Evolving Automatic Target Detection Algorithms that Logically Combine Decision Spaces

Karl Benson
Defence Evaluation and Research Agency,
DERA Malvern, St Andrews Road,
Worcestershire WR14 3PS, UK
kabenson@dera.gov.uk

Abstract

In this paper a novel approach to performing classification is presented. Discriminant functions are constructed by combining selected features from the feature set with simple mathematical functions such as $+, -, \times, \div, max, min$. These discriminant functions are capable of forming nonlinear discontinuous hypersurfaces . For multimodal data more than one discriminant function *may* be combined with logical operators before classification is performed. An algorithm capable of making decisions as to whether a combination of discriminant functions is needed to classify a data sample, or whether a single discriminant function will suffice, is developed. The algorithms used to perform classification are not written by a human. The algorithms are learnt, or rather evolved, using Evolutionary Computing techniques.

1 Introduction

Traditional decision theoretic methods strive to perform classification via the use of *discriminant functions*. Given M functions $d_1(\mathbf{x}), \ldots, d_M(\mathbf{x})$, M classes $\omega_1, \ldots, \omega_m$, and a vector of features $\mathbf{x} = (x_1, \ldots, x_n)$. The functions $d_1(\mathbf{x}), \ldots, d_M(\mathbf{x})$ are known as discriminant functions if $d_i(\mathbf{x}) > d_j(\mathbf{x})$, when $\mathbf{x}$ belongs to class ω_i. Common practice is to construct a decision boundary between two classes using the single discriminant function $d_{ij}(\mathbf{x}) = d_i(\mathbf{x}) - d_j(\mathbf{x}) = 0$. This has the property that $d_{ij}(\mathbf{x}) > 0$ if $\mathbf{x}$ is from class ω_i, and $d_{ij}(\mathbf{x}) < 0$ if $\mathbf{x}$ is from class ω_j [6, pp 579–580]. This approach works well when two classes are linearly separable and unimodal. However, when two classes are not linearly separable, overlap, or are multimodal, deriving a discriminant function that discriminates between the two classes is neither trivial nor obvious.

One approach to this problem is to learn the discriminant function from training data. Two commonly used method that achieve this are MLP neural networks and decision trees which form discriminant functions from multiple intersecting hyperplanes. In the research presented in this paper the discriminant functions are learnt using Evolutionary Computing techniques and take the form of hypersurfaces which may be nonlinear and discontinuous if necessary.

BMVC 2000 doi:10.5244/C.14.69

2 Evolutionary Computing

Biological evolution may be modeled as a two step process [11]. Step one is selection, and step two is random variation. Individuals living within an environment have a set of behaviours which are a response to this environment. Some of these behaviours are better suited to the environment and are able to exploit it. However, some individuals will exhibit behaviours that are not suitable for the environment and so they perish whilst others survive. This is selection. The survivors reproduce, either sexually or asexually, and pass to their offspring the genetics responsible for their behavioural traits. No individual is a perfect copy of its parent(s), since individual genotypes are subject to mutation. This random variation leads to new behaviours that may, or may not be better suited to the environment. The evolutionary cycle is then repeated.

Evolutionary Computing (EC) is a field of research concerned with optimization and search algorithms that mimic biological evolution [3]. These algorithms are collectively known as Evolutionary Algorithms (EAs). In this paper an EA is used to search the space of ATD algorithms for an algorithm with high performance. The procedure for achieving this is as follows. Initially, a population of candidate algorithms is randomly created. The population is then evaluated on a training image. This training image, or more specifically the feature vectors of the training image, is the populations environment. Due to their random creation, these initial algorithms do not perform ATD very well. However, some perform better than others, i.e. exhibit behaviours which are better suited to the environment. The top $x\%$ of the algorithms are then selected to be the parents of the next generation of algorithms. The selected algorithms are then randomly mutated to produce children which replace the bottom $(100 - x)\%$ of the population of algorithms. This process is repeated over many generations resulting in a population of high performing ATD algorithms.

2.1 Genetic Programming (GP)

One field of research in EC is GP [9]. GPs are encoded as tree structures in prefix notation. The internal nodes of a GP are known as the functions of the GP, and the leaf nodes are known as the terminals of the GP. If for example the GP function set was $\{+, \times\}$, and the terminal set was $\{a, b\}$ (which would be the feature vector in our application) then a possible GP we may construct is $\times \times a\ b + a\ a \equiv (a \times b) \times (a + a) = 2ba^2$. If we were to mutate the first symbol of our prefix tree (i.e $\times$) to a $+$ to form a new GP (an offspring) we would create $+ \times a\ b + a\ a = a(b + 2)$. For our application GP is used to learn discriminant functions. If the offspring GP provides better class discrimination than its parent, then it may be retained as a parent of future generations, else it may be culled.

3 The Environment Manipulating Mutable Automaton (EMMA)

3.1 EMMA's Architecture

As with any human written procedural program, the ATD algorithms undergoing evolution have a main program, and functions that are called by the main program. To facilitate

the mutation of the algorithms, the main program takes the form of a Finite State Automaton (FSA), and the functions are encoded GPs. Each state of the FSA has a GP embedded within it, in addition to two logical functions which are used to explicitly gather evidence as to a targets classification.

An offspring EMMA is produced by applying one, or all of the following mutations. Add a state, delete a state, change the start state, change a transition(s), change a state logical function(s), change a GP function(s), change a GP terminal(s), extend a GP sub-tree, and shrink a GP sub-tree. The use of these mutations allow the evolutionary process to design the architecture of the algorithm, a task normally performed by a human programmer.

3.2 Performing Classification with EMMA

The GP terminal set of EMMA is the feature set of the problem in hand. The features are combined using the GP function set to form discriminant functions. Selection of the GP functions and terminals is guided by the evolutionary process. Thus, feature selection is achieved as the algorithms evolve, a process which normally needs to be performed explicitly before a classification algorithm is trained.

The logical functions are used to combine the decision spaces when an objects classification is not clear cut. In real world imagery all targets do not appear the same. Targets may be large or small, bright or dull, or obscured. This being the case, a classifier that can perform in-class discrimination as well as inter-class discrimination is desirable. If we were to derive a discriminant function to perform target non-target discrimination, then the features that gave the best inter-class discrimination would be used. However, the feature vector may also contain features that give good in-class discrimination. The EMMA architecture allows the use of more than one discriminant function when performing classification. Each may have different features, thus allowing them to perform different tasks. The decisions made by these discriminant functions may then be combined logically before arriving at a classification. For example, let us assume that the first discriminant function, F_1, called by EMMA specializes in bright targets and the second, F_2, in dull. Correct classification of bright and dull targets could then be achieved with the statement **if** F_1 **OR** F_2 **then** target. Statements such as these are implicit in the EMMA architecture due to the FSA component.

EMMA is executed as follows assuming the start state to be state 0 (denoted q_0). A feature vector $\mathbf{x} = (x_1, \ldots, x_n)$ to be classified is presented to EMMA. The state discriminant function F_0, which may use 1 to n of the features, is then evaluated and is treated as the state input, and, in the start state, as the state output. A positive return from F_0 is treated as a logical true (binary 1), and a negative return as a logical false (binary 0). A transition to another state, say q_k, is then executed based on the value returned by F_0. On entering q_k, F_k is executed, and again the value returned is mapped to 0 or 1. The returned value is then combined with the output of the last state via one of the state logical functions, and forms the state output. Again, a transition to a new state occurs based on the value returned by F_k. This procedure continues until the maximum number of user defined transitions has occurred, or the halt state is entered. The final output is used as the classification, 1 indicating target, and 0 indicating non-target. The functionality of a state and its associated transitions is depicted in Table 1.

EMMA's architecture has a number of properties worthy of note. Classification of

State	Input I	Output O_i	Next State q_n
q_k	F_k	$O_i = \begin{cases} O_{i-1} \text{ AND } F_k, \text{ if } F_k < 0 \\ O_{i-1} \text{ OR } F_k, \text{ if } F_k \geq 0 \end{cases}$	$q_n = \begin{cases} q_m, \text{ if } F_k < 0 \\ q_h, \text{ if } F_k \geq 0 \end{cases}$

Table 1: Representation of a state q_k and its transitions.

$I : F_k \mapsto \begin{cases} 0 \text{ if } F_k < 0 \\ 1 \text{ if } F_k \geq 0 \end{cases}$	$c : O_i \mapsto O_{i-1}$ NAND I
$a : O_i \mapsto O_{i-1}$ AND I	$d : O_i \mapsto O_{i-1}$ NOR I
$b : O_i \mapsto O_{i-1}$ OR I	$e : O_i \mapsto O_{i-1}$ XOR I

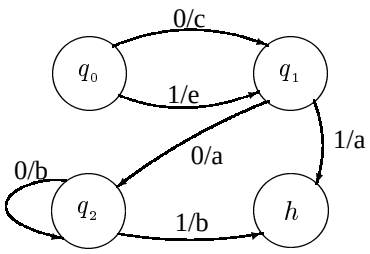
Table 2: Mappings used for shorthand notation.

multimodal or overlapping data is obtainable. This is because no one discriminant function need be learnt for any one class. If for example a class is bimodal, a discriminant function for each of the clusters can be learnt and concatenated with a logical OR. To achieve this EMMA has the ability to make a decision as to which discriminant function to call next, and how to logically combine it with previously called discriminant functions, based on the classification the current discriminant function is assigning to the sample. In addition, since the discriminant functions are constructed from the features, two samples from the same class may cause a discriminant function to return very different values. This in turn influences EMMA's decision as to which discriminant function to call next, and consequently causes different logical combinations to be formed. Thus, no two samples from the same class need be classified using the same features. Instead the most appropriate features for that sample are used. To the author's knowledge these properties, and EMMA's versatility, are not found explicitly in the literature, making this a novel approach. To clarify the classification process, a worked example is presented in Section 3.3.

3.3 Example Classification Problem

To enable visualization of the classification process, a two class example, with a two dimensional feature space is presented. The example chosen is that of classifying two intertwined spirals, which has been a challenge for pattern classification algorithms, and has been the subject of much work in the Neural Network community [1, 2, 10]. Each spiral is composed of 97 points, has a radius of 6.5, and has 3 revolutions. For this problem the feature vector $\mathbf{x} = (x_1, x_2)$ consists of the x, y coordinates of the points that form the spirals. Figure 2(a) shows the spirals, in which the circles represent class one and the squares represent class two. In Figure 2 (e) and (f), an incorrectly classified point from class one is represented by a star, and incorrectly classified point from class two is represented by a diamond. The GP terminal set used by EMMA for this problem is {x, y, $\mathcal{R}$,}, where x, y are the coordinates of a point on the spiral, and $\mathcal{R} \in (-1, 1)$ is a random constant, and the function set used is $\{+, -, \times, \div, \sin, \cos\}$.

Figure 1 shows an evolved three state EMMA, and its functions F_0, F_1, and F_2, that classifies each point of the two spirals correctly. This EMMA is used to demonstrate how classification is achieved. For completeness whilst working through the example,



$$F_0 = 4.180636962 \sin\left(3.251366387\, x + \frac{y^2}{x}\right)$$

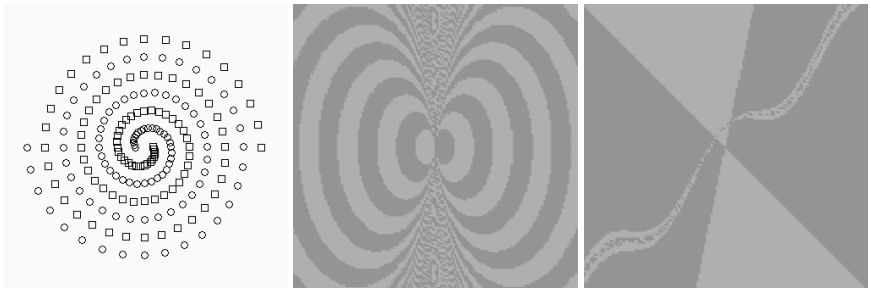
$$F_1 = \frac{(x - 0.205963\, y)\cos\left(\frac{\sin(0.192915\, x)}{\cos(x) - y + x}\right)}{(x + y)}$$

$$F_2 = \sin(3\, y - x)\,(-\cos(0.55221\, x) - \cos(\cos(xy)))$$

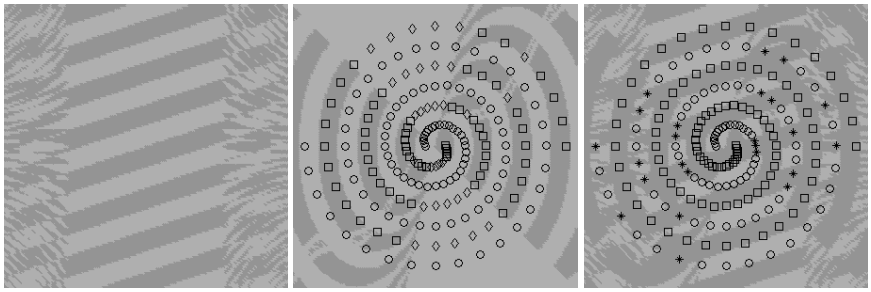
Figure 1: EMMA that solves the spiral problem. The start state is q_0. The notation α/β denotes an input of α and an output of β.

a full description of the state transitions and outputs is given rather than referring to the mappings of Table 2.

On entering the start state q_0, F_0 is executed and becomes both the input I and current output O_i. At this stage O_{i-1} and I cannot be logically combined by NAND or XOR (this states logical functions) since there has been no previous output O_{i-1}. A transition to state q_1 then occurs. On entering state q_1, F_1 is executed and provides the state input. If $F_1 \geq 0$ then the halt state is entered and the output is $O_i = O_{i-1}$ AND $F_1 \equiv F_0$ AND F_1. Referring to Figure 2 (c) and (e), the halt state is entered for all points which lay in the positive region defined by F_1, and their classification is given by F_0 AND F_1. All of these points are now correctly classified, and EMMA has determined that no further evaluation of them is necessary. If $F_1 < 0$ then a transition to state q_2 occurs, and the output is again F_0 AND F_1. On entering state q_2, F_2 is executed producing the state input. If $F_2 \geq 0$ then the halt state is entered and the output is $O_i = O_{i-1}$ OR $F_2 \equiv (F_0$ AND $F_1)$ OR F_2. Referring to Figure 2 (f), any point from class two that was incorrectly classified whilst in the previous state is now correctly classified. If $F_2 < 0$ then EMMA remains in the same state until the maximum number of allowed state transitions has been performed. EMMA then halts correctly classifying the points of class one which have not already been classified. A more efficient EMMA would have moved straight to the halt state for $F_2 < 0$ and still correctly classified the remaining points. However, no evolutionary pressure was applied to force halting before the maximum number of state transitions occurred, and so this repeated looping is expected.



(a) Two spirals to classify (b) F_0 (c) F_1



(d) F_2 (e) F_0 AND F_1 (f) (F_0 AND F_1) OR F_2

Figure 2: The output of the EMMA shown in Figure 1. The dark regions represent a positive output from the function, and the light a negative.

4 Application to Real World ATD Problem

EMMA was applied to the detection of ships within images produced from Synthetic Aperture Radar (SAR) carried on board the ERS-1[1] satellite. This problem was chosen since it has been tackled before by Foulkes and Booth [4, 5] using both Multi-Layer Perceptron (MLP), and Kohonen neural networks, and Howard et al. using two staged GP [8, 7]. Foulkes and Booth achieved their best results using the Kohonen neural network (which contained 12×12 nodes), and so these results are used as one of the benchmarks when assessing EMMA's performance. Before training their NNs Foulkes and Booth first ran a statistical bases target detection algorithm over the images. This detector did detect all targets but had a high false alarm rate. Foulkes and Booth trained their NNs on the positive returns from this detector thus reducing the task of the NNs from that of performing target detection on the whole image, to one of removing the false alarms of the statistical based detector. A similar approach was taken by Howard et al. They used GP to construct a ship detector but found that it gave a high false alarm rate. They then evolved a second GP that re-classified the mistakes made by the first. The results of this

[1] ERS: European Remote Sensing.

S_0	pixel intensity
S_1	average pixel intensity of a 3×3 window
S_2	average pixel intensity of a 5×5 window
S_3	average pixel intensity of a 7×7 window
S_4	average pixel intensity of a 9×9 window
S_5	average pixel intensity of the perimeter of a 3×3 window
S_6	average pixel intensity of the perimeter of a 5×5
S_7	average pixel intensity of the perimeter of a 7×7 window
S_8	average pixel intensity of the perimeter of a 9×9 window
S_9	standard deviation of pixel intensity of the perimeter of a 3×3 window
S_{10}	standard deviation of pixel intensity of the perimeter of a 5×5 window
S_{11}	standard deviation of pixel intensity of the perimeter of a 7×7 window
S_{12}	standard deviation of pixel intensity of the perimeter of a 9×9 window
S_{13}	$S_1 - S_6$
S_{14}	$S_1 - S_7$
S_{15}	$S_1 - S_8$

Table 3: Statistical features used for ship detection.

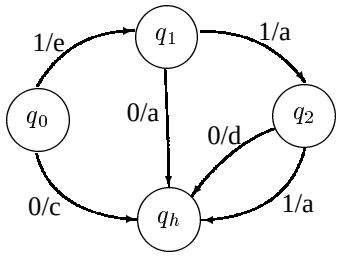
two stage GP strategy are used as the second benchmark for EMMA.

Five images were used in both these studies which were split into training and test sets. The training set consisted of a single training image and two validation images, and the test set consisted of two test images. The same images and ground truth were used to evaluate EMMA. On inspecting the images provided by Foulkes many of the ships indicated in the ground truth could not be seen (by a human). Within the training image there was supposed to be 77 ships. Of these the author felt that only 57 could be safely classified as ships, and so during training the remaining 20 were designated as ocean. This view was also taken by Howard et al.

To gauge the performance of their algorithms, Foulkes and Booth, and Howard et al. used the well established figure of merit, $FOM = \frac{N_{tt}}{N_{fa}+N_{gt}}$, where N_{tt} is the number of true target detections, N_{fa} is the number false alarms, and N_{gt} is the total number of ships in the ground truth data set [12, p 75]. For consistency this figure of merit is also used to gauge the performance of EMMA. Sixteen statistical features $S_0, \ldots, S_{15}$, were used in the ship detection problem which are the same as those used by Howard et al. These features form the GP terminal set and are given in Table 3. The GP function set used was $+, -, \times, \div, max,$ and min, where max returns the maximum of its two arguments, and min the minimum.

5 Experimental Results

An EMMA algorithm evolved to perform ship detection is shown in Figure 3, and its output on an unseen test image is shown in Figure 4. A comparison between this EMMA, the Kohonen NN, and the two stage GP is shown in Table 4. Table 4 shows that the performance of EMMA compares very favorably with the performance of both the Kohonen NN and two stage GP. The evolved EMMA is a very simple algorithm with only three computational states, and hence three discriminant functions. As can be seen from



$$
\begin{aligned}
F_0 =& (S_0 + S_{15}) - (S_{12} + 7.215) \\
F_1 =& min[S_{14}, S_{15}] + min[S_0, S_{12}] + S_0 \\
& - \left(min \left[5.095, \frac{S_{12}}{S_2} \right] + S_2 + S_8 + S_{11} + S_{12} + S_3 S_{12} \right) \\
F_2 =& S_{10} + \frac{S_9 S_{14} + S_{13} - S_6}{S_8 + S_{12} + S_{13} - 3.364}
\end{aligned}
$$

Figure 3: EMMA evolved to perform ship detection.

	FOM		
Image	Kohonen NN	Two stage GP	EMMA
Test 1	0.67	0.67	0.75
Test 2	0.72	0.69	0.83

Table 4: Performance comparison of Kohonen NN, two stage GP, and EMMA.

Figure 3 the algorithm may perform classification at any one of three stages. This allows a data sample to be classified with as little as three features, or up to eleven features depending on what the algorithm decides is most appropriate for that sample. Thus, feature selection has occurred at two levels. At a top level, eleven of the possible sixteen features have been selected for use in the problem, and at the individual data sample level where between three and eleven features may be used depending on the particular sample being classified.

6 Conclusions

In this paper a novel method of logically combining decision spaces has been presented. These decision spaces are formed by discriminant functions constructed from selected features from the feature set. Feature selection, the construction of the discriminant functions, and the construction of the algorithm architecture were all evolved (learnt) using Evolutionary Computing techniques. The evolved algorithms are very simple and were shown to out perform both Kohonen and MLP[2] neural networks, and two stage GP on the

[2]The Kohonen NN was shown to out perform an MLP in [4]

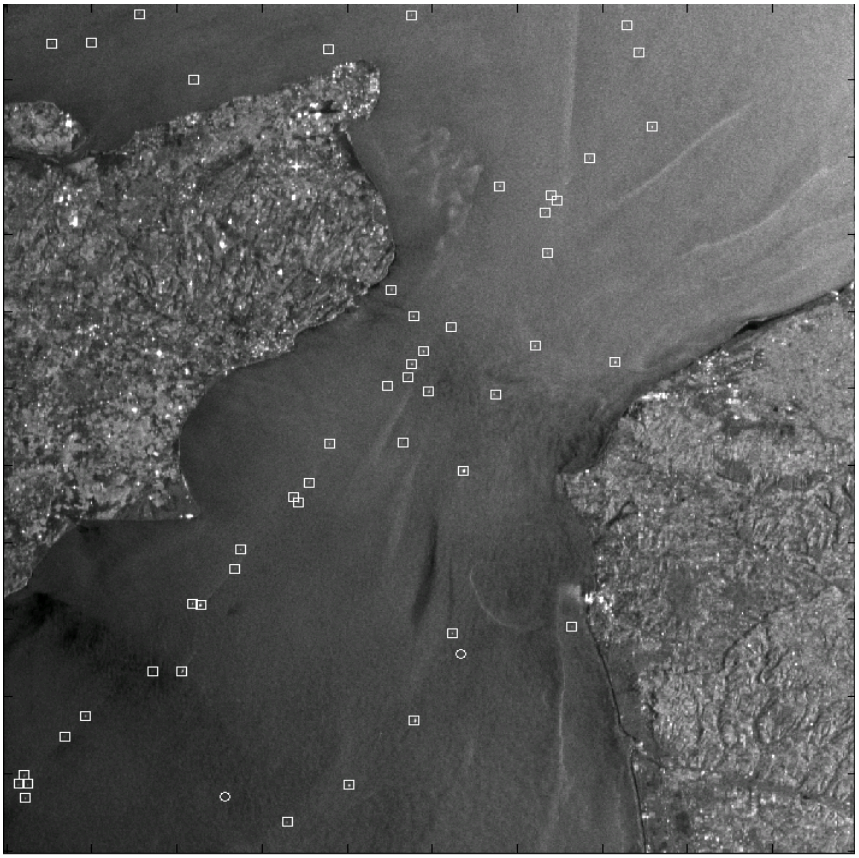


Figure 4: Output of EMMA on Test image 2. The squares are the true detections, and the two circles are the False detections.

problem of detecting ships in SAR imagery of the English channel, when gauging their performance with the figure of merit presented in Section 4.

Further research is now being carried out in which the FSA component of EMMA is being replaced by a Turing Machine.

7 Acknowledgements

The author would like to thank David Booth, James Cubillo, and Colin Reeves for their support whilst carrying out this research; and Steve Foulkes for providing the imagery and ground truth data.

References

[1] G Carpenter, S Grossberg, N Markuzon, J Raynolds, and D Rosen. Fuzzy artmap: A neural network architecture for incremental supervised learning of analog multi-dimensional maps. *IEEE Transactions on Neural Networks*, 3:698–713, 1992.

[2] S E Fahlman and C Lebiere. The cascade-correlation learning architecture. In Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan Kauffman, 1990.

[3] David B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 2(1):3–14, January 1994.

[4] Stephen B. Foulkes and David Booth. Ship detection in ESR-1 and radarsat SAR images using self-organising neural networks. Technical Report DERA/LS1/TR980309/1.0, DERA, April 1998.

[5] Stephen B. Foulkes and David Booth. 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*, 2000.

[6] Rafael C Gonzalez and Richard E Woods. *Digital Image Processing*. Addison Wessley, 1992.

[7] Daniel Howard, Simon C. Roberts, and Richard Brankin. Evolution of ship detectors for satellite SAR imagery. In *Genetic Programming, Second European Workshop, EuroGP'99*, pages 135–148. Springer, 1999.

[8] Daniel Howard, Simon C. Roberts, and Richard Brankin. Target detection in SAR imagery by genetic programming. *Advances in Engineering Software*, 30:303–311, 1999.

[9] John R Koza. *Genetic Programming: on the programming of computers by means of natural selection*. The MIT Press, 1992.

[10] Kevin J Lang and Michael J Witbrock. Learning to tell two spirals apart. In *Proceedings of the 1988 Connectionist Summer Schools*. Morgan Kaufman, 1988.

[11] E Mayr. *Toward a New Philosophy of Biology: Observations of an Evolutionist*. Cambridge,MA., 1988.

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