Visual Learning by Coevolutionary Feature Synthesis

Krzysztof Krawiec and Bir Bhanu, *Fellow, IEEE*

***Abstract*—In this paper, a novel genetically inspired visual learning method is proposed. Given the training raster images, this general approach induces a sophisticated feature-based recognition system. It employs the paradigm of cooperative coevolution to handle the computational difficulty of this task. To represent the feature extraction agents, the linear genetic programming is used. The paper describes the learning algorithm and provides a firm rationale for its design. Different architectures of recognition systems are considered that employ the proposed feature synthesis method. An extensive experimental evaluation on the demanding real-world task of object recognition in synthetic aperture radar (SAR) imagery shows the ability of the proposed approach to attain high recognition performance in different operating conditions.**

***Index Terms*—Automatic programming, feature extraction, genetic algorithms, pattern recognition.**

I. Introduction

As it is widely known, computers are good at storing, transmitting, and low-level processing of data, but their ability of autonomous knowledge acquisition falls far behind that of humans. This becomes an important bottleneck of many practical applications, where human assistance is impossible or expensive. Therefore, adaptation, learning, and self-organization are probably the only ways to cope with the enormous data flood that overwhelms these applications.

Most real-world learning tasks are inherently complex. This complexity results not only from the large volume of data that the learning algorithm often needs to process but from the information incompleteness and imperfectness as well. As a result, the number of hypotheses that have to be considered is usually large if no additional constraints are given. Therefore, the design of a learning algorithm mostly consists of *modeling* and *constraining* its capabilities so that it is effective in solving the problem. To develop a useful classifier/recognition system, which successfully solves the given learning task on one hand and avoids overfitting to the training data on the other, some assumptions have to be made. These assumptions concern training data and hypothesis representation and are known in the Machine Learning (ML) community as *inductive bias* and *representation bias*, respectively.

Visual learning is the process of autonomous acquisition of knowledge from training image data that aims at solving a given visual task (e.g., recognition, detection, tracking, etc.). Here, the complexity of the learning process grows even further. The volumes of data to be processed are usually larger than in standard ML problems and the data itself exhibits spatial nature, while the particular elementary "granules" (e.g., bitmap pixels) convey little information. Therefore, the aforementioned 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 this is general knowledge concerning vision [*background knowledge* (BK)], for instance, basic concepts like pixel proximity, edges, regions, primitive features, etc. However, in most cases, this is not enough, and 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 extensive use of DK to attain a competitive performance level. This is, however, a double-edged sword, as the more DK the method uses, the more specific it becomes, and the less general and transferable the knowledge it acquires will be.

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 decompose it into components ("building blocks"). 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 the module's "desired output" is available, this requirement is hard to meet. Therefore, in this paper, we propose to employ the paradigm of cooperative coevolution [1], as it does not require explicit specification of objectives for each of the components.

In Section II, we describe related work and our contributions. Section III presents coevolutionary construction and representation of feature extraction procedures. Section IV describes experimental results for various architectures of the recognition system. Finally, Section V presents the conclusions of this paper.

II. Related Work and Contributions

Current recognition systems are mostly open loop, and human input in the design of these systems is still predominant. Only a few contributions, which are summarized in Table I, attempt to close the feedback loop of the learning process at the highest (e.g., recognition) level *and* propose methods that cope with the demanding requirements of real-world tasks. Among them, mostly [3] and [4] exhibit some commonalities with the approach presented in this paper (e.g., use of off-the-shelf operators). Note that only a few approaches [3]–[5], [7]–[10] have been reported that learn using *raw* images as the training data and produce the *entire* object recognition system. Moreover,

Manuscript received August 11, 2003; revised October 24, 2004. This work was supported in part by the Polish Committee for Scientific Research under Grant KBN 3 T11C 050 26 and by Air Force Research Laboratory under Grant F33615-99-C-1440. The contents of the information do not necessarily reflect the position or policy of the U. S. Government. This paper was recommended by Associate Editor Lawrence O. Hall.

K. Krawiec is with the Institute of Computing Science, Poznań University of Technology, Poznań, Poland (e-mail: krawiec@cs.put.poznan.pl).

B. Bhanu is with the Center for Research in Intelligent Systems, University of California, Riverside, CA 92521 USA (e-mail: bhanu@cris.ucr.edu).

Digital Object Identifier 10.1109/TSMCB.2005.846644

TABLE I
RELATED WORK IN VISUAL LEARNING (EC—EVOLUTIONARY COMPUTATION, GP—GENETIC PROGRAMMING, LGP—LINEAR GENETIC PROGRAMMING, CC—COOPERATIVE COEVOLUTION, NN—NEURAL NETWORK)

Reference		Approach	Experimental task	Training data
(Draper, 1993)	[2]	Learning recognition graphs	Recognize buildings	High-level CV concepts
(Segen, 1994)	[13]	Learning of object models	Hand gesture recognition	Graphs extracted from images
(Johnson, 1995)	[11]	EC (GP)	Locating hand in human body silhouette	Binary silhouettes
(Teller & Veloso, 1995)	[10]	EC (GP variant)	Face recognition	Raw images (grayscale)
(Peng & Bhanu, 1998)	[8]	Reinforcement learning	Segmentation of in/outdoor scenes	Raw images (color)
(Peng & Bhanu, 1998)	[9]	Delayed reinforcement learning	Segm. And feature extraction, in/outdoor	Raw images (color)
(Draper et al., 1999)	[4]	Markov processes	Rooftop classification in aerial imaging	Raw images (grayscale)
(Bhanu & Peng, 2000)	[5]	Biased reinforcement learning	Segmentation of outdoor scenes	Raw images (color)
(Draper et al., 2001)	[6]	Markov processes	Recognize common household objects	Raw images (color)
(Krawiec, 2001)	[24]	EC (GP)	Handwriting recognition	Raw images (grayscale)
(Rizky et al., 2002)	[18]	Hybrid EC (GP+NN)	Target recognition in radar modality	1-D radar signals
(Maloof et al., 2003)	[12]	Standard ML/PR classifiers	Rooftop detection in aerial imagery	Fixed set of scalar features
(Bulitko et al., 2003)	[3]	Markov processes	Forest inventorization	Raw images (color)
(Levner et al., 2003)	[14]	Markov processes	Forest inventorization	Raw images (color)
(Lin & Bhanu, 2004)	[15]	EC (CC+GP)	Object recognition in radar modality	Raw image (grayscale)
This contribution		EC (CC+LGP)	Object recognition in radar modality	Raw image (grayscale)

some of the methods [11]–[13] use domain-specific knowledge (DK) and are highly specialized toward a particular application. For a comparison of the effect of domain-specific versus general knowledge in a framework of evolutionary computation for object recognition, see [15].

The approach proposed in this paper is based on evolutionary computation (EC). The methods listed in Table I involve blackboard architecture, case-based reasoning, reinforcement learning, and Markov processes, to mention the most predominant. The EC paradigm has found applications in image processing and analysis [15], [17]. It has been found to be 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 [3], [4], [10], [11], [18] 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 of i) *procedural representation* of features for recognition, ii) utilization of *coevolutionary computation* for induction of image representation, iii) a *learning process* that optimizes the image feature definitions, prior to classifier induction, and iv) application to SAR images.

III. COEVOLUTIONARY CONSTRUCTION AND REPRESENTATION OF PROCEDURES

A. Feature Extraction Procedures

It is widely recognized that the scalability of basic learning methods with respect to the complexity of the task is usually limited. For instance, one can successfully train a neural network to recognize characters, but this result cannot be extrapolated to, e.g., interpretation of outdoor scenes. This observation motivated several recent research trends in related disciplines, e.g., multiagent systems in Artificial Intelligence, ensembles of classifiers in Machine Learning, and multiple classifiers in Pattern Recognition. Results obtained there clearly indicate that further progress cannot be made without resorting to some higher level learning methodologies that inherently involve *modularity* (see, e.g., [19]).

To provide modularity, we propose to use *cooperative coevolution* (CC) [1], which, besides being appealing from the theoretical viewpoint, has been reported to yield interesting results in some experiments [20]. As opposed to EC, where the individual solutions coexist in the same population, in CC, one maintains *many* populations, with each individual in population encoding only a *part* of the solution to the problem (see Table II). 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 the evolutionary algorithm proceed in each population independently. As the algorithm advances and joint evaluation takes place, each population specializes in and works on a specific part of the problem. Thus, this methodology is especially useful for *modular problems*, where it is possible to decompose the task, but the resulting components are not independent (for more thorough study on task modularity, see, e.g., [21]).

According to Wolpert's "No Free Lunch" theorem [22], 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 identify and benefit from those features.

The *high complexity* and *decomposable nature* of the task are important features of visual learning. Cooperative coevolution seems to fit it 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. This makes CC especially appealing to the problem of visual

TABLE II
COMPARISON OF EC AND CC ALGORITHMS (MAJOR DIFFERENCES IN BOLDFACE)

Evolutionary Computation (EC)	Cooperative Coevolution (CC)
solution ≡ an individual in population	solution ≡ an **organism** composed of individuals selected from different populations

initialize population
loop
  evaluate individuals
  store best individual
  select mating candidates
  recombine parents and use their offspring as the next generation
until stopping condition
return best individual
	initialize **populations**
	loop
	  evaluate **organisms** O and assign fitness to individuals in populations
	  store **best organism**
	  ***for each* population**
	  select mating candidates
	  recombine parents and use their offspring as the next generation
	  end for
	until stopping condition
	return **best organism** O^*

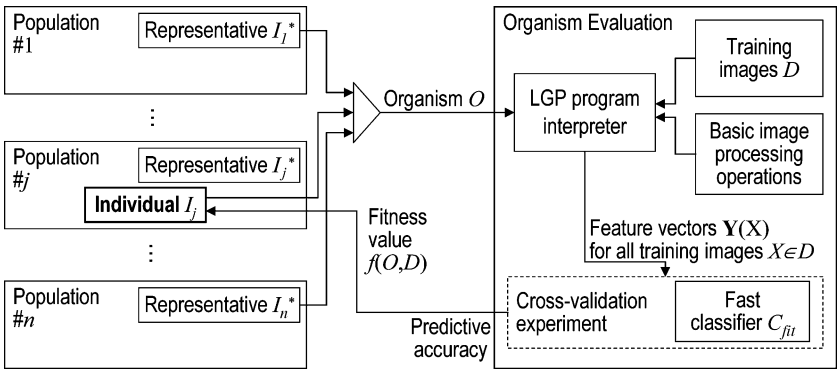


Fig. 1. Evaluation of an individual I_i from ith population.

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 [23], we provide experimental evidence for the superiority of CC-based feature construction over the standard EC approach in a conventional machine learning setting (learning from examples described by fixed-length vectors of features). In [24], we extend this idea to visual learning and come to analogous conclusion. Following the *feature-based object recognition* paradigm, we split the object recognition process into two steps: *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 a finite number of pairwise disjoint decision classes D_i, $i = 1 \ldots l$.

We pose visual learning as a search in the space of *image representations* (sets of features). In the coevolutionary run, n populations cooperate in the task of building the complete image representation, where each population is 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 constitutes a *module* (component) of the complete solution and encodes a single *feature extraction procedure*, i.e. mapping from the space of input images to the space of scalar features. For clarity, details of this encoding are provided in a separate Section III-B.

The coevolutionary search proceeds in all populations independently, except for the evaluation phase, which is shown in Fig. 1. To evaluate an individual I_j from population $\#j$, we first provide the remaining part of the representation. For this purpose, representatives I_i^* are selected from all the remaining populations $i \neq j$. A representative I_i^* of the ith population is defined here in a way that has been reported to work best [20]. It is the best individual w.r.t. the previous evaluation. In the first generation of an 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

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

Then, the feature extraction procedures encoded by individuals from O are "run" (see Section III-B) for all images X from the training set D. The feature values $\mathbf{y}$ computed by them are concatenated, building the compound feature vector $\mathbf{Y}$:

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

Feature vectors $\mathbf{Y}(X)$, which are computed for all training images $X \in D$, together with the images' decision class labels constitute the dataset:

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

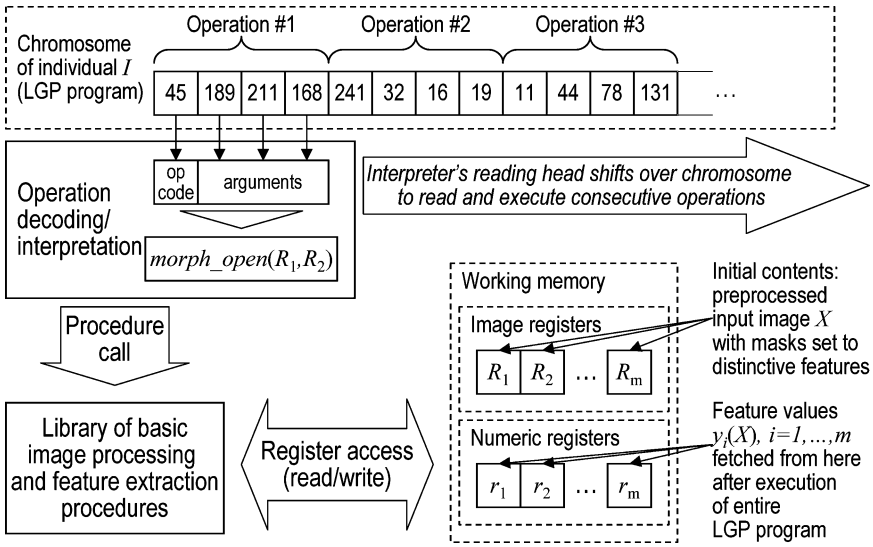


Fig. 2. Execution of LGP code contained in individual's I chromosome (for a single image X).

Finally, cross-validation, i.e. multiple train-and-test procedure is carried out on these data. For the sake of speed, here, we use a fast classifier C_{fit} that is usually much simpler than the classifier used in the final recognition system. The resulting predictive recognition ratio in (4), shown at the bottom of the page, becomes the evaluation of the organism O, which is subsequently assigned as the fitness value $f()$ to the individual I_{j}, concluding its evaluation process in (4), where card(.) denotes cardinality of a set. Using this evaluation procedure, the coevolutionary search proceeds until some stopping criterion is met. The final outcome of the coevolutionary run is the best found organism/representation O^*.

B. 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) [25], which is a hybrid of genetic algorithms (GAs) and genetic programming (GP).

The LGP representation combines advantages of both GP and GA. The individual's *chromosome* is a fixed-length string of bytes, representing a sequential program composed of (possibly parameterized) basic operations that are given *a priori*; this feature makes LGP similar to GP. On the other hand, as opposed to GP, where tree-like expressions are maintained, LGP encodes such procedures in the form of a fixed-length sequence that is essentially equivalent to GA representation at the chromosome level. LGP encoding is, therefore, more "positional," i.e., the evolutionary process tends to bind some "meaning" to a particular code fragment. As a consequence, the standard crossover operator used in LGP exchanges mutually corresponding code fragments. In GP, on the contrary, the standard crossover picks at random subtrees in parent solutions and exchanges them. This action most often affects unrelated code fragments, and such crossover is, in most cases, nothing more than mere mutation. Therefore, LGP is more resistant to the destructive effect of crossover that may occur in "regular" GP [25].

In the proposed approach, LGP procedures are encoded in fixed-length sequences of bytes and work on both the image and scalar data. A feature extraction procedure accepts an image X as input and yields a vector $\mathbf{y}$ of scalar values as the result. Its elementary operations are effectively calls to image processing functions, to feature extraction functions, and to other data processing operations. They work on *registers* and may use them for both input as well as output arguments. *Image registers* (R_{j}, $j = 1 \ldots m$) store processed images, whereas *numeric registers* (r_{j}, $j = 1 \ldots m$) keep intermediate scalar results and final feature values. For simplicity, the numbers of both types of registers are controlled by the same parameter m. Each image register is single channel (grayscale), has the same dimensions as the input image X, and maintains a rectangular *mask*. The mask may be used by an operation (if it is *local*) and then limits the processing to its area. *Global* operations ignore masks and operate on the entire image.

Each chunk of four consecutive bytes in the chromosome 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"), and d) arguments—references to registers to fetch input data and store the result. As an illustration, Fig. 2 shows the execution at the moment of carrying out the following operation: (a) morphological opening, (b) applied locally, with (c) a mask of size 14×14, to the image (d) fetched from image register R_1 pointed out by the first argument and storing the result in image register R_2 pointed out by the second argument. Other registers are not affected by this operation.

$$f(I_j, D) = f(O, D) = \frac{\text{card}\left(\{\langle \mathbf{Y}(X), i\rangle, \forall X \in D_i \wedge C_{\text{fit}}\left(\mathbf{Y}(X)\right) = i, \forall D_i, i = 1, \ldots, l\}\right)}{\text{card}(D)} \tag{4}$$

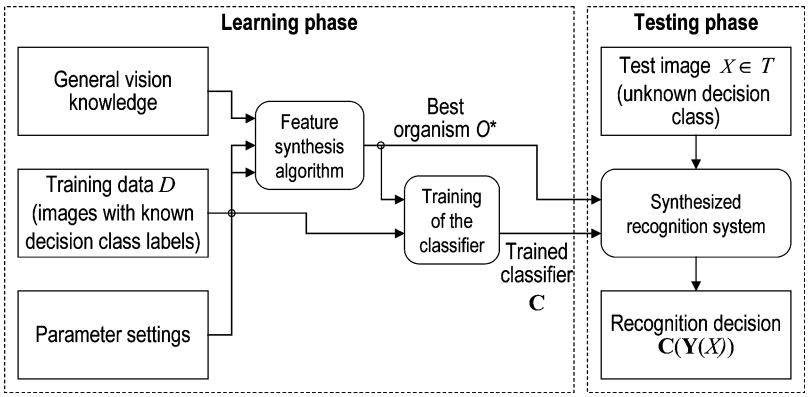


Fig. 3. Framework of the computational experiment.

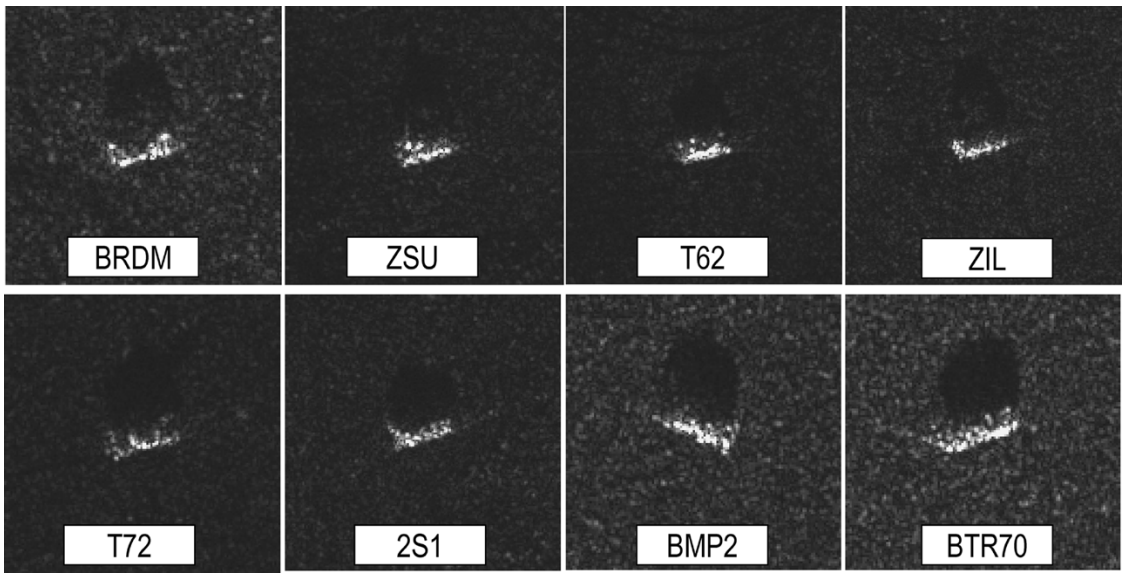


Fig. 4. Exemplary SAR images used in the learning experiment.

The set of elementary operations may be adjusted to a particular application and vision task. However, to demonstrate the universal character of the proposed approach, in the experiments involving real-world images described in this paper, only general-purpose image processing and feature extraction procedures have been used (see Table III in Section IV-A).

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

1) *Initialization*: Each of the m image registers R_{i} is set to an image resulting from preprocessing X by an image filter (global processing). The choice of these filters is determined by an initial fragment (4 bytes) of an individual's chromosome (see the example in Section IV-B2 for a detailed explanation). The masks of registers are set to the m most salient local features (here: bright "blobs") found in the image. Numeric registers r_{i} 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.

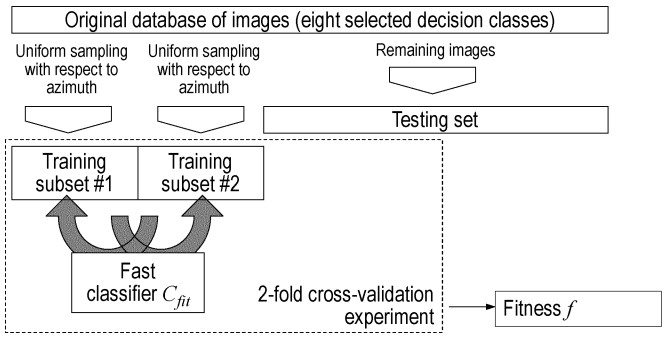


Fig. 5. Division of the original data into training and testing part and its usage for fitness computation.

3) *Exploitation*: The scalar values $y_{\mathrm{j}}(I, X), j = 1, \ldots, m$ contained in the m numeric registers after program execution are interpreted as the output yielded by I for image X. The values are gathered to form an individual's output vector

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

that is subject to subsequent processing [see (2)–(4)].

TABLE III
ELEMENTARY LGP OPERATIONS USED IN THE EXPERIMENTS

Image → Image	Image → Scalar value
Convolution filters (for mask sizes 3x3 and 5x5): Prewitt, Sobel, Laplacian, Gaussian, Highpass, Lowpass, Sharpening	**Image norms:** Dot product, L_1 (city-block), L_2 (Euclidean)
Other filters: Median filter, Min filter, Thresholding, Normalized cross-correlation	**Feature extraction operations:** Spatial 2D moments (up to 3rd order), Central 2D moments (up to 3rd order), Normalized central 2D moments (up to 3rd order), Mass center, Location of the brightest pixel, Location of the darkest pixel,
Image transforms: 2-D Fast Fourier Transform	Number of non-zero pixels, Sum, Average, Standard deviation of pixel intensities,
Morphological operations: Erosion, Dilatation, Opening, Closing	
Image arithmetic (pixelwise): Absolute difference, Addition, Subtraction, Multiplication,	
Image logic operations (pixelwise): And, Or, Xor	
Mask-related operations: Set rectangular mask, Set mask upper left corner, Set mask lower right corner, Shift mask in specific direction, Get mask height, Get mask width, Get mask mid X, Get mask mid Y	
Scalar value → Scalar value	
Scalar arithmetic: + - * /	
Scalar functions: Max, Min, Abs, Sgn, If (conditional expression), Sin, Cos, Tan, Exp, Log	

IV. EXPERIMENTS WITH DIFFERENT SYSTEM ARCHITECTURES

A. Data and Parameter Setting

The primary objective of the computational experiment is to evaluate the proposed approach within different experimental settings, including varying task difficulty (binary and multiple-class tasks, different number of decision classes) and sensitivity to object distortions. The overall framework of all the experiments described hereafter is presented in Fig. 3. In the learning phase, we perform evolutionary synthesis of feature extraction procedures described in Section III. Different training data is used for this purpose, whereas the parameter settings and vision knowledge (the set of elementary operations) remains the same. After finishing the evolutionary run, the best organism O^* found in the search is used to compute features for all images from the training set. Next, a classifier $\mathbf{C}$ is trained on that data. The best organism O^* together with the trained classifier constitute the complete recognition system. In the testing phase, the recognition system is applied to the test images to verify its recognition performance.

In all the following experiments, the decision tree induction algorithm C4.5 [26] (with the default parameter setting as specified in [27]) is used in the feature synthesis algorithm as the classifier $\mathrm{C}_{\mathrm{fit}}$ for fitness computation [see (4)]. This choice was mostly motivated by C4.5's low time complexity of both hypothesis (tree) induction (training) and hypothesis querying (testing). Time is a critical factor here, as training and testing takes place for each evaluated individual in each generation.

After feature synthesis, we need to train the classifier on all of the training data; as this is done once per recognition system, a more sophisticated (and more time-consuming in training and/or querying) classifier $\mathbf{C}$ may be used here (compared to $\mathrm{C}_{\mathrm{fit}}$). Here, we use two classifiers. In some runs, we use a C4.5 decision tree inducer as $\mathbf{C}$. In other experiments, $\mathbf{C}$ is a support vector machine (SVM) with polynomial kernels of degree 3 (trained by means of the sequential minimal optimization algorithm [28] with complexity parameter set to 10).

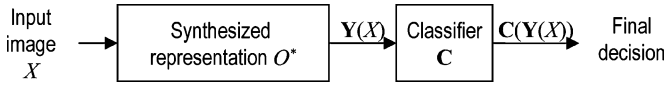


Fig. 6. Top-level architecture of basic recognition system.

As an experimental testbed, we chose the demanding task of object recognition in SAR images. There are several difficulties that make recognition in this modality extremely hard:

- nonliteral nature of images, i.e., the images appear different than the images in the visible band of the spectrum, and one needs to understand the physics of SAR image formation in order to interpret them;
- poor visibility of objects—usually only prominent scattering centers are visible;
- low persistence of features under rotation;
- low resolution and high levels of noise.

The data source is the MSTAR public database [29], containing real images of several objects taken at different azimuths and at 1-ft spatial resolution. In this study, we consider only images acquired at 15° elevation angle. Fig. 4 presents selected representatives of all eight decision classes used in the experiments: BRDM truck (armored personnel carrier), ZSU gun, T62 tank, ZIL truck, T72 tank, 2S1 gun, BMP2 tank, and BTR transporter. From the original complex (two-channel) SAR images, we extract the magnitude (real) component and crop it to 48 × 48 pixels. No other form of preprocessing is applied.

Fig. 5 outlines the division of the original MSTAR collection into training and test data. For the ith decision class, $i = 1 \ldots l$, $l = 8$, its representation D_i in the training data $D = \bigcup_{i=1}^{l} D_i$ consists of *two* subsets of images sampled from the original MSTAR database. For both subsets, this sampling is made as uniformly as possible with respect to a 6° azimuth step. Training set D, therefore, always contains $2 * (360/6) = 120$ images from each decision class, so its total size is $120 * l$, where l is the number of decision classes. The corresponding test set T contains all the remaining images (for a given object and elevation angle) from the original MSTAR collection; there are 155 of them for each decision class. In this way, the training and test

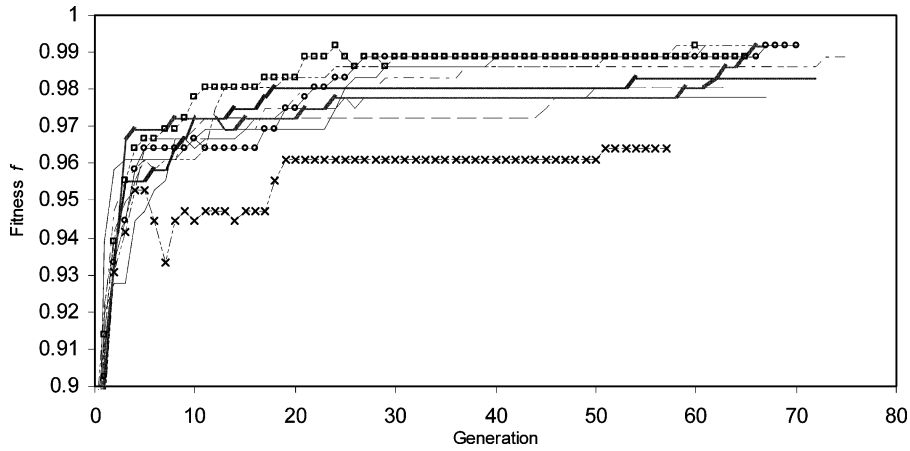


Fig. 7. Fitness charts of the best individuals for the evolutionary learning process run on the $B2$ problem.

sets are strictly disjoint, and the learning task is well represented by the training set as far as the azimuth is concerned. Therefore, the results presented in the following approximate well the results that could be obtained using multiple train-and-test procedures like cross-validation.

In experiments, all the parameters have been kept constant (if not stated otherwise) to provide a comparison of results. For simplicity, we decided to keep the number m of image registers and numeric registers as low as possible. This implies setting $m = 2$, as some of the elementary LGP operations are binary and need two registers to fetch input arguments. The number of cooperating populations n is set to 4. These settings together imply that a single coevolutionary synthesis process produces an organism O^* encoding eight feature definitions, with each population delivering two of them. Thus, there are $n = 4$ vectors $\mathbf{y}$ in (2), and each one of them contains two elements (feature values). In each population, an individuals' chromosome is a string of 36 bytes that corresponds to $(36 - 4)/4 = 8$ LGP operations (first four bytes of chromosome encode register initialization; see Section IV-B2 for details).

The experiments have been run on the implementation of the method that offers 70 elementary LGP operations listed in Table III. They mostly consist of functions imported from the Intel Image Processing library (IPL, [30]), and OpenCV library [31] and encompass image processing, mask-related operations, feature extraction, and arithmetic and logic operations. The core modules of the system have been implemented in Java, with the Java Native Interface providing communication with libraries written in C (OpenCV) and machine code (IPL). The high-level part of the approach (evolutionary computation and machine learning) uses the publicly available Evolutionary Computation in Java package (ECJ, [32]) and Waikato Environment for Knowledge Analysis (WEKA, [27]), which is also written in Java.

Other parameters of the experiment have been set as follows—mutation operator: one-point, probability 0.1; crossover operator: one-point, probability 1.0, chromosome cutting allowed at every point; selection operator: tournament selection with tournament pool size: 5 (a compromise between 2 and 7 used commonly in genetic algorithms and genetic programming, respectively); single population size: 200 individuals; time limit for evolutionary search: 4000 s (Pentium PC with a 1.4-GHz processor). This setting for time limit was motivated by the practical aspect of our approach. We evaluate the quality of results that may be obtained within a reasonable time of approximately 1 hr, and we want to enable a fair comparison of different variants of the method.

B. Binary Classification Task

To illustrate the proposed approach, let us first consider the simple two-class experimental setting. The overall architecture of the recognition system for this case is depicted in Fig. 6. It consists of two modules: i) the best feature extraction procedure O^* constructed using the approach described in Section III and ii) a classifier trained using those features.

The task is the recognition of the positive decision class D^+, represented here by the BRDM vehicle. The objects representing the remaining categories build up the negative class D^-, $D^+ \cap D^- = \emptyset$. We run several experiments of different difficulty, starting with D^-, containing images from single class (ZSU); let us denote this task by $B1$. Next, we define subsequent tasks, which are denoted hereafter as $B2$ to $B7$, by extending D^- by other vehicles in the following order: T62, Zil131, T72, 2S1, BMP2, and BTR70. In all these tasks, D^+ remains constant and contains images of the BRDM vehicle.

On each of these seven binary classification problems $B1 \ldots B7$, ten independent synthesis processes have been run to provide a statistical significance of obtained results. Each run started with a different, randomly created, initial population of feature synthesis programs. Fig. 7 presents fitness charts of the best individuals for the evolutionary learning process run on the $B2$ problem, i.e., BRDM (D^+) versus ZSU and T62 (D^-). Each data series depicts a complete evolutionary run; as each run is repeated ten times, there are ten series in Fig. 7. All learning processes attain fitness over 0.9 within the first three generations. The fitness $f(O^*)$ of the best organisms found in O^* varies from 0.964 to 0.992, depending on the run. Runs end up in different generations (57th to 75th), as they evolve feature extraction procedures that require different amounts of time when executed, and the stopping condition concerns the time limit (4000 s). Note that this learning process seems to be quite

TABLE IV
TRUE POSITIVE (TP) AND FALSE POSITIVE (FP) RATIOS FOR BINARY RECOGNITION TASKS (TESTING SET, **SINGLE** RECOGNITION SYSTEMS). TABLE PRESENTS AVERAGES OVER TEN INDEPENDENT SYNTHESIS PROCESSES AND THEIR 0.95 CONFIDENCE INTERVALS

	C4.5		SVM	
Task	TP	FP	TP	FP
B1	.987 ± .007	.042 ± .016	.966 ± .032	.022 ± .019
B2	.960 ± .013	.040 ± .011	.935 ± .027	.010 ± .005
B3	.892 ± .025	.035 ± .005	.929 ± .030	.017 ± .005
B4	.901 ± .023	.036 ± .007	.929 ± .032	.013 ± .002
B5	.860 ± .030	.033 ± .007	.880 ± .039	.014 ± .005
B6	.733 ± .055	.026 ± .004	.762 ± .063	.018 ± .009
B7	.654 ± .041	.039 ± .006	.610 ± .069	.012 ± .004

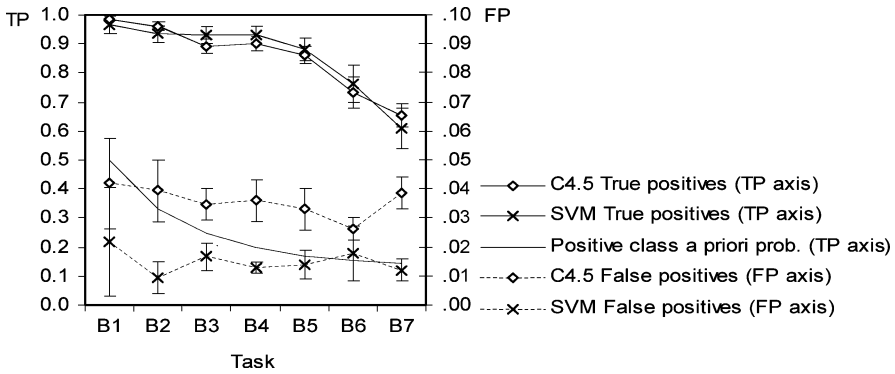


Fig. 8. True positive (TP) and false positive (FP) ratios for binary recognition tasks (testing set, **single** recognition systems). Chart presents averages over ten independent synthesis processes and their 0.95 confidence intervals.

resistant to the problem of local minima: several runs exhibit improvement, even after long periods of leveling off.

The fitness graphs in Fig. 7 reflect the evolution and behavior of the recognition systems on the training data. The performances of the synthesized recognition systems on the test data are presented in Table IV and Fig. 8, which give true positive ($\mathrm{TP} = \mathrm{P}(\text{positive decision}|\text{positive example})$) and false positive ($\mathrm{FP} = \mathrm{P}(\text{positive decision}|\text{negative example})$) ratios, i.e., their averages and 0.95 confidence intervals over ten independent runs. It may be observed that in all the experiments, recognition systems using C4.5 and SVM perform similarly. At the first sight, this may seem surprising, taking into account the simplicity of C4.5, especially its limited capability of fusing and combining attributes. On the other hand, the synthesized features are especially well suited for C4.5, as this induction algorithm is used for fitness computation in the process of feature synthesis. In terms of machine learning, the features generated are *biased* toward C4.5.

Not surprisingly, the true positive ratios of synthesized recognition systems decrease as the complexity of the problem, controlled by the number of other objects used in the negative class, increases, lowering the *a priori* probability of the positive class (see Fig. 8). Nevertheless, the results obtained are still impressive if we keep in mind that the classifier operates in the space spanned over only eight scalar features computed by the best organism O^* from the raw, difficult-to-recognize, raster images. Let us also point out that objects BMP2 and BTR70, which are used in problems $B6$ and $B7$ and are the last two instances of the problem, resemble visually the positive class BRDM to a great extent (see Fig. 4). Note also that *a priori* probabilities of the positive class in these instances are relatively low, amounting to 0.15 and 0.14, respectively.

In terms of false positives, all the synthesized systems perform well. Here, SVM outperforms C4.5 in a statistically significant way (significance level 0.01), exceeding 2% FP only for problem $B1$ (recognizing BRDM (D^+) versus ZSU (D^-)). Compared to C4.5, SVM reduces the FP rate relatively by 32% (for task $B6$) to 75% (for task $B2$).

1) Adaptive Cooperative Coevolution: The results presented in Table IV and Fig. 8 have been obtained with $n = 4$ populations, each of them evolving $m = 2$ features. Determining the number of populations n required to attain acceptable performance on a particular task prior to test set evaluation may be hard in general. Therefore, we developed a variant of the approach, which we termed *adaptive cooperative coevolution* (ACC), which adapts the number of cooperating populations to the difficulty of the problem. The coevolutionary algorithm starts with a single population ($n = 1$). In this special case, the organism on which the algorithm works is composed of a single part (individual). In this configuration, evolution proceeds until saturation, i.e., until the fitness of the best organism does not improve for a certain number of generations (here: 5). In such a case, a new, randomly initialized population is added to the cooperation, and the evolutionary process continues with

TABLE V
TRUE POSITIVE (TP) AND FALSE POSITIVE (FP) RATIOS FOR BINARY RECOGNITION TASKS (TESTING SET, **SINGLE** RECOGNITION SYSTEMS, **ADAPTIVE CC**). MEAN AND MAXIMAL NUMBER OF INDIVIDUALS THAT FORM THE BEST ORGANISM FOUND IN RUNS. AVERAGES OVER TEN INDEPENDENT SYNTHESIS PROCESSES, AND THEIR 0.95 CONFIDENCE INTERVALS ARE SHOWN

	C4.5		SVM		# of learned populations	
Task	TP	FP	TP	FP	Mean	Maximum
B1	.973 ± .012	.050 ± .019	.981 ± .010	.012 ± .008	5.6	7
B2	.969 ± .011	.033 ± .010	.972 ± .012	.013 ± .008	5.1	7
B3	.904 ± .025	.036 ± .008	.940 ± .026	.014 ± .006	5.8	6
B4	.888 ± .031	.026 ± .006	.908 ± .035	.021 ± .011	5.3	6
B5	.816 ± .036	.028 ± .006	.856 ± .038	.015 ± .003	5.0	6
B6	.736 ± .038	.037 ± .008	.723 ± .058	.018 ± .006	4.9	6
B7	.652 ± .062	.027 ± .007	.698 ± .082	.014 ± .003	4.4	5

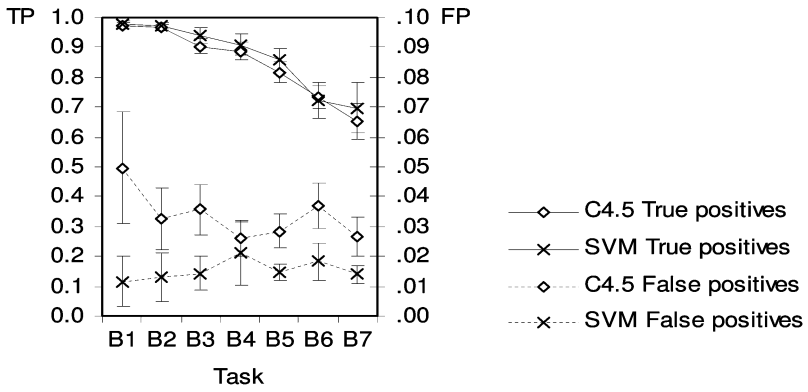


Fig. 9. True positive and false positive ratios for binary recognition tasks (testing set, **single** recognition systems, **adaptive CC**). Chart presents averages over ten independent synthesis processes and their 0.95 confidence intervals.

two populations. Consecutive saturations of the evolutionary search cause the addition of other populations. However, with n populations at hand, the extension to $n+1$ populations is allowed only if an improvement has been observed since the insertion of the nth population.

Table V and Fig. 9 present results of the evolutionary runs carried out using the above algorithm. Table V also depicts the mean and maximum number of individuals that form the best organism found in the ACC runs. For instance, for the B1 task, the ACC ends up with 5.6 populations on the average, which corresponds to $5.6 \times m = 11.2$ features. These figures decrease as the complexity of the problem grows. This is due to the fact that the runs on more difficult problems usually last for a smaller number of generations (where the fitness function is more time consuming). As a result, within the fixed time limit of 4000 s per evolutionary run, the ACC algorithm has fewer opportunities to add new populations onto the difficult problems. This phenomenon is illustrated in Fig. 10, where each point represents the final state of the evolutionary run concerning tasks B1–B7. Clearly, for stopping criteria other than time, the more difficult tasks would result in a significantly greater number of populations.

These results suggest that the test set performance of the recognition systems synthesized using ACC do not differ much from those obtained using standard CC. The slight differences

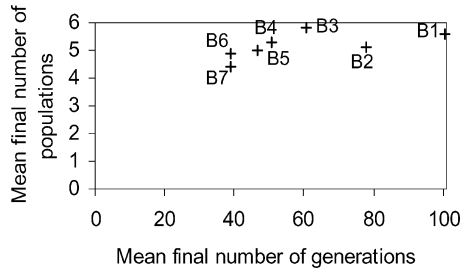
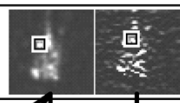
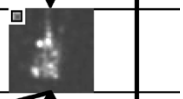
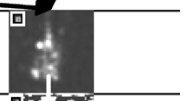


Fig. 10. Final number of populations and generations reached in ACC experiment for learning tasks of different difficulty (B1–B7).

that are observed in both TP and FP ratios are not statistically significant. We can, therefore, draw a positive conclusion that ACC allows the attainment of results that are not worse than those obtained by CC, with the advantage of relieving the system designer from estimating the number of cooperating populations n.

2) Illustrative Example of Evolved Procedures: To illustrate the process of synthesis of feature extraction procedures, we present an example of a complete evolved organism. The solution given here is the best organism O^* found in one of the learning processes concerning binary classification tasks described at the beginning of Section IV-B—more precisely, the $B1$ task (BRDM (D^+) versus ZSU (D^-)). This particular

TABLE VI
PROCESSING CARRIED OUT BY ONE OF THE EVOLVED ORGANISMS (INDIVIDUAL 1 OF 4; SEE TEXT FOR DETAILS)

			Numeric registers		Image registers	
	Operation	Arguments	r1	r2	R1	R2
	Initial register contents (input image after initial, genome-dependent preprocessing)		20.0	-3.0		
1	Scalar multiplication	r2, r1, [r1]	-60.0			
2	Move mask to minimum brightness	[R1], [r2], [r1]	126.0	3.0		
3	Normalized central moment	R1, [r1]	0.0			
4	Image dot product (pixelwise, global)	R1, R2, [r2]		2577.0		
5	Median filter	R1, [R2]				
6	Average brightness	R2, [r1]	2.1			
7	Move mask's lower right corner to specified point	[R1], r2, r1				
8	Highpass filter 5x5 (global)	R2, [R1]				
Final feature values			**2.1**	**2577.0**		

organism has perfect fitness (1.0) on the training set and attains TP and FP ratios of 0.974 and 0.058, respectively, on the testing set, when combined with C4.5 classifier, and 0.974 and 0.0, respectively, when used together with an SVM classifier.

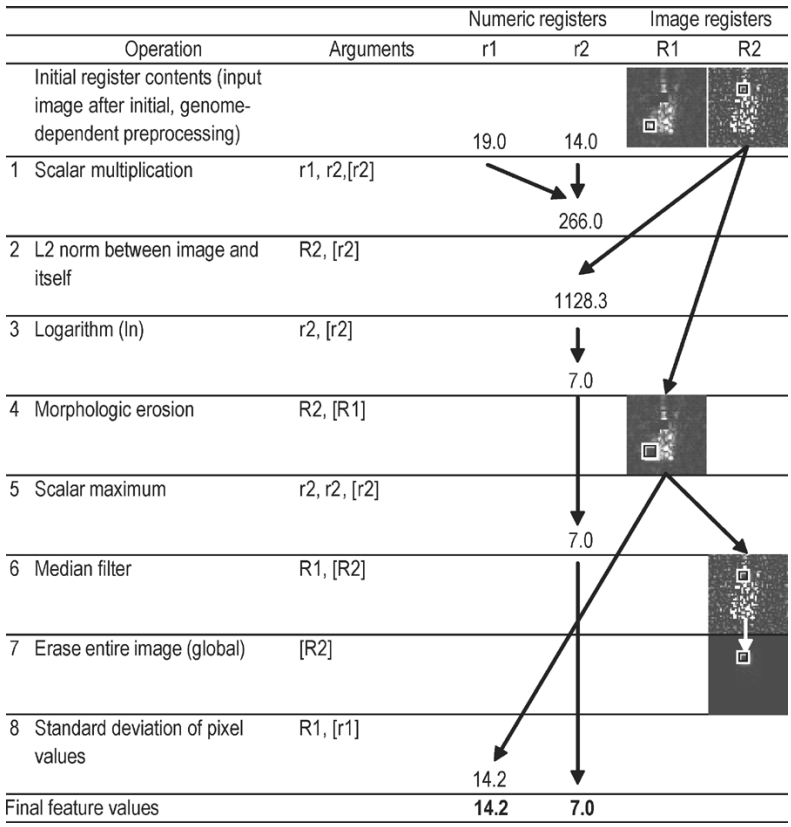
Tables VI to IX illustrate the processing carried out by this organism for a selected image representing the negative class (ZSU in this experiment), taken at 6° azimuth (see Fig. 11). These tables depict LGP procedures encoded by particular individuals of the considered organism. Each row in these tables corresponds to the execution of a single elementary operation.

The first table row contains the initial register contents, which is different for particular individuals, as it is determined by the individual's LGP code (chromosome). Technically, the LGP execution engine, before carrying out the procedure, initializes the image registers by the original input image passed through one of 12 unary (Image → Image) operations (filters). The first chunk of the LGP code determines which filter is to be applied to which image register. The masks of the registers are initially set to the brightest spot, and the numeric registers are initialized by mask coordinates. This chromosome-dependent initialization method proved useful in preliminary experiments, speeding up the convergence by enabling the LGP code to start with an already preprocessed input image and providing more diversity among individuals. It also causes the effective LGP code to be shorter by one chunk (4 bytes); this is why, although originally, the parameter determining LGP code length has been set to nine operations (implying a chromosome length of 36 bytes), the effective number of operations is 8.

The original binary LGP code is not presented, as it would not be readable. Rather than that, in each row, the first column presents the textual description of the operation being carried out, whereas the second column contains the argument lists. An argument list contains references to numeric registers (r_1 and r_2) and/or image registers (R_1 and R_2). Registers in square brackets are "output" or "input–output" arguments, i.e., their contents change when the operation is executed; if no brackets are present, the register referred to is "input" (read only) and is used only to fetch the input data for the operation. Each of the last four table columns corresponds to a particular register and illustrates how its contents change during LGP procedure execution. For clarity, tables contain values only at those places where the contents of a register have been changed; a blank table cell means that the value of the register has not been influenced by a particular operation. Arrows illustrate data flow or, in other words, dependencies between particular nodes of the processing graph.

Small black-and-white boxes inside images mark the current position of the image mask. Local operations limit their processing to that mask; global ones ignore them. Mask position and size may be controlled by the LGP procedure, either explicitly (see, for instance, operation #7 in Table VI and operation #5 in Table IX) or as a side effect of some image processing operations (e.g., operation #4 in Table VII). As a consequence,

TABLE VII
PROCESSING CARRIED OUT BY ONE OF THE EVOLVED ORGANISMS (INDIVIDUAL 2 OF 4; SEE TEXT FOR DETAILS)



a particular LGP procedure may apply and use a different mask position/size, depending on the input image. Any violations of required ranges of scalar values (e.g., mask corner coordinate exceeding the actual image dimension) are handled by modulo operation.

Note that some operations involve some extra parameters that are not fetched from the registers but are encoded directly in the LGP code. For clarity, such "constant" parameters are not shown in these examples. For instance, they determine the orders of geometrical moments to be computed (see operation #3 in Table VI and operation #6 in Table VIII).

It may be observed that due to the heuristic nature of evolutionary search, only a part of LGP code is effective, i.e., produces feature values that are fetched from numeric registers after execution of the entire procedure. Parts of the LGP procedure constitute "dead code" that does not influence the final feature values. This phenomenon takes place when an operation writes to an image register that is not being read until the end of the entire procedure execution (e.g., operations #7 and #8 in Table VI), or the register contents (image or numeric) becomes overwritten by a subsequent operation without being read (e.g., operation #1 in Table VI). Seemingly superfluous, this redundancy is a normal and positive phenomenon characteristic to all variants of genetic programming. In fact, the dead code fragments (which are often referred to as *introns* at the chromosome level) are extremely important, as they i) increase individual's immunity to destructive mutation and crossover operations (see remarks in Section III-B) and ii) enable the evolutionary process to work "in the background" on novel code fragments without affecting the individuals' current fitness.

For the input image considered here, the four consecutive individuals return feature values of, respectively, 2.1 and 2577, 14.2 and 7.0, 343 and 4386817, and 0 and 0. These eight feature values concatenated together, according to (2), build up the final feature vector, which is given by

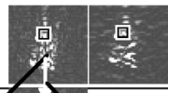
$$\mathbf{Y} = [2.1\ 2577\ 14.2\ 7.0\ 343\ 4386817\ 0\ 0]^{\mathrm{T}} \tag{6}$$

which is subsequently passed to the classifier. Both C4.5 and SVM yield a correct decision for this image, pointing to the "ZSU" decision class.

C. Multiple-Agent Approach

One way of boosting the overall performance and compensating for the suboptimal character of representations elaborated by the evolutionary process is the incorporation of a *multiagent methodology*. In this framework, agents correspond to classifiers. The basic prerequisite for the agents' fusion to become beneficial is their *diversification*. This may be ensured by using homogenous classifiers with different parameter settings, homogenous classifiers with different training data (e.g., bagging [33]), heterogeneous classifiers, etc. Here, the diversification is naturally provided by the random nature of the genetic search. As already mentioned, we run *many* genetic searches that start from different initial states (initial populations). In the basic approach, which is described in Section IV-B, the synthesized feature sets have been used in separate recognition systems. Here,

TABLE VIII
PROCESSING CARRIED OUT BY ONE OF THE EVOLVED ORGANISMS (INDIVIDUAL 3 OF 4; SEE TEXT FOR DETAILS)

			Numeric registers		Image registers	
	Operation	Arguments	r1	r2	R1	R2
	Initial register contents (input image after initial, genome-dependent preprocessing)		14.0	14.0		
1	Shift the mask towards adjacent local brightness maximum	[R1], [r2]		24.5		
2	Highpass filter (global)	R1, [R2]				
3	Scalar multiplication	r1, r2, [r1]	343.0			
4	Scalar minimum	r2, r2, [r2]		24.5		
5	Central moment	R2, [r2]		6798.5		
6	Central moment (global)	R2, [r2]		4386817		
7	Exclusive OR of a pair of images (pixelwise, global)	R1, R1, [R2]				
8	Count non-zero pixels (global)	R2, [r1]				
	Final feature values		**343.0**	**4386817**		

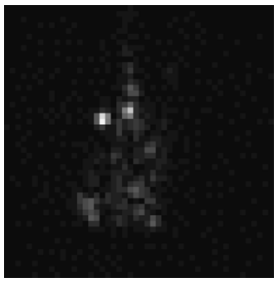


Fig. 11. Image of the ZSU class taken at 6° azimuth angle (cropped to input size, i.e., 48 × 48 pixels).

on the contrary, the best representation O^* evolved in each run becomes a part of a single *subsystem* in the *compound* recognition system's architecture (see Fig. 12). Each subsystem shown in Fig. 12 is equivalent to the system depicted in Fig. 6. The particular classifiers used in particular subsystems will be hereafter referred to as *base classifiers*.

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. In particular, inconsistency among the base classifiers may lead to ties and cause an example to remain unclassified (assigned to the "No decision" category). 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. Here, n_{sub} has been set to 10, as there were ten runs in each of the experiments described in Section IV-B1.

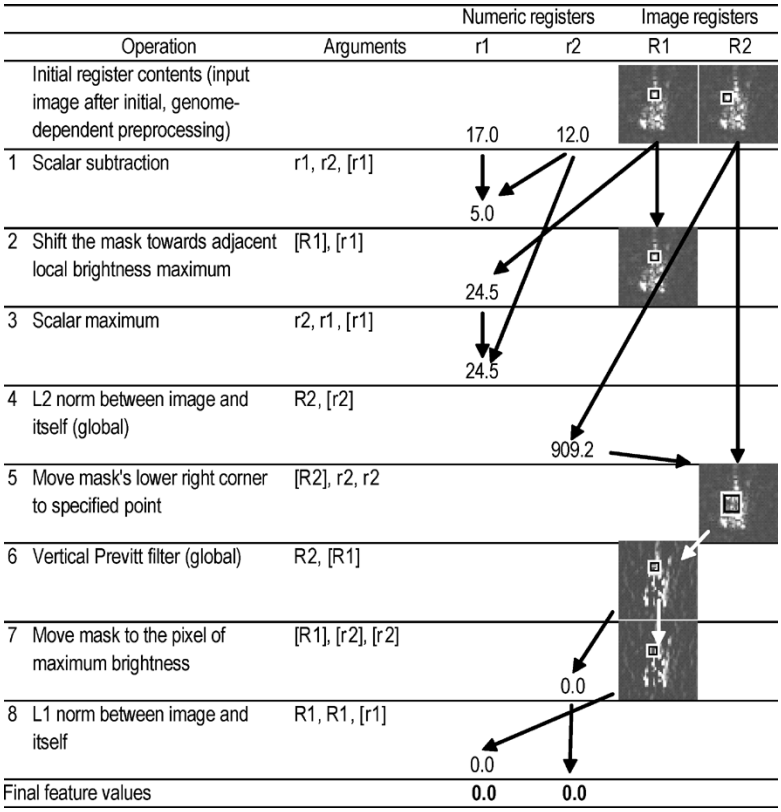
Table X and Fig. 13 present TP and FP ratios of the compound recognition systems built using the described procedure. Quite naturally, the cooperation of ten classifiers using different features makes the compound recognition system superior to all the single recognition systems examined in the previous subsection. This applies to both C4.5 and SVM, as well as to both performance measures: true positives and false positives. In particular, here, the FP ratios are approximately one order of magnitude smaller than in the case of single recognition systems.

D. Feature Fusion

Fusing different information sources by aggregating outputs of multiple classifiers is not the only way of benefiting from the synergy of feature extraction procedures evolved in independent runs. The other approach described in this subsection operates at an earlier stage of information processing and may be referred to as *feature fusion*. Here, the idea consists of gathering all the features synthesized in several independent evolutionary runs (i.e., the best organisms O^* found) into one extensive description, which is subsequently fed into a single classifier. According to the parameter setting used, this means assembling the $n_{\mathrm{sub}} = 10$ descriptions, each composed of eight features, to create a feature vector composed of 80 features. The motivation is that as each run is biased by the choice of initial random population, particular runs produce significantly different feature sets. Thus, feature fusion may lead to synergy.

Table XI and Fig. 14 present the performance of recognition systems incorporating feature fusion. When comparing these results with those presented in previous tables and figures, one

TABLE IX
PROCESSING CARRIED OUT BY ONE OF THE EVOLVED ORGANISMS (INDIVIDUAL 4 OF 4; SEE TEXT FOR DETAILS)



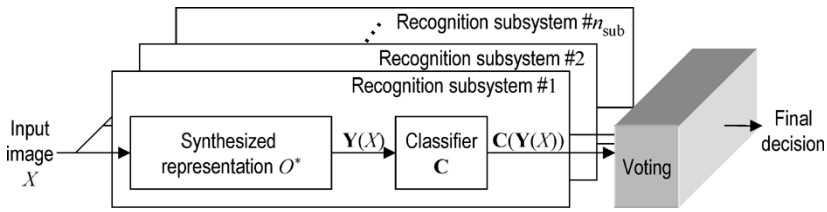


Fig. 12. Architecture of the compound recognition system.

TABLE X
TRUE POSITIVE AND FALSE POSITIVE RATIOS FOR BINARY RECOGNITION TASKS (TESTING SET, **COMPOUND** RECOGNITION SYSTEMS)

	C4.5		SVM	
Task	TP	FP	TP	FP
B1	1.000	.000	1.000	.000
B2	1.000	.000	.981	.000
B3	.955	.006	.981	.002
B4	.955	.006	.974	.000
B5	.955	.004	.961	.001
B6	.792	.002	.896	.004
B7	.721	.005	.708	.001

comes to the conclusion that feature fusion does not seem to work well for C4.5. This is due to the fact that decision tree inducers cannot benefit from a large number of features/attributes. Each attribute used at a particular stage of tree induction means an extra tree node that splits the training set into disjoint subsets so that support (confidence) of decisions being made at a tree node decreases with its depth in the tree.

On the contrary, here, the SVM proves its ability to aggregate and combine different, possibly many, attributes by mapping them into another space. The TP ratios deteriorate only slightly when the difficulty of the task increases, and the FPs are almost negligible. Up to problem $B5$ (BRDM versus {ZSU, T62, Zil131, T72, 2S1}), the recognition systems using this classifier perform almost perfectly and are superior to all the results presented earlier in this paper.

E. Multiple-Class Recognition

From a practical viewpoint, our interest is not limited to binary classification only. To investigate the ability of the proposed approach to handle multiple class-recognition tasks, we use several datasets with increasing numbers of decision classes, similarly to the binary classification experiment (Section IV-B-1). The simplest problem, which is denoted hereafter by "D2," involves $l = 2$ decision classes: BRDM (D_1) and

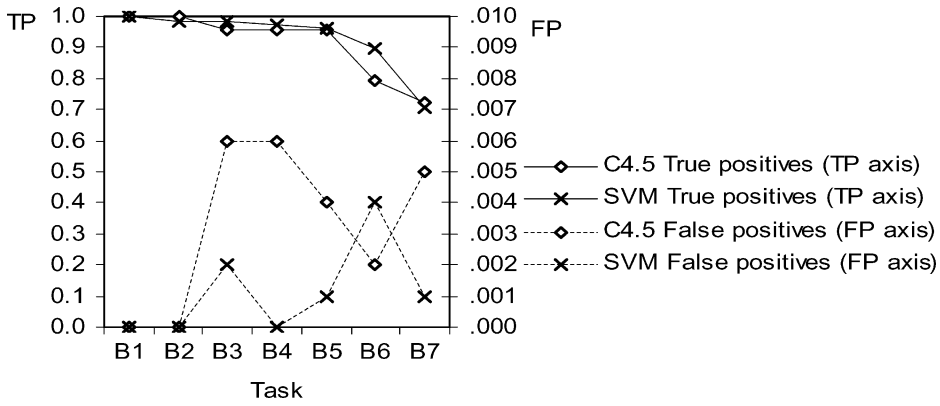


Fig. 13. TP and FP ratios for binary recognition tasks (testing set, **compound** recognition systems).

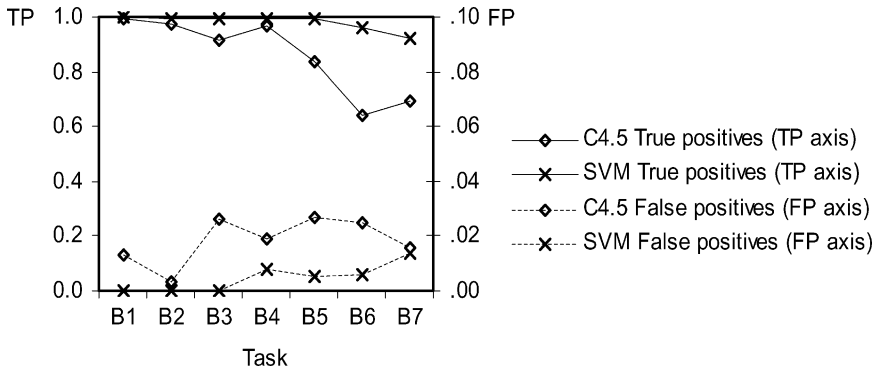


Fig. 14. TP and FP ratios for binary recognition tasks (testing set, recognition systems using **feature fusion**).

TABLE XI
TRUE POSITIVE AND FALSE POSITIVE RATIOS FOR BINARY RECOGNITION TASKS (TESTING SET, RECOGNITION SYSTEMS USING **FEATURE FUSION**)

	C4.5		SVM	
Task	TP	FP	TP	FP
B1	.994	.013	1.000	.000
B2	.974	.003	.994	.000
B3	.916	.026	.994	.000
B4	.968	.019	.994	.008
B5	.838	.027	.994	.005
B6	.643	.025	.961	.006
B7	.695	.016	.922	.014

ZSU (D_2). Consecutive problems are created by adding the decision classes up to $l = 8$ in the same order as in the experiment described in Section IV-B: T62 (classes $D_1 - D_3$, problem D3), Zil131 (classes $D_1 - D_4$, problem D4), T72 (classes $D_1 - D_5$, problem D5), 2S1 (classes $D_1 - D_6$, problem D6), BMP2 (classes $D_1 - D_7$, problem D7), and BTR70 (classes $D_1 - D_8$, problem D8). In this task, the architecture of the compound recognition system is the same as the one used in Section IV-C (see Fig. 12); however, this time, each recognition system makes a decision concerning $l \geq 2$ decision classes. The number of subsystems/voters is $n_{\text{sub}} = 10$. Each subsystem is a result of an independent evolutionary run that started from

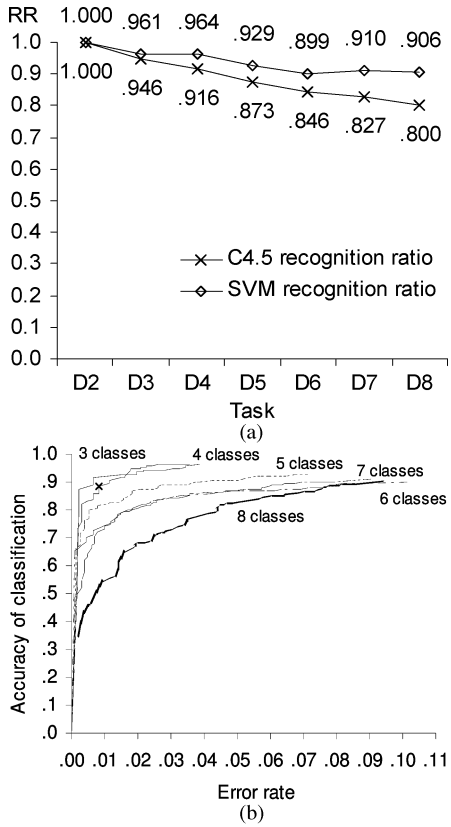


Fig. 15. (a) Test set recognition ratio as a function of number of decision classes. (b) Error-accuracy curves for different number of decision classes (base classifier: SVM).

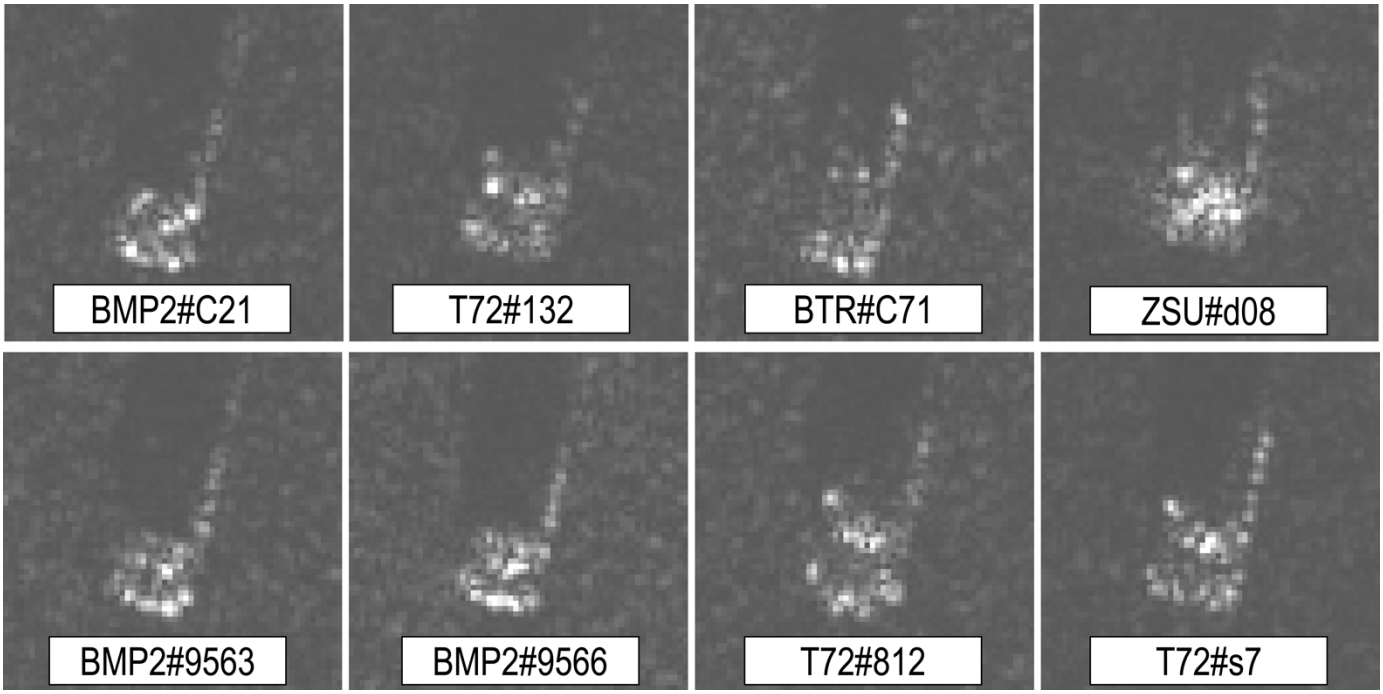


Fig. 16. Representative images of objects used in experiments concerning object variants (all pictures taken at 191° aspect/azimuth, cropped to central 64 × 64 pixels, and magnified to show details).

a different initial population. Simple voting (argmax-like) is used.

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. Fig. 15(a) presents the *accuracy of classification* (recognition) rate, i.e., $n_{\mathrm{c}}/(n_{\mathrm{c}} + n_{\mathrm{e}} + n_{\mathrm{u}})$, as a function of the number of decision classes. It can be observed that the scalability of the proposed approach with respect to the number of decision classes depends heavily on the base classifier. Here, SVM clearly outperforms C4.5. The major dropoffs of accuracy occur when T72 tank and 2S1 self-propelled gun (classes D_5 and D_6, respectively) are added to the training data; this is probably due to the fact that these objects are similar to each other (e.g., both have gun turrets) and significantly resemble the T62 tank (class D_3). On the contrary, introducing consecutive classes D_7 and D_8 (BMP2 and BTR60) did not affect the performance much; more than this, an improvement of accuracy is even observable for class D_7.

Fig. 15(b) shows the curves obtained, for the recognition systems using SVM as a base classifier, by introducing and modifying the confidence threshold that controls the voting among base classifiers (these graphs may be regarded as a generalization of *receiver operating characteristics* (ROC) curves to $l > 2$ decision classes). The higher this threshold, the more classifiers are required to vote for a particular class to make a final decision. Too few votes causes an example to remain unclassified. In this chart, the *error rate* is defined as $n_{\mathrm{e}}/(n_{\mathrm{c}}+n_{\mathrm{e}}+n_{\mathrm{u}})$. In addition, here, the results are encouraging, as the curves do not drop rapidly as the error rate decreases. By modifying the confidence threshold, one can easily control the characteristic of the recognition system, for instance, lower the error rate by accepting a reasonable *rejection rate* $n_{\mathrm{u}}/(n_{\mathrm{c}} + n_{\mathrm{e}} + n_{\mathrm{u}})$. Fig. 15(b) shows the experimental results concerning $l = 3$ and more decision classes. As for the two decision classes, the accuracy was 1.0, and error rate amounted to 0.0.

In [15], a similar experiment has been described involving coevolution and standard genetic programming using predefined *primitive* features. In one of the reported experiments, three classes of objects have been subject to recognition: BRDM, D7 (bulldozer), and T62. This task is essentially equivalent to our D3 problem with the ZSU gun replaced by D7. The test set recognition ratio amounted to 0.843–0.970, depending on the number of primitive features and number of coevolving populations used. For a recognition task concerning five decision classes (BRDM, D7, T62, Zil131, and ZSU), which is equivalent to our D5 problem with the ZSU gun replaced by the D7 bulldozer, the authors of [15] report a test set recognition ratio in the range 0.579–0.825, depending on the number of the primitive features and the number of coevolving populations used. Therefore, according to results quoted in Fig. 15, the approach proposed in this paper is not worse than the approach described in [15] when recognizing three objects, and it is significantly better when recognizing five objects. This comparison also suggests the better scalability of our method. Note that feature fusion is used here.

F. Recognizing 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 configuration variants of the same vehicle often vary significantly, yet they should be assigned to the same decision class. To provide a comparison with a human-designed recognition system, we use the conditions of the experiment reported in [34]. In particular, we synthesized recognition systems using the following decision classes

TABLE XII
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 #	Total #	[#C21]	[#132]	decision	[#C21]	[#132]	[#C71]	[#d08]	decision
BMP2	[#9563,9566]	391	**295**	18	78	**293**	27	27	1	43
T72	[#812,s7]	386	4	**330**	52	12	**323**	1	9	41

(the number following a "#" sign represents an objects' variant, which in most cases is equivalent to a vehicle's serial number):

— Two-class problem: BMP2#C21, T72#132;
— Four-class problem: 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). Therefore, as in the previous experiments, the training and testing sets are disjoint. Fig. 16 shows representative images of these decision classes taken at the same azimuth.

The results of the test set evaluation shown in the confusion matrices (see Table XII) 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 TP equals 0.804 and 0.793 for two- and four-class systems, respectively. For the case of forced recognition, the human-designed recognition algorithms described in [34] attain recognition ratios of 0.958 and 0.942, respectively. Note that in the test, we have not used any "confusers," Di.e., test images from different classes than those present in the training set, as opposed to [34], where BRDM has been used for that purpose. Synthesizing features for recognizing object variants is challenging, and further work is needed.

V. CONCLUSIONS

In this contribution, we provide experimental evidence for the possibility of synthesizing, without or with little human intervention, a feature-based recognition system that effectively recognizes 3-D objects. It is to be noted that these encouraging results are obtained in the demanding field of SAR image recognition, where the acquired images only roughly depict the underlying 3-D structure of an object.

The proposed approach learns without resorting to the database of object models, without explicit estimation of object pose, without any hand-tuning of basic operations specifically to this application, and, in particular, without introducing any SAR-specific concepts or features like 'scattering center'. Thus, the approach may be considered to be fully domain-independent. The learning process requires a minimum amount of training data, i.e., images and their class labels only, and does not rely on domain-specific knowledge, using only general vision-related knowledge encoded in basic operations.

As there is no need for, usually time-consuming, indexing/matching of the recognized image with models from the database, the recognition speed of the synthesized systems is high. The average time required by the entire recognition process for a single 48×48 image, starting from the raw image and ending up at the decision, ranged on the average from 2.2 to 20.5 ms for single classifiers and compound recognition systems, respectively (on a PC with a Pentium 1.4-GHz processor). This impressive recognition speed makes our approach suitable for real-time application.

As no domain-specific knowledge has been used, this approach may be extended to other visual learning tasks. This includes making appropriate changes to the fitness function, and modifying the library of elementary operations to provide for effective utilization of background knowledge.

REFERENCES

[1] M. A. Potter and K. A. De Jong, "Cooperative coevolution: an architecture for evolving coadapted subcomponents," *Evol. Comput.*, vol. 8, pp. 1–29, Spring 2000.
[2] B. Draper, A. Hanson, and E. Riseman, "Learning blackboard-based scheduling algorithms for computer vision," *Int. J. Pattern Recogn. Artificial Intell.*, vol. 7, pp. 309–328, 1993.
[3] V. Bulitko, G. Lee, I. Levner, L. Li, and R. Greiner, "Adaptive image interpretation: a spectrum of machine learning problems," in *Proc. ICML Workshop Continuum Labeled to Unlabeled Data Machine Learning Data Mining*, Washington, DC, 2003, pp. 1–8.
[4] B. Draper, J. Bins, and K. Baek, "ADORE: adaptive object recognition," *Videre*, vol. 4, no. 1, pp. 86–99, Winter 2000.
[5] B. Bhanu and J. Peng, "Adaptive integrated image segmentation and object recognition," *IEEE Trans. Syst., Man, Cybern. C*, vol. 30, no. 4, pp. 427–441, Nov. 2000.
[6] B. Draper, U. Ahlrichs, and D. Paulus, "Adapting object recognition across domains: a demonstration," in *Proc. Int. Conf. Vision Syst.*, Vancouver, BC, 2001, pp. 256–267.
[7] K. Krawiec, "On the use of pairwise comparison of hypotheses in evolutionary learning applied to learning from visual examples," in *Machine Learning and Data Mining in Pattern Recognition. Lecture Notes in Artificial Intelligence*, P. Perner, Ed. Berlin, Germany: Springer Verlag, 2001, vol. 2123, pp. 307–321.
[8] J. Peng and B. Bhanu, "Closed-loop object recognition using reinforcement learning," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, no. 2, pp. 139–154, Feb. 1998.
[9] ——, "Delayed reinforcement learning for adaptive image segmentation and feature extraction," *IEEE Trans. Syst,. Man, Cybern. C*, vol. 28, no. 3, pp. 482–488, Aug. 1998.
[10] A. Teller and M. Veloso, "A controlled experiment: evolution for learning difficult image classification," in *Proc. Seventh Portuguese Conf. Artificial Intelligence. Lecture Notes in Computer Science*, vol. 990. Berlin, Germany, 1995, pp. 165–185.
[11] M. P. Johnson, "Evolving Visual Routines," M.S. thesis, Sch. Architecture Planning, Mass. Inst. Technol., Cambridge, MA, 1995.
[12] M. A. Maloof, P. Langley, T. O. Binford, R. Nevatia, and S. Sage, "Improved rooftop detection in aerial images with machine learning," *Machine Learning*, vol. 53, no. 1–2, pp. 157–191, Oct.–Nov. 2003.
[13] J. Segen, "GEST: a learning computer vision system that recognizes hand gestures," in *Machine Learning. A Multistrategy Approach. Volume IV*, R. S. Michalski and G. Tecuci, Eds. San Francisco, CA: Morgan Kaufmann, 1994, pp. 621–634.

[14] I. Levner, V. Bulitko, L. Li, G. Lee, and R. Greiner, "Toward automated creation of image interpretation systems," in *Proc. 16th Australian Joint Conference on Artificial Intelligence. Lecture Notes in Artificial Intelligence*, vol. 2903, T. D. Gedeon and L. C. C. Fung, Eds. Berlin, Germany, 2003, pp. 653–665.
[15] Y. Lin and B. Bhanu, "Evolutionary feature synthesis for object recognition," *IEEE Trans. Syst., Man, Cybern. C (Special Issue on Knowledge Extraction and Incorporation in Evolutionary Computation)*, vol. 35, no. 2, May 2005.
[16] B. Bhanu and S. Lee, *Genetic Learning for Adaptive Image Segmentation*. Amsterdam, The Netherlands: Kluwer, 1994.
[17] A. Ghosh and S. Tsutsui, Eds., *Advances in Evolutionary Computing*. Berlin, Germany: Springer-Verlag, 2002.
[18] M. Rizki, M. Zmuda, and L. Tamburino, "Evolving pattern recognition systems," *IEEE Trans. Evol. Comput.*, vol. 6, pp. 594–609, Dec. 2002.
[19] T. Dietterich, "Machine learning research: four current directions," *Artificial Intell. Mag.*, vol. 18, no. 4, pp. 97–136, Winter 1997.
[20] R. P. Wiegand, W. C. Liles, and K. A. De Jong, "An empirical analysis of collaboration methods in cooperative coevolutionary algorithms," in *Proc. Genetic Evolutionary Comput. Conf.*, San Francisco, CA, 2001, pp. 1235–1242.
[21] R. A. Watson, "Compositional Evolution," Ph.D. dissertation, Graduate Sch. Arts Sci., Brandeis Univ., Waltham, MA, 2002.
[22] D. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.
[23] B. Bhanu and K. Krawiec, "Coevolutionary construction of features for transformation of representation in machine learning," in *Proc. Workshop Coevolution, Genetic Evol. Comput. Conf.*, New York, 2002, pp. 249–254.
[24] K. Krawiec and B. Bhanu, "Visual learning by evolutionary feature synthesis," in *Proc. 20th Int. Conf. Machine Learning (ICML)*, Menlo Park, CA, 2003, pp. 376–383.
[25] W. Banzhaf, P. Nordic, R. Keller, and F. Francine, *Genetic Programming. An Introduction. On the Automatic Evolution of Computer Programs and its Application*. San Francisco, CA: Morgan Kaufmann, 1998.
[26] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1992.
[27] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques With Java Implementations*. San Francisco, CA: Morgan Kaufmann, 1999.
[28] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods—Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds. Cambridge, MA: MIT Press, 1998.
[29] T. Ross, S. Worell, V. Velten, J. Mossing, and M. Bryant, "Standard SAR ATR evaluation experiments using the MSTAR public release data set," in *Proc. SPIE Algorithms Synthetic Aperture Radar Imagery V*, vol. 3370, Orlando, FL, 1998, pp. 566–573.
[30] *Intel Image Processing Library: Reference Manual*, Intel Corp., 2000.
[31] *Open Source Computer Vision Library: Reference Manual*, Intel Corp., 2001.
[32] ECJ Evolutionary Computation System, S. Luke. (2002). *http://cs.gmu.edu/~eclab/projects/ecj/* [Online]
[33] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, pp. 123–140, Aug. 1996.
[34] B. Bhanu and G. Jones, "Increasing the discrimination of SAR recognition models," *Opt. Eng.*, vol. 12, pp. 3298–3306, Dec. 2002.

Krzysztof Krawiec received the M.S. and Ph.D. degrees in computer science from Poznań University of Technology, Poznań, Poland, in 1993 and 2000.

Since 1993, he has been with University of Computing Science, Poznań University of Technology, as an Assistant Professor. From 2002 to 2003, he was a Visiting Researcher at the Center for Research in Intelligent Systems, University of California, Riverside. He is the co-author of *Evolutionary Synthesis of Pattern Recognition Systems* (New York: Springer, 2005). His research interests include pattern recognition, visual learning, evolutionary computation, and data mining.

Dr. Krawiec is a member of Association for Image Processing, Polish Member Society of the International Association of Pattern Recognition. In 2000, he received the award for the best Ph.D. thesis from the Association for Image Processing. Since 2002, he has been a member of the Task Force on Coevolution, Technical Committee on Evolutionary Computation, of the IEEE Neural Networks Society.

Bir Bhanu (S'72–M'82–SM'87–F'95) received the S.M. and E.E. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, the Ph.D. degree in electrical engineering from the Image Processing Institute, University of Southern California, Los Angeles, and the M.B.A. degree from the University of California, Irvine.

Dr. Bhanu was the founding Professor of electrical engineering and served its first Chair at the University of California, Riverside (UCR). He has been the Cooperative Professor of Computer Science and Engineering and Director of Visualization and Intelligent Systems Laboratory (VISLab) since 1991. He has also served as the founding Director of an interdisciplinary Center for Research in Intelligent Systems (CRIS) at UCR since 1998. Previously, he was a Senior Honeywell Fellow at Honeywell Inc., Minneapolis, MN. He was on the faculty of the Department of Computer Science, University of Utah, Salt Lake City, and has worked at Ford Aerospace and Communications Corporation, Newport Beach, CA; INRIA, Rocquencourt, France; and IBM San Jose Research Laboratory, San Jose, CA. He has been the principal investigator of various programs for the Defense Advanced Research Projects Agency, the National Aeronautics and Space Administration, the National Science Foundation, the Air Force Office of Scientific Research, the Army Research Office, and other agencies and industries in the areas of learning and vision, image understanding, pattern recognition, target recognition, biometrics, navigation, image databases, and machine vision applications. He is the co-author of the books *Evolutionary Synthesis of Pattern Recognition Systems* (New York: Springer, 2005), *Computational Algorithms for Fingerprint Recognition* (Boston, MA: Kluwer, 2004), *Genetic Learning for Adaptive Image Segmentation* (Kluwer, 1994), and *Qualitative Motion Understanding* (Kluwer, 1992), and the co-editor of the book *Computer Vision Beyond the Visible Spectrum* (New York: Springer, 2004).

Dr. Bhanu received two outstanding paper awards from the Pattern Recognition Society and has received industrial and university awards for research excellence, outstanding contributions, and team efforts. He has been on the editorial board of various journals and has edited special issues of several IEEE transactions and other journals. He holds 11 U.S. and international patents and over 230 reviewed technical publications in the areas of his interest. He was General Chair for IEEE Workshops on Applications of Computer Vision, Chair for the DARPA Image Understanding Workshop, General Chair for the IEEE Conference on Computer Vision and Pattern Recognition, and Program Chair for the IEEE Workshops on Computer Vision Beyond the Visible Spectrum. He is a Fellow of the American Association for the Advancement of Science, the International Association of Pattern Recognition, and the International Society for Optical Engineering.