Evolving Finite State Machines with Embedded Genetic Programming for Automatic Target Detection

Karl Benson
Defence Evaluation and Research Agency,
DERA Malvern, St Andrews Road,
Worcestershire WR14 3PS, UK
Tel: +44(0)1684 894580
kabenson@dera.gov.uk

Abstract- This paper presents a model comprising Finite State Machines (FSMs) with embedded Genetic Programs (GPs) which co-evolve to perform the task of Automatic Target Detection (ATD). The fusion of a FSM and GPs allows for a control structure (main program), the FSM, and sub-programs, the GPs, to co-evolve in a symbiotic relationship. The GP outputs along with the FSM state transition levels are used to construct confidence intervals that enable each pixel within the image to be classified as either target or non-target, or to cause a state transition to take place and further analysis of the pixel to be performed. The algorithms produced using this method consist of nominally four GPs, with a typical node cardinality of less than ten, that are executed in an order dictated by the FSM. The results of the experimentation performed are compared to those obtained in two independent studies of the same problem using Kohonen Neural Networks and a two stage Genetic Programming strategy.

1 Introduction

In the model presented in this paper each state of the FSM has two GPs (Koza, 1992) embedded within it that cause the state transitions to take place, and provide the input and output alphabet of the FSM. The FSM is thus a function of the GPs embedded within its states, and so the hybrid FSM with embedded GP has been denoted FSM(GP) just as we would denote a function f of x as $f(x)$ in mathematics. This framework forms a symbiotic relationship between the FSM and the GPs which co-evolve together. The FSM must evolve transitions that enable the most useful GPs to be executed. Also it must facilitate the appropriate number of states to achieve the objective, and transition levels that work with the GP outputs to form confidence intervals. At the same time the GPs must evolve to produce an output that works with the FSM transition levels. They must also search and combine the function and terminal set for productive combinations that can achieve sub-tasks. In addition, they must co-evolve with each other to act on the current environment, and, if necessary, cause a transition to a state that contains a GP that may be more able to act on the current environment. To the author's knowledge the fusion of FSMs and GP has only been used once before by Ashlock (Ashlock and Richter, 1997; Ashlock, 1997) to play the game of divide the dollar, and has never been applied to the field of computer vision. The work presented in this paper extends that of Ashlock (Ashlock and Richter, 1997; Ashlock, 1997) in several ways to incorporate the following.

1. Self adaptive parameters.
2. The ability to add and delete states.
3. The ability to construct confidence intervals.
4. Mutation types are extended from four to fourteen.
5. The number of GPs per state are extended from one to two.

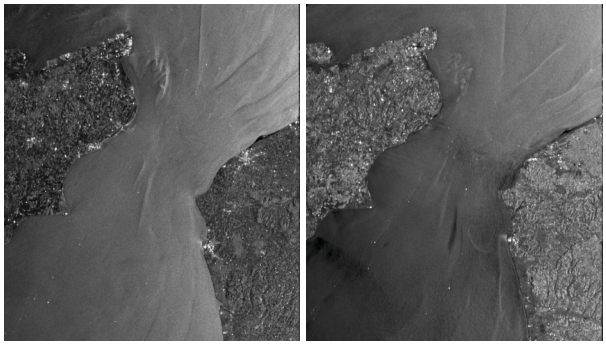
2 The Test Problem

The problem on which to test the FSM(GP) is that of Automatic Target Detection (ATD). More specifically, the detection of ships within 100 meter resolution Synthetic Aperture Radar (SAR) images of a section of the English Channel known as the Dover Straits. The images were captured between July and November 1992 by the ERS-1 satellite[1]. This problem has been tackled before by Foulkes et al. (Foulkes and Booth, 2000), and by Howard et al. (Howard et al., 1999a; Howard et al., 1999b), and their studies are used as a benchmark for the research presented in this paper. Foulkes et al. used both a self-organising Kohonen neural network and a Multi-Layer Perceptron (MLP) neural network in their research, obtaining their best results with the self-organising Kohonen neural network. Howard et al. used genetic programming with a two stage strategy as follows: In the first phase GP evolved a detector to locate as many ships as possible. However, it was found that a large amount of ocean pixels were also classified as ships and so a second GP was evolved that could correctly reclassify these errors as ocean. The two GPs were then fused to perform the ATD. The images used by Foulkes et al., Howard et al., and in this paper are shown in Figure 1.

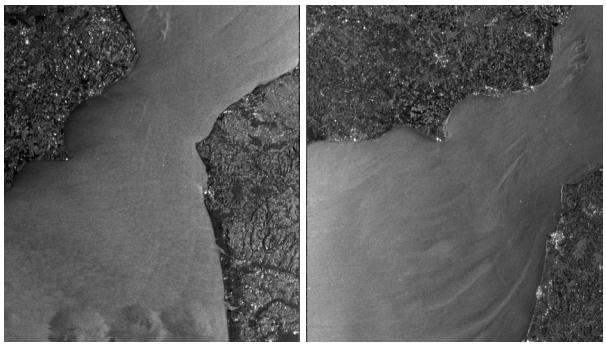
3 Finite State Machines (FSMs)

A FSM consists of a finite number of internal states which map a set of input symbols to a set of output symbols. When an input symbol is presented to the FSM, the current state will produce an output symbol based on the input, and a transition may then be made to a new state. Table 1 shows a three state FSM with start state 0 (denoted q_0), an input alphabet (0, 1) and output alphabet (A, B, C). An input of 011010110 into this FSM would produce the output CBACBCBAC. FSMs

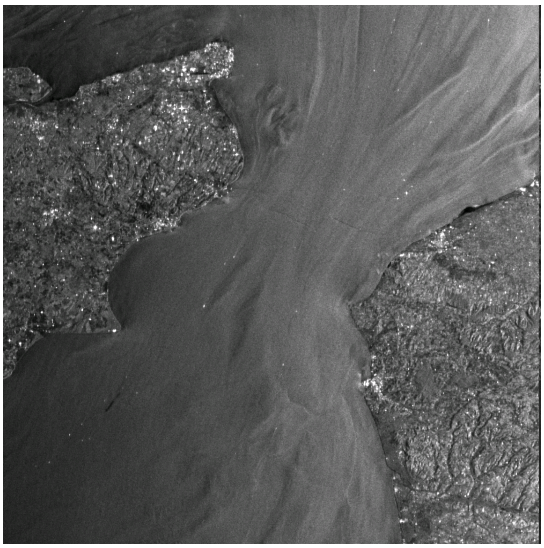
[1] ERS - European Remote Sensing.



(a) Test Image 1. (b) Test Image 2.



(c) Validation Image 1. (d) Validation Image 2.



(e) Training Image.

Figure 1: Imagery of the English channel used in the ATD experimentation.

State	Input	Output	Next State
q_0	0	C	q_1
	1	A	q_2
q_1	0	B	q_1
	1	B	q_0
q_2	0	C	q_1
	1	A	q_2

Table 1: State transition table of a three state FSM.

were first used in Evolutionary Programming (EP) by Fogel (Fogel, 1962; Fogel, 1964) with the goal of creating artificial intelligence. Fogel proposed that "intelligent behaviour requires the composite ability to predict one's environment coupled with a translation of the predictions into a suitable response in light of a given goal". The aim of Fogel's original work was to evolve an algorithm that when presented with a finite set of symbols, could predict the next symbol in the sequence. FSMs were used by Fogel as a useful representation of the algorithm.

4 Finite State Machine with Embedded Genetic Programming: FSM(GP)

As explained in Section 3 state transitions in a FSM take place according to an environmental input. The environment in our case is the image, or more precisely, the pixel intensity values. The number of pixel intensity values in any one image is large and usually lies in the range (0,255). To cater for all of the possible intensity values an input alphabet size of 256 would be required. The search space of FSMs that have an input alphabet of this size is very large, and can be computed using the formula $N = (n^a b^a)^n$, where a and b are the cardinalities of the input and output alphabets respectively, and n is the number of states (Atmar, 1976). If we limited the output alphabet to a cardinality of 2 and the number of states to 2, then with an input alphabet cardinality of 256 it is possible to construct 2^{1024} different FSMs. This is a large search space, and working with a smaller input alphabet than pixel intensities is desirable. In the FSM(GP) it is the GPs that interact with the environment, and so their output is used as the input to the FSM element of the FSM(GP). However, the outputs from the GP in this research are real valued numbers, which are greater in quantity than pixel intensity values within an image. Therefore, a wrapper (see equation 1) on the GP outputs is implemented to reduce the FSM input alphabet. An alternative would have been to quantize pixel intensity values, but this would lose information on small differences in the pixel intensities. As the objective is to distinguish between targets and non-targets, it would seem reasonable to apply evolutionary pressure to the FSM(GP) in such a manner as to force the GP outputs to be as different as possible for targets and non-targets, and at the same time to use these outputs to cause state transitions. To achieve this, the state transitions and the

halting of the FSM(GP) occur using the following wrapper -

$$I = \begin{cases} 0 & \text{if } 0 \leq O_t < T_{si} \\ 1 & \text{if } -T_{si} < O_t < 0 \\ Halt & \text{if } |O_t| \geq T_{si} \end{cases} \quad (1)$$

where I is the input to the next state, O_t is the GP output, and T_{si} is the transition threshold of state i. The state transition thresholds are chosen uniformly at random from the interval (3,15) during the creation of the FSM(GP), and may then be mutated during the run by adding Gaussian noise. A GP output $|O_t| \geq T_{si}$ causes the FSM(GP) to move to the halt state. In this state if $O_t \geq T_{si}$ then the pixel being processed is designated as a target, else if $O_t \leq -T_{si}$ then the pixel being processed is designated as a non-target. The state transition thresholds also act as a confidence interval. If $|O_t| < T_{si}$ then the algorithm does not have the confidence to classify the pixel under consideration as target or non-target and so further state transitions occur until a state is entered where a GP resides that produces an output $|O_t| \geq T_{si}$ enabling the pixel to be classified. For the FSM(GP) to function correctly, each GP must give an output based on what it has seen of the environment. This enables the FSM to change to a state that contains a GP that can usefully act upon the current environment by forcing another transition to take place (thus repeating the cycle), or move to the halt state. Thus the GPs must co-evolve to help each other in the decision of designating a pixel as target or non-target, and to enable state changes to occur. At the same time the FSM element must evolve to 'connect' co-operating GPs, and have enough states (and thus GPs) to achieve the task presented to it. Thus the FSM element and the GP element of the FSM(GP) have a symbiotic relationship.

5 Performance and Fitness

A well established metric used to evaluate an ATD algorithms performance is the Figure Of Merit (FOM) given by equation 2 (NAC, 1994). The FOM is defined in terms of the number of correctly classified targets, true positives (TP), and the number of non-targets classified as targets - false positives (FP).

$$FOM = \frac{TP}{FP + \text{number of targets}} \quad (2)$$

As can be seen from equation 2 the FOM increases with an increase in target detection and a decrease in false detection.

Three fitness functions were considered during this research - (a) the FOM (b) the Pareto optimal set of true positives and true negatives, and (c) the function shown in equation 3. Using the FOM as the fitness function produced results that were no worse than those produced by the other two functions, but the FOM only applies evolutionary pressure to increase target detection and decrease false detections, which is what we want, but it does not force the Evolutionary Algorithm (EA) to produce a FSM(GP) that can also detect non-targets (ocean). It was found that when using the FOM as the fitness function the FSM(GP) would halt for targets, but complete the maximum number of state transitions allowed when presented with ocean. This did not hinder the accuracy of the detector in any way, but it is more computationally expensive. The second strategy was to select the Pareto optimal set (Goldberg, 1989, pp 197–198) of TPs and TNs at each generation. This did cause the FSM(GP) to halt when presented with ocean pixels, but did not perform as well as the fitness function shown in equation 3, which is a weighted sum of the FOM for targets, and the FOM for ocean pixels.

$$fitness = \frac{\alpha\, TP}{FP + N_t} + \frac{\beta\, TN}{FN + N_o} \quad (3)$$

where N_t = number of targets, N_o = number of ocean pixels, TN = true negatives, and FN = false negatives. This fitness function produced detectors as accurate as those using the FOM as the fitness function, but with the added benefit that the FSM(GP)s it produced also halted when presented with ocean pixels, thus producing more efficient ATD algorithms. In all experimentation carried out α and β were set to 0.5, on the assumption of equal importance of detecting target and non-target. The maximum number of state transitions allowed was set at 10.

6 Mutation of FSM(GP)

In this research an Evolutionary Programming (EP) approach was adopted for the modification of an individual in the population. That is, the method of modification is mutation. The following mutations are performed on the FSM(GP)

1. Add a state.
2. Delete a state.
3. Change the start state.
4. Mutate a transition.
5. Cycle the states.
6. Mutate a state transition threshold.
7. Perform headless chicken crossover on the states.
8. Exchange two GPs from different states.
9. Replace a GP within a state with a new randomly created GP.
10. Perform headless chicken crossover on the GPs.
11. Grow a sub-tree.
12. Shrink a sub-tree.
13. Mutate a tree terminal.
14. Mutate a tree function.

Since there are 14 possible forms of mutation, choosing the rate at which each of them should be performed is a difficult task. To overcome this problem the mutation probabilities are co-evolved along with the FSM(GP), that is, adaptive parameters are used. The parameter update rule chosen is part of the lognormal method of Schwefel (Schwefel, 1981). In particular the equation $\sigma_i' = \sigma_i e^{\tau' N(0,1)+\tau N_i(0,1)}$ that is used to update the standard deviations of the strategy parameters in Evolutionary Strategies. The reader may at first find this questionable - after all this equation was designed to update standard deviations, not the probability of a mutation. But we see that it does have the desired properties to solve our problem of not knowing the correct levels at which to set the mutation probabilities. The global factor $e^{\tau' N(0,1)}$ allows for an overall change in mutability. At the beginning of an EA run we would suspect that a high rate of mutability would be desirable whilst the population of solutions climb to an optimum, and that at the end of a run, when the solutions are near optimal, we would desire only small changes in the solution. This global factor allows this to occur. However we still do not know which of the mutations is having the most beneficial effect at any one time during an EA run. The inclusion of the term $e^{\tau N_i(0,1)}$ will allow for changes of the individual mutation probabilities, and thus the mutations having the most beneficial effect at that stage of the EA run will be carried forward to the next generation. So recasting Schwefel's standard deviation equation as one for the updated probability P_i' of mutation i taking place, we have

$$\begin{aligned} P_i' &= P_i\, e^{\tau' N(0,1)+\tau N_i(0,1)} \\ \tau &= \frac{1}{\sqrt{2\sqrt{n}}} \\ \tau' &= \frac{1}{\sqrt{2n}} \\ n &= \text{number of possible mutations.} \end{aligned} \tag{4}$$

We should note that equation 4 also provides a means of controlling the amount of mutations an individual receives. For example, if $P_i = 1 \ \forall\, i$ then the individual would receive n mutations (14 in our current case), and if $P_i = 0 \ \forall\, i$ it would receive none. If $P' > 1$ then it is reset to 1.

7 GP Parameters

7.1 Terminal Set

The terminal set for this problem was chosen to be the pixel intensity and a combination four sets of statistics calculated around the center pixel of a 9×9 sub-image. The terminal set is a subset (the statistics) of those used by Howard et al. (Howard et al., 1999a; Howard et al., 1999b). The reason for using this terminal set is that a direct comparison of the research presented in this paper and the research presented in (Howard et al., 1999a; Howard et al., 1999b) may be drawn thus providing a benchmark. If an alternative terminal set were chosen then better or worse performance of the algorithms developed in this paper could be attributed to a superior or inferior terminal set and not the algorithm itself, thus a fair comparison could not be made. The approach of using statistics centred on a pixel is similar to that taken by Poli (Poli, 1996a; Poli, 1996b), who uses moving averages. Poli also calculated the pixel statistics before running the EA to speed up the evaluation of an individual. This approach was also used in this research. Each pixel under consideration was treated as the centre pixel of a 9×9 sub-image on which a shift in the mean and standard deviation was performed before the statistics were calculated. The statistics used are listed below and the terminal set calculated from these statistics is given in Table 2.

1. The intensity value of the centre pixel - PIX
2. The average pixel intensity of an n×n sub-image centered at $P(x, y)$ - $Av_{n\times n}$
3. The average pixel intensity of the perimeter of an n×n sub-image centred at $P(x, y)$ - $PAv_{n\times n}$
4. The standard deviation in pixel intensity of the perimeter of an n×n sub-image centred at $P(x, y)$ - $PStd_{n\times n}$
5. The difference of the average pixel intensity of an n×n sub-image and the average pixel intensity of the perimeter of an r×r sub-image both centred at $P(x, y)$ - $DAv_{n,r}$

7.2 Function Set

The function set used is shown in Table 2, where s_i and s_j represent two arbitrary terminals. As can be seen from Table 2 the function set consists of the very simple mathematical functions $+$, $-$, $\div$, and $\times$, along with max, min, and neg. The first four functions were chosen to allow the EA to construct and exploit useful combinations of the terminal set. Since the FSM(GP) must act within thresholds, we need to equip the GPs with functions that will enable the FSM(GP) to move between these thresholds. Hence we include the GP functions max to enable a maximum output, min to enable a minimum output, and neg to assist in the production of negative outputs needed for the designation of non-targets.

8 Selection method

Two selection methods were considered. The first considered was a $(\mu + \lambda)$-EP approach. In the $(\mu + \lambda)$-EP approach the population (which contains $\mu + \lambda$ individuals) is first evaluated and each individual is assigned a fitness value. Each population member is then placed in a tournament with k other members and assigned a tournament score proportional to its fitness compared to the other k in the tournament. The population is then ranked according to their tournament score, and the top μ individuals go forward to the next generation along with λ offspring produced from these μ parents. The second

Functions			
Name	Definition	Name	Definition
$+$	$s_i + s_j$	$-$	$s_i - s_j$
$\div$	$\frac{s_i}{s_j}$	$\times$	$s_i \times s_j$
max	s_i if $s_i > s_j$, else s_j	neg	$-s_i$
min	s_i if $s_i < s_j$, else s_j		
Terminals			
Name	Definition	Name	Definition
s_0	PIX	s_8	$PAv_{9\times9}$
s_1	$Av_{3\times3}$	s_9	$PStd_{3\times3}$
s_2	$Av_{5\times5}$	s_{10}	$PStd_{5\times5}$
s_3	$Av_{7\times7}$	s_{11}	$PStd_{7\times7}$
s_4	$Av_{9\times9}$	s_{12}	$PStd_{9\times9}$
s_5	$PAv_{3\times3}$	s_{13}	$DAv_{3,5}$
s_6	$PAv_{5\times5}$	s_{14}	$DAv_{3,7}$
s_7	$PAv_{7\times7}$	s_{15}	$DAv_{3,9}$

Table 2: Function and Terminal Set.

method used is one which shall be denoted $(1+\lambda)^n$-EP. In the $(1+\lambda)^n$-EP approach a population of size n is evaluated and each individual in the population is assigned a fitness score. Thereafter, at each generation each individual in the population produces λ offspring. The λ offspring are then ranked against their siblings only, and the fittest offspring replaces its parent if its fitness is greater than or equal to that of its parent. Using this methodology no interaction between population members takes place. This promotes diversity in the population, and each population member's lineage is guaranteed to continue.

9 Experimental Results

Five runs using a (250+250)-EP, and five runs using a $(1+10)^{50}$-EP were performed. Each run was executed for 500 generations. The maximum number of states was set to 5, and the maximum GP tree depth was set to 4. At each generation the population was presented with 57 targets and a random sample of 1000 ocean pixels from the training image. At each generation g, the population member with the greatest fitness, the Best Of Generation (BOG), was applied to the two validation images shown in Figure 1. If the BOG's average fitness over the validation images was greater than the average fitness of the BOG at generation $g-1$ over the validation images, then the BOG of generation g was designated the Best Of Run (BOR) individual so far. The reason for using only 57 of the 77 possible targets is that the author believes the remaining 20 targets were in fact ocean, and could not be safely classified (by a human) as targets. This view was also shared by Howard et al. (Howard et al., 1999a; Howard et al., 1999b). The best individual found over the ten runs performed is shown in Table 3, and was found using the $(1+10)^{50}$-EP at generation 101 of run 3. The start state of this FSM(GP) is zero, and an initial input of zero is applied. The GP functions evolved in this FSM(GP) are given in equations 5, where F_{ij} denotes function i of state j. It is worth noting that function F_{21} was never executed when the FSM(GP) was presented with the five images used in this paper. A comparison of this FSM(GP) and the results obtained by Howard et al. (Howard et al., 1999a; Howard et al., 1999b) and Foulkes et al. (Foulkes and Booth, 2000) on the two unseen test images of Figure 1 are shown in Table 4. Figure 2 shows the output produced by the FSM(GP) depicted in Table 3 on test image 1. The white squares are TPs and the black circles are the FPs.

$$
\begin{aligned}
F_{10} &= min\left[(s_0 - s_4), \frac{s_{15}}{s_6}\right] - max[max(s_1, s_4), (s_3 - s_6)] \\
F_{20} &= min[(s_{11} - s_4), 1] - max[(s_0 + s_7), s_{13}] \\
F_{11} &= (s_0 + s_1) - (s_1 s_{12} + s_4) \\
F_{21} &= s_{10}\{min[min(s_7, s_{12}), (s_2 - s_{10})]\}
\end{aligned}
\tag{5}
$$

State	Input I	Output	Next State
q_0	$0 \le I < 0.07$	F_{10}	q_0
	$-0.07 < I < 0$	F_{20}	q_1
	$\lvert I\rvert \ge 0.07$	I	Halt
q_1	$0 \le I < 6.36$	F_{11}	q_0
	$-6.36 < I < 0$	F_{21}	q_1
	$\lvert I\rvert \ge 6.36$	I	Halt

Table 3: FSM(GP) evolved to perform automatic ship detection.

10 Discussion

A method of co-evolving GPs within an overall control structure, a Finite State Machine, to perform the task of ATD has been presented. The FSM(GP) is analogous to most human written programs in that there is a main program, the FSM, and functions that are called by the main program, the GPs. The FSM(GP) achieves the task of ATD in two ways:

1. A search over the provided features (terminal set) is performed. Useful features are combined as a GP to create a sub-program that performs part of the overall task of ATD. Each GP co-evolves with the other GPs within the FSM to perform a different sub-task.

2. A FSM is evolved that controls how many GPs are available (since they are embedded in its states) and the order in which the GPs are executed. The FSM's state transition thresholds are mutated during the run. The resulting state transition thresholds, in conjunction with the GP output, allows confidence intervals to be formed. The GP outputs, along with these confidence intervals, enable the FSM(GP) to cluster targets and non-targets into their appropriate classes.

	FOM		
Image	Kohonen NN	Two stage GP	FSM(GP)
Test 1	0.67	0.67	0.74
Test 2	0.72	0.69	0.8

Table 4: Performance comparison of Kohonen NN, two stage GP, and FSM(GP) on two unseen test images.

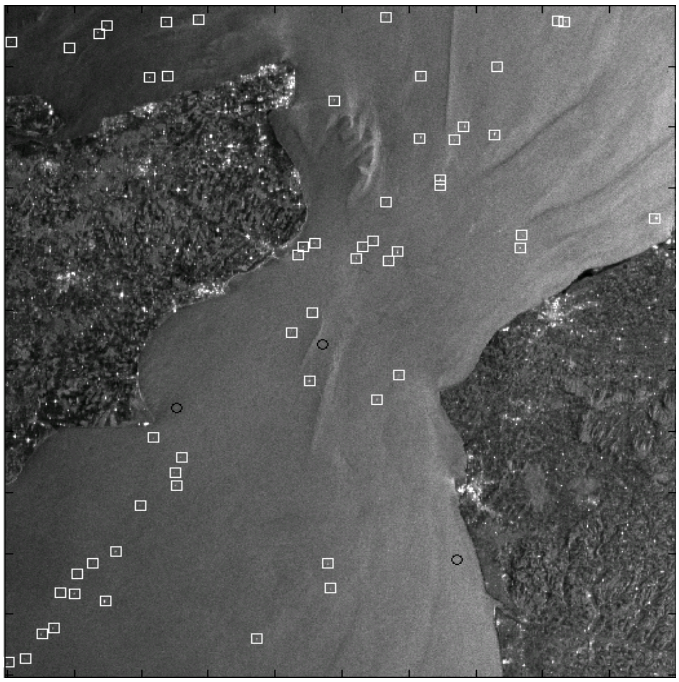


Figure 2: Output of FSM(GP) on Test image 1. The white squares are the TPs, and the black circles are the FPs.

It has been shown that the approach taken in this paper can result in an algorithm that consists of a few small GPs that are very easy to read and understand. Conventional GP however has a tendency (in non-toy problems) to create very large tree structures that make it impossible to determine exactly how the resulting program performs its given task. When encoded into an in service computer vision system these evolved algorithms should be very efficient owing to their simplicity.

This limited investigation has shown promising results. The FSM(GP)s evolved compare favourably to both Kohonen Neural Networks and two stage Genetic Programming. The true test of an algorithm is over the test set were it is applied to unseen images. Here the FOMs achieved by the FSM(GP) are greater than those achieved using Kohonen NNs, or a two stage GP strategy.

11 Current and Future Work

The work described in this paper has now been extended to a new version of the FSM(GP) that has no transition thresholds. A transition to a new state is based on a positive or negative output from the GP. In addition each state now contains two logical functions that combines the GP output of the current state with the outputs of the GPs of previously visited states. This concatenation of the GP outputs via logical functions allows highly complex decision spaces to be formed. The next stage of development is to provide the FSM(GP) with a memory. The memory will be modeled as a tape, thus mapping the Finite State Machine to a Turning Machine.

12 Acknowledgments

The author would like to thank Steve Foulkes for providing the imagery and ground truth target positions used in this research, and David Booth, James Cubillo, and Colin Reeves for editing this paper. Thanks also go to Mike Molland of MFM software for providing C++ expertise.

References

Ashlock, D. (1997). GP-automata for dividing the dollar. In Koza, J. R., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M., Iba, H., and Riolo, R. L., editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 18–26, Stanford University, CA, USA. Morgan Kaufmann.

Ashlock, D. and Richter, C. (1997). The effect of splitting populations of bidding strategies. In Koza, J. R., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M., Iba, H., and Riolo, R. L., editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 27–34, Stanford University, CA, USA. Morgan Kaufmann.

Atmar, W. (1976). *Speculation on the Evolution of Intelligence and its Possible Realization in Machine Form*. Sc.D dissertation, New Mexico State University, Las Cruces.

Fogel, L. J. (1962). Autonomous automata. *Industrial Research*, 4:14–19.

Fogel, L. J. (1964). *On the organization of intellect*. PhD thesis, UCLA.

Foulkes, S. B. and Booth, D. (2000). Ship detection in ESR-1 and radarsat SAR images using self-organising neural networks. In *Procceding of the amrs workshop on ship detection in coastal waters*.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, & Machine Learning*. Addison Wesley.

Howard, D., Roberts, S. C., and Brankin, R. (1999a). Evolution of ship detectors for satellite SAR imagery. In *Genetic Programming, Second European Workshop, EuroGP'99*, pages 135–148. Springer.

Howard, D., Roberts, S. C., and Brankin, R. (1999b). Target detection in SAR imagery by genetic programming. *Advances in Engineering Software*, 30:303–311.

Koza, J. R. (1992). *Genetic Programming: on the programming of computers by means of natural selection*. The MIT Press.

NAC (1994). Autonomous long-range IR target acquisition. Technical Report AC/243(Panel 3)TR/12, North Atlantic Council.

Poli, R. (1996a). Genetic programming for image analysis. Technical Report CSRP-96-1, University of Birmingham, UK.

Poli, R. (1996b). Genetic programming for image analysis. In *Proceedings of the First Annual Conference on Genetic Programming*, pages 363–368.

Schwefel, H. P. (1981). *Numerical Optimization of Computer Models*. John Wiley, Chichester, U.K.