Z. Wu and R. Leahy, "Image segmentation via edge contour finding: A graph theoretic approach," in *Proc. IEEE Computer Vision and Pattern Recogn. Conf. '92*, pp. 613–619.