



0031-3203(94)E0028-J

AN EDGE DETECTION TECHNIQUE USING GENETIC ALGORITHM-BASED OPTIMIZATION

SUCHENDRA M. BHANDARKAR,† YIQING ZHANG‡ and WALTER D. POTTER††

† Department of Computer Science and ‡ Artificial Intelligence Programs, 415 Boyd Graduate Studies Research Center, University of Georgia, Athens, GA 30602-7404, U.S.A.

(Received 22 March 1993; in revised form 7 March 1994; received for publication 15 March 1994)

Abstract—In this paper we present a genetic algorithm-based optimization technique for edge detection. The problem of edge detection is formulated as one of choosing a minimum cost edge configuration. The edge configurations are viewed as two-dimensional *chromosomes* with fitness values inversely proportional to their costs. The design of the *crossover* and the *mutation* operators in the context of the two-dimensional chromosomal representation is described. The *knowledge-augmented* mutation operator which exploits knowledge of the local edge structure is shown to result in rapid convergence. The incorporation of *meta-level* operators and strategies such as the *elitism* strategy, the *engineered conditioning* operator and *adaptation* of mutation and crossover rates in the context of edge detection are discussed and are shown to improve the convergence rate. The genetic algorithm with various combinations of meta-level operators is tested on synthetic and natural images. The performance of the genetic algorithm-based cost minimization technique is compared both qualitatively and quantitatively with local search-based and simulated annealing-based cost minimization approaches. The genetic algorithm-based technique is shown to perform very well in terms of robustness to noise, rate of convergence and quality of the final edge image.

Genetic algorithm Edge detection Cost minimization

1. INTRODUCTION

Edge detection is an important task in computer vision. It is the front-end processing stage in object recognition and image understanding systems. The accuracy with which this task can be performed is a crucial factor in determining overall system performance. Although the ultimate goal of any edge detection task is to detect edges in a given gray scale image, there are many ways in which to accomplish this task. Rosenfeld and Kak,⁽¹⁾ Torre and Poggio⁽²⁾ and Peli and Mallah⁽³⁾ present an excellent overview of this topic.

Most edge detection operators such as the gradient operator, the Laplacian operator or the Laplacian-of-Gaussian operator are very simple. In these operators, the edge pixels are defined to be those where the first-order derivative of the pixel intensity values exceeds a certain threshold or the second-order derivative of the pixel intensity values undergoes a zero crossing. The first- and second-order derivatives of the pixel intensity values are approximated by the appropriate differences. These operators are only suitable for detecting limited types of edges and are highly susceptible to noise often resulting in fragmented edges. More recent edge detection schemes can be classified as based on optimal filtering,⁽⁴⁻⁸⁾ residual analysis,⁽⁹⁾ surface fitting,^(10,11) anisotropic diffusion^(12,13) and sequential contour tracing.⁽¹⁴⁻¹⁶⁾ 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. One of the reasons is that most of these edge detection techniques are based on a very restricted definition of what constitutes an edge and in fact most of them are based on the assumption that the edges in the image are step intensity edges. This makes generalization of these techniques to other edge types such as texture edges or intensity edges with arbitrary profiles very difficult. Moreover, these aforementioned approaches consider the edge detection problem as one that is based upon the response of the edge detector at a single pixel location, i.e. the nature of the edge structure around a given pixel in the edge image is largely ignored. So although the performance of the edge detector in terms of signal-to-noise ratio and localization accuracy is optimized at each individual pixel location, the edge image as a whole could still be unsatisfactory, i.e. causing the resulting edges to be thick, fragmented and perceptually non-intuitive.

More recently, Tan *et al.*^(17,18) and Acton and Bovik⁽¹⁹⁾ have cast the problem of edge detection as one of cost minimization. Their work attempts to overcome the aforementioned shortcomings of existing edge detection techniques by formulating a definition of an edge that is general enough to include most edge types. Their work also improves on existing edge detection techniques by explicitly considering the local edge structure in the neighborhood of the hypothesized edge pixels. They define an edge to be a boundary in an image that separates two regions that have significantly dissimilar characteristics. The cause of the dissimilarity could be due to factors such as the geometry

of the object, surface reflectance, viewpoint, and illumination. This definition of an edge includes a wide variety of edge types. They also require that the edges detected as a result of minimizing the cost function be thin, continuous, long and most importantly, occupy an accurately computed location, and partition the dissimilar regions in the best possible manner.

Tan *et al.*⁽¹⁷⁾ have implemented a comparative cost function approach to edge detection. They assign to an edge image a cost measure which is a function of the local edge structure described in terms of region dissimilarity, continuity, fragmentation, and edge thickness. The edge detection algorithm is based on a simple local search technique (also known as hill climbing or greedy search). The local search algorithm compares two edge configurations (edge images) which are different only at one pixel, and selects the better one. The cost factors are computed by searching a decision tree. The comparative cost function approach to edge detection based on local search is shown to yield good results. Tan *et al.*⁽¹⁸⁾ have also implemented a simulated annealing approach to edge detection. Simulated annealing is a stochastic optimization technique derived from Monte Carlo methods in statistical mechanics.^(20,21) In their simulated annealing algorithm, each edge image S_k is assigned a cost function $F(S_k)$ which is to be minimized. A sequence of decreasing positive numbers, T_k , called a temperature schedule, represents the manner in which the algorithm is *annealed* or *cooled* towards a global optimum. The simulated annealing algorithm compares two edge images, but selects a new edge image stochastically. Given the current edge image X_k , it generates a candidate edge image, Y_k , which is nearly identical to X_k except in a local 3×3 window. Y_k is selected as the new state X_{k+1} with a probability $\exp(\max(F(Y_k) - F(X_k), 0.0)/T_k)$ or $X_{k+1} = X_k$ otherwise. Since an edge configuration is selected stochastically based on the relative goodness of one edge image vs another, the simulated annealing algorithm is less prone to being trapped in an undesired local optimum. The edge detection technique based on cost minimization via simulated annealing was also shown to yield good results. Acton and Bovik⁽¹⁹⁾ use a *mean field* neural network to determine a minimum cost edge configuration that provides thin and continuous edge contours while also providing accurate localization. Anisotropic diffusion is used to provide localized edge data that partitions dissimilar regions in the best manner possible. The mean field neural network is used to optimize a cost function that incorporates criteria such as region dissimilarity, edge continuity and edge thinness. Mean field annealing, which can be looked upon as a deterministic approximation to simulated annealing, is used to determine a minimum cost edge configuration. The combination of anisotropic diffusion with mean field annealing is shown to yield good results with fairly low computational burden and high parameter insensitivity.

In this paper, we present a genetic algorithm-based cost minimization approach to edge detection. The

genetic algorithm, the conceptual basis of which lies in Darwin's *Theory of Natural Selection* (better known as *the survival of the fittest*), is a heuristic search/optimization technique for obtaining the best possible solution in a vast solution space. Given certain information about the problem domain, genetic algorithms focus on a subset of promising solutions which would eventually lead to convergence to a globally optimal solution. Genetic algorithms are robust in that they are relatively unaffected by the presence of spurious local optima in the solution space. Because of their ability to perform effective search, genetic algorithms have been successfully applied to problems in different disciplines. This has led to a great deal of research in the theoretical and practical aspects of genetic algorithms. Many advanced genetic algorithm operators have been designed for performing effective search of the solution space and speeding the convergence of solutions. Problem-solving methodologies based on genetic algorithms have been acknowledged as effective problem solving tools by many experts in different areas. This motivates us to consider the genetic algorithm as a candidate optimization technique that could be used to perform edge detection via cost minimization.

In this paper, we have proposed and implemented a genetic algorithm-based cost minimization approach to edge detection. We have also implemented both the local search-based and the simulated annealing-based approaches to edge detection, as described in references (17, 18), and compared their results with those of the genetic algorithm approach. We have used both synthetic and natural images to compare the performance of all the three aforementioned cost minimization approaches to edge detection. In this paper, we present experimental results for two intensity images: a synthetic *checker-board* image and a natural *computer-terminal* image acquired using a CCD video camera. In the case of the synthetic image we compute the *Pratt Figure of Merit* for the resulting edge images and use it as a metric for comparing the performance of the three edge detection approaches. For the real image, the *Pratt Figure of Merit* of the resulting edge image could not be easily computed. Hence, for the *computer-terminal* image we evaluate and compare the final edge images by computing their costs using the cost function. All three algorithms are run on the same set of synthetic and natural gray scale images which are corrupted with zero mean, additive Gaussian noise with different variance levels.

This paper consists of five sections. In Section 1, we have stated and described the problem. In Section 2, we describe our proposed genetic algorithm-based cost minimization approach to edge detection. In Section 3, we briefly describe, for the sake of completeness, the local search-based and the simulated annealing-based cost minimization approaches to edge detection. In Section 4, we discuss and compare the results obtained by the three different cost minimization approaches to edge detection. In Section 5, we conclude the paper and suggest future research directions.

2. GENETIC ALGORITHM-BASED OPTIMIZATION FOR EDGE DETECTION

A gray scale image is a two-dimensional array of pixels $G(m, n)$, $1 \leq m \leq m_{max}$, $1 \leq n \leq n_{max}$, with each pixel having a gray scale value typically in the range between 0 and 255 (i.e. 8 bits per pixel). In our case $m_{max} = n_{max} = 256$. An edge image (or an edge configuration), is a Boolean or binary array of pixels $S(m, n)$, $1 \leq m \leq m_{max}$, $1 \leq n \leq n_{max}$ with each pixel having a value 0 or 1. If $S(m, n) = 1$ the pixel is an *edge pixel* and a *non-edge pixel* otherwise. In the following subsections, we describe what we mean by the *validity* of an edge structure in a local neighborhood of an edge pixel and also describe the cost function associated with an edge configuration. Our cost function formulation is similar to that described in the work of Tan *et al.*^(17,18) but is included here for the sake of completeness. We also describe our genetic algorithm-based approach for minimization of the cost function and discuss the various issues related to our genetic algorithm-based edge detection technique.

2.1. Valid local edge structures

In order to determine the validity of the local edge structure, we examine each pixel in an edge image and evaluate the local edge structure in a 3×3 window centered around the pixel. An edge structure is deemed either *valid* or *invalid* based on the criteria described below:

A valid edge structure falls into one of the following categories:⁽¹⁷⁾

- (1) An edge pixel has none or one other neighboring edge pixel.
- (2) An edge pixel has two other neighboring edge pixels and the resulting edge structure does not turn by an angle $\theta \geq 45^\circ$ as shown in Fig. 1.
- (3) An edge pixel has three other neighboring edge pixels and forms one of the eight edge structures shown in Fig. 2.
- (4) An edge pixel has four other neighboring edge pixels and forms one of the two edge structures shown in Fig. 3.

All other local edge structures are considered invalid.

2.2. Computation of cost for an edge image

The point cost of an edge image $S(m, n)$ at pixel site

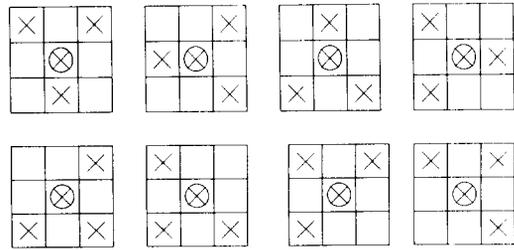


Fig. 2. Valid three-neighbor edge structures.

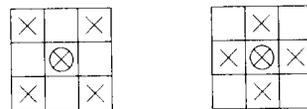


Fig. 3. Valid four-neighbor edge structures.

l is a weighted sum of cost factors:

$$F(S, l) = \sum_i w_i C_i(S, l) \tag{1}$$

The total cost $F(S)$ of an edge image S is the sum of the point costs at each pixel site in the image:

$$F(S) = \sum_l F(S, l) = \sum_l \sum_i w_i C_i(S, l) \tag{2}$$

In particular, if we examine a pair of edge images S_i and S_j which are nearly identical except in the window $W(l)$ centered around a pixel site l , a comparative cost function can be defined as follows:

$$\Delta F(S_i, S_j, l) = \sum_{W(l)} \sum_k w_k [C_k(S_i, l) - C_k(S_j, l)] = \sum_{W(l)} \sum_k \Delta C_k(S_i, S_j, l) \tag{3}$$

where $0 \leq C_k \leq 1$ and $w_k \geq 0$. The C_k s are the cost factors and the w_k s are the associated weights. The window $W(l)$ can be as small as a single pixel l or as large as a 3×3 window centered at pixel site l . If $\Delta F(S_i, S_j, l) < 0$, S_i is a better configuration whereas if $\Delta F(S_i, S_j, l) > 0$, S_j is a better configuration. If $\Delta F(S_i, S_j, l) = 0$ then both configurations are deemed identical in terms of cost.

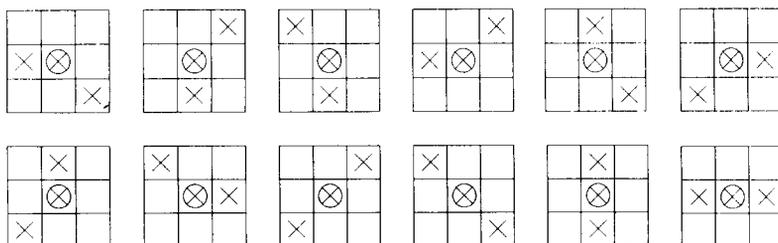


Fig. 1. Valid two-neighbor edge structures.

2.3. Cost factors associated with the cost function

The cost factors in the cost function $F(S, l)$ or the comparative cost function $\Delta F(S_i, S_j, l)$ are computed by examining the local edge structure of the edge image about each pixel and also the image data. There are five cost factors to be considered.⁽¹⁷⁾ These are:

- (i) Cost factor $C_d(S, l)$ based on region dissimilarity.
- (ii) Cost factor $C_t(S, l)$ based on for edge thickness.
- (iii) Cost factor $C_c(S, l)$ based on edge curvature.
- (iv) Cost factor $C_f(S, l)$ based on edge fragmentation.
- (v) Cost factor $C_e(S, l)$ based on the number of edge pixels detected.

Each of these cost factors are elaborated upon in the following subsections.

2.3.1. Cost factor based on region dissimilarity. In order that the detected edges partition dissimilar regions in the image in the best possible manner, a cost measure based on region dissimilarity $C_d(S, l)$ is assigned to each pixel. This cost measure is intended to emphasize edge pixels that constitute the common boundary of dissimilar regions and simultaneously penalize non-edge pixels. Before we compute $C_d(S, l)$, we carry out a dissimilarity enhancement of the input image to obtain an enhanced image $D = \{D(i, j); 1 \leq i, j \leq N\}$ such that $0 \leq D(i, j) \leq 1$. The enhanced image D is thus a collection of pixels wherein each pixel value is proportional to the degree of region dissimilarity that exists at that pixel site. Values of $D(i, j)$ close to 1 denote pixels at the boundary of dissimilar regions that could be considered as highly likely candidates for being classified as edge pixels. Coming up with an enhanced image entails the following: (a) formulating well defined regions of interest on either side of the edge structure and (b) formulating a function that

measures the dissimilarity between the regions of interest.

The regions of interest are defined with respect to a set of basis edge structures. In our case we have used the valid two-neighbor edge structures (Fig. 4) to constitute the basis set. A pair of dissimilar regions, R_1 and R_2 , is defined for each edge structure in the basis set. Figure 4 shows the 12 valid two-neighbor edge structures that constitute the basis set and the associated regions of interest R_1 and R_2 on either side of each edge structure. A function $f(R_1, R_2)$ that measures the degree of dissimilarity between the regions R_1 and R_2 is then defined. This measure could be a simple difference of gray level averages in regions R_1 and R_2 or, depending on the specific application, could be a more complex measure based on the statistical distribution or structural properties of the gray level values. In our case, a simple difference of gray level averages between the regions was used as the value of $f(R_1, R_2)$.

The enhanced image $D(l)$ is obtained using the following procedure:⁽¹⁸⁾

- (1) All pixels in $D(l)$ are initially set to zero.
- (2) At each pixel site l , we perform steps (a) and (b):

(a) We fit each of the edge structures in the basis set shown in Fig. 4 to the pixel site l . The pixel site l is chosen to be the center pixel for each fitted edge structure from the basis set. We then calculate the difference in average gray level values $f(R_1, R_2)$ between regions R_1 and R_2 as indicated in Fig. 4 for each fitted edge structure. The edge structure that results in the maximum value of $f(R_1, R_2)$ is deemed to be the best fitting edge structure. Note that each edge structure in the basis set contains exactly three edge pixels. For the best fitting edge structure, the sites of the three edge pixels are denoted as l, l_1 and l_2 (Fig. 4).

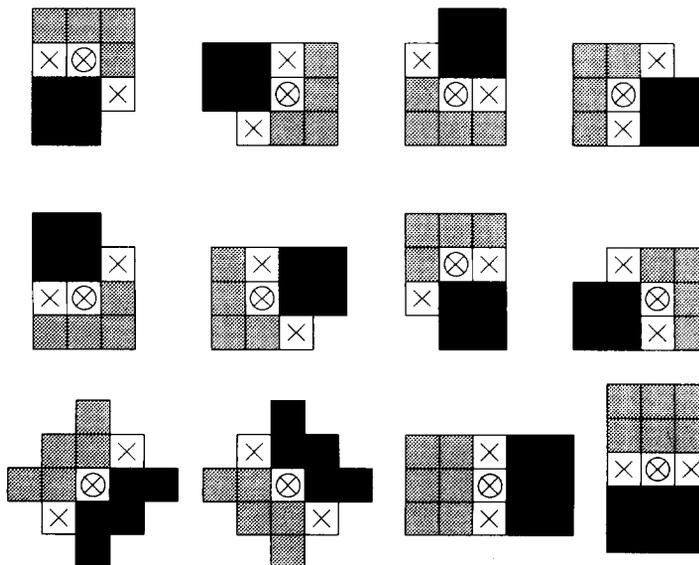


Fig. 4. Regions of interest for valid two-neighbor edge structures.

(b) Next, we perform non-maxima suppression by shifting the location of the best fitting edge structure in a direction determined by the edge structure. For vertical, horizontal, and diagonal edge structures, the shift operation is performed by moving the edge location by one pixel in each of the directions perpendicular to the edge. For other edge structures, the shift operation is performed by moving the edge location by one pixel in each of the four directions: up, down, left, and right as shown in Fig. 5. For each shifted edge structure, we determine the new regions R_1 and R_2 and the corresponding value of $f(R_1, R_2)$. We consider one of the following two cases:

(i) If no larger value of $f(R_1, R_2)$ results from shifting the best fitting edge structure, we set $\delta = f(R_1, R_2)/3$ where $f(R_1, R_2)$ is computed from the best fitting edge structure. We increment the value of each of the pixels $D(l)$, $D(l_1)$ and $D(l_2)$ by δ .

(ii) If the shifting results in a larger value of $f(R_1, R_2)$ then the pixel values $D(l)$, $D(l_1)$ and $D(l_2)$ are left unaltered.

(3) We truncate the values of $D(l)$ to a maximum value of 1 at all sites l .

Step (3) ensures that the final dissimilarity values in image $D(l)$ lie in the range $[0, 1]$.

The cost factor $C_d(S, l)$ is computed using the enhanced image $D(l)$ and the edge image $S(l)$. The cost factor

$C_d(S, l)$ is computed by examining the edge image $S(m, n)$ in a local 3×3 window centered at l . If the pixel $S(l)$ is an edge pixel, we set $C_d(S, l) = 0$, else we set $C_d(S, l) = D(l)$.

2.3.2. *Cost factor based on edge thickness.* The cost factor $C_t(S, l)$ based on edge thickness, is assigned to each pixel location. A *thick edge pixel* is a pixel the presence of which causes multiple links between neighboring (edge) pixels in the edge structure as shown in Fig. 6. An edge structure in which there exist thick edge pixels is called a *thick edge*. The cost factor based on edge thickness, $C_t(S, l)$ is determined by considering the pixel at site l in the edge image $S(l)$. If l is a thick edge pixel, $C_t(S, l) = 1$, otherwise, $C_t(S, l) = 0$. This cost factor is designed to penalize thick edges and favor thin edges.

2.3.3. *Cost factor based on edge curvature.* The cost factor $C_c(S, l)$ based on edge curvature is designed to penalize *kinky edges* and favor straight edges. A cost $C_c(S, l)$ is assigned to a non-endpoint edge pixel at site l in the edge image S (an endpoint is an edge pixel that has at most one neighboring edge pixel). If there is a pair of neighboring edge pixels in S connected via the pixel at site l such that the resulting edge structure turns by more than 45° , we set $C_c(S, l) = 1$. If there is no pair of neighboring edge pixels that causes the

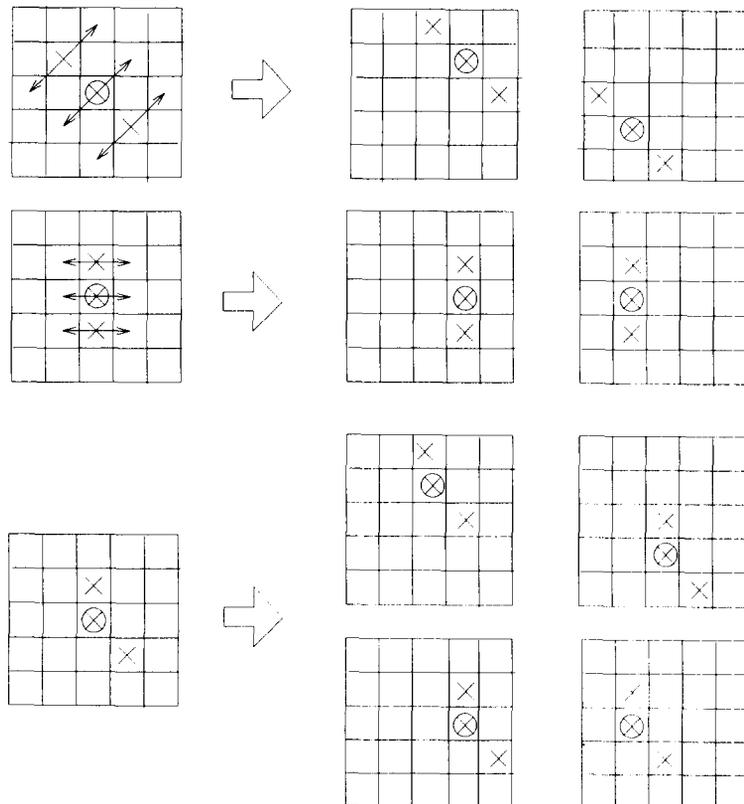


Fig. 5. Examples of non-maximum suppression.

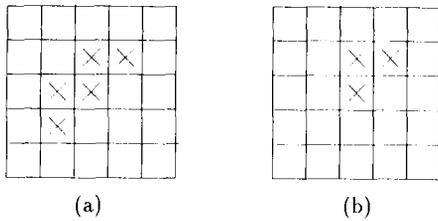


Fig. 6. Examples of thick edges (a) with 5 pixels (b) with 3 pixels.

resulting edge structure to turn by more than 45°, but there exists a pair of edge pixels that causes the resulting edge structure to turn by 45°, we set $C_c(S, l) = 0.5$. If there is no pair of neighboring edge pixels that causes the resulting edge structure to turn by more than 0° we set $C_c(S, l) = 0$.

2.3.4. Cost factor based on edge fragmentation. The cost factor $C_f(S, l)$, is designed for reducing the fragmentation of the resulting edge structure. If the edge pixel at site l in the edge image S does not have any neighboring edge pixel, we set $C_f(S, l) = 1$. If the edge pixel at site l has only one neighboring edge pixel, we set $C_f(S, l) = 0.5$. For all other cases, we set $C_f(S, l) = 0$.

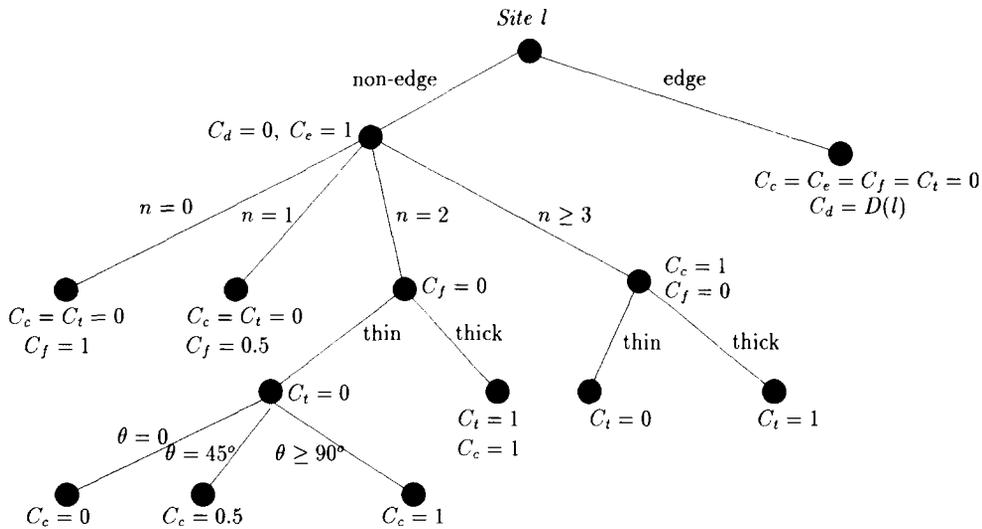
2.3.5. Cost factor based on number of edge pixels. The cost factor $C_d(S, l)$ favors the detection of edge pixels where $f(R_1, R_2)$ is non-zero, thus leading to the detection of an excessive number of edge pixels. A cost factor $C_e(S, l)$ is formulated to suppress this tendency. If pixel l is an edge pixel, we set $C_e(S, l) = 1$ otherwise $C_e(S, l) = 0$.

Each of the cost factors can be seen to place different and often conflicting requirements on the final edge

image. The final edge image therefore can be seen to be one that best satisfies all the constraints imposed by the cost factors, i.e. it can be looked upon as a problem in constrained optimization. A decision tree search can be performed when computing each of the cost factors. Figure 7 shows a decision tree for determining the cost factors $C_k, k \in \{d, t, c, f, e\}$ at each pixel site l . In Fig. 7, n denotes the number of neighboring edge pixels for a pixel site l and θ the angle associated with a two-neighbor edge structure as described in Section 2.1. The total cost of an edge image S can be calculated using equation (1) where w_k s are the weights associated with the corresponding cost factor. At this time, the numerical values for these weights are chosen heuristically. Deriving analytical expressions that relate the weights w_k to parameters such as signal-to-noise ratio, however, would be an important and challenging problem. It is recommended⁽¹⁷⁾ that $w_d(S, l) = 2.00, w_e(S, l) = 1.00, w_c(S, l) = \{0.25, 0.50, 0.75\}, w_f(S, l) = \{2.00, 3.00, 4.00\}$, and $w_t(S, l) = 2w_f(S, l) - w_c(S, l) + w_d(S, l) - w_e(S, l) + 0.01$.

2.4. Genetic algorithms—an overview

Genetic algorithms are a problem solving methodology based on Darwin's theory of evolution. Although genetic algorithms attempt to emulate the process of natural evolution, it may not be possible to draw *exact* parallels between the genetic algorithm and the process of natural evolution since the latter may be far too complex. The work in this paper treats the genetic algorithm from the standpoint of constrained optimization and hence does not attempt a faithful reproduction of the process of natural or biological evolution.



n = No. of Neighboring Pixels
 θ = Angle associated with the edge structure
 $D(l)$ = Region Dissimilarity Measure

Fig. 7. Decision tree for computing the cost function $F(S, l)$.

The two key factors critical to the success of genetic algorithms are (a) the manner in which the possible solution to the problem is encoded or represented and (b) the manner in which the possible solution is evaluated in the context of the problem domain.⁽²²⁾ In a genetic algorithm, the possible solutions to a problem in a certain problem domain are encoded as *chromosomes*.

In most problem domains a chromosome can be represented by a bit string of 1s and 0s as shown in Fig. 8(a). In the context of edge detection, a chromosome is represented by a two dimensional array of 1s and 0s as shown in Fig. 8(b). A *population* of chromosomes, also called the *gene pool*, contains possible solutions to the problem that is being solved. The size of a population is an important factor that determines the richness of a gene pool. The larger the size of the population, the richer the gene pool, and the faster the prospective solutions will appear in the population. There are many ways to generate an initial population of chromosomes. The simplest way is to generate the chromosomes randomly.

An *objective function* is used to evaluate the performance of chromosomes in the population. The objective function takes a chromosome as input and returns a number which is a measure of the chromosome's performance (also referred to as the *fitness* value of the chromosome). The fitness value determines the probability that a given chromosome will survive into and be selected for reproduction in the succeeding generation(s).

The choice of the objective function greatly influences the performance of the genetic algorithm. As pointed out, the objective function determines the probability that a given chromosome will survive into and be selected for reproduction in the succeeding generation(s). Moreover, it is also important to enlarge the difference between two competitive chromosomes whose objective function values are very close to each other. This is a situation that occurs quite often when the chromosomes in the population have converged to a certain extent. One such process that adjusts the relative fitness values of chromosomes in the population is called *normalization*. For example, in a *normalized* population, the best chromosome would have a score of 100 whereas the most defective one would have a score of 0. Consider a population of 6 chromosomes, each having an objective function (i.e. fitness) value of

$F_i = 94, 95, 96, 97, 98,$ and 99 . In such a population, the objective function values of the chromosomes are very close, therefore, all the chromosomes will have nearly the same probability of being selected for the succeeding generation(s), i.e. $94/\sum F_i, 95/\sum F_i, 96/\sum F_i, 97/\sum F_i, 98/\sum F_i,$ and $99/\sum F_i$. We can normalize the objective function so that the members of such a population can have objective function values $F_i = 0; 20, 40, 60, 80,$ and 100 , thus enlarging the difference in fitness values between competitive chromosomes. This ensures that each chromosome will have a vastly different chance of selection for the succeeding generations, i.e. $0/\sum F_i, 20/\sum F_i, 40/\sum F_i, 60/\sum F_i, 80/\sum F_i,$ and $100/\sum F_i$. Alternatively, one can raise the objective function value to some predefined power for each chromosome. For example, for a given population of 5 chromosomes, each having an objective function value of $F_i = 11, 12, 13, 14,$ and 15 , we can raise the fitness value of each chromosome to the power 2 so that each chromosome will have a greatly different chance of survival, i.e. $F_i^2/\sum F_i^2$. However, it must be noted that if a normalization technique is overemphasized, it will lead to the elimination of alternative genetic material from the gene pool. This could result in a predominance of a single strain in the gene pool, thereby causing premature convergence which should be avoided.

Chromosomes are improved upon in the following generation using a process which resembles natural evolution. This process is called *reproduction*. There are three basic operations in the reproduction of chromosomes: *selection, crossover* and *mutation*. The selection operation selects chromosome pairs for mating. One way of implementing the selection operation is by means of the *biased roulette wheel* in which each chromosome in the population has a roulette wheel slot sized in proportion to its fitness value.⁽²³⁾ During roulette wheel selection, two mates are selected for reproduction with probability values in direct proportion to their fitness values. Therefore, fitter chromosomes will contribute, on an average, a greater number of offsprings in the succeeding generation(s) whereas the least fit chromosomes will be eliminated in the selection process.

Crossover is a process by which new chromosomes are created from existing ones during reproduction. The crossover operator makes copies of the two mates selected by the roulette wheel selection operator, chooses a site (or several sites) randomly or according to certain prespecified criteria, and exchanges parts of chromosomes (Fig. 9). After crossover, genetic material in the parent chromosomes is recombined and new (and hopefully fitter) child chromosomes are created. The crossover operation is the primary mechanism used by the genetic algorithm in the exploration of the search space and is therefore assigned a comparatively high probability of occurrence.

The *mutation* operation represents the phenomena of rare chance in the evolution process. The population of chromosomes in a genetic algorithm will converge to a certain extent over a few generations due to

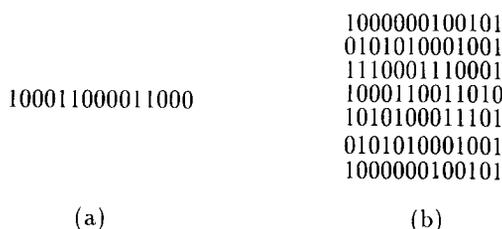


Fig. 8. Representation of chromosomes (a) one-dimensional (b) two-dimensional.

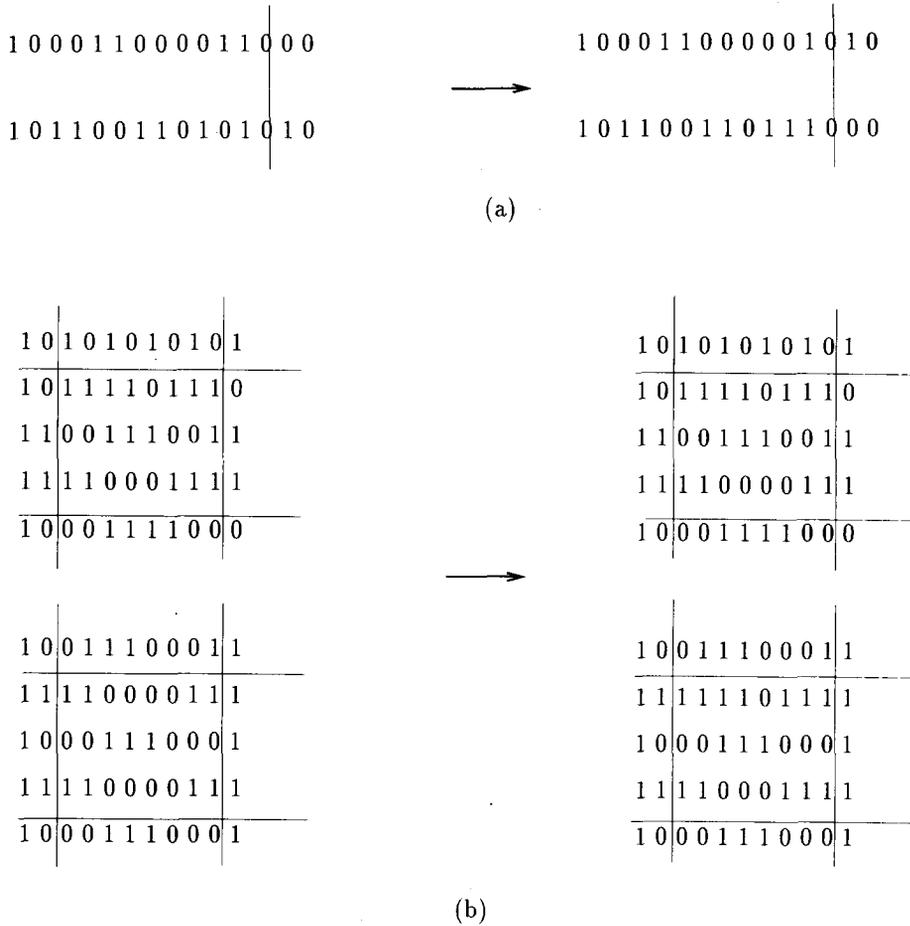


Fig. 9. Crossover operation (a) one-dimensional crossover, (b) two-dimensional crossover.

effective mate selection and crossover. However, in the process of doing so, the gene pool may lose some useful genetic material which may prevent convergence to a globally optimum solution. The role of the mutation operator is to introduce new genetic material to the gene pool, thus preventing the inadvertent loss of useful genetic material in earlier phases of evolution. In problem domains where the chromosome is represented by a bit string, the mutation operator could be as simple as flipping the bit value at a randomly chosen position in the bit string. In contrast to the crossover operation, mutation is assigned a relatively low value of probability, thus signifying a relatively infrequently occurring event.

In the context of optimization, mutation prevents a genetic algorithm from being trapped in an undesired local optimum. The problem of edge detection and contour grouping when cast as a constrained optimization problem has been shown to be ill-posed in the sense of Haddamard by Marroquin *et al.*⁽²⁴⁾ A constrained optimization problem is considered to be well-posed in the sense of Haddamard if (a) a solution exists, (b) the solution is unique and (c) the solution is well-behaved, i.e. varies continuously with the problem parameters. An ill-posed problem fails to satisfy one

or more of the aforementioned criteria. The problem of edge detection and contour grouping is shown to be ill-posed due to the non-convexity of its solution space⁽²⁴⁾ resulting in the presence of several undesired local optima. It is necessary in this case to use a combinatorial optimization technique that has the capability to escape from an undesired local optimum. The mutation operator gives the genetic algorithm the capability to escape from an undesired local optimum.

At the time of writing of this paper, there does not exist a systematic or analytical means of selecting the values for the probability of crossover or mutation operators. In most cases, the crossover and mutation probabilities are selected heuristically and fine-tuned over several runs of the genetic algorithm. In our case we selected the initial crossover and mutation probabilities based on the recommendations of Goldberg⁽²³⁾ which were then modified and fine-tuned over several runs of the genetic algorithm.

The probabilities assigned to crossover and mutation can also be made to change over the course of evolution. For example, during the initial stages of evolution when the population or gene pool contains vastly different chromosomes, the crossover operator is typically assigned a higher probability value, whereas the muta-

tion operator is assigned a relatively lower probability value. After the chromosomes in the population have converged to a certain extent, the crossover operation does not introduce significant genetic changes in the population and hence the mutation operation becomes a major source of introduction of new genetic material in the population. At this stage, we can lower the value of crossover probability and raise the value of the mutation probability. A genetic algorithm in which genetic operators are assigned probabilities dynamically is called an *adaptive* genetic algorithm. It is to be noted that all three operations: selection, crossover and mutation are *stochastic* and *not deterministic*. The stochastic nature of these operations is what prevents the genetic algorithm from being trapped in an undesired local optimum.

A genetic algorithm is implemented as an iterative program. It starts from an initial population, which in most cases is generated randomly, and undergoes a series of iterations, until a stopping criterion is met. A stopping criterion such as "there is no better chromosome to be found in the most recent several generations" could be used. The stopping criterion is a heuristic since strictly speaking, the evolution process is unending. A simple genetic algorithm could be described as follows:^(2,3)

1. An initial population of chromosomes is created randomly, and each individual is evaluated by the objective function.
2. Two mates are selected for reproduction with probabilities in proportion to their fitness values using roulette wheel selection.
3. Crossover and mutation operators are applied to the selected mates, and offsprings are generated.
4. Each individual offspring is evaluated by the objective function.
5. Steps 2–4 are repeated until an entirely new population of chromosomes is generated.
6. The previous population is replaced by the new population.
7. If the stopping criterion is not met, go to step 2, otherwise return the best chromosome in the new population as the solution to the problem.

There are some significant similarities and differences between the genetic algorithm and simulated annealing. The genetic algorithm and simulated annealing are similar in the sense that they are both stochastic optimization techniques with the ability to escape from undesired local optima. Simulated annealing relies on the principles of generating a stationary Markov chain of solution states with a Boltzmann distribution at a given temperature and a suitably chosen annealing schedule to arrive at a globally optimum solution. The genetic algorithm, on the other hand, relies on the diversity of a population of candidate solutions to arrive at a globally optimum solution via the crossover and mutation operations. Since, in simulated annealing, a new candidate solution is generated by perturbing an existing solution, simulated annealing could be

looked upon as a genetic algorithm with a population of unit size that employs only a mutation operator. However, simulated annealing has a temperature-controlled stochastic local search or hill climbing mechanism which is implemented using the Metropolis function.^(20,21) The local search or hill climbing mechanism is conspicuously absent in the genetic algorithm. The crossover operation is purely stochastic and can prove potentially disruptive when the population of chromosomes has converged to a certain extent. The genetic algorithm, however, possesses the ability to use high quality subsolutions as building blocks for subsequent solutions which simulated annealing lacks. At the time of writing of this paper, only simulated annealing possesses a proof of asymptotic convergence whereas the genetic algorithm does not. However, the convergence of simulated annealing entails a logarithmic cooling schedule⁽²⁰⁾ which is computationally intensive and hence rarely used in practice. With linear or geometric annealing schedules, that are often used in practice, the simulated annealing algorithm shares with the genetic algorithm, the tendency of premature convergence to an undesired local optimum.

2.5. The proposed genetic algorithm-based approach to edge detection

Our proposed genetic algorithm-based optimization approach to edge detection consists of two parts: (a) edge enhancement, which has been described in Section 2.3.1 and (b) application of the genetic algorithm. In our genetic algorithm approach, the chromosomes in the population are represented by two-dimensional binary arrays of 1s or 0s. A 1 represents an edge pixel whereas a 0 represents a non-edge pixel. The chromosomes are an explicit representation of the edge images. We have chosen a population size of 512 for images of size of 256×256 pixels so that our workstation could have enough memory for this task without disrupting other jobs. The initial chromosome population was generated randomly from the enhanced image. It is also possible to generate the initial population deterministically by sampling the results of several edge detectors such as the Sobel edge detector, gradient edge detector, Laplacian-of-Gaussian edge detector, etc. However, by choosing the initial population randomly we remove the possible bias that might result from the choice of edge detectors, the outputs of which are sampled. Since the genetic algorithm relies on the diversity within an existing population to search for a globally optimum solution, we remark that large population sizes are especially beneficial for the task of edge detection because of the obvious need for richness in the gene pool, resulting primarily from the large chromosome size and the large magnitude of the search space. The gene pool in our case contains only a negligible fraction of the entire search space, i.e. 512 images out of a possible $2^{256 \times 256} = 2^{65536} \approx 2 \times 10^{19728}$ edge configurations. This fact has certain repercussions on the convergence of the genetic algorithm as will be elaborated upon in Section 4.

Each chromosome in the population is associated with a cost obtained by performing a decision tree search shown in Fig. 7. We calculate the fitness value of each chromosome based on its relative ranking in the entire population:

$$\text{fitness}[i] = (\text{cost}[\text{worst}] - \text{cost}[i])^n \quad (4)$$

where *worst* denotes the least fit chromosome found in the present generation. During the earlier phases of evolution, we set $n = 2$. After the solutions converge to a certain extent, we make n successively larger—up to $n = 5$.

We perform simple roulette wheel selection to select mates for reproduction based on the relative fitness value of each chromosome. During the crossover process, we randomly select two sites along the X -dimension, two sites along the Y -dimension, and perform crossover. The mutation operator flips the labeling at a pixel location, from $0 \rightarrow 1$ or $1 \rightarrow 0$ with a prespecified mutation probability. We set the initial crossover rate to 0.6 and the initial mutation rate to 0.008. If after 15 generations, no better chromosome can be found, we assume that the present population contains the best edge image, i.e. the edge image corresponding to the lowest cost.

2.6. Advanced and meta-level genetic algorithm operators

In addition to the components of a simple genetic algorithm, we also apply to our optimization scheme (a) the *elitism* strategy,⁽²⁵⁾ (b) the *Engineered Conditioning* (EC) operator⁽²⁶⁾ and (c) the *Intelligent or Knowledge-augmented Mutation* operator. We show that these meta-level operators help to accelerate the convergence of the population of solutions to the desired optimum. We also adapt the basic genetic algorithm parameters during the course of evolution via dynamic assessment of the performance of the genetic algorithm.

2.6.1. Elitism strategy in a genetic algorithm. *Population replacement* in a genetic algorithm resembles the process of natural evolution in which the older generation is replaced by the newer generation. In the simplest population replacement scheme, the entire previous population is replaced by the new generation. The drawback of such a replacement is that although mates are selected according to their relative fitness values in the population, there exists a non-zero probability that good chromosomes in the current generation may still not be selected for the succeeding generation. An *elitism* strategy ensures that the best chromosome(s) is one generation survive(s) into the following generation, thus preventing a possible inadvertent loss of such chromosome(s). Although the elitism strategy may increase the rate by which a population may be dominated by a highly fit chromosome or a set of chromosomes, it appears to improve the overall performance of the genetic algorithm.⁽²⁵⁾ We show that this observation is true in our case as well. In order to prevent the inadvertent loss of the best chromosomes

due to stochastic roulette selection, we employ a meta-level elitism strategy which ensures that the best chromosome in the current generation always survives into the succeeding generation.

2.6.2. Intelligent or knowledge-augmented mutation.

In a traditional genetic algorithm, the mutation operator just flips a bit (from 0 to 1 or from 1 to 0) randomly without the knowledge of the chromosome structure in the neighborhood of the bit being flipped. In our approach, mutation is not just for preventing loss of useful genetic material, rather, mutations are performed more *intelligently* by exploiting the local edge structure so that they will help solutions converge faster. Our mutation strategies are selected and performed based on the examination of the local neighborhood in a 3×3 window centered at a randomly chosen pixel location. In particular, for valid two-neighbor local edge structures, the most frequently encountered valid local edge structures in an edge image, we define a *mutation complexity* which is a weighted sum of the number of *pixel shifts* and *pixel flips* carried out during the mutation operation. A pixel flip is an action by which a pixel's label is reversed, i.e. $1 \rightarrow 0$, or $0 \rightarrow 1$. A pixel shift is an action by which an edge pixel is shifted one position in either of the four directions: up, down, left, or right. A pixel shift is equivalent to two pixel flips (itself and one of its neighboring pixels). For a given local edge structure, valid or invalid, there could be several possible mutation options, each of which is assigned a probability based on the validity of the edge structure before and after mutation, the thickness and the curvature of the edge structure before mutation, and the mutation complexity if the edge structure happens to be a valid two-neighbor edge structure. However, it is very difficult to arrive at an exact formula, based on which the mutation probabilities could be calculated. Apart from the intuitive notion of what a valid edge structure is, several heuristic guidelines are followed in order to determine the probability distribution of the possible mutations. Sample runs show the following guidelines to be very effective:

1. Mutations that result in straight local edge structures are assigned a higher probability.
2. Mutations that result in local edge structures that turn by 45° are assigned a higher probability than those that turn by more than 45° .
3. Resulting valid local edge structures are more favored than invalid local edge structures.
4. For resulting valid two-neighbor local edge structures, those with higher mutation complexity are assigned a lower probability, and vice versa.
5. A certain non-zero probability is assigned to a mutation that would cause the resulting local edge structure to be an empty 3×3 window.
6. A certain non-zero probability is assigned to a random mutation in a 3×3 window.

In guidelines 5 and 6, probabilities are determined based on the validity of the existing local edge struc-

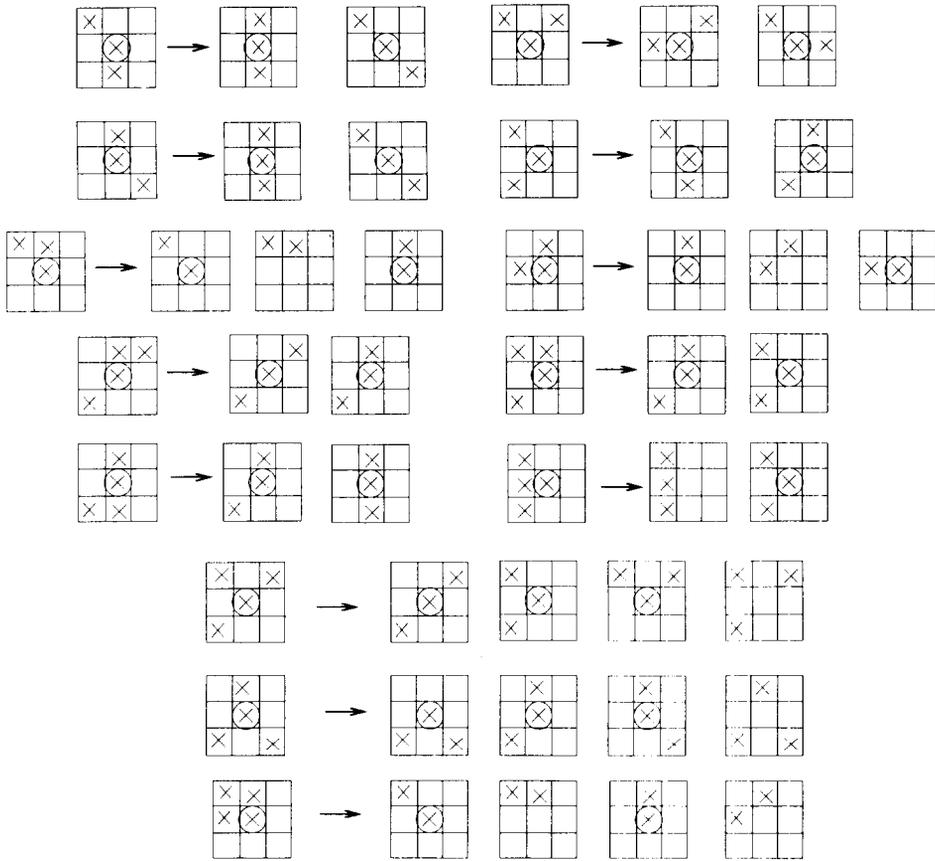


Fig. 10. Examples of mutation strategies.

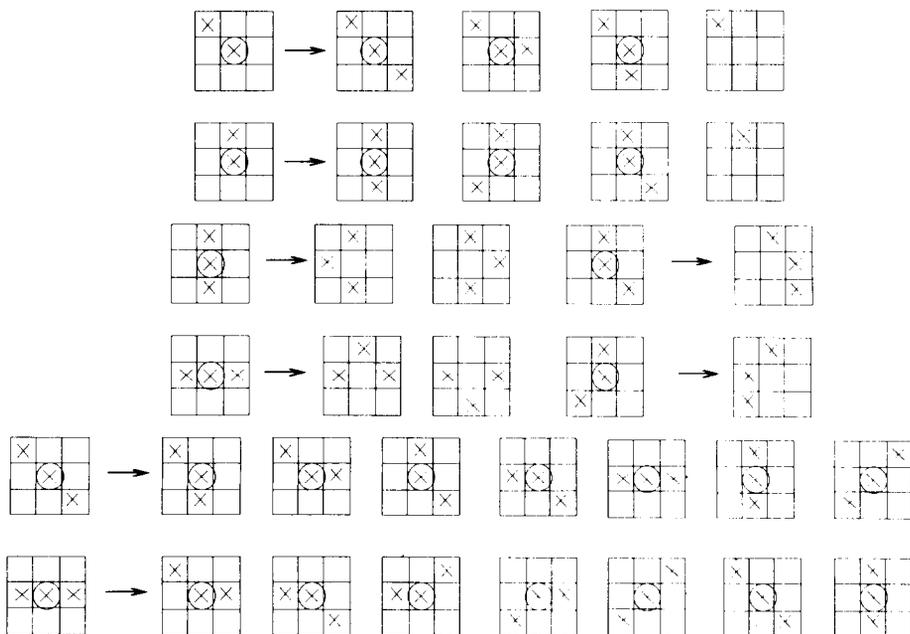


Fig. 11. Examples of mutation strategies (continued).

tures. If the existing local edge structure is valid, we assign a lower probability to guidelines 5 and 6 otherwise, the probability is higher. In Figs 10 and 11 we show mutation strategies for each specific edge structure.

2.6.3. The engineered conditioning operator. In our genetic algorithm-based optimization approach, we also employ an *Engineered Conditioning* (EC) meta-level operator that works in combination with the basic genetic algorithm operators. The Engineered Conditioning (EC) operator is an operator that can be used for local improvement in the search space.⁽²⁶⁾ With the application of the EC operator, the best chromosomes in the population are conditioned so that they may acquire the strength and the characteristics of stronger neighbors. This is in contrast to conventional genetic algorithms that improve reliability by exploiting the global nature of the genetic algorithm. We incorporate the EC operator in our genetic algorithm-based optimization approach with the expectation that the local improvement capabilities of the EC operator would result in faster convergence.⁽²⁶⁾

The EC operator represents a greedy or local search of the solution space. We take the best chromosome from the population and modify a small portion of it. This small portion is a 3×3 window centered around a pixel location chosen at random. The modification is performed stochastically based on the knowledge of the edge structure in a local neighborhood. This action is similar to that of intelligent mutation. The difference is that in the EC operation, we compare the modified chromosome with the original and if the conditioned chromosome is found to be better than the original, we substitute the original with the conditioned chromosome, otherwise we put the original back into the population. In this sense, the EC operation is deterministic (since it represents a deterministic local or greedy search) whereas the intelligent mutation operator retains a stochastic flavor. In our case, since the chromosomes are large in size, we condition five percent of the pixel locations chosen randomly in the best chromosome (edge image).

2.6.4. Adaptation of basic genetic algorithm operators. Finally, we employ a mechanism that adjusts the crossover and the mutation rates based on dynamic assessment of the performance of the genetic algorithm. During the initial stages of evolution, we assign a high probability value to the crossover operator and a very low probability value to the mutation operator. After five generations when the chromosomes in the population converge towards a highly fit chromosome and there is no better chromosome to be found, we lower the crossover rate (by 10% of the original value) and raise the mutation rate (to five-times the original value). In this case, mutation is the major source of introduction of new genetic material to the population. After 10 generations when no better chromosome can be found, we lower the crossover rate (by 10% of the original value) and raise the mutation rate once again

(to five-times the original value). After 15 consecutive generations if there is still no better chromosome to be found, we assume that the best chromosome in the present population corresponds to an approximate global optimum. The rates by which the crossover and mutation rates were adapted, were chosen empirically over several runs of the genetic algorithm. At this time, there does not exist a means of analytically deciding these adaptation rates.

In addition to the basic genetic algorithm operators, we have incorporated different combinations of advanced operators in our genetic algorithm-based edge detection technique:

1. Elitism strategy.
2. Intelligent or knowledge-augmented mutation.
3. Engineered conditioning.
4. Elitism strategy, intelligent mutation, and engineered conditioning.
5. Integrated genetic algorithm in which elitism strategy, intelligent mutation, engineered conditioning and adaptive probabilities for the basic genetic operators (crossover and mutation) are incorporated.

For implementations wherein no elitism strategy is incorporated, we set the initial crossover probabilities to 0.6 and the initial mutation probability to 0.008. For implementations wherein elitism strategy is employed, we set the initial crossover probability to 0.8 and the initial mutation probability to 0.02. In implementations wherein the elitism strategy is used, one can afford to have a higher probability of crossover and mutation since the elitism strategy ensures that any good chromosomes will not be lost in succeeding generations due to the crossover and mutation operators.⁽²⁵⁾ The aforementioned values for the crossover and mutation probabilities were determined empirically over several runs of the genetic algorithm.

3. LOCAL SEARCH AND SIMULATED ANNEALING APPROACHES TO EDGE DETECTION

In this section, for the sake of completeness, we describe briefly the local search⁽¹⁷⁾ and simulated annealing⁽¹⁸⁾ approaches to edge detection.

3.1. Local search-based cost minimization approach to edge detection

In the local search-based cost minimization approach to edge detection, the cost factors in the comparative cost function $\Delta F(S_i, S_j, l)$ are computed by performing a decision tree search as shown in Fig. 7. The local search-based cost minimization approach is an iterative algorithm in which a comparative cost function $\Delta F(S_i, S_j, l)$ is used as a criterion for determining the better of two edge configurations S_i and S_j which are identical except at one pixel location l . A new edge configuration is generated from the current edge configuration by complementing one of the pixels in the current edge configuration. The pixel site to be complemented may

be selected either randomly or in a raster scan fashion. The new edge configuration replaces the current edge configuration if the former is found to be better (i.e. has a lower cost value). The algorithm terminates when a simple stopping criterion is satisfied. The stopping criterion could be based on the number of better configurations found after K iterations.⁽¹⁷⁾ If this number does not exceed a certain minimum, the program is halted.

3.2. Simulated annealing-based cost minimization approach to edge detection

The simulated annealing approach to edge detection is based on a stochastic iterative algorithm consisting of two phases. The first phase is the edge enhancement procedure described in Section 2.3.1. The second phase is the actual simulated annealing procedure.^(20,21) Starting from an initial edge configuration X_0 which can be generated randomly, the simulated annealing procedure generates a stationary Markov chain of edge configurations $\{X_k\}$. A cost function $F(X_k)$, which is to be minimized, is used to evaluate the edge configuration X_k . During the simulated annealing process, for a given present configuration X_k , the algorithm stochastically selects a transformation strategy and generates a candidate configuration Y_k based on the selected transformation strategy. The transformation strategy is selected from a transformation distribution $R: \{p_1, p_2, p_3, p_4, p_5\}$ where p_i is the probability of selection of the i th transformation strategy.

The transformation is performed in a 3×3 window centered at pixel site l selected either at random or in a raster scan fashion. In strategy 1, the center pixel at site l is complemented. In strategy 2, we complement the center pixel at site l as well as one of the eight neighboring pixels which is selected randomly. In strategy 3 and strategy 4, the transformations are performed based on the specific local edge structure ob-

served in the 3×3 window (Figs 12 and 13). In strategy 5, we assign values to the pixels in the 3×3 window at random. It is recommended⁽¹⁸⁾ that $p_1 = 200/1024$, $p_2 = 300/1024$, $p_3 = 200/1024$, $p_4 = 200/1024$, and $p_5 = 124/1024$.

The decision of selecting either configuration X_k or configuration Y_k as the next state X_{k+1} for the following iteration is not made deterministically via a straightforward comparison of the cost function values of the configurations X_k and Y_k , but stochastically, based on a sampling of the Boltzmann distribution via the Metropolis function.^(17,18) If configuration Y_k has a cost function value smaller than that of configuration X_k , configuration Y_k is selected as configuration X_{k+1} . If configuration Y_k has a cost function value larger than that of configuration X_k , configuration Y_k is selected as X_{k+1} with probability p determined by the Metropolis function $p = \exp(-\max(\Delta F(Y_k, X_k), 0)/T_k)$, where $\Delta F(Y_k, X_k) = F(Y_k) - F(X_k)$.

The set $\{T_k\}$ is called an annealing schedule, and is a sequence of strictly monotonically decreasing non-negative numbers such that $T_1 > T_2 > \dots > T_n$ and $\lim_{k \rightarrow \infty} T_k = 0$. The asymptotic convergence (i.e. in the limit $k \rightarrow \infty$) of simulated annealing is guaranteed for a logarithmic annealing schedule of the form $T_k = T_1 / (1 + \ln k)$ where $k \geq 1$. However, in practice, the logarithmic annealing schedule is far too slow and hence we have used a geometric annealing schedule of the form $T_k = (1 - \alpha)^k T_1$ where α is a positive real number close to zero. We have empirically found the values of $\alpha = 0.05$ and $T_1 \in [5, 100]$ to be well suited for our purpose. As $\lim_{k \rightarrow \infty} T_k = 0$, $p \rightarrow 0$, i.e. no better configuration Y_k can be found and X_k is always chosen as X_{k+1} . We refer to $\lim_{k \rightarrow \infty} T_k$ as T_∞ and deem it to be the freezing

temperature. The state corresponding T_∞ , which represents the best edge configuration, is said to be frozen. Practically speaking, when T_k is small enough,

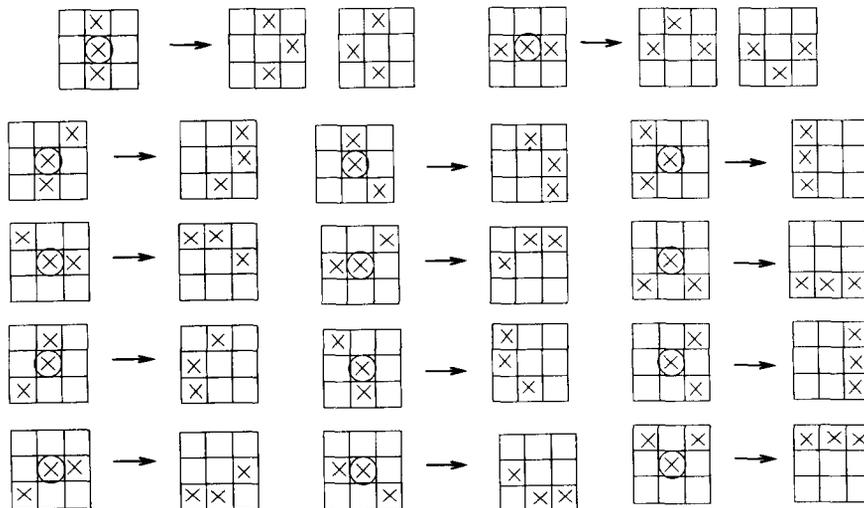


Fig. 12. Examples of transformation strategy 3.

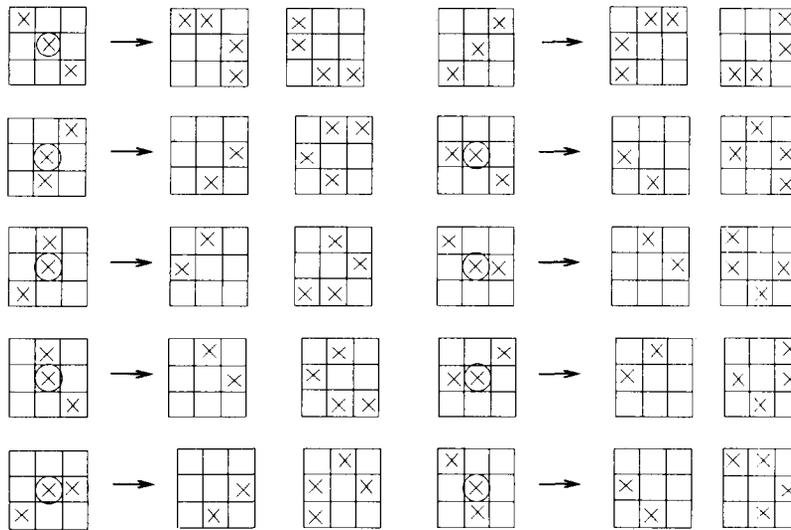


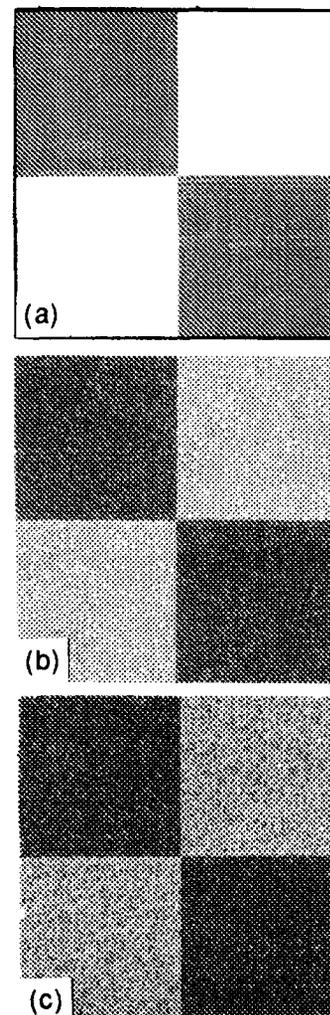
Fig. 13. Examples of transformation strategy 4.

we may consider the state to be *frozen*. Such a state represents a relatively low-cost edge configuration and the result in most cases is very satisfactory. For most practical purposes, when $T_k = 0.03$, we may assume that the state is frozen. This is the stopping criterion for the simulated annealing algorithm in our case. The cost factors in the cost function F , in the case of the simulated annealing approach, are computed in a manner identical to that in the case of the local search or the genetic algorithm approach.

4. EXPERIMENTAL RESULTS AND COMPARISON OF DIFFERENT APPROACHES

We have implemented and tested the local search-based, simulated annealing-based and genetic algorithm-based optimization techniques for edge detection on synthetic and real intensity images. All three approaches are based on different mathematical models of constrained optimization; however, they all cast the edge detection problem as one of minimizing the cost of an edge image. They use both the image data and the local edge structure information measured in the form of edge continuity, edge thinness and region dissimilarity. Consequently, the edges detected by all these three approaches are expected to be well localized, continuous and thin.

In this section, we compare both qualitatively and quantitatively the results obtained by these three approaches. In this paper we present experimental results for two intensity images: a synthetic *checker-board* image and a real *computer-terminal* image acquired using a CCD video camera. In order to test the robustness to noise of each of the edge detection techniques, we corrupt the checker-board and the computer-terminal images with additive Gaussian noise with zero mean and variances $\sigma^2 = 25$ and 100. In the case of the

Fig. 14. A checker-board image, (a) without noise, (b) with noise of variance $\sigma^2 = 25$, (c) with noise of variance $\sigma^2 = 100$.

synthetic image we compute the Pratt Figure of Merit for the resulting edge images and use it as a metric for comparing the performance of the three edge detection approaches. For the real image, the Pratt Figure of Merit could not be easily computed. Hence, for the computer-terminal image we evaluate and compare the final edge images by computing their costs using the cost function. Figure 14(a), (b) and (c) shows the checker-board image without noise, with noise of variance $\sigma^2 = 25$ and with noise of variance $\sigma^2 = 100$, respectively. Figure 15(a), (b) and (c) shows the com-

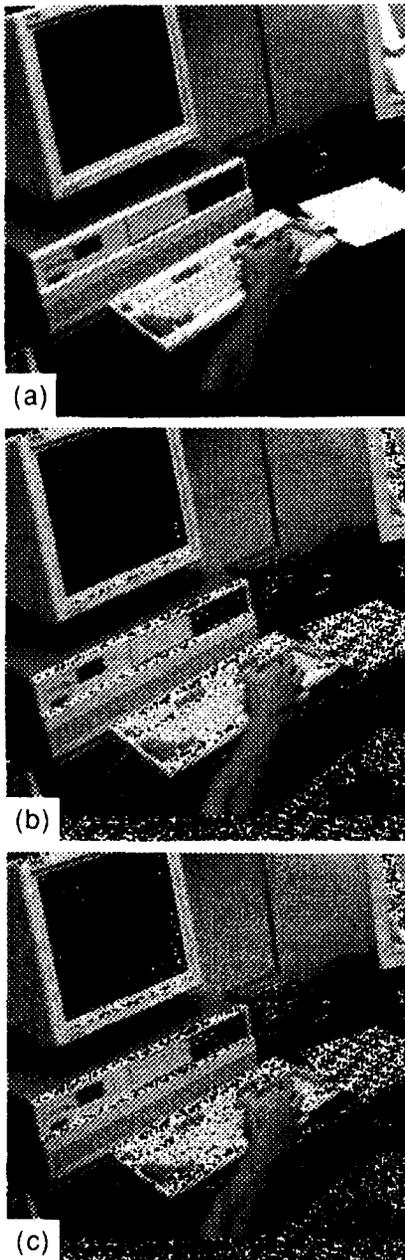


Fig. 15. A computer-terminal image, (a) without noise, (b) with noise of variance $\sigma^2 = 25$, (c) with noise of variance $\sigma^2 = 100$.

puter-terminal image without noise, with noise of variance $\sigma^2 = 25$ and with noise of variance $\sigma^2 = 100$, respectively.

In Fig. 16(a), (b) and (c), and Fig. 17(a), (b) and (c), we show the results of local search-based edge detection technique on the checker-board image and the computer-terminal image, respectively. In Fig. 18(a), (b) and (c) and Fig. 19(a), (b) and (c), we show the results of simulated annealing-based edge detection technique on the checker-board image and the computer-terminal image, respectively. In Fig. 20(a), (b) and (c) and Fig. 21(a), (b) and (c), we show the results of the integrated genetic algorithm on the checker-board image and the computer-terminal image, respectively. Although we experimented with different combinations of basic, advanced and meta-level genetic algorithm operators described in Section 2, the integrated genetic algorithm was found to be the best.

As mentioned previously, we use the Pratt Figure of Merit⁽¹⁷⁾ as a performance metric to quantitatively

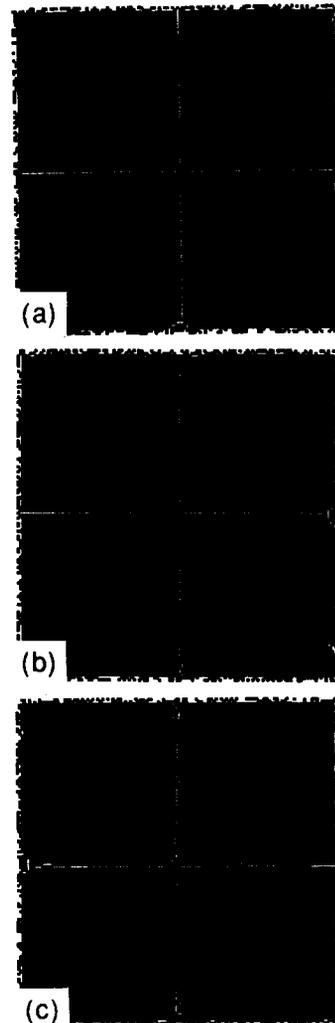


Fig. 16. Edge image using local search on the checker-board image, (a) without noise, (b) with noise of variance $\sigma^2 = 25$, (c) with noise of variance $\sigma^2 = 100$.

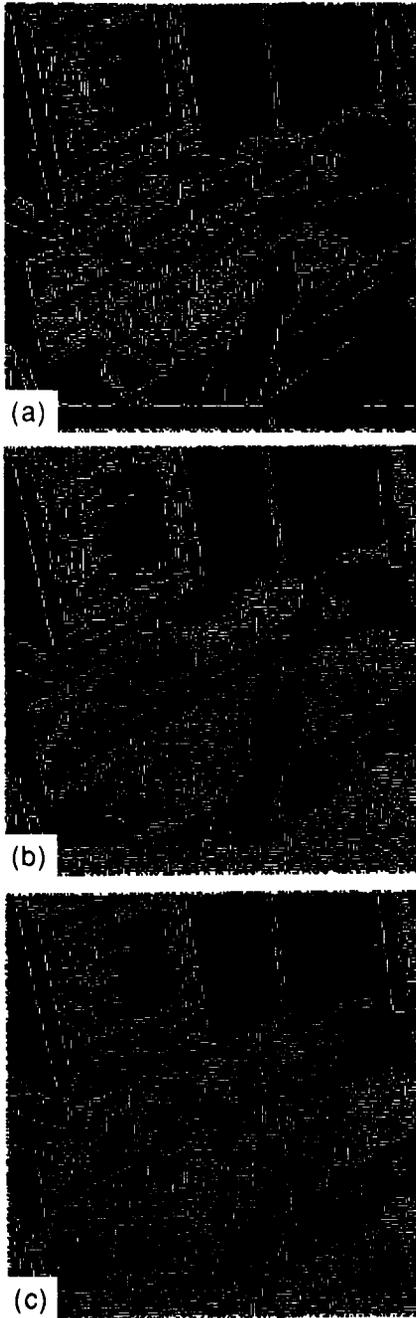


Fig. 17. Edge image using local search on the computer-terminal image, (a) without noise, (b) with noise of variance $\sigma^2 = 25$, (c) with noise of variance $\sigma^2 = 100$.

evaluate the edge images in the case of the synthetic checker-board image. The Pratt Figure of Merit (PFM) is a number in the range [0, 100] and is a good measure for evaluating edges in simple gray scale images in which the edge locations and the edge continuity are known *a priori*. The Pratt Figure of Merit is given by:

$$PFM = 100 \cdot \left(\frac{1}{I_M} \sum_{i=1}^{I_d} \frac{1}{1 + \alpha \delta_i^2} \right) \quad (5)$$

where $I_M = \max(I_d, I_i)$, where I_i is the number of ideal

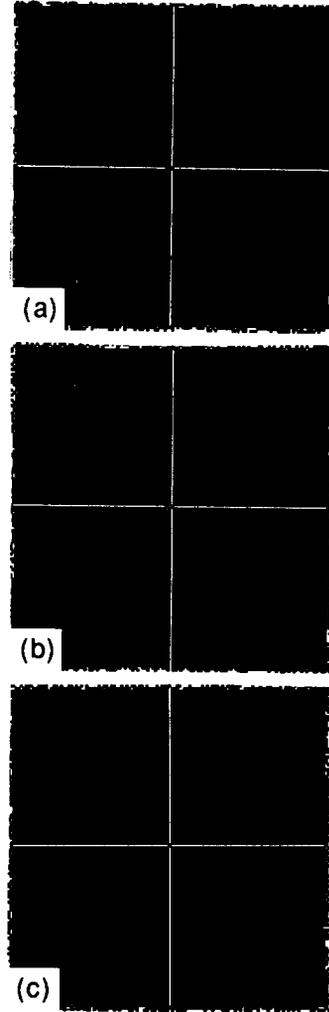


Fig. 18. Edge image using simulated annealing on the checker-board image, (a) without noise, (b) with noise of variance $\sigma^2 = 25$, (c) with noise of variance $\sigma^2 = 100$.

edge points, I_d is the number of detected edge points, δ_i is the deviation (i.e. distance) of the i th detected edge point from the ideal edge and α is a scaling factor. A higher value for the Pratt Figure of Merit denotes a better edge image. In the case of the real computer-terminal image we use the edge cost as computed using the cost function as a performance metric. In this case a lower value for the cost denotes a better edge image. We plot the performance metric as a function of the number of iterations of the local search or simulated annealing algorithm or the number of generations of the genetic algorithm. This permits us to compare the convergence rates of each of the various approaches as well as the quality of final edge image produced by each approach. We show the performance of the local search approach, simulated annealing approach, simple genetic algorithm approach and integrated genetic algorithm approach which incorporates advanced and meta-level operators for the synthetic checker-board image without noise [Fig. 22(a)] and

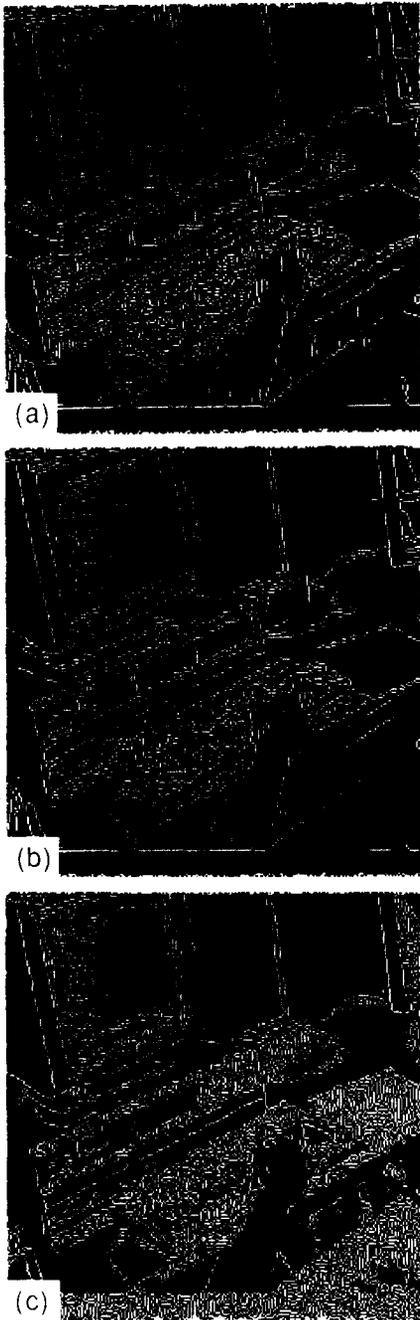


Fig. 19. Edge image using simulated annealing on the computer-terminal image, (a) without noise, (b) with noise of variance $\sigma^2 = 25$, (c) with noise of variance $\sigma^2 = 100$.

with noise of variance $\sigma^2 = 100$ [Fig. 22(b)] and for the natural computer-terminal image without noise [(Fig. 23(a)) and with noise of variance $\sigma^2 = 100$ [Fig. 23(b)].

On the whole, the simulated annealing algorithm and the integrated genetic algorithm were found to work the best on a general class of images. Due to the rather lengthy annealing schedule, the simulated annealing algorithm does not converge rapidly. However, since the simulated annealing algorithm generates

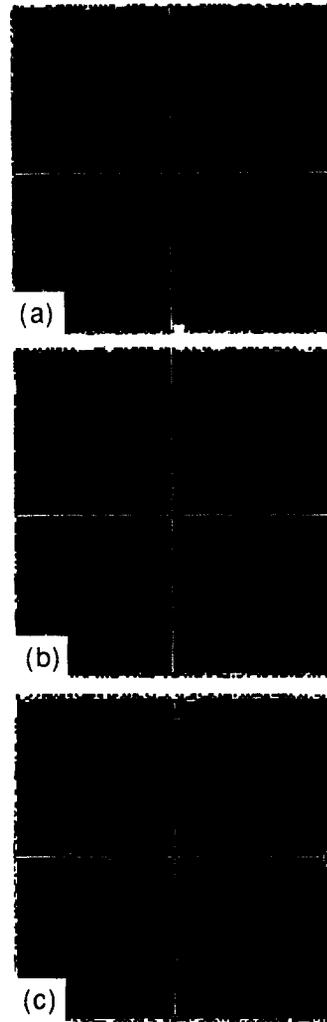


Fig. 20. Edge image using the integrated genetic algorithm on the checker-board image, (a) without noise, (b) with noise of variance $\sigma^2 = 25$, (c) with noise of variance $\sigma^2 = 100$.

the candidate states in a stochastic manner, it is less susceptible to being trapped in an undesired local minimum. We refer to the transformations (i.e. perturbations) carried out at a single temperature value in the simulated annealing algorithm as a single iteration. In our case, we have carried out $256 \times 256 = 65,536$ transformations for a single iteration (i.e. at a single temperature value). It takes about 40–60 iterations of the simulated annealing algorithm to get an edge image in which the contents of the image can be distinguished. In about 100–150 iterations, the edge image is an ideal one in which the edges are thin, continuous, and correctly localized.

Among the various combinations of genetic algorithm operators, the integrated genetic algorithm was found to perform the best. In our integrated genetic algorithm approach, we obtain after a few generations an edge image (corresponding to the best chromosome in the population) in which the quality of the edges detected is equivalent to that of the edges obtained after several

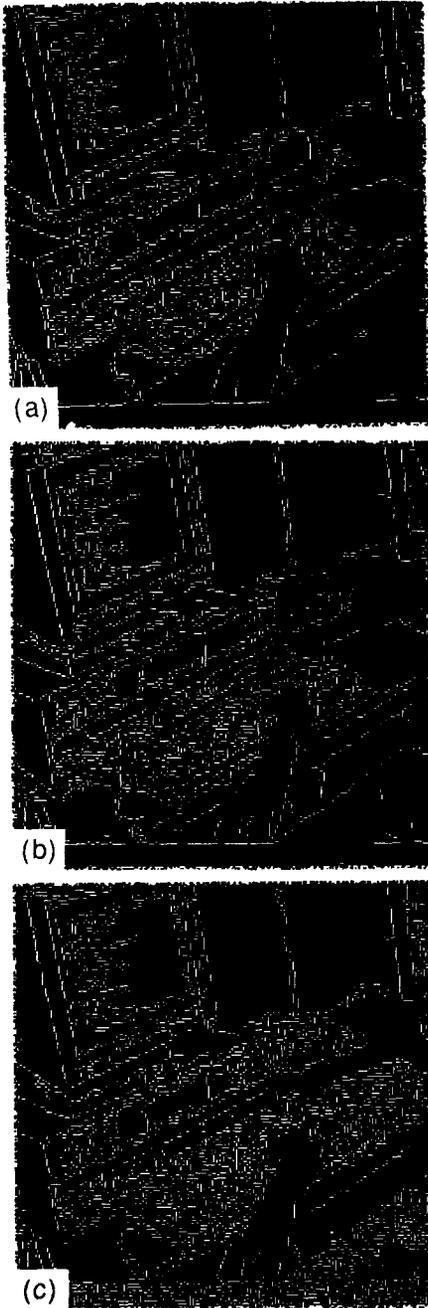


Fig. 21. Edge image using the integrated genetic algorithm on the computer-terminal image, (a) without noise, (b) with noise of variance $\sigma^2 = 25$, (c) with noise of variance $\sigma^2 = 100$.

iterations of the simulated annealing algorithm. That is to say, the solutions were seen to approach an approximate global minimum much faster in the integrated genetic algorithm as compared to the simulated annealing algorithm. The simple genetic algorithm with crossover and random mutation was found to perform very poorly in terms of convergence rate as is evident from the graphs in Fig. 22(a), (b) and Fig. 23(a), (b). This could be attributed to the fact that the population of 512 chromosomes represents a neg-

ligible fraction of the search space of $2^{65,536}$ possible edge configurations. The population size, in our case, is inadequate in being able to truly represent the diversity of the underlying search space. Since a simple genetic algorithm relies on the diversity within the existing population (i.e. richness in the gene pool) to search for better solutions, it is not surprising that the simple genetic algorithm performs so poorly in the context of our problem. The incorporation of the knowledge-augmented mutation operator and the meta-level genetic operators discussed in Section 2.6 were found to be necessary to hasten convergence. Each new generation of the genetic algorithm involved the application of the selection, crossover and mutation operations, and in the case of the integrated genetic algorithm, the application of the meta-level genetic operators as well, on an existing population of 512 chromosomes (i.e. edge images). Experimentally, the computational complexity of each new generation of the genetic algorithm was found to be roughly comparable to a single iteration of the simulated annealing algorithm with 65,536 transformations per iteration (i.e. for each temperature value).

The local search algorithm was found to have the fastest rate of convergence. Each iteration of the local search algorithm involved $256 \times 256 = 65,536$ perturbations carried out over the entire image in a faster-scan manner. Each iteration for the local search was found to be 15–18 times faster than a single iteration for the simulated annealing algorithm or a single generation of the genetic algorithm. For an ordinary noise-free synthetic checker-board image, an edge image in which the dominant edges could be distinguished was obtained in 2 iterations and after 5 iterations, an acceptable edge image was obtained for which the Pratt Figure of Merit was approximately 92. Between 5 and 10 iterations, the Pratt Figure of Merit was seen to improve very slightly and after 10 iterations there was no improvement at all. The maximum Pratt Figure of Merit obtained for a noisy synthetic checker-board image was 89 for $\sigma^2 = 25$ and 72 for $\sigma^2 = 100$. In the case of the natural computer-terminal image, the local search algorithm was found to display similar behavior. It took 2 iterations to get an edge image in which the dominant edges could be distinguished, (for example, the contour of computer terminal) and in 5 iterations, a generally acceptable edge image was obtained. Between 5 and 8 iterations, the cost of the edge image was found to change very little. After 8 iterations, the cost of edge image was found to be more or less stable. However, in places where the image data is more complicated or busy (for example, the rim of computer desk in the computer terminal image), the local search algorithm could not distinguish the edges very clearly.

On examining the cost of the resulting edge images, one can see that the lowest-cost edge image obtained using the local search algorithm is not as good the lowest-cost edge images obtained by the other two techniques. There is a sizable gap in terms of cost (or the Pratt Figure of Merit) between the lowest-cost edge

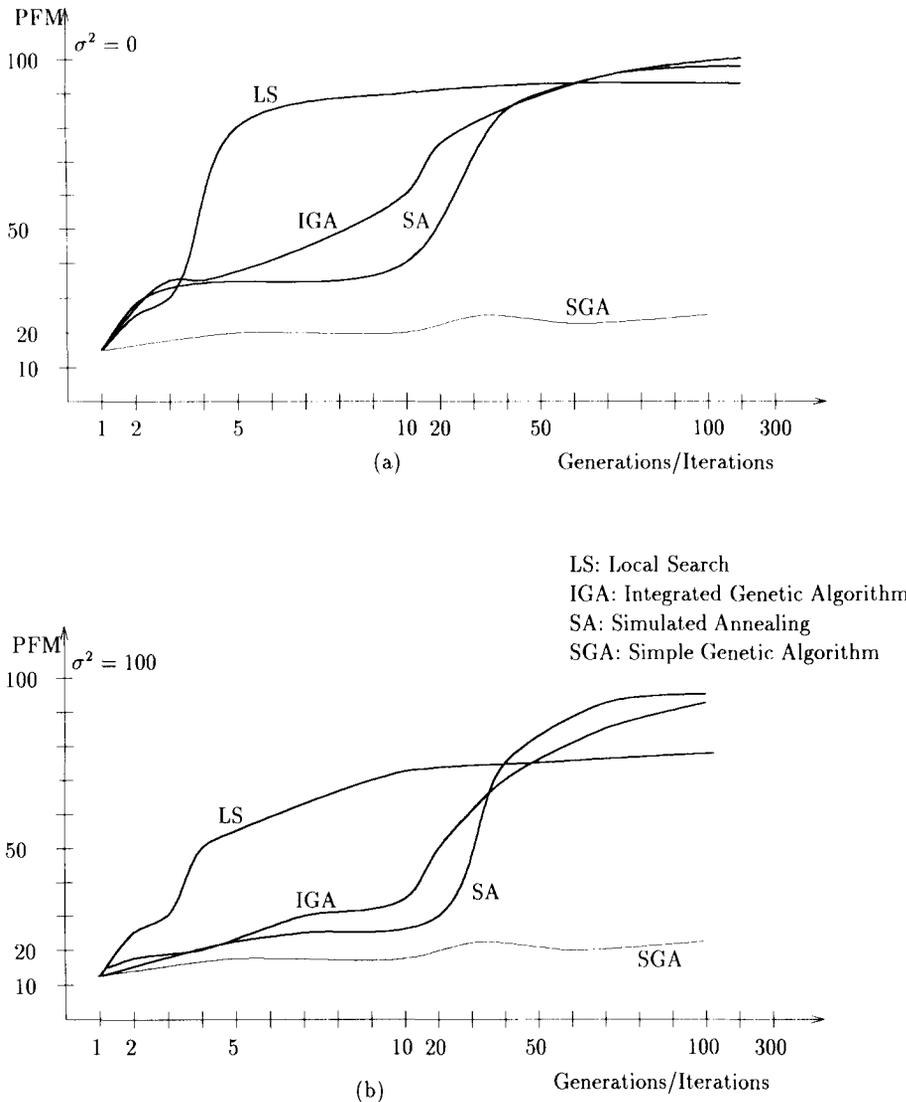


Fig. 22. Quantitative comparison based on the Pratt Figure of Merit for the various edge detection techniques on the synthetic checker-board image, (a) without noise, (b) with noise of variance $\sigma^2 = 100$.

image obtained by the local search algorithm and those obtained by the simulated annealing algorithm and the integrated genetic algorithm. This could be attributed to the tendency of the local search algorithm to get trapped in an undesired local minimum.

In dealing with noisy images, the simulated annealing algorithm and the integrated genetic algorithm were found to perform the best. For the synthetic checker-board image with noise with $\sigma^2 = 25$, the Pratt Figure of Merit was as high as 100 and for noise with $\sigma^2 = 100$, the Pratt Figure of Merit was over 90 for both simulated annealing and the integrated genetic algorithm. The graphs in Fig. 22(a), (b) and 15(a), (b) show the relative convergence rates in terms of either the Pratt Figure of Merit (checker-board image) or the cost (computer-terminal image) as well as the relative quality of the final edge images for all the three techniques.

5. CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

In this paper, we implemented three different cost minimization approaches to edge detection. The local search approach is simple and efficient and in some sense can be considered a special case of the other two approaches. However, the local search approach may not find a globally optimal solution but only one that is locally optimal. The simulated annealing algorithm and the integrated genetic algorithm approach were seen to produce the best results. We intend to extend our work described in this paper in the following areas:

1. Since most low-level image processing operations can be performed in parallel and genetic algorithm operators such as selection, crossover and mutation are inherently parallel in nature, we intend to parallelize our genetic algorithm-based approach to edge detection.

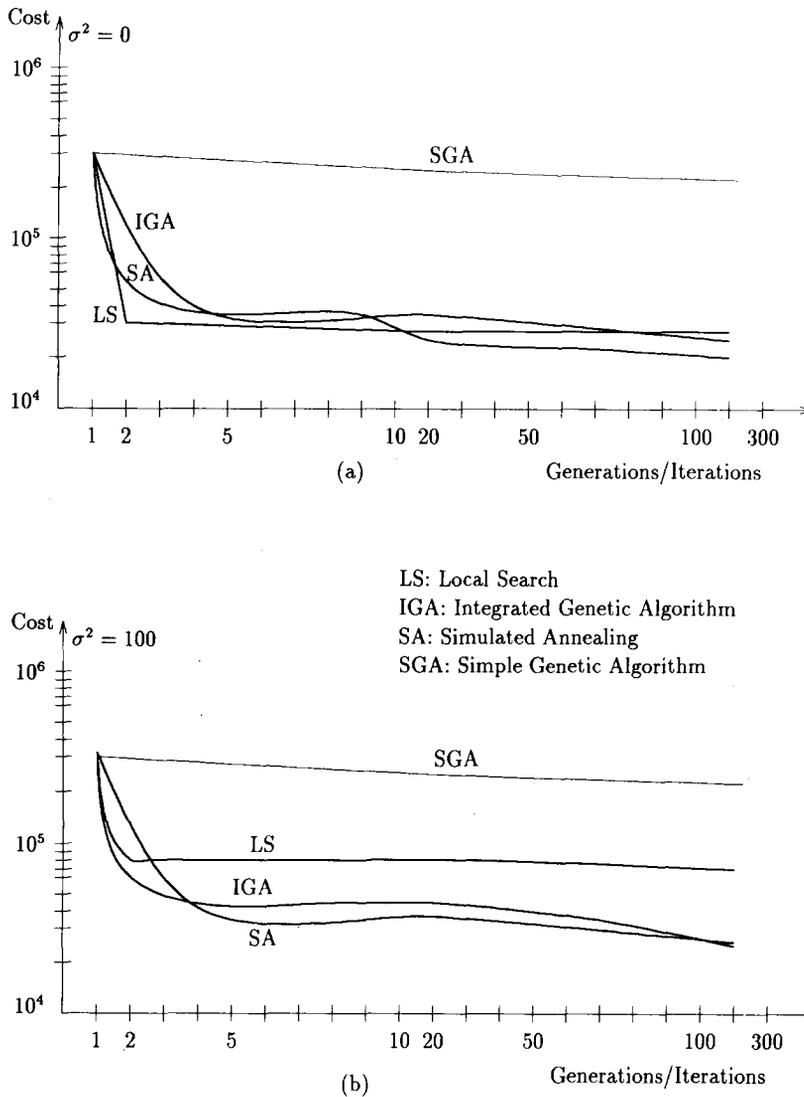


Fig. 23. Quantitative comparison based on the cost of the resulting edge image for the various edge detection techniques on the real computer-terminal image, (a) without noise, (b) with noise of variance $\sigma^2 = 100$.

2. We also intend to design more effective meta-level genetic algorithm operators so that more efficient search can be performed in the vast solution space. This paper has already demonstrated how meta-level genetic algorithm operators help to achieve faster convergence.

3. The chromosome representation used in this paper is an explicit representation of the edge image. Though simple and direct, this representation need not be the most efficient. Alternative (and hopefully better) representation schemes need to be investigated.

In conclusion, we feel that genetic algorithm-based optimization techniques have a major role to play in image processing and computer vision. With the use of suitable parallel hardware, genetic algorithms can be used to design robust processing techniques for most vision applications.

6. SUMMARY

In this paper we have presented a genetic algorithm-based cost minimization technique for edge detection. We have formulated the problem of edge detection as one of choosing a minimum cost edge configuration. The edge configurations are viewed as two-dimensional chromosomes with fitness values inversely proportional to their costs. The design of the crossover and the mutation operators in the context of the two-dimensional chromosomal representation is described. We have designed a knowledge-augmented mutation operator which exploits knowledge of the local edge structure in the edge image. The knowledge-augmented mutation operator is shown to result in rapid convergence. The incorporation of meta-level operators and strategies such as the elitism strategy, the engineered conditioning operator and adaptation of mutation

and crossover rates in the context of edge detection are discussed and are shown to improve the convergence rate. The genetic algorithm with various combinations of meta-level operators is tested on synthetic and natural images. The performance of the genetic algorithm-based cost minimization technique is compared both qualitatively and quantitatively with local search-based and simulated annealing-based cost minimization approaches. The genetic algorithm-based technique is shown to perform very well in terms of robustness to noise, rate of convergence and quality of the final edge image.

Acknowledgements—This research was partially supported by a Faculty Research Grant (1991) to Dr Suchendra M. Bhandarkar by the University of Georgia Research Foundation Inc., Athens, GA, 30602. The authors wish to thank the anonymous reviewers for their insightful comments and helpful suggestions on the earlier version of this paper.

REFERENCES

1. A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, Vols. 1 and 2. Academic Press, New York (1982).
2. V. Torre and T. A. Poggio, On edge detection, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-8**, 147–163 (1986).
3. T. Peli and D. Malah, A study of edge detection algorithms, *Comput. Graphics Image Process.* **20**, 1–21 (1982).
4. F. M. Dickey and K. S. Shanmugam, Optimum edge detection filter, *Appl. Optics* **16**, 145–148 (1977).
5. J. Canny, A computational approach to edge detection, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-8**, 679–698 (1986).
6. D. Lee and G. W. Wasilkowski, Discontinuity detection and thresholding—a stochastic approach, in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 208–214 (June 1991).
7. J. Shen and S. Castan, An optimal linear operator for step edge detection, *CVGIP: Graph. Models Image Process.* **54**, 112–133 (1992).
8. D. Marr and E. Hildreth, Theory of edge detection, *Proc. R. Soc. London* **B207**, 187–217 (1980).
9. M. H. Chen, D. Lee and T. Pavlidis, Residual analysis for feature detection, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-13**, 30–40 (1991).
10. R. M. Haralick, Digital step edges from zero crossing of second directional derivatives, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-6**, 58–68 (1984).
11. V. Nalwa and T. Binford, On detecting edges, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-8**, 699–714 (1986).
12. P. Perona and J. Malik, Scale space and edge detection using anisotropic diffusion, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-12**, 629–639 (1990).
13. P. Saint-Marc, J. Chen and G. Medioni, Adaptive smoothing: a general tool for early vision, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-13**, 514–529 (1991).
14. G. P. Ashkar and J. W. Modestino, The contour extraction problem with biomedical applications, *Comput. Graphics Image Process.* **7**, 331–355 (1978).
15. U. Montanari, On the optimal detection of curves in noisy pictures, *Commun. ACM* **14**, 335–345 (1971).
16. A. Martelli, An application of heuristic search to edge and contour detection, *Commun. ACM* **19**, 73–83 (1976).
17. H. L. Tan, S. B. Gelfand and E. J. Delp, A comparative cost function approach to edge detection, *IEEE Trans. Systems Man Cybern.* **19**, 1337–1349 (1989).
18. H. L. Tan, S. B. Gelfand and E. J. Delp, A cost minimization approach to edge detection using simulated annealing, *IEEE Trans. Pattern Anal. Mach. Intell.* **14**, 3–18 (1991).
19. S. T. Acton and A. C. Bovik, Anisotropic edge detection using mean field annealing, *Proc. IEEE Intl Conf. Acoustics, Speech and Signal Processing*, Vol. II, pp. 393–396 (1992).
20. S. Geman and D. Geman, Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-6**, 721–741 (1984).
21. S. Kirkpatrick, C. D. Gelatt Jr and M. P. Vecchi, Optimization by simulated annealing, *Science*, **220**, 671–680 (1983).
22. L. Davis and M. Steenstrup, Genetic Algorithms and Simulated annealing: an overview, in *Genetic Algorithms and Simulated Annealing* (edited by L. Davis). Morgan Koffman, Los Altos, California (1987).
23. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Co., Reading, MA (1989).
24. J. Marroquin, S. Mitter and T. Poggio, Probabilistic solution of ill-posed problems in computational vision, *J. Am. Statist. Ass.* **82**, 76–89 (1987).
25. L. Davis, A genetic algorithms tutorial, in *Handbook of Genetic Algorithms* (edited by L. Davis). Van Nostrand Reinhold, New York (1991).
26. W. D. Potter, J. A. Miller, B. E. Tonn, R. V. Gandham and C. N. Lapena, Improving the reliability of heuristic multiple fault diagnosis via the EC-based genetic algorithm, *J. Appl. Intell.* **2**, 5–19 (1992).

About the Author—SUCHENDRA M. BHANDARKAR received his B.Tech. in Electrical Engineering from the Indian Institute of Technology, Bombay, India, in 1983, and his M.S. and Ph.D. in Computer Engineering from Syracuse University, Syracuse, New York, in 1985 and 1989, respectively. He was a Syracuse University Fellow for the academic years 1986–1987 and 1987–1988. He is currently an Assistant Professor in the Department of Computer Science at the University of Georgia, Athens, Georgia. He is a member of the Institute of Electrical and Electronic Engineers (IEEE), American Association of Artificial Intelligence (AAAI), Association of Computing Machinery (ACM) and Society of Photo-Optical and Instrumentation Engineers (SPIE). He is a coauthor of the book *Object Recognition from Range Images* (Springer, Berlin 1992). He is also a member of the honor societies Phi Kappa Phi and Phi Beta Delta. His research interests include computer vision, pattern recognition, image processing, artificial intelligence and parallel algorithms and architectures for computer vision and pattern recognition. He has over 40 published research articles in these areas.

About the Author—YIQING ZHANG received his B.S. in Electrical Engineering from the Shanghai University of Science and Technology, Shanghai, P.R. China, in 1986 and his M.S. in Artificial Intelligence from the University of Georgia, Athens, Georgia, in 1992. He was an Assistant Engineer for Optical Fiber Communi-

cations at the Transmission Lines Research Institute, Shanghai, P.R. China, from 1986 to 1990. He is currently a Software Engineer at Automated Design Systems, Atlanta, Georgia. His research interests are in computer vision, image processing, graphical user interface design and database systems.

About the Author—WALTER D. POTTER is an Associate Professor of Computer Science and the Graduate Coordinator for the Artificial Intelligence Programs at the University of Georgia in Athens, Georgia. His research interests include genetic algorithms, multiple fault diagnosis, expert database systems, hyper-semantic data modeling, and advanced information system design. He has worked, as a research fellow, with the Intelligent Cognitive Systems Group at Oak Ridge National Laboratory in Oak Ridge, Tennessee and with the Army Research Institute at Fort Gordon, Georgia. More recently, Dr Potter has been working with the Advanced Technologies Group at the Westing house Savannah River Company, Dr Potter received the B.S. degree (1974) from The University of Tennessee in Business Administration and the M.S. (1981) and Ph.D. (1987) from The University of South Carolina, both in Computer Science. While at South Carolina, he was the first recipient of the H.W. Close Graduate Fellowship in Computer Science. He is a member of the AAAI, ACM, the IEEE Computer Society, and has served as the Secretary of the IEEE Computer Society technical committee on Data Engineering.