Genetic Algorithms for Object Localization in a Complex Scene

Daniel L. Swets
A714 Wells Hall
Michigan State University
East Lansing, Michigan 48824
(517) 355-9319
FAX (517) 336-1061
swetsd@cps.msu.edu

Bill Punch
A394 Engineering Building
Michigan State University
East Lansing, Michigan 48824
(517) 353-3541
FAX (517) 336-1061
punch@cps.msu.edu

Abstract

The localization of objects of interest in a scene is one of the major problems facing the computer vision module. Genetic algorithms are a search technique for dealing with a very large search space, such as the one encountered in image segmentation. This work describes a technique for using genetic algorithms for object localization in a complex scene. A brief review of genetic algorithms is given, followed by the specifics of using a genetic algorithm for this image segmentation application. The results from several experiments show that this approach is a viable method for image segmentation.

Keywords

Vision, Image Segmentation, Learning, Genetic Algorithms.

Declaration

This paper has not already been accepted by and is not currently under review for a journal or another conference. Nor will it be submitted for such during IJCAI's review period.

1 Introduction

In order to provide machines the ability to interact in complex, real-world environments, sensory data must be presented to the machine [13]. One such module dealing with sensory input is the visual data processing module, also known as the computer vision module. A central task of this computer vision module is to recognize objects from images of the environment [34].

One of the major problems facing the computer vision module is the localization of objects of interest. For a navigation system, this may involve the localization of landmarks or threats; for a robotic system, this may mean the discovery of objects to manipulate; for an object recognition system, this will include finding objects in the input image to attempt to recognize. An object recognition system may be functioning in a world where the objects being sought are known and well-defined, but their relative position and size in the images being received from the sensor are not. In addition, the images being presented to the recognition system may well be cluttered with other, irrelevant objects that the system properly wishes to ignore.

Finding the objects of interest is often called *segmenting* the image. This segmentation is not necessarily an easier problem than recognition itself [34], and many efforts have been made to produce a robust image segmentation system with varying amounts of success. For example, thresholding is a common technique in image segmentation. This technique uses an image function

$$f_t(x,y) = \begin{cases} b_0 & \text{if} f(x,y) < t \\ b_1 & \text{if} f(x,y) \geq t \end{cases}$$

to determine whether an image pixel belongs in the background or foreground class [30]. If t is constant throughout the image, the technique is called *global thresholding*; if the image is divided into several subimages and a different t is used on each subimage, the method is called *local thresholding*. The following is a brief treatment of various reported thresholding techniques for image segmentation.

The *p-tile* method [9] assumes that images consist of dark objects on light backgrounds. The percentage p of the object in the image is known; t is chosen so that at least $(100 - p)\%$ of the pixels fall into the "foreground" category. Images which have disjoint object and background sets will have a bimodal histogram; a technique called the *mode method* [23] finds t in the valley of the histogram. But since these valleys may not be found, the *histogram concavity* [28] method defines the threshold at the concavities of the gray-level histograms. For this method, the gray-level histogram is considered as a two-dimensional object; the convex hull of this "object" is computed, and the concavities of the histogram are found by looking at the difference between this convex hull and the actual histogram. Ostu's method [22] uses discriminant analysis to partition the pixels into background and foreground classes.

Several entropic methods have been proposed [24] [25] [16] [15], in which an optimal threshold is computed by applying the principles of information theory. Moment-preserving methods (e.g., [32]) seek to keep the moments of the images constant.

These types of global thresholding methods depend solely on the first-order gray level statistics of the images. Other methods work with second-order gray-level statistics (e.g., [14] [18] [8]), relaxation methods (e.g., [29] [36] [3]), and combinations of threshold operators (e.g., [2] [35]).

Often, however, the global thresholding techniques prove to be insufficient for many types of images. For these, local thresholding methods (e.g., [6] [7] [21] [10] [27]) or a multi-thresholding method (e.g., [5] [33] [19]) have been proposed.

Genetic Algorithms have also recently been making their way into the computer vision literature. To date, GAs have been used for parameter tuning (e.g., [17] [1] [4]) and feature extraction [31]; the actual image processing tasks have been performed using other methods.

This work proposes the use of genetic algorithms to locate an object of interest in a complex scene. As such, the complete segmentation problem is given to the genetic algorithm to solve.

Before the approach is laid out, a brief review of genetic algorithms is given. Some issues that arose during the experimentation are then addressed, and results are given to show the functionality of this type of approach. Finally, some concluding remarks are given.

2 Brief Review of Genetic Algorithms

This section gives a brief overview of genetic algorithm fundamentals. Goldberg [11] gives a wonderful introduction to genetic algorithms, and the reader is referred to this source for further information.

A genetic algorithm is an optimization technique that operates on a *population* of individual solutions. Each individual solution, also called a *string* in the population, represents a proposed solution to the problem being solved. The theories of natural selection are applied to this population, and subsequent generations of the population are obtained by applying selection, reproduction, and mutation operators (among possible others) to the population. With these operators, the population of solutions is gently pushed towards a good–and hopefully the optimum–solution to the problem.

The GA designer provides a fitness function to evaluate the fitness of each individual solution; this fitness function is used to propagate "good" individuals into the next generation. Some set of these fit individuals are chosen for a *crossover* operation, which recombines the strings of the parents into new children, trying to build up healthier strings in the process. The mutation

operator randomly alters some element of an individual in order to further enhance the population.

Crossover is the name given to a simple reproduction operation because of the way that the parent strings are recombined. Usually one or two common points in a pair of parents are chosen at random. For a one-point crossover, the portion of the parent strings to the right of the crossover point can be called the *crossover area*; for a two-point crossover, the area between the points is the crossover area. One child is formed by taking the crossover area from one parent and the non-crossover area from the other parent–"crossing-over" the parent strings. The other child is formed by reversing the process. The theory behind doing this operation is called the *schema theorem* [11]; short, low-order portions of strings contributing to fit individuals are thus created.

There are several issues of paramount importance in the design of a genetic algorithm. First and foremost, the fitness function of the system must be designed appropriately to select good individuals, since it is the only window that the population has to the outside world. Furthermore, the representation of the strings making up the population of individuals must be selected with great care. The way that strings recombine and are propagated into the next generation is inherently linked to the way they are represented and interpreted. Finally, the selection of appropriate genetic operators for the system is fundamental to obtaining a good solution in a reasonable time.

3 Approach

The system introduced in this work proceeds in two phases: a training phase and a testing phase. The training phase has a set of labeled images, each containing a single object of interest such that the object shown in the image has a standard size, position, and orientation. The image dimensions for identical objects in the training set of images is also fixed. During the training phase, a simple "object mean" image is generated. Each pixel in the object mean image represents the average pixel value of all the training images for that object at that pixel position.

The test phase has a set of arbitrary images. During the test phase, no constraints on the input images are made; instead, a subimage is extracted from the test image and projected to the standard image dimensions used for each object of interest during the training phase. A simple image distance is computed between the normalized subimage extracted from the test image and the learned object mean images. The subimage with the smallest distance from a particular object mean is taken to be the best subimage to be fed to the object recognition system. This subimage extraction and size normalization will handle differences in scale and position in the test images.

Occlusion can be handled by choosing a set of vertices within the subimage

being extracted, drawing a polygon connecting these vertices, and masking the exterior of the polygon. Alternatively, the subimage extraction could be done using a set of vertices of a polygon directly, but then the size and position of the polygon in the standard-sized image would have to be determined. Implementation was more straightforward using a polygon within an extracted rectangle.

3.1 Genetic Algorithm Parameters

The task of locating a particular object of interest in a complex scene is quite simple when cast in the framework of genetic algorithms. The brute-force method for finding an object in a scene is to examine all positions and sizes, with varying degrees of occlusion of the objects, to determine whether the extracted subimage matches a rough notion of what is being sought. This method is immediately dismissed as far to computationally expensive to achieve. The use of the genetic algorithm methodology, however, can raise the brute-force setup to an elegant solution to this complex problem. Since the genetic algorithm approach does well in very large search spaces by working only with a sample of the available population, the computational limitation of the brute-force method using full search space enumeration does not apply. So the parameters of the brute-force method for object localization are examined in order to formulate this problem in terms applicable to genetic algorithms.

Let (u_x, u_y) and (b_x, b_y) represent the upper left and lower right corner of the object of interest, respectively. Also let $p_1, p_2, \ldots, p_n$ be a list of n vertex points of the n-sided polygon mask. Then for an input image whose dimensions are $h \times w$,

$$\begin{aligned} u_x &\in [0, w] \\ u_y &\in [0, h]. \end{aligned}$$

Once the point (u_x, u_y) has been determined,

$$\begin{aligned} b_x &\in (u_x, w] \\ b_y &\in (u_y, h]\,. \end{aligned}$$

When $(u_x, u_y), (b_x, b_y)$ have been specified, the subimage to be extracted of dimensions $h_s = b_y - u_y \times w_s = b_x - u_x$ has been determined. Then the points of the occlusion mask, $p_i = (p_{i,x}, p_{i,y}), i = 1, 2, \ldots, n$, must satisfy

$$\begin{aligned} p_{i,x} &\in [0, w_s] \\ p_{i,y} &\in [0, h_s]\,. \end{aligned}$$

The coordinates of the relevant points can be encoded as a bit string of b-bit numbers, where b can hold the larger of (h, w), concatenated together. However, analysis done by Punch *et al.* [26] on a similar encoding problem showed that little change occurred during crossover with such a representation, resulting in slower evolution to an optimal fitness value. Their approach was to provide a "relative encoding" of the coordinates found on the strings, where the first coordinate on the string is an absolute position and all the following coordinates are relative offsets from the previous point on the string. That way, crossover at a particular point in the string will result in a repositioning of all the points found after the crossover position, giving a much larger change in the overall fitness. Their recommendations for relative encoding were used for encoding the coordinates.

The evaluation function in this system extracts the coordinates from the bit string, performs the subimage extraction and polygon masking, and computes a distance measure from the object mean of the current object of interest. The distance measure between the test image x and the object mean μ is given by

$$d(x, \mu) = e^{-\frac{1}{2}\frac{(x-\mu)^T(x-\mu)}{\sigma}}$$

where x, μ are treated as vectors in row-raster fashion and σ is some user-specified system parameter to allow for the differences among pixel gray-level values. $d(x, \mu)$ always falls in the range of $[0, 1]$, and is maximized when the distance between the test image x and the object mean under consideration μ is minimized.

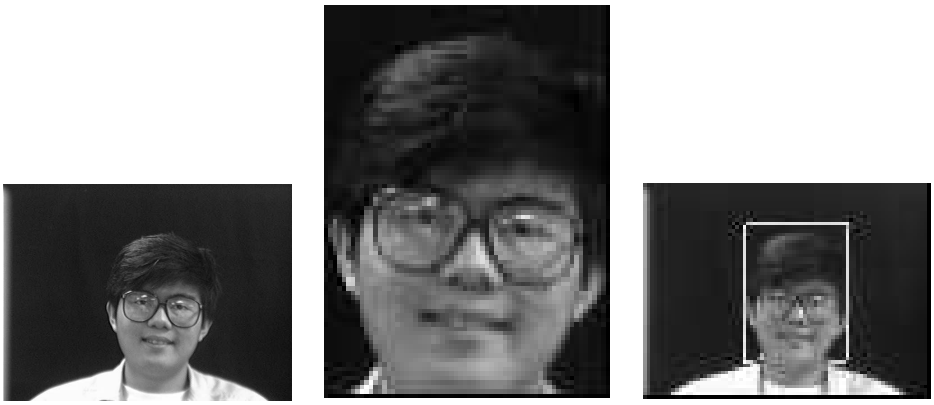
A penalty function for illegal point combinations will ensure that these invalid solutions receive little weight in the population of possible strings. The penalty function is addressed in section 4.1.2.

4 Experiments

The experiments were performed using the Genetic Algorithm Optimized for Portability and Parallelism (GALOPPS) system developed at the Michigan State University Intelligent Systems Laboratory. This system is available via the world-wide web at http://isl.cps.msu.edu/GA/ [12].

4.1 Verification of the method

The first set of experiments centered around the verification of the proposed method. The training image set consisted of a single, hand-segmented image of a face as shown in figure 1(b). For the test phase, the original image was given to the genetic algorithm to segment. During this technique verification



(a) Original image (b) Hand-segmented face for training (c) GA-segmented face image

Figure 1: Experiment 1, original image, hand-segmented object of interest for the training phase, and the GA-segmented image found during the test phase. The GA-segmented image was found after only 229 generations with a population size of 100.

stage, several important issues were identified and addressed, including various representations, different evaluation function parameters, and pertinent penalty functions.

4.1.1 Representations

The original design of this system had each chromosome laid out as shown in figure 2. Each field was fully used; the *Class* field indicated to which class mean the extracted subimage should be compared.

Several problems were uncovered in the use of this representation and interpretation. First, the training set of images had a particular aspect ratio for the

bits	0-7	8-15	16-23	24-39	40-56	57-end
Meaning	Class	UL x	UL y	BR x	BR y	Polygon list
Interpretation	Class mean to test [0-255]	x, y-Position [-128, 128]		x, y offsets [-128, 128]		x, y offsets [-128, 128]

Figure 2: Layout of chromosomes in original design. Here, UL is the upper-left corner of the bounding box, and BR is the bottom-right corner. Each polygon list entry consists of two 8-bit points for the x and y coordinate of the point, respectively.

$x : y$ dimensions. When allowing all four of $(UL_x, UL_y), (BR_x, BR_y)$ to evolve on the chromosome, mostly useless subimages were generated. Though any sized subimage could be warped to the dimensions given in the training set, such subimages do not appear similar to the learned class mean image unless the aspect ratios for these images match. So a penalty function was generated to penalize those individuals for whom the aspect ratio was incorrect; nearly every chromosome had a stiff penalty applied to it, and the results were of little use. Additionally, laying the chromosome out in this manner results in much more work than is required if the aspect ratio of the training set is to be honored. For the training set, the aspect ratio is given by

$$a = \frac{h_t}{w_t},$$

where h_t and w_t are the height and width of the training images, respectively. Then in the test phase, for a given (UL_x, UL_y) and BR_x, an appropriate BR_y can be computed by $BR_y = a(BR_x - UL_x) + UL_y$. Thus the complexity of the problem is reduced without loss of generality.

Another problem encountered was that many of the bounding box points represented in a population did not intersect the original image being segmented at all. This was due to the fact that the original image did not necessarily have dimensions conforming to the powers of 2, as the fields of the chromosomes were interpreted. Thus, many of the values in the fields were useless and required unnecessary extra work to filter out.

These and other problems were addressed with the improved chromosome layout shown in figure 3. The values were extracted from the chromosomes

bits	0-7	8-15	16-23	24-39	40-end
Meaning	Class	UL x	UL y	BR x	Polygon list
Interpretation	Class mean to test [0-numclasses]	x, y-offset from image center [-128, 128]		x offset [0, 128]	x, y offsets [0, Sub Width] or [0, Sub Height]

Figure 3: Improved layout of chromosomes.

using the minimum number of bits in each field; the remaining bits were treated as padding bits.

4.1.2 Penalty and Evaluation Functions

Because the representation used allows for some invalid combinations of parameters, a penalty function was used to reduce the contributions of these invalid combinations without losing their primordial material likely to help

form better individuals in future generations. A myriad of penalty functions were attempted, but the following one provided acceptable results without completely throwing out the invalid chromosomes. For every field of the chromosome, once the proper number of bits to be use was ascertained, a check was made to determine whether the extracted field lay outside its valid range. As the fields are extracted, a penalty value is maintained to indicate the penalty that a particular chromosome is given. This penalty value is incremented by the square of the distance from a valid field value as the value is extracted from the chromosome. For the case of the *Class mean* field, when an invalid field value was encountered, the class used for comparison was taken modulo the number of classes available. For the coordinate values, the nearest valid value was taken as the value to use when the chromosome held an invalid value in the field.

So the evaluation function took the following form:

$$d(x,\mu) = 1 + e^{-\frac{1}{2}\frac{(x-\mu)^T(x-\mu)}{\sigma}} - \left(1 - e^{\text{penalty}^{-2}}\right)$$

This function always gives a value in the range $[0, 2]$.

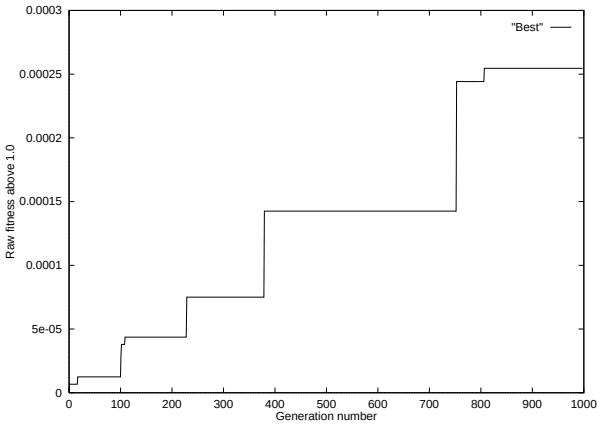
4.1.3 Results of the Method Verification Experiments

The results of running the genetic algorithm are given in figure 1(c). A population size of 100 was used for a maximum of 1000 generations; only about 300 generations proved to be useful, however. Figure 4 (a) shows the convergence measures for this experiment. The parameters for this experiment are shown in figure 5.

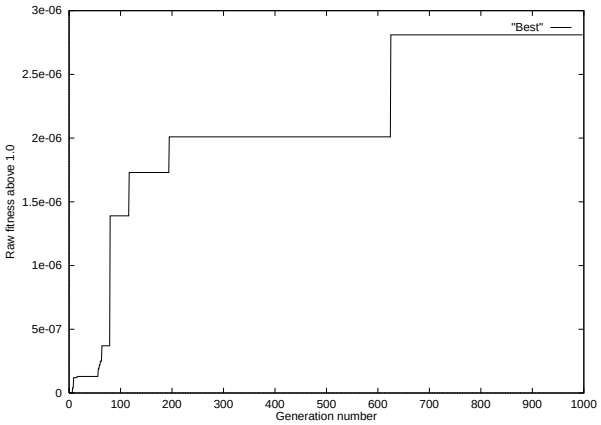
The final coordinates found by the genetic algorithm were only one pixel different from the hand-segmented image. The relatively low evaluation function measures were due to the difference in resolution between the hand-segmented image and the image used by the genetic algorithm. The training image came from an image resolution of 240×328 scaled to 88×64 pixels; the one used by the genetic algorithm came from an image resolution of 14×55 scaled to 88×64 pixels. So one pixel in the image used by the genetic algorithm represented the average pixel value from a block of approximately 4×4 pixels in the original class mean image.

4.2 Method Generalization Experiment

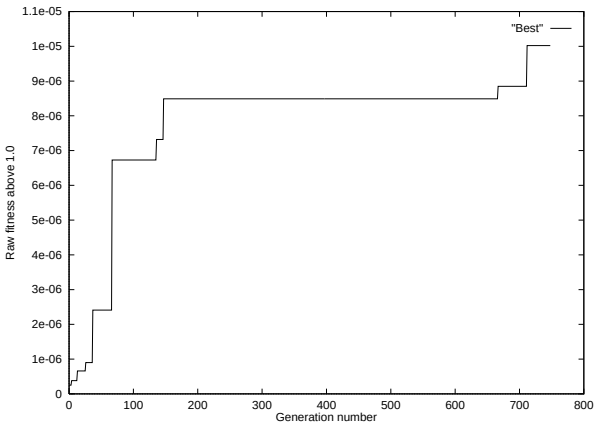
Following the success of the technique verification experiment, the genetic algorithm was tested on a slightly more difficult problem. For this experiment, a tile of ten distinct faces was created as the image for the genetic algorithm to segment, using the same training set as was used in experiment 1.



(a) Experiment 1: method verification



(b) Experiment 2: method generalization



(c) Experiment 3: natural scene experiment

Figure 4: Convergence graphs for the raw fitness of the best individual for each generation.

P_c	0.60
P_m	0.0625
l	32 bits
Scale	2.0
Population size	100
Maximum generations	1000

Figure 5: Parameters used for experiment 1. P_c is the probability for crossover; P_m is the probability for mutation; l is the length of the chromosome; Scale is the scaling factor used for the ratio of the best : mean individual. The mutation rate was set so that on average, two bits will be flipped per individual in each generation.

The resulting segmentation and the GA parameters used are shown in figure 6. The convergence measures for this experiment are shown in figure 4 (b).

(a) Best segmentation found.

P_c	0.60
P_m	0.0625
l	32 bits
Scale	1.5
Pop. size	1000
Max gens	1000

(b) Parameters used for this experiment.

Figure 6: Face tile segmentation using the Genetic Algorithm.

Though a face different from the one trained was found as the best solution, a valid segmentation of this image for future recognition tasks was achieved.

4.3 Natural scene experiment

This experiment allowed the genetic algorithm operate on a natural scene to try to locate a face in a crowd. The crowd image and the segmentation found by the genetic algorithm are shown in figure 7. The convergence measures for this crowd segmentation are shown in figure 4 (c). As this experiment shows, the genetic algorithm is capable of finding a valid segmentation in a complex, natural scene.

(a) Best segmentation found by the GA.

P_c	0.60
P_m	0.0625
l	44 bits
Scale	1.5
Pop. size	1000
Max gens	750

(b) Parameters used for this experiment.

Figure 7: Crowd segmentation using the Genetic Algorithm.

5 Conclusions and Future Work

This work proposes a technique for image segmentation using genetic algorithms on the images directly. The experiments show that the genetic algorithm performs well in finding areas of interest even in a complex, real-world scene. Genetic Algorithms are adaptive to their environment, and as such this type of a method is appealing to the vision community who must often work in a changing environment.

In order for the described procedure to be more generally applicable, however, several improvements should be considered. Grey coding the fields would greatly improve the mutation operation by ensuring that flipping a bit only changes a field's value by one value. The way this system was implemented, a bit changed could cause a large change in the value. DeJong-style crowding [11] could be used to find multiple objects of interest in an image. Since there may be many objects in a natural scene, this segmentation into multiple objects is necessary to make the technique useful for real-world images. Finally, if the class mean being considered is already on the chromosome, object recognition could be combined with this segmentation step so that the segmentation that best matches the object mean specified on the chromosome would not only indicate a segmentation for possible further verification, but also an indication of the object class.

Timing improvements could be made by utilizing the implicit parallelization of multiple independent generations evolving at the same time [20]. Different areas of the search space are explored in each of these populations; oc-

casional swapping of individuals among populations could be done to improve diversity and gently push all the populations to a global maximum.

References

[1] J. W. Bala and H. Wechsler, "Learning to detect targets using scale-space and genetic search," in *Proceedings, 5th Int. Conf. Genetic Algorithms*, (Urbana-Champaign), 1993.

[2] M. R. Bartz, "Optimizing a video processor for OCR," in *Proceedings International Joint Conference on AI*, pp. 79–90, 1969.

[3] B. Bhanu and O. Faugeras, "Segmentation of images having unimodal distributions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-4, pp. 408–419, 1982.

[4] B. Bhanu, S. Lee, and J. Ming, "Self-optimizing image segmentation system using a genetic algorithm," in *Proceedings, 4th Int. Conf. Genetic Algorithms*, (San Diego), 1991.

[5] S. Boukharouba, J. M. Rebordao, and P. L. Wendel, "An amplitude segmentation method based on the distribution function of an image," *Comput. Vision Graphics Image Process.*, vol. 29, pp. 47–59, 1985.

[6] C. K. Chow and T. Kaneko, "Boundary detection of radiographic images by a threshold method," in *Proceedings IFIP Congress*, pp. 130–134, 1971.

[7] C. K. Chow and T. Kaneko, "Automatic boundary detection of left ventricle from cineaogiograms," *Computat. Biomed. Res.*, vol. 5, pp. 338–410, 1972.

[8] F. Deravi and S. K. Pal, "Gray level thresholding using second-order statistics," *Pattern Recognition Letters*, vol. 1, pp. 417–422, 1983.

[9] W. Doyle, "Operation useful for similarity-invariant pattern recognition," *J. Assoc. Comput. Mach.*, vol. 9, pp. 259–267, 1962.

[10] S. M. X. Fernando and D. M. Monro, "Variable thresholding applied to angiography," in *Proceedings 6th Int. Conf. Pattern Recognition*, 1982.

[11] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.

[12] E. D. Goodman, *GALOPPS–Genetic ALgorithm Optimized for Portability and Parallelism System.* Genetic Algorithms Research and Applications Group, Michigan State University, East Lansing, Michigan 48824, 2.20 ed., 1994.

[13] W. E. L. Grimson, *Object Recognition by Computer: The Role of Geometric Constraints.* The MIT Press, 1990.

[14] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Texture features for image classification," *IEEE Trans. Systems Man Cybernet.*, vol. SMC-3, pp. 610–621, 1973.

[15] G. Johannsen and J. Bille, "A threshold selection method using information measures," in *Proceedings 6th International Conference on Pattern Recognition*, (Munich, Germany), pp. 140–143, 1982.

[16] J. N. Kapur, P. K. Sahoo, and A. K. C. Wong, "A new method for gray-level picture thresholding using the entropy of the histogram," *Computer Vision, Graphics, and Image Processing*, vol. 29, pp. 273–285, 1985.

[17] A. J. Katz and P. R. Thrift, "Generating image filters for target recognition by genetic learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, pp. 906–910, September 1994.

[18] R. L. Kirby and A. Rosenfeld, "A note on the use of (gray level, local average gray level) space as an aid in thresholding selection," *IEEE Trans. Systems Man Cybernet.*, vol. SMC-9, pp. 860–864, 1979.

[19] R. Kohler, "A segmentation system based on thresholding," *Comput. Graphics Image Process.*, vol. 15, pp. 319–338, 1981.

[20] S.-C. Lin, W. F. Punch, and E. D. Goodman, "Coarse-grain parallel genetic algorithms: Categorization and new approaches," in *Sixth IEEE Symposium on Parallel Distributed Processing*, (Dallas, Texas), pp. 28–37, October 1994.

[21] Y. Nakagawa and A. Rosenfeld, "Some experiments on variable thresholding," *Pattern Recognition*, vol. 11, pp. 191–204, 1979.

[22] N. Ostu, "A threshold selection method from gray-level histogram," *IEEE Trans. Systems Man Cybernet.*, vol. SMC-8, pp. 62–66, 1978.

[23] J. M. S. Prewitt and M. L. Mendelsohn, "The analysis of cell images," in *Ann. New York Acad. Sci.*, vol. 128, pp. 1035–1053, New York Acad. Sci., New York, 1966.

[24] T. Pun, "A new method for gray-level picture thresholding using the entropy of the histogram," *Signal Process.*, vol. 2, pp. 223–237, 1980.

[25] T. Pun, "Entropic thresholding: A new approach," *Computer Vision, Graphics, and Image Processing*, vol. 16, pp. 210–239, 1981.

[26] W. Punch, E. Goodman, R. Averill, M. Pei, L. Chia-Shun, D. Ying, D. Redder, and V. Kureichik, "Research applications using genetic algorithms," Tech. Rep. CAD-94, Michigan State University Genetic Algorithm Group, East Lansing, Michigan 48824, 1994.

[27] T. Pun, "A new method for gray-level picture thresholding using the entropy of the histogram," *Signal Process.*, vol. 2, pp. 223–237, 1980.

[28] A. Rosenfeld and P. De La Torre, "Histogram concavity analysis as an aid in threshold selection," *IEEE Trans. Systems Man Cybernet.*, vol. SMC-13, pp. 231–235, 1983.

[29] A. Rosenfeld and R. C. Smith, "Thresholding using relaxation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-3, pp. 598–606, 1981.

[30] P. Sahoo, S. Soltani, A. K. C. Wong, and Y. C. Chen, "A survey of thresholding techniques," *Computer Vision, Graphics, and Image Processing*, vol. 41, pp. 233–260, 1988.

[31] W. A. Tackett, "Genetic programming for feature discovery and image discrimination," in *Proceedings, 5th Int. Conf. on Genetic Algorithms*, (Urbana-Champaign), 1993.

[32] W. Tsai, "Moment-preserving thresholding: A new approach," *Computer Vision, Graphics, and Image Processing*, vol. 29, pp. 377–393, 1985.

[33] S. Wang and R. M. Haralick, "Automatic multithreshold selection," *Comput. Vision Graphics Image Process.*, vol. 25, pp. 46–67, 1984.

[34] J. J. Weng, "SHOSLIF: The hierarchical optimal subspace learning and inference framework," Tech. Rep. CPS 94-15, Michigan State University, Department of Computer Science, A714 Wells Hall, East Lansing, Michigan 48824, March 1994.

[35] N. Wolfe, "A dynamic thresholding scheme for quantization of scanned image," in *Proceedings Automatic Pattern Recognition*, pp. 143–162, 1969.

[36] S. Zucker, R. Hummel, and A. Rosenfeld, "An application of relaxation labelling to line and curve enhancement," *IEEE Trans. Comput.*, vol. C-26, pp. 394–403, 1977.