Genetic Programming for Object Detection

Jay F. Winkeler
Electrical and Computer Engineering Dept.
University of California at Santa Barbara
Santa Barbara, CA 93106-9560
E-mail: jay@iplab.ece.ucsb.edu

B. S. Manjunath
Electrical and Computer Engineering Dept.
University of California at Santa Barbara
Santa Barbara, CA 93106-9560
E-mail: manj@ece.ucsb.edu

ABSTRACT

This paper examines genetic programming as a machine learning technique in the context of object detection. Object detection is performed on image features and on gray-scale images themselves, with different goals. The generality of the solutions discovered, over the training set and over a wider range of images, is tested in both cases. Using genetic programming as a means of testing the utility of algorithms is also explored. Two programs generated using different features are hierarchically combined, improving the results to 1.4% false negatives on an untrained image, while saving processing.

1. Introduction

How can one formulate a general method for detecting specific objects in a cluttered scene? A general solution is impossible since the algorithms will be target specific. Any strategy for detecting faces in an image, for example, cannot be expected to detect airplanes as well. Just like a person, a computer needs different strategies to detect different objects. Machine learning provides a general approach for solving task-specific problems like target detection.

As a machine explores its environment in the context of a task it gathers exploitable knowledge about which actions lead to success. The specific knowledge depends on the task, but the learning method is task-independent. Machine learning allows strategies tailored to a particular data set to be developed without any simplifying assumptions. A learning machine handles any adaptive fine tuning necessary, and may produce novel algorithms. Machine learning can provide general solutions to task-specific problems.

Genetic programming (GP) is capable of learning strategies based on the rewards received as it attempts to perform a task. Given a set of tools, GP efficiently searches the space of possible programs and finds those that perform the specified task well. Because the only information GP receives about its task is the fitness measures of its programs, the learning method is the same whether one is interested in locating faces or planes. Only the training set is modified.

GP differs from other learning methods in that the data does not have to be manipulated into a special format, and any algorithm coded as a function can be used as a tool in the solution. GP exhibits this flexibility because it works directly with coded functions to perform the task. For image processing, commands as clear as “shift image left” or “Gabor filter image” can be included. Because specific algorithms can be included, their utility at the task can be tested. The cost of using these different functions can easily be taken into account with GP. Another advantage occurs when the evolved code is terse: it can be interpreted piecewise and examined for specific strategies.

2. Previous Work

Not much work has been done with image processing using GP because of the huge amounts of processing involved. GP typically runs thousands of programs through hundreds of generations to arrive at a solution. When these programs start processing images rather than scalars, the computations become immense. A training set of 50 by 50 pixel images where the average program in a population of 500 performs 50 operations requires 62.5 million operations per generation for every image trained. For a task such as object detection, thousands of training examples may be required for a representative set. The promise of GP holds true in that the effort comes from a computer rather than a human, but for images this represents a great deal of effort. The results reported thus far, however, are encouraging.

Harris and Buxton (1996) evolved edge detectors which outperformed Canny’s edge detector, particularly on non-ideal step edges of the type included in the training set. They worked in 1D rather than 2D to save on processing effort.

Poli (1996) used GP to segment MR images of the brain and reported better results than with a neural network. He attempted to keep the processing down to a reasonable level by including precomputed results and using tools that do minimal processing. His work used minimum and maximum functions to allow for morphological operations, and “shift”

functions for filter design. Poli thresholded the result of tested programs before scoring because forcing the GP to produce binary output is inefficient. The filter's output will have to be classified into object and non-object pixels anyway, so selecting a threshold tests the actual filter output.

Soule et al. (1996) examined how to use selective pressure to cut down on code growth. The code in GP tends to grow exponentially in size while the fitness grows linearly. Since images require huge amounts of computational effort, code growth must be constrained. Selective pressure punishes code based on the number of nodes (function calls) in the evolved program, using the machinery that already exists to create a pressure on longer programs.

Tackett (1993) used image features to locate vehicles in cluttered terrain. The genetic program created a better strategy using primitive features directly rather than statistical features derived from these primitive features. Using a limited number of features rather than the entire gray-scale image reduces the amount of processing required.

Other work using GP with images has been done by Koza (1992), Nordin and Banzhaf (1996), Daida et al. (1996) and Swets et al. (1995).

3. Tools

The `Comp` function is a new tool which enhances the capabilities of the `Min`, `Max`, `Avg`, and Logical functions in creating nonlinear filters. The `Comp` function compares two images pixel by pixel, returning an image with a value of 0 where the two images are equal, 1 where the first image has a larger value, and -1 where the second image has a larger value. Consider trying to locate all the global maxima in an image. In three or four generations, the GP evolves the code `(+ (Comp(Img Max(Img))) 1)`, which marks only the global maxima. This tool allows the evolving programs to easily select portions of an image based on intensity.

Table 1: Optional Functions

Name	Definition
`G(XYZ)`	Scale space interactions with Gabor filters
`PGA(X)`	Big window peer grouping (PG).
`P(WXYZ)`	Big window peer grouping, then scale space interactions with Gabor filters

The three optional tools shown in Table 1 are included so their utility at object detection can be tested. Each is expected to be useful for the detection task, but their relative utility is unknown. The GP will discover which tools seem more useful for object detection as the tools are included or not for training runs. If including any tool increases performance, it is in some way fit for the specific task.

The scale space interactions with Gabor filters tool, described in Manjunath et al. (1996), implements a feature detection model. Two different Gabor filters are applied to an image, and the absolute value of the difference in their responses is the interaction result. `G(XYZ)` represents the difference of the responses of a Gabor filter at scale X and a filter at scale Y, both at orientation Z. In scaling, 1 represents the largest filter, i.e. the most smoothing. $X \in [1, 3]$, $Y \in [X, 4]$, and $Z \in [1, 8]$, where an increase of one represents a change of 22.5 degrees counterclockwise in filter orientation. One is horizontal and five is vertical.

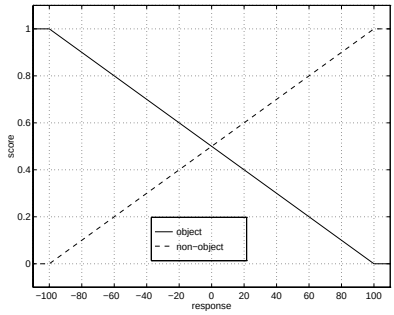

Figure 1: Response versus fitness score curves.

The peer grouping (PG) tool from Hewer et al. (1997) is a means of smoothing an image within similar regions. At each pixel the m values closest to the center pixel's value are chosen from an n by n window centered on the pixel. The average value of these peers replaces the center pixel's value and the process is repeated until convergence. In big window PG, the entire image is the window, so the averaging ignores spatial location. Now sorting has to be done only initially. `PGA(X)` represents a big window PG image where X% of the total pixels in the image are chosen as peers. In the following experiments, 20 iterations are performed in all cases.

The final tool combines the two other optional tools. `P(WXYZ)` represents an image peer grouped with W% of the total pixels in the image as peers, and then operated on by the scale space model interactions, as in `G(XYZ)`.

Table 2: Tableau for Feature-based Detection

Objective	Partition faces from non-faces
Function set	+,-, *,% (protected division)
Terminal set	constants 0.5, 1, 1.5,...10 52 features from image
Fitness	See Figure 1
Parameters	Population Size: 500
Selection	Demetic (20 individuals/deme) Migration rate: 5%
Mutation Rate	1%

4. Experiments

The experiments use GP to learn a strategy for detecting faces in a cluttered scene. The algorithm could be applied to any other detection problem without modification.

Two experiments are performed, examining different methods of processing images with GP, and involving dif-

ferent goals. The first experiment processes statistical features extracted from the image to determine if the specifically sized image is a face. Only one scale is trained, so the entire scene must be searched through scale space to detect all faces. A large set of faces and non-faces is trained in hopes of discovering a general solution.

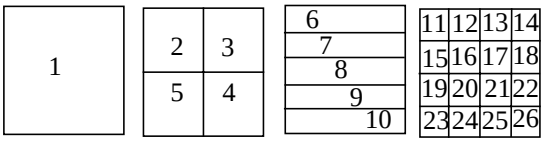

Figure 2: Regions of the image for feature extraction.

The second experiment processes gray-scale images, searching the image once for faces at all scales. Only a single image with multiple faces is used for training. Four different training runs are made, including different tools each time, so that the utility of these tools may be tested.

The experiments were run with GPC++, a public domain genetic programming system developed by Fraser (1994).

4.1 Feature-Based Detection

The first experiment creates programs which process "features" extracted from 20 by 20 pixel regions of either face or non-face images. Every face or non-face image is partitioned four times, as shown in Figure 2. The average value and standard deviation of these regions are used as two separate features, for a total of 52 features per image. Table 2 shows the details of the parameters for these experiments.

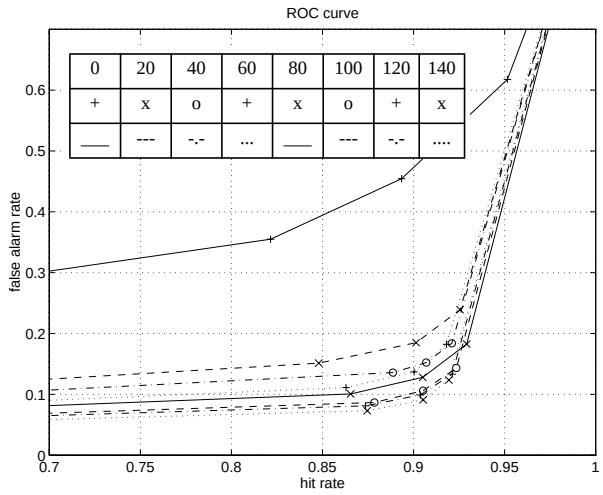

Figure 3: The ROC curves for every 20 generations.

The training set includes 1464 face images, including 137 faces from the FERET database and 45 faces from 24 images taken with two different cameras in two different environments. The faces are scaled, with constant aspect ratio, to be approximately the same size. Each face is represented 8 times in the training set, using the mirror image and low-pass filtered versions (3x3, 9x9, and 15x15 smoothing masks) of both the image and its mirror. The training set also includes 36300 non-faces which are cut from variously subsampled versions of 15 images taken with the same cameras and environments as the non-FERET faces.

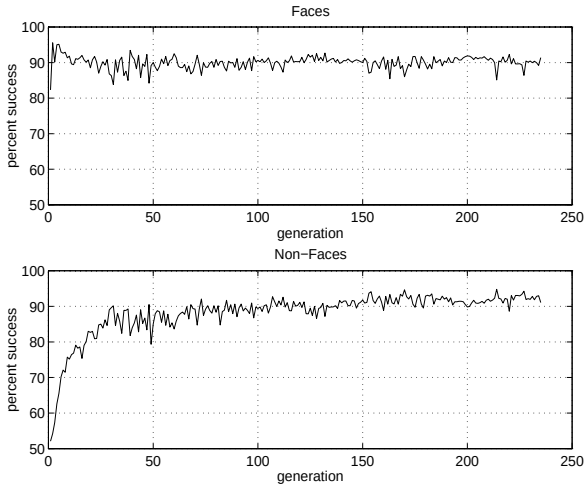

Figure 4: Percent success at identifying faces.

For such a large training set it is impossible to train on every example every generation, so the GP may begin to converge to solutions before a representative set of the images have been experienced. Premature convergence can be diminished by demetic grouping selection, which keeps the code from becoming too tightly focused in a particular area of program space. The use of 'islands' of programs developing independently, except for a few wanderers who may travel to another island and breed, accomplishes this goal.

Table 3: Tableau for gray-scale object detection

Objective	Detect faces in image.
Parameters	Population Size: 500 Maximum Generations: 40
Selection	Tournament (size 5)
Mutation Rate	0%

Training is performed on 300 randomly selected examples each generation: 150 faces and 150 non-faces. Selection is without repetition. The average fitness over the 300 examples is reported as the fitness of the program.

The receiver operating characteristic (ROC) curves shown in Figure 3 move toward an ideal receiver as training progresses. These curves piecewise linearly approximate the ROC curves with five points: when the face selection threshold is -100, 0, 100, and plus and minus infinity. The data points represent the average results for three successive best-of-generation programs over the entire training set. The performance of these programs at distinguishing faces from non-faces is obviously improving. Figure 4 shows the percentage of success at correctly identifying faces and non-faces in the entire training set as training progresses.

The best-of-generation programs for generations 198 to 202 have average fitness 183.4 with 3495.2 nodes. The only

features used by these five programs relate to the areas 4, 10 through 19, and 23. Regions 11 to 18 represent the statistics most closely related to eye areas, which are known to be useful in detecting faces. The discovered program mimics human behavior in focusing on eye areas.

Table 4: Standard Function Tool Box

Name	Arity	Definition
Img	0	Image.
Im(X)	0	Low pass filtered image (X by X neighborhood).
(X)	0	Constant value X. NX = -X.
+, -, *	2	Add, subtract, multiply.
%	2	Protected division.
Sh(X)	1	Shift input up, down, etc.
Add3	3	Add three inputs.
Comp	2	Compare inputs (see text).
Not	1	Logically NOTs input, assuming values>0 were 1s.
And, Or	2	Logically ORs, ANDs.
Avg, Min, Max	1	Returns a constant value based on input.

The generality of the evolved code is tested using the best of generation program for generation 200. This program has 3651 nodes and had a fitness score of 179 during training. This program is not the best-of-run individual, but can be considered representative. Using zero as the face selection threshold, 130 out of 137 faces from the training set (94.9%) are classified as faces. The code correctly classifies 128 out of 137 untrained images (93.4%) of the same individuals and 36 out of 50 faces (72%) images of unknown individuals in the FERET database. Given a mixed set of trained and untrained non-face images, the program classifies 109915 out of 116509 images as non-faces. Even if the 2420 trained examples included in this set have all been correctly classified, the program achieves better than 94% correct classification for unknown non-faces. A good strategy for partitioning known faces from non-faces has been evolved and it does fairly well for unknown faces too.

4.2 Gray-Scale Images

The next experiment trains on a single image with the faces marked by hand. The task is to locate all six faces, which are at different scales and include a hand drawn face. Table 3 and Table 4 show the parameters for the run.

Fitness is calculated with

$$N(1-0.5(F_p/M_p)-0.5(F_n/M_n)) \quad (1)$$

where F_p is the number of false positives observed and M_p is the number of false positives possible. The n subscript denotes equivalent values for false negatives. N is the number of pixels in the image. As more false positives are possible, individual false negatives are weighted more heavily.

Selective pressure is used to keep the code terse and save processing. The programs are allowed 50 nodes and then penalized heavily, piecewise linearly with code length. This selective pressure is so severe as to degrade performance, but limits the processing to acceptable levels.

Figure 5: The solution found by the evolved program. Bright areas indicate potential faces.

The tools available in this experiment are designed to work as efficiently as possible. Many tools, including low pass filtering, peer grouping (PG), and scale space interactions with Gabor filters, have been precomputed. None of the image processing tools used in this experiment can increase the size of the images they process.

Figure 6: Response to an individual not in training set.

Figure 5 shows the results for the best individual after 45 generations. A few false negatives are evident: portions of the face near the eyes and the entire hand-drawn face. The false positives include some of the skin areas in the image, as well as some of the high frequency areas.

The evolved program to locate these face regions reads:

```
((- (- (Comp (Add3 Im9 2 N9) (- Im15 Im3)) (ShL (% (ShD(ShD(- Im9 8))) (- Im9 8)))) (% (ShD (ShD (ShL (- Im9 8)))) (Add3 (Add3 Im9 2 N9) (Min (Min (ShL (ShL (% (- N8 (Neg Im3)) (% 8 N3)))))) 2))))
```

Studying this code can reveal the evolved strategy. Three

major pieces exist, the results of second and third portions being subtracted from the first. The first segment selects all but dark regions surrounded by bright. The next block removes regions that do not have a specific intensity texture. The final block of code removes the brighter regions of the image except under specific brightness conditions. This code selects as faces the dark but not darkest regions of the image that have a face-like texture of intensities.

Figure 7: Response to the face of an individual not in training set with a hand occluding portions of the face.

The code generated during training can be informative, too. Generation 0 selects regions with sudden changes in intensity no matter how minimal. Generation 1 selects only regions that have larger intensity changes. Generations 2 and 3 keep uniform regions in the smoothed image which are not the darkest areas. By Generation 4, the faces are beginning to be marked as darker regions in the image which have a certain texture. Subsequent generations fine tune the texture.

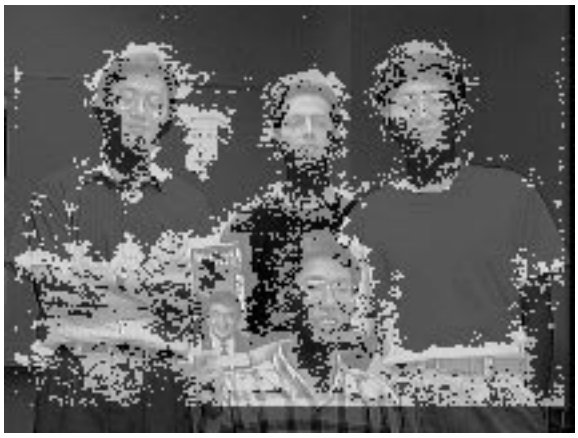

Figure 8: The best resulting image for run 2.

The evolved code is applied to other images taken with the same camera under similar lighting conditions. Only one of the three individuals in the images was included in the training set image, and the scale of the faces differs from those in the training image by up to a factor of two. The algorithm locates the six faces in the four images, but also has a number of false positives. Figure 6 shows one example result. The code has not been exposed to the texture of the partition board, which is selected as face. Had this texture appeared in the training image, a different approach would have been learned. Figure 7 shows how the program selects a partially occluded face, rejecting the obscuring hand.

Table 5: Face detection after 40 generations.

Run	Fitness	Optional Tools
1	25117	None
2	27296	`G(XYZ)`
3	26327	`PGA(X)`
4	27787	All (29 Generations)

Four different training runs are made with the same face image, the details are shown in Table 5. Figure 9 shows how fitness improves as training progresses. The code evolved when all optional tools were available performs the best, while the code denied all optional tools performs the worst. The generated code shows that for runs 2 and 3, the optional tools are used extensively, while for run 4 only the scale space interaction tools are used. Since run 4 becomes the same as run 2 with different random numbers, it is not examined further. The best choice then is the scale space interaction tools, while the PG tools have some utility at the task.

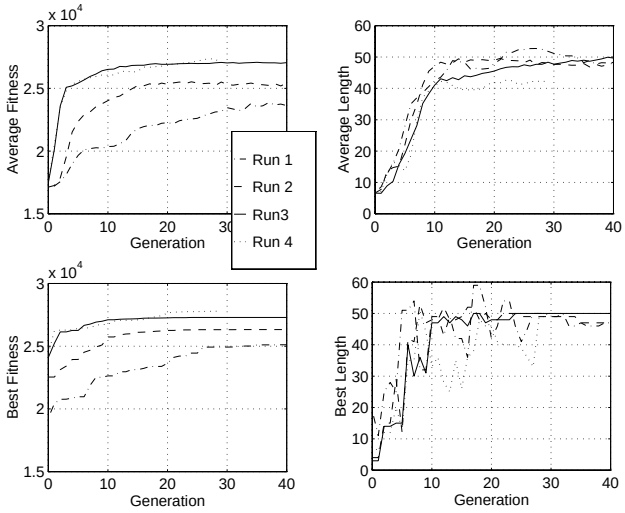

Figure 9: Comparison of fitness using different tools

This “best” code produces Figure 8, and is terse enough to be interpreted in a piecewise manner. The code selects regions with edges in the horizontal and diagonal direction that are near areas with edges in the horizontal but not in the diagonal direction.When this evolved filter is applied to untrained images, it becomes obvious that this filter is not as general as the one evolved without the optional tools. The filter responds mainly to horizontal edges, which were nearly all face-related in the training image. In other images, this strategy causes false positives. Given a more representative training image, the results would have been more general.

4.3 Hierarchical combination

The code from the first experiment does well at locating a specific scale of face, but has to be applied to images multiple times to find the faces at every scale. The code from the second experiment locates faces at all scales based solely on intensity, but exhibits an undesirable number of false positives. An elegant solution might combine the two evolved programs in a hierarchical manner. After the gray-scale program has processed the image, the more time consuming search through scale space can be limited to the potential-face regions discovered by this first filter. For the image shown in Figure 6, this cuts the processing on the second level of the hierarchy nearly 75%. If the scale can be predicted as well, the savings are much greater.

Figure 10: The results for an untrained image and hierarchical processing. Bright areas indicate potential face centers. The square indicates the scale for the search.

Because the two face detection methods are evolved based on different information about the image, they are unlikely to have similar false positives. Figure 10 shows the results for processing an untrained image with the evolved code in a hierarchical manner, and assuming a nearly correct estimate of face scale. The false negative rate is cut from 8.2% (assuming the hits on the nose are correct) to 1.4%. The programs discovered are complementary in this case.

5. Conclusions

GP trades human insight for computer iterations in attempting to solve a class of problems. Training is expensive, but the solutions discovered may be general for object detection. A combination of results evolved using different features can improve performance and save processing.

Testing the utility of different tools cannot be performed by learning mechanisms which do not allow functions themselves to be included. The GP has multiple means of testing, and will even suggest potential uses of the tool.

The main problems with the programs discovered herein arise from mistakes in the training set. The training set should be more representative, and the question arises of how to select non-objects from the training set. In neural network literature, Rowley et al. (1996) suggest a bootstrapping method for selecting non-objects. Recent experiments show that this method does not work in GP. An alternative approach may be the topic for future research.

Acknowledgments

Portions of the research in this paper use the FERET database of facial images collected under the ARPA/ARL FERET program.

Bibliography

Daida, J., Bersano-Begey, F., Ross, S. and Vesecky, J. 1996. Computer Assisted Design of Image Classification Algorithms: Dynamic and Static Fitness Evaluation in a Scaffolded Genetic Programming Environment. *Genetic Programming 1996: Proceedings of the First Annual Conference*. 279-284.

Fraser, A. 1994. *Genetic Programming in C++*. University of Salford, Cybernetics Research Institute. Technical Report 040.

Harris, C. and Buxton, B. 1996. Evolving Edge Detectors with Genetic Programming. *Genetic Programming 1996: Proceedings of the First Annual Conference*. 309-314.

Hewer, G., Kenney, C., Manjunath, B.S. and Schoyen, R. 1997. Peer Group Image Processing for Segmentation. working paper.

Koza, J. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA: The MIT Press.

Manjunath, B.S., Shekhar, C. and Chellappa, R. April 1996. A New Approach to Image Feature Detection with Applications. *Pattern Recognition*. 29(4) 627-40.

Nordin, P. and Banzhaf, W. 1996. Programmatic Compression of Images and Sound. *Genetic Programming 1996: Proceedings of the First Annual Conference*. 345-350.

Poli, R. 1996. Genetic Programming for Image Analysis. *Genetic Programming 1996: Proceedings of the First Annual Conference*. 363-368.

Rowley, H., Baluja, S. and Kanade, T. 1996. Neural Network-Based Face Detection. *IEEE Conference on Computer Vision and Pattern Recognition*. 203-208.

Soule, T., Foster, J. and Dickinson, J. 1996. Code Growth in Genetic Programming; *Genetic Programming 1996: Proceedings of the First Annual Conference*. 215-223.

Swets, D., Punch, B. and Weng, J. 1995. Genetic Algorithms for Object Recognition in a Complex Scene. *International Conference on Image Processing*. Vol. 2 595-598.

Tackett, W. 1993. Genetic Programming for Feature Discovery and Image Discrimination. *Proceedings of the Fifth International Conference on Genetic Algorithms*. 303-309.