Coevolution and Linear Genetic Programming for Visual Learning

Krzysztof Krawiec[1] and Bir Bhanu

Center for Research in Intelligent Systems
University of California, Riverside, CA 92521-0425, USA
{kkrawiec,bhanu}@cris.ucr.edu

Abstract. In this paper, a novel genetically-inspired visual learning method is proposed. Given the training images, this general approach induces a sophisticated feature-based recognition system, by using cooperative coevolution and linear genetic programming for the procedural representation of feature extraction agents. The paper describes the learning algorithm and provides a firm rationale for its design. An extensive experimental evaluation, on the demanding real-world task of object recognition in synthetic aperture radar (SAR) imagery, shows the competitiveness of the proposed approach with human-designed recognition systems.

1 Introduction

Most real-world learning tasks concerning visual information processing are inherently complex. This complexity results not only from the large volume of data that one usually needs to process, but also from its spatial nature, information incompleteness, and, most of all, from the vast number of hypotheses that have to be considered in the learning process and the 'ruggedness' of the fitness landscape. Therefore, the design of a visual learning algorithm mostly consists in *modeling* its capabilities so that it is effective in solving the problem. To induce useful hypotheses on one hand and avoid overfitting to the training data on the other, some assumptions have to be made, concerning training data and hypothesis representation, known as *inductive bias* and *representation bias,* respectively. In visual learning, these biases have to be augmented by an extra '*visual bias*', i.e., knowledge related to the visual nature of the information being subject to the learning process. A part of that is general knowledge concerning vision (*background knowledge,* BK), for instance, basic concepts like pixel proximity, edges, regions, primitive features, etc. However, usually a more specific *domain knowledge* (DK) related to a particular *task/application* (e.g., fingerprint identification, face recognition, etc.) is also required.

Currently, most recognition methods make intense use of DK to attain a competitive performance level. This is, however, a double-edged sword, as the more

[1] On a temporary leave from Institute of Computing Science, Poznań University of Technology, Poznań, Poland.

DK the method uses, the more specific it becomes and the less general and transferable is the knowledge it acquires. The contribution of such over-specific methods to the overall body of knowledge is questionable.

Therefore, in this paper, we propose a general-purpose visual learning method that requires only BK and produces a complete recognition system that is able to classify objects in images. To cope with the complexity of the recognition task, we break it down into components. However, the ability to identify building blocks is a necessary, but not a sufficient, precondition for a successful learning task. To enforce learning in each identified component, we need an *evaluation function* that spans over the space of all potential solutions and guides the learning process. Unfortunately, when no *a priori* definition of module's 'desired output' is available, this requirement is hard to meet. This is why we propose to employ here cooperative coevolution [10], as it does not require the explicit specification of objectives for each component.

2 Related Work and Contributions

No general methodology has been developed so far that effectively automates the visual learning process. Several methods have been reported in the literature; they include blackboard architecture, case-based reasoning, reinforcement learning, and automatic acquisition of models, to mention the most predominant. The paradigm of evolutionary computation (EC) has also found applications in image processing and analysis. It has been found effective for its ability to perform global parallel search in high-dimensional search spaces and to resist the local optima problem. However, in most approaches the learning is limited to parameter optimization. Relatively few results have been reported [5,8,13,14], that perform visual learning in the deep sense, i.e., with a learner being able to synthesize and manipulate an entire recognition system.

The major contribution of this paper is a general method that, given only a set of training images, performs visual learning and yields a complete feature-based recognition system. Its novelty consists mostly in (i) *procedural representation* of features for recognition, (ii) utilization of *coevolutionary computation* for induction of image representation, and (iii) *a learning process* that optimizes the image feature definitions, prior to classifier induction.

3 Coevolutionary Construction of Feature Extraction Procedures

We pose visual learning as the search of the space of image representations (sets of features). For this purpose, we propose to use *cooperative coevolution* (CC) [10], which, besides being appealing from the theoretical viewpoint, has been reported to yield interesting results in some experiments [15]. In CC, one maintains many populations, with individuals in populations encoding only a *part* of the solution to the problem. To undergo evaluation, individuals have to be (temporarily) combined with individuals from the remaining populations to form an *organism* (solution). This joint evaluation scheme forces the populations to cooperate. Except for this evaluation step, other steps of evolutionary algorithm proceed in each population independently.

According to Wolpert's 'No Free Lunch' theorem [17], the choice of this particular search method is irrelevant, as the average performance of any metaheuristic search over a set of all possible fitness functions is the same. In the real world, however, not all fitness functions are equally probable. Most real-world problems are characterized by some features that make them specific. The practical utility of a search/learning algorithm depends, therefore, on its ability to detect and benefit from those features.

The *high complexity* and *decomposable nature* of the visual learning task are such features. Cooperative coevolution seems to fit them well, as it provides the possibility of breaking up a complex problem into components *without specifying explicitly the objectives for them*. The manner in which the individuals from populations cooperate *emerges* as the evolution proceeds. In our opinion, this makes CC especially appealing to the problem of visual learning, where the overall object recognition task is well defined, but there is no *a priori* knowledge about what should be expected at intermediate stages of processing, or such knowledge requires an extra effort from the designer.

In [3], we provide experimental evidence for the superiority of CC-based feature construction over standard EC approach in the standard machine learning setting; here, we extend this idea to visual learning. Following the *feature-based recognition* paradigm, we split the object recognition process into two modules: *feature extraction* and *decision making*. The algorithm learns from a finite training set of examples (images) D in a *supervised* manner, i.e. requires D to be partitioned into finite number of pairwise disjoint decision classes D_i.

In the coevolutionary run, n populations cooperate in the task of building the complete image *representation*, with each population responsible for evolving one component. Therefore, the cooperation here may be characterized as taking place at the *feature level*. In particular, each individual I from a given population encodes a single *feature extraction procedure*. For clarity, details of this encoding are provided in Section 4.

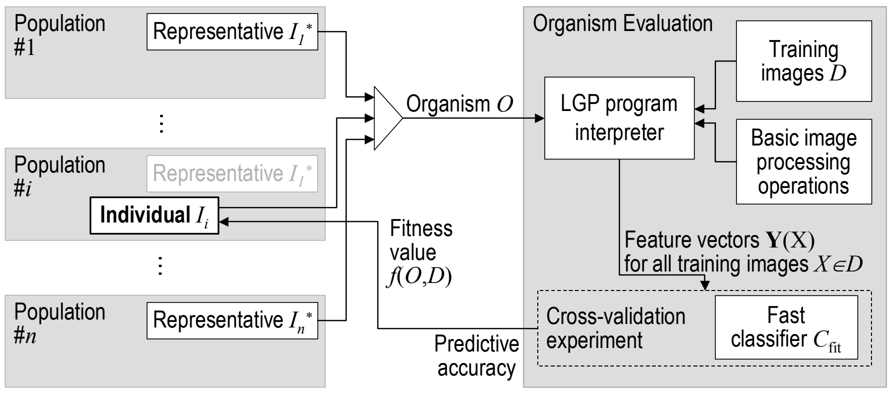

Fig. 1. The evaluation of an individual I_i from i^{th} population.

The coevolutionary search proceeds in all populations independently, except for the evaluation phase, shown in Fig. 1. To evaluate an individual I_j from population #*j*, we first provide for the remaining part of the representation. For this purpose, representatives I_i^* are selected from all the remaining populations *i≠j*. A representative I_i^* of *i*th population is defined here in a way that has been reported to work best [15]: it is the best individual w.r.t. the previous evaluation. In the first generation of evolutionary run, since no prior evaluation data is given, it is a randomly chosen individual.

Subsequently, I_j is temporarily combined with representatives of all the remaining populations to form an organism

$$O = \left\langle I_1^*, \ldots, I_{j-1}^*, I_j, I_{j+1}^*, \ldots, I_n^* \right\rangle . \quad \textbf{(1)}$$

Then, the feature extraction procedures encoded by individuals from *O* are 'run' (see Section 4) for all images *X* from the training set *D*. The feature values **y** computed by them are concatenated, building the compound feature vector **Y**:

$$\mathbf{Y}(X) = \left\langle \mathbf{y}(I_1^*, X), \ldots, \mathbf{y}(I_{j-1}^*, X), \mathbf{y}(I_j, X), \mathbf{y}(I_{j+1}^*, X), \ldots, \mathbf{y}(I_n^*, X) \right\rangle . \quad \textbf{(2)}$$

Feature vectors **Y**(*X*), computed for all training images *X*∈*D*, together with the images' decision class labels constitute the dataset:

$$\{ \langle \mathbf{Y}(X), i \rangle : \forall X \in D_i, \forall D_i \} \quad \textbf{(3)}$$

Finally, cross-validation, i.e. multiple train-and-test procedure is carried out on these data. For the sake of speed, we use here a fast classifier C_{fit} that is usually much simpler than the classifier used in the final recognition system. The resulting predictive recognition ratio (see equation 4) becomes the evaluation of the organism *O,* which is subsequently assigned as the fitness value to $f()$ the individual I_j, concluding its evaluation process:

$$\begin{aligned} f(I_j, D) &= f(O, D) = \\ &= card\left(\left\{\left\langle \mathbf{Y}(X), i \right\rangle, \forall X \in D_i \wedge C(\mathbf{Y}(X)) = i, \forall D_i, \right\}\right) / \; card(D) \end{aligned} \quad \textbf{(4)}$$

where *card*() denotes cardinality of a set. Using this evaluation procedure, the coevolutionary search proceeds until some stopping criterion (usually considering computation time) is met. The final outcome of the coevolutionary run is the best found organism/representation O^*.

4 Representation of Feature Extraction Procedures

For representing the feature extraction procedures as individuals in the evolutionary process, we adopt a variety of Linear Genetic Programming (LGP) [1], a hybrid of genetic algorithms (GA) and genetic programming (GP). The individual's genome is a fixed-length string of bytes, representing a sequential program composed of (possibly parameterized) basic *operations* that work on images and scalar data. This representation combines advantages of both GP and GA, being both procedural and more resistant to the destructive effect of crossover that may occur in 'regular' GP [1].

A feature extraction procedure accepts an image *X* as input and yields a vector **y** of scalar values as the result. Its operations are effectively calls to image processing and feature extraction functions. They work on *registers*, and may use them for both input as well as output arguments. *Image registers* store processed images, whereas *real-number registers* keep intermediate scalar results features. Each image register has single channel (grayscale), the same dimensions as the input image *X*, and maintains a rectangular *mask* that, when used by an operation, limits the processing to its area. For simplicity, the numbers of both types of registers are controlled by the same parameter *m*.

Each chunk of four consecutive bytes in the genome encodes a single operation with the following components:

(a) operation code,
(b) mask flag – decides whether the operation should be global (work on the entire image) or local (limited to the mask),
(c) mask dimensions (ignored if the mask flag is 'off'),
(d) arguments: references to registers to fetch input data and store the result.

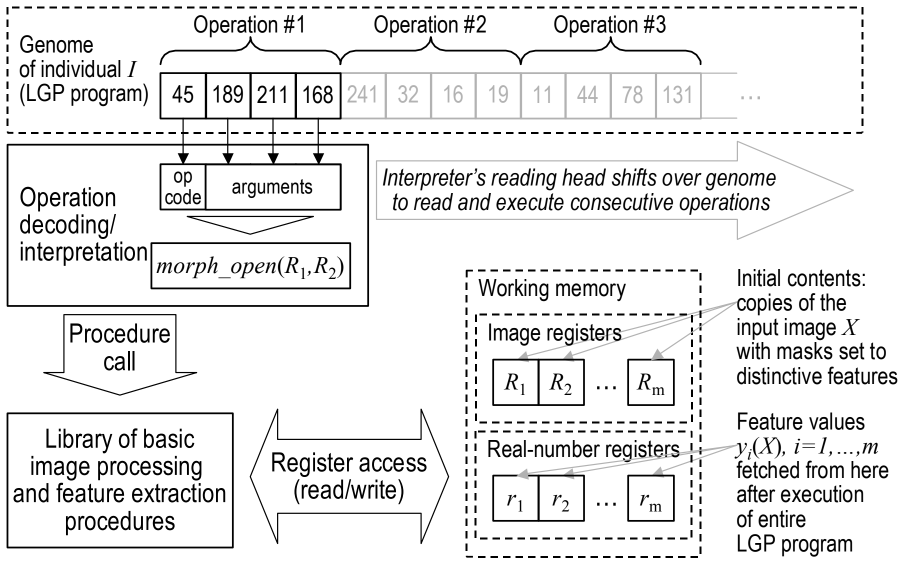
Genome
of individual I
(LGP program)
Operation #1
Operation #2
Operation #3
45
189
211
168
241
32
16
19
11
44
78
131
…
op code
arguments
Operation decoding/ interpretation
$morph\_open(R_1,R_2)$
Interpreter's reading head shifts over genome to read and execute consecutive operations
Procedure call
Library of basic image processing and feature extraction procedures
Register access (read/write)
Working memory
Image registers
R_1 R_2 … R_m
Real-number registers
r_1 r_2 … r_m
Initial contents: copies of the input image X with masks set to distinctive features
Feature values $y_i(X)$, $i=1,…,m$ fetched from here after execution of entire LGP program

Fig. 2. Execution of LGP code contained in individual's *I* genome (for a single image *X*).

Fig. 2 shows the execution at the moment of executing the following operation: morphological opening (a), applied locally (b) to the mask of size 14×14 (c) to the image fetched from image register pointed by argument #1, and storing the result in image register pointed by argument #2 (d). There are currently 70 operations implemented in the system. They mostly consist of calls to functions from Intel Image

Processing and OpenCV libraries, and encompass image processing, mask-related operations, feature extraction, and arithmetic and logic operations.

The processing of a single input image $X \in D$ by the LGP procedure encoded in an individual I proceeds as follows (Fig. 2):

1. *Initialization*: Each of the m image registers is set to X. The masks of images are set to the m most distinctive local features (here: bright 'blobs') found in the image. Real-number registers are set to the center coordinates of corresponding masks.
2. *Execution*: the operations encoded by I are carried out one by one, with intermediate results stored in registers.
3. *Interpretation*: the scalar values $y_j(I,X)$, $j=1,\ldots,m$, contained in the m real-value registers are interpreted as the output yielded by I for image X. The values are gathered to form an individual's output vector

$$\mathbf{y}(I,X) = \langle y_1(I,X),\ldots,y_m(I,X)\rangle, \qquad (5)$$

that is subject to further processing described in Section 3.

5 Architecture of the Recognition System

The overall recognition system consists of: (i) the best feature extraction procedures O^* constructed using the approach described in Sections 3 and 4, and (ii) classifiers trained using those features.

We incorporate a multi-agent methodology that aims to compensate for the suboptimal character of representations elaborated by the evolutionary process and allows us to boost the overall performance.

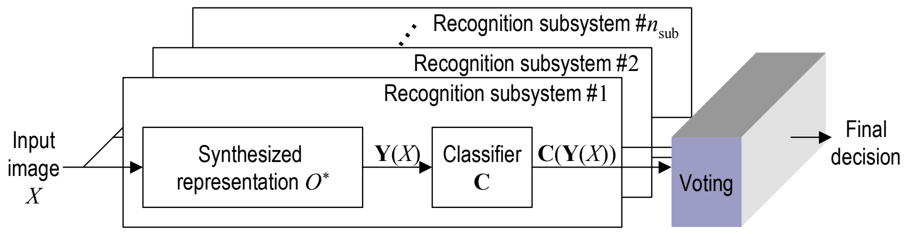

Fig. 3. The top-level architecture of recognition system.

The basic prerequisite for the agents' fusion to become beneficial is their *diversification*. This may be ensured by using homogenous agents with different parameter settings, homogenous agents with different training data (e.g., bagging [4]), heterogeneous agents, etc. Here, the diversification is naturally provided by the random nature of the genetic search. In particular, we run *many* genetic searches that start from different initial states (initial populations). The best representation O^* evolved in each run becomes a part of a single *subsystem* in the recognition system's architecture (see Fig. 3). Each subsystem has two major components: (i) a representation O^*, and (ii) a classifier **C** trained using that representation. As this classifier training is done once per subsystem, a more sophisticated classifier **C** may be used here (as compared to the classifier C_{fit} used in the evaluation function).

The subsystems process the input image X independently and output recognition decisions that are further aggregated by a simple majority voting procedure into the final decision. The subsystems are therefore homogenous as far as the structure is concerned; they only differ in the features extracted from the input image and the decisions made. The number of subsystems n_{sub} is a parameter set by the designer.

6 Experimental Results

The primary objective of the computational experiment is to test the scalability of the approach with respect to the number of decision classes and its sensitivity to various types of object distortions. As an experimental testbed, we choose the demanding task of object recognition in synthetic aperture radar (SAR) images. There are several difficulties that make recognition in this modality extremely hard:

- poor visibility of objects – usually only prominent scattering centers are visible,
- low persistence of features under rotation, and
- high levels of noise.

The data source is the MSTAR public database [12] containing real images of several objects taken at different azimuths and at 1-foot spatial resolution. From the original complex (2-channel) SAR images, we extract the magnitude component and crop it to 48×48 pixels. No other form of preprocessing is applied.

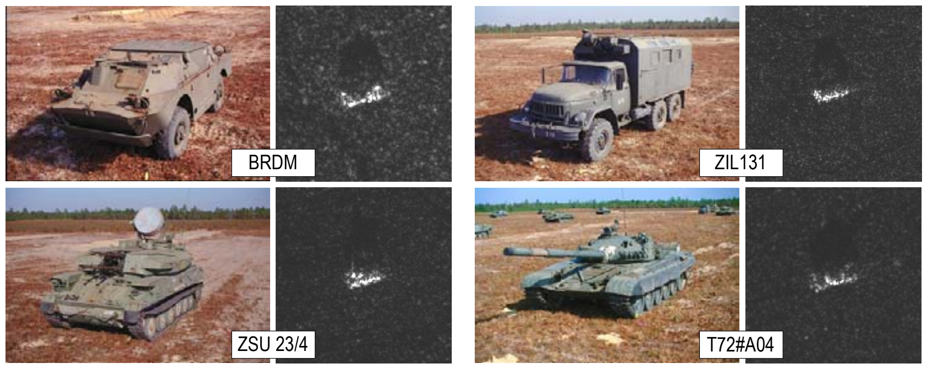

Fig. 4. Selected objects and their SAR images used in the learning experiment.

The following parameter settings are used for each coevolutionary run: number of subsystems n_{sub}: 10; classifier C_{fit} used for feature set evaluation: decision tree inducer C4.5 [11]; mutation operator: one-point, probability 0.1; crossover operator: one-point, probability 1.0, cutting allowed at every point; selection operator: tournament selection with tournament pool size = 5; number of registers (image and numeric) m: 2; number of populations n: 4; genome length: 40 bytes (10 operations);

single population size: 200 individuals; time limit for evolutionary search: 4000 seconds (Pentium PC 1.4 GHz processor).

A *compound* classifier **C** is used to boost the recognition performance. In particular, **C** implements the '1-vs.-all' scheme, i.e. it is composed of *l base classifiers* (where *l* is the number of decision classes), each of them working as a binary (two-class) discriminator between a single decision class and all the remaining classes. To aggregate their outputs, a simple decision rule is used that yields final class assignment only if the base classifiers are consistent and indicate a single decision class. With this strict rule, any inconsistency among the base classifiers (i.e., no class indicated or more than one class indicated) disables univocal decision and the example remains unclassified (assigned to 'No decision' category).

The system's performance is measured using different base classifiers (if not stated otherwise, the classifier uses default parameter settings as specified in [16]):

- support vector machine with polynomial kernels of degree 3 (trained using sequential minimal optimization algorithm [9] with complexity parameter set to 10),
- nonlinear neural networks with sigmoidal units trained using backpropagation algorithm with momentum,
- C4.5 decision tree inducer [11].

Scalability. To investigate the scalability of the proposed approach w.r.t. to the problem size, we use several datasets with increasing numbers of decision classes for a 15-deg. depression angle, starting from l=2 decision classes: BRDM2 and ZSU. Consecutive problems are created by adding the decision classes up to l=8 in the following order: T62, Zil131, a variant A04 of T72 (T72#A04 in short), 2S1, BMP2#9563, and BTR70#C71.

For i^{th} decision class, its representation D_i in the training data D consists of *two* subsets of images sampled uniformly from the original MSTAR database with respect to a 6-degree azimuth step. Training set D, therefore, always contains 2*(360/6)=120 images from each decision class, so its total size is 120*l. The corresponding test set T contains all the remaining images (for a given object and elevation angle) from the original MSTAR collection. In this way, the training and test sets are strictly disjoint. Moreover, the learning task is well represented by the training set as far as the azimuth is concerned. Therefore, there is no need for multiple train-and-test procedures here and the results presented in the following all use this single particular partitioning of MSTAR data.

Let n_c, n_e, and n_u, denote respectively the numbers of test objects correctly classified, erroneously classified, and unclassified by the recognition system. Figure 5(a) presents the *true positive rate*, i.e. $P_{tp}=n_c/(n_c+n_e+n_u)$, also known as *probability of correct identification* (PCI), as a function of the number of decision classes. It can be observed, that the scalability depends heavily on the base classifier, and that SVM clearly outperforms its rivals. For this base classifier, as new decision classes are added to the problem, the recognition performance gradually decreases. The major drop-offs occur when T72 tank and 2S1 self-propelled gun (classes 5 and 6, respectively), are added to the training data; this is probably due to the fact that these objects are visually similar to each other (e.g., both have gun turrets) and significantly resemble the T62 tank (class 3). On the contrary, introducing consecutive classes 7 and 8 (BMP2 and BTR60) did not affect the performance much; more than this, an improvement of accuracy is even observable for class 7.

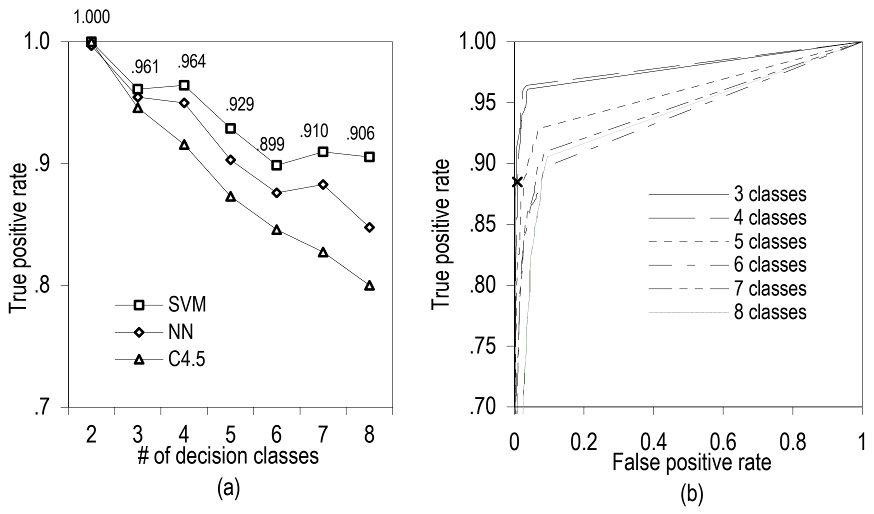

Fig. 5. (a) Test set recognition ratio as a function of number of decision classes. (b) ROC curves for different number of decision classes (base classifier: SVM).

Figure 5(b) shows the *receiver operating characteristics* (ROC) curves obtained, for the recognition systems using SVM as a base classifier, by modifying the confidence threshold that controls whether the classifier votes. The false positive rate is defined here as $P_{fp}=n_e/(n_c+n_e+n_u)$. Again, the results support our method: the curves do not drop rapidly as the false positive rate decreases. Therefore, very high *accuracy of classification*, i.e., $n_c/(n_c+n_e)$, may be obtained when accepting a reasonable *rejection rate* $n_u/(n_c+n_e+n_u)$. For instance, for 4 decision classes, when P_{fp}=0.008, P_{tp}=0.885 (see marked point in Fig. 5(b)), and, therefore, rejection rate is 1-(P_{fp}+P_{tp})=0.107, the accuracy of classification equals 0.991.

Object variants. A desirable property of an object recognition system is its ability to recognize different variants of the same object. This task may pose some difficulties, as configurations of vehicles often vary significantly. To provide a comparison with human-designed recognition system, we use the conditions of the experiment reported in [2]. In particular, we synthesized recognition systems using:

- 2 objects: BMP2#C21, T72#132,
- 4 objects: BMP2#C21, T72#132, BTR70#C71, and ZSU23/4.

For both of these cases, the testing set includes two *other* variants of BMP2 (#9563 and #9566), and two *other* variants of T72 (#812 and #s7).

The results of the test set evaluation shown in the confusion matrices (Table 1) suggest that, even when the recognized objects differ significantly from the models provided in the training data, the approach is still able to maintain high performance.

Here the true positive rate P_{tp} equals 0.804 and 0.793, for 2- and 4-class systems, respectively. For the cases where a decision can be made (83.3% and 89.2%, respectively), the values of classification accuracy, 0.966 and 0.940, respectively, are comparable to the forced recognition results of the human-designed recognition algorithms reported in [2], which are 0.958 and 0.942, respectively. Note that in the test, we have not used 'confusers', i.e. test images from different classes that those present in the training set, as opposed to [2], where BRDM2 armored personnel carrier has been used for that purpose.

Table 1. Confusion matrices for recognition of object variants.

		Predicted class							
		2-class system			*4-class system*				
Test objects		BMP2	T72	No	BMP2	T72	BTR	ZSU	No
Object	*Serial #*	[#C21]	[#132]	decision	[#C21]	[#132]	[#C71]	[#d08]	decision
BMP2	[#9563,9566]	**295**	18	78	**293**	27	27	1	43
T72	[#812,s7]	4	**330**	52	12	**323**	1	9	41

7 Conclusions

In this contribution, we provide experimental evidence for the possibility of synthesizing, without or with little human intervention, a feature-based recognition system which recognizes 3D objects at the performance level that can be comparable to handcrafted solutions. Let us emphasize that these encouraging results are obtained in the demanding field of SAR imagery, where the acquired images only roughly depict the underlying 3D structure of the object.

There are several major factors that contribute to the overall high performance of the approach. First of all, the paradigm of *coevolution* allows us to decompose the task of representation (feature set) construction into several semi-independent, cooperating subtasks. In this way, we exploit the inherent modularity of the learning process, without the need of specifying explicit objectives for each developed feature extraction procedure. Secondly, the approach manipulates LGP-encoded feature extraction *procedures*, as opposed to most approaches which are usually limited to learning meant as parameter optimization. This allows for learning sophisticated features, which are novel and sometimes very different from expert's intuition, as may be seen from example shown in Figure 6. And thirdly, the *fusion at feature and decision level* helps us to aggregate sometimes contradictory information sources and build a recognition system that is comparable to human-designed system performance with a bunch of simple components at hand.

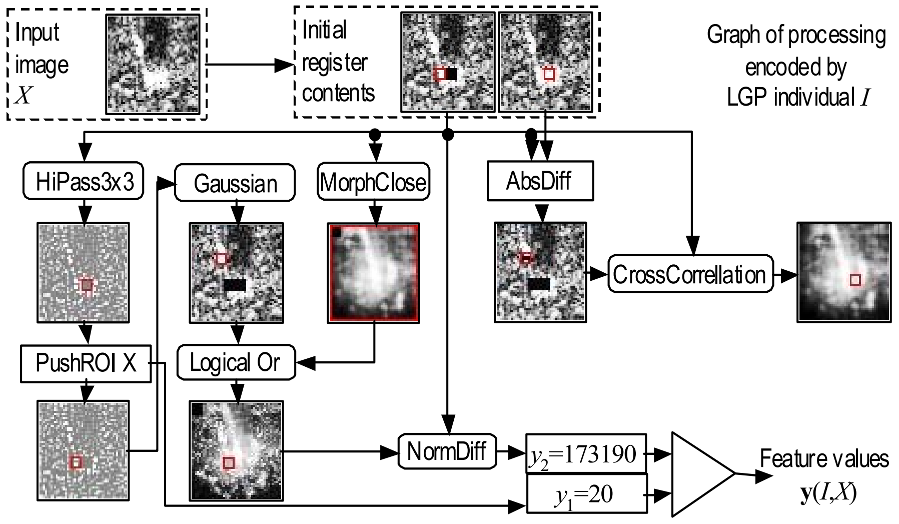

Fig. 6. Processing carried out by one of the evolved procedures shown as a graph (small rectangles in images depict masks; boxes: local operations; rounded boxes: global operations).

Acknowledgements

This research was supported by the grant F33615-99-C-1440. The contents of the information do not necessarily reflect the position or policy of the U. S. Government. The first author is supported by the Polish State Committee for Scientific Research, research grant no. 8T11F 006 19. We would like to thank the authors of software packages: ECJ [7] and WEKA [16] for making their software publicly available.

References

1. Banzhaf, W., Nordic, P., Keller, R., Francine, F.: Genetic Programming. An Introduction. On the automatic Evolution of Computer Programs and its Application. Morgan Kaufmann, San Francisco, Calif. (1998)
2. Bhanu, B., Jones, G.: Increasing the discrimination of SAR recognition models. Optical Engineering 12 (2002) 3298–3306
3. Bhanu, B. and Krawiec, K.: Coevolutionary construction of features for transformation of representation in machine learning. Proc. Genetic and Evolutionary Computation Conference (GECCO 2002). AAAI Press, New York (2002) 249–254
4. Breiman, L.: Bagging predictors. Machine Learning 24 (1996) 123–140
5. Draper, B., Hanson, A., Riseman, E.: Knowledge-Directed Vision: Control, Learning and Integration. Proc. IEEE 84 (1996) 1625–1637
6. Krawiec, K.: On the Use of Pair wise Comparison of Hypotheses in Evolutionary Learning Applied to Learning from Visual Examples. In: Perner, P. (ed.): Machine Learning and Data

Mining in Pattern Recognition. Lecture Notes in Artificial Intelligence, Vol. 2123. Springer Verlag, Berlin (2001) 307–321.
7. Luke, S.: ECJ Evolutionary Computation System. http://www.cs.umd.edu/projects/plus/ec/ecj/ (2002)
8. Peng, J., Bhanu, B.: Closed-Loop Object Recognition Using Reinforcement Learning. IEEE Trans. on PAMI 20 (1998) 139–154
9. Platt, J.: Fast Training of Support Vector Machines using Sequential Minimal Optimization. In: Schölkopf, B., Burges, C., Smola, A. (eds.): Advances in Kernel Methods - Support Vector Learning. MIT Press, Cambridge, Mass. (1998)
10. Potter, M.A., De Jong, K.A.: Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents. Evolutionary Computation 8 (2000) 1–29
11. Quinlan, J.R.: C4.5: Programs for machine learning. Morgan Kaufmann, San Mateo, Calif. (1992)
12. Ross, T., Worell, S., Velten, V., Mossing, J., Bryant, M.: Standard SAR ATR Evaluation Experiments using the MSTAR Public Release Data Set. SPIE Proc.: Algorithms for Synthetic Aperture Radar Imagery V, Vol. 3370, Orlando, FL (1998) 566–573
13. Segen, J.: GEST: A Learning Computer Vision System that Recognizes Hand Gestures. In: Michalski, R.S., Tecuci, G., (eds.): Machine Learning. A Multistrategy Approach. Volume IV. Morgan Kaufmann, San Francisco, Calif. (1994) 621–634
14. Teller, A., Veloso, M.: A Controlled Experiment: Evolution for Learning Difficult Image Classification. Proc. 7th Portuguese Conference on Artificial Intelligence. Springer Verlag, Berlin, Germany (1995) 165–176
15. Wiegand, R.P., Liles, W.C., De Jong, K.A.: An Empirical Analysis of Collaboration Methods in Cooperative Coevolutionary Algorithms. Proc. Genetic and Evolutionary Computation Conference (GECCO 2001). Morgan Kaufmann, San Francisco, Calif. (2001) 1235–1242
16. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Francisco, Calif. (1999)
17. Wolpert, D., Macready, W.G.: No Free Lunch Theorems for Search. Tech. Report SFI-TR-95-010, The Santa Fe Institute (1995)