Visual Learning by Evolutionary and Coevolutionary Feature Synthesis

Krzysztof Krawiec, *Member, IEEE*, and Bir Bhanu, *Fellow, IEEE*

***Abstract*—In this paper, we present a novel method for learning complex concepts/hypotheses directly from raw training data. The task addressed here concerns data-driven synthesis of recognition procedures for real-world object recognition. The method uses linear genetic programming to encode potential solutions expressed in terms of elementary operations, and handles the complexity of the learning task by applying cooperative coevolution to decompose the problem automatically at the genotype level. The training coevolves feature extraction procedures, each being a sequence of elementary image processing and computer vision operations applied to input images. Extensive experimental results show that the approach attains competitive performance for three-dimensional object recognition in real synthetic aperture radar imagery.**

***Index Terms*—Computer vision (CV), cooperative coevolution (CC), evolutionary computation (EC), machine learning (ML), pattern recognition, visual learning.**

I. Introduction

VISUAL learning is the process of autonomous acquisition of knowledge from visual data, aimed at improving the future performance of an intelligent system. As such, it is a vital component of any real-world application that engages visual information, for instance, enabling an autonomous robot to move safely by building a "mental map" of its environment, making it possible for a medical diagnostic support system to learn how to discriminate cases of different diseases represented by microscopic images, or helping an intelligent multimedia interface to adapt to the changing operating environments and different users.

Visual learning requires synergy between computer vision (CV) and machine learning (ML). It is a challenging domain, with difficult real-world tasks, for several reasons.

- First, visual learning is a *complex* (modular) task. It starts from raw image data and ends with a final decision in a specific scenario, usually involving several intermediate steps of information processing. This implies a need for decomposition, which is nontrivial in itself.
- Second, the visual training data is *represented* in a way that is inconvenient for most standard ML methods, which work best with a low number of scalar attributes (features), and generally do not consider the relationships (e.g., spatial) among attributes. Raster images are, on the contrary, large, structured, two-dimensional arrays. Therefore, specialized procedures and operators are required to access, aggregate, and transform the input (e.g., image filtering, segmentation, feature extraction) into an appropriate form.
- Third, the *amount of data* that has to be processed during training is usually much higher than in standard ML applications (a few or a few dozen attributes versus thousands or millions of pixels). This imposes significant requirements on the effectiveness of the search in the hypothesis space.
- Finally, the real-world image data are usually *noisy* and contain plenty of *irrelevant* factors that have to be sieved out during learning.

This paper describes an approach for recognizing objects in real-world images that addresses *all* the above issues and attempts to solve them by using important ideas from ML, evolutionary computation (EC) and computer vision, and combining them in a novel way. It is organized as follows. Section II describes motivations for this research, discusses related work in visual learning, and identifies contributions of this paper. Section III presents in detail the proposed methodology of evolutionary visual feature synthesis. Section IV describes experimental settings, implementation of the proposed method, and results in a difficult domain of real-world object recognition in radar modality. Finally, Section V provides theoretical and experimental conclusions of this paper.

II. Motivation, Related Work, and Contributions

A. Motivation

The primary motivation for the research described in this paper is the lack of a general methodology for the design and development of recognition systems. The manual design of such systems for most real-world tasks is tedious, time-consuming, and expensive. Though satisfactory in performance in constrained situations, the handcrafted solutions are usually limited in the scope of applicability and have poor adaptation ability in practical applications (for instance, real-world outdoor navigation with changing environmental conditions). As the difficulty of real-world problems approached in practice increases, the above limitations become severe obstacles. In some aspects, this is similar to the way the complexity of the software development process made the developers struggle until the software engineering offered appropriate techniques.

B. Related Work

The interest in visual learning research has been so far rather limited in both ML and CV communities, although the impor-

Manuscript received July 6, 2006; revised October 11, 2006. This work was supported in part by the Air Force Research Laboratory under Grant F33615-99-C-1440 and in part by Research Grant DS-91-427. The contents of the information do not necessarily reflect the position or policy of the U.S. Government.

K. Krawiec is with the Institute of Computing Science, Poznań University of Technology, 60965 Poznan, 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/TEVC.2006.887351

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

Reference	Approach	Experimental task	Training data
[7]	Learning recognition graphs	Recognizing buildings	High-level CV concepts
[28]	Learning of object models	Hand gesture recognition	Graphs extracted from images
[10]	EC (GP)	Locating hand in human body silhouette	Binary silhouettes
[29]	EC (GP variant)	Face recognition	Raw images (grayscale)
[21]	Reinforcement learning	Segmentation of in/outdoor scenes	Raw images (color)
[22]	Delayed reinforcement learning	Segmentation, feature extraction, in/outdoor	Raw images (color)
[5]	Biased reinforcement learning	Local/global segmentation and recognition	Raw images (color)
[12]	EC (GP)	Handwritting recognition	Raw images (grayscale)
[26]	Hybrid EC (GP+NN)	Target recognition in radar modality	1-D radar signals
[17]	Standard ML/PR classifiers	Rooftop detection in aerial imagery	Fixed set of scalar features
[8]	Coevolution of polynomials	Surface reconstruction in stereo vision	Real-world stereo pairs
[13]	EC (phenotypic CC+LGP)	Object recognition in radar modality	Raw images (grayscale)
[16]	Coevolutionary GP	Object recognition in radar modality	General + SAR-specific features
This paper	EC (genotypic CC+LGP)	Object recognition in radar modality	Raw images (grayscale)

tance of vision in the development of intelligent systems has been well recognized. Contemporary recognition systems are mostly open-loop and the human input is still predominant in the design of such systems. In most approaches reported in the literature, learning is limited to parameter optimization [4] that usually concerns a particular processing step, such as image segmentation, feature extraction, etc. Only a few contributions, summarized in Table I, attempt to close the feedback loop of the learning process at the highest (e.g., recognition) level *and* test the proposed approach in a real-world setting. Note that, to the best of our knowledge, only a few approaches [5], [12], [13], [16], [22], [23], [30] have been reported that learn using *raw* images as training data, and therefore, produce the *entire* object recognition system. Moreover, some of these methods [10], [18], [29] use domain-specific knowledge and are highly specialized towards a particular application.

C. Previous Work and Contributions of This Paper

This paper continues our previous research [6], [13], [14] on the use of cooperative coevolution (CC) [25] for the decomposition of feature synthesis task for pattern recognition and computer vision problems. In particular, in [13] we considered the task of coevolutionary feature synthesis where the cooperation between individuals from different populations takes place at the *phenotype level (phenotypic CC)*. Technically, each population maintained by the CC algorithm was devoted to the development of a single feature extraction procedure. The genotype-phenotype mapping that produces the actual feature extraction procedure took place in each population independently. Then, the resulting feature extraction procedures were used together to describe the training images in terms of feature values, which were subsequently fed into a ML classifier that performed the actual recognition.

The results obtained in [13] were encouraging and proved, among others, the scalability of phenotypic CC with respect to the number of decision classes. We also observed that the recognition performance depends heavily on the ability of the classifier to make use of *synergy*, i.e., to benefit from *combination* of feature values. This makes it difficult to judge whether the performance of the system as a whole should be attributed to the "intelligence" of the classifier, or to the nature of the coevolutionary process. In other words, the actual manner in which the populations cooperate becomes intricate due to the intermediation of the classifier.

For this reason, in this paper we propose a different approach, where the cooperation takes place at the *genotype level (genotypic CC)*. The individuals (bit strings) evolved in particular populations are first concatenated. The resulting compound bit string is subject to genotype-phenotype mapping, which results in a single feature extraction procedure that is evaluated in the same way as in the phenotypic CC. This makes the interpopulation cooperation tighter, more direct, and easier to explain and investigate. Thanks to that, in experimental part, we perform a series of experiments on a medium-size recognition task, focused on the parameterization of the coevolutionary process. Among others, we analyze the sensitivity of algorithm convergence to the number of cooperating populations and to the number of registers (local variables) used by the feature extraction procedures. This provides an in-depth insight into the coevolutionary process, as opposed to [13], where we focused on scalability, metaclassifiers, and on the ability to recognize distorted objects, in binary recognition as well as in multiple-class recognition tasks. The differences between this paper and [13] are briefly summarized in Table II.

The particular contributions of this paper are the following.

1) This paper continues and deepens the research on a novel *general methodology of automatic learning/synthesis of recognition procedures*, initially proposed in [13]. This methodology: (a) uses raw image data for training; (b) does not require domain-specific knowledge; and (c) attains good performance on a complex, real-world object recognition task. Learning proceeds with a database of training examples (images) partitioned into decision classes and a set of general-purpose image processing and feature extraction operators. The method uses EC to search in the space of image

TABLE II
COMPARISON OF THIS PAPER AND OUR PREVIOUS CONTRIBUTION ON RELATED TOPIC [13]

	Previous approach [13]	This paper
Subject and focus	Recognition performance, robustness in the presence of distorted data, scalability with respect to the number of classes	Dynamics and sensitivity to parameter setting, searching for an optimal coevolutionary setup
Method	Phenotypic CC	Genotypic CC
Experimental data	SAR, various subsets and classes	SAR, fixed 3-class data set
Experimental variants	Feature fusion, multi-agent approach	Changing the number of populations and the number of registers

representations (features) and relies on representation inspired by linear genetic programming (LGP) [1] to encode the feature extraction procedures as EC individuals.

2) We focus on the pioneering genotypic CC, a variant of the paradigm of EC [25], to handle the complexity of the task, and investigate the sensitivity of search convergence to the setting of particular parameters such as the number of coevolving populations and the number of registers used for storing intermediate results.
3) We use real-image data to demonstrate our approach and provide a comparison of performance between the genotypic coevolutionary approach (CC) and genetic algorithm (GA).

III. TECHNICAL APPROACH

Our approach, proposed in its general form in [3], [13], and [14], evolves feature extraction procedures and operates in a learning-from-examples paradigm. The learner/inducer acquires knowledge autonomously from the set of training examples (raster images) $D\,(D \subset U)$, which is a representative sample taken from the dataset U that forms the recognition problem under consideration. The technique solves a *classification* problem and learns in a *supervised* manner, i.e., requires D to be partitioned into mutually disjoint *decision classes* D_c. The output of the learner is the synthesized recognition system that implements the *feature-based recognition* paradigm, with processing split into two stages: *feature extraction*, carried out by a feature extraction procedure S, and *decision making*, performed by a trained classifier C. The interface between these modules is a vector $\mathbf{Y}$ of k scalar features.[1]

The novelty of this methodology is in integrating image processing and feature extraction into the learning process. Therefore, the learner is able to design an *intermediate (internal) image representation*, i.e., globally or locally processed versions of the input image that selects or emphasizes those visual features which are appropriate for solving the task at hand. Thus, the learner is not forced to compute the features directly from the input image, but is allowed to create processed (globally or locally) versions of the input image and use them as the source of (global or local) features. The *entire* recognition system becomes a learner's *hypothesis* in ML terms, as opposed to most approaches where hypothesis space is spanned over a fixed space of predefined features. This paradigm, often referred to as *feature synthesis* has also been known in ML literature as feature construction or constructive induction [19].

By feature extraction procedure S, we mean a function that, given an image $I \in \Im$ from the domain of raster images $\Im$, produces its description in terms of at most k scalar features, where k is an upper limit imposed on the number of features

$$S : \Im \to \Re^k.$$

A well-designed feature extraction procedure (module) S is clearly a necessity for attaining high recognition rate, and that is why its design is usually so demanding and resource-consuming. To automate that design and include it in the learning loop, we pose it as a *search problem* in the discrete space of all feature extraction procedures Ω, with each search state corresponding to a unique feature extraction procedure $S \in \Omega$. This search is guided by a maximized *evaluation (fitness) function* $f : \Omega \times U \to \langle 0, 1 \rangle$, such that, given the training data $D \subset U$, $f(S, D)$ is an estimate, for all examples from U, of the probability of correct recognition using S (details on f are provided later).

In the above setting, the task of synthesizing the *globally optimal* image representation S_{opt} given the training data D can be formalized as

$$S_{\text{opt}} = \arg\max_{S \in \Omega} f(S, D).$$

In our learning-from-examples setting, f is defined by the training data D and their partitioning into decision classes D_c, so its characteristics are not known *a priori*. In such a case, the task of finding S_{opt} has exponential time complexity with respect to the number k of scalar features and with respect to the number of possible realizations of particular scalar features. This prohibits the use of an exhaustive search algorithm even for relatively small values of k.

Heuristic or metaheuristic search is, therefore, the only plausible method that can be applied to the synthesis task that can yield reasonably good *suboptimal solutions* S^*, $f(S^*, D) < f(S_{\text{opt}}, D)$ in polynomial time. In fact, for some problems, the solutions found during the heuristic search may even be globally optimal; however, as we do not know the upper bound of recognition performance in realistic settings, we cannot discern them from the suboptimal ones.

A. Representation of Evolving Recognition Procedures

The overall architecture of the learning system, presented in Fig. 1 shows clear distinction between general-purpose evolutionary engine and domain-oriented fitness function f. For each

[1] In this paper, we use a different notation than in [13]: S for solutions (previously O), X for individuals (previously I), and I for images (previously X).

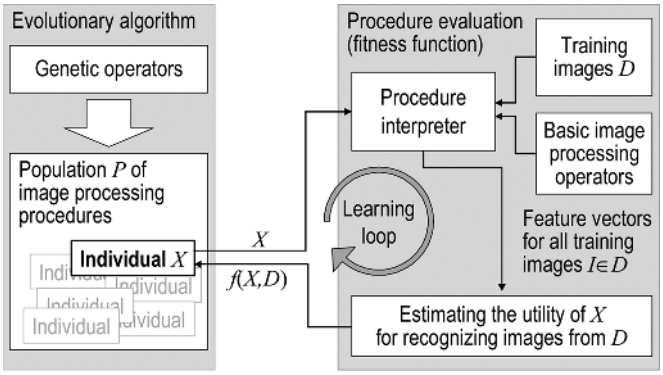


Fig. 1. The overall architecture of our learning system.

evaluated individual X, fitness function interprets it as feature extraction procedure, runs it on all images from the training set D, and estimates the predictive accuracy of the computed feature values. In this way, f provides feedback to the search process and closes the learning loop.

To effectively search the space Ω of feature extraction procedures, the learning proceeds in the framework of EC (Fig. 2). The evolutionary algorithm (EA) maintains a population P of procedures that are encoded in individuals (solutions) X, which are modified and recombined during the evolutionary search. The procedures compete with each other by means of their fitness values f, which reflect the utility of particular image representation for solving the problem posed by the training data D. The best procedure S^* found in the evolutionary run becomes the feature extraction module of the final synthesized recognition system.

An important issue that influences the performance of the proposed approach is the representation of individuals. To speed up the convergence of the search process and provide the system with basic knowledge, we assume that certain elementary building blocks are given *a priori* to the learner in the form of basic image processing, feature extraction, and feature transformation operations.

The representation framework for the proposed evolutionary learning is inspired by a variation of LGP [1]. LGP is a hybrid of GAs and genetic programming (GP). Here, an individual's chromosome encodes a feature extraction procedure S as a fixed-length string of numbers that is interpreted as a sequential program. The procedure is composed of (possibly parameterized) basic operations that work on input data (images). The major advantage of this linear representation is low susceptibility to destructive crossovers and avoidance of code bloat problems, which are important considerations for GP.

The technical encoding of procedures has been originally proposed in [13] and may be summarized as follows.

- A procedure S is encoded by a fixed-length string of bytes (valued [0...255]) that defines a sequence of *operations*, i.e., image processing and feature extraction steps.
- The particular operations work on *registers* (working variables) that are used for both input and output during procedure execution. *Image registers* store processed images (i.e., entities with domain in $\Im$), whereas *scalar registers* (real-number) store scalar features (i.e., entities with domain in $\Re$). All image registers have the same dimensions as the input image I. Each image register, apart from storing the image, maintains the position of a single rectangular *mask* that may or may not be used by a particular operation. To keep the number of parameters reasonably low, there are k image registers and k number registers, with the latter ones corresponding to features.
- An operation is encoded in a chunk of four consecutive chromosome bytes, with the following elements (see Fig. 3).
  - (i) *Operation code (opcode)*, which univocally identifies the operation to be executed.
  - (ii) *Mask flag*—decides whether the operation should be performed globally (work on the entire image) or locally (limited to the mask).
  - (iii) *Mask dimensions* (ignored if mask flag is "off"). Mask *location* is maintained by image registers and depends on the actual image contents.
  - (iv) The remaining part of the encoded operation describes the *argument list*. For most operations (e.g., $Prewitt\_3{\times}3$, *MedianFilter*, *Erode*), each argument is interpreted as identifier (number) of image or scalar register to fetch input data (input argument) or store the result (output argument). The opcode determines the number and types of arguments. However, for some operations (e.g., scalar arithmetic), an argument may directly encode a scalar constant. The distinction between register reference and scalar constant is made independently for each argument by testing its most significant bit reserved for that purpose (not marked in Fig. 3).

As an example, the second instruction (Op. #2) of the procedure shown in Fig. 3 illustrates the encoding of image thresholding operation (opcode 22) performed locally in image fragment limited by rectangular mask (mask flag 1) of size 14 (square mask dimension), using threshold value fetched from scalar register #1 (pointed by argument #1), on the image fetched from image register #2 (pointed by argument #3), and storing the result in image register #2 (pointed by argument #3). An operation may use the same register(s) for input and output, thus overwriting its input arguments by output arguments. For each operation, at least one argument must refer to register(s) that serve as output(s) for the operation.

To guarantee the correct execution of the encoded procedure, each chromosome element is interpreted modulo cardinality of its domain. For instance, with 70 elementary operations at hand, the opcode 82 will be interpreted as $82 \bmod 70 = 12$. Therefore, *all* individuals are feasible and there is no need for possibly time-consuming correction of syntactically incorrect solutions, as *any* string of bytes implements a correct feature extraction procedure.

Technically, the elementary GP operations used in this paper are implemented by means of common image processing and feature extraction procedures imported from Intel Image Processing Library [9] and OpenCV Library [21]. The set of operations is virtually the same as the one used in our former work on phenotypic CC for feature synthesis [13]; for the complete

```
procedure EvolutionaryAlgorithm(D)
P := random_initialization()          // population initialization
S*.fit := 0                           // best solution found so far
repeat
    for each X ∈ P                    // evaluation loop
        X.fit := f(X, D)              // evaluate solution/representation X using training set of images D
        if X.fit > S*.fit then S* := X   // update the best solution
    end for
    P' := ∅                           // subsequent population
    while card(P') < card(P)          // recombination loop
        M := selection(P)             // build mating pool
        P' := P' ∪ crossover(M)       // mate parents and insert offsprings into next population
    end while
    P := mutation(P')                 // mutate randomly selected solutions
until S*.fit = 1 or stopping_conditions()   // other stopping conditions involve time
return S*                             // return the best solution found
```

Fig. 2. Standard EA applied to the feature synthesis task ($card(.)$—set cardinality). This figure serves as reference for the CC algorithm (Fig. 4).

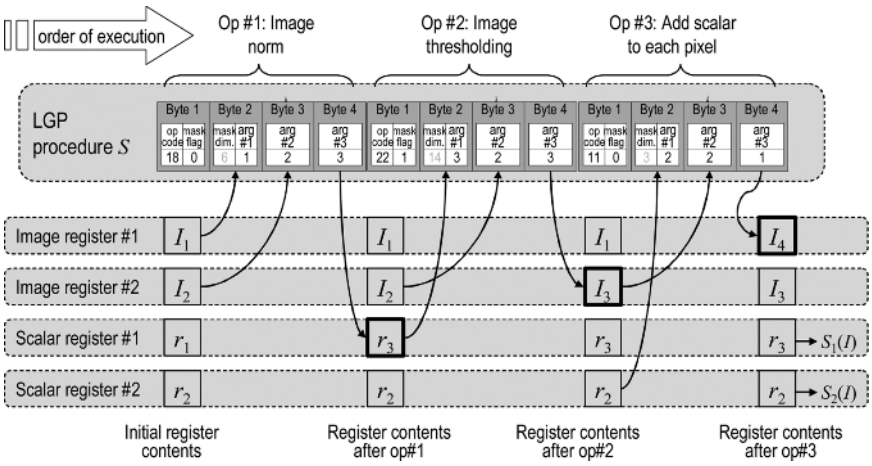


Fig. 3. Illustration of an exemplary LGP procedure S and its interpretation during execution for a single image I. Thick register frame denotes change of register's value.

list of operations, the reader is referred to [13, Table iii]. These operations may be grouped into the following categories.

- image processing operations: unary ($\Im \to \Im$) and binary ($\Im \times \Im \to \Im$);
- mask-related operations ($\Im \to \Im$);
- feature extraction operations:
  - unary: producing a single output value ($\Im \to \Re$), or a pair of numbers ($\Im \to \Re^2$);
  - binary $\Im \times \Im \to \Re$.

Apart from these image-related operations, we provided also unary ($\Re \to \Re$) and binary ($\Re \times \Re \to \Re$) operations implementing basic arithmetic and logic.

For simplicity, there are no separate registers for storing logical values. The logical operations use the scalar registers, with zero value being interpreted as logical "false," and nonzero value interpreted as logical "true."

Given the above settings, a procedure S processes a single input image I in following steps.

1) *Initialization* of register contents: Each of the k image registers is set to I. The locations of masks are set to consecutive salient features found in the image, ordered with respect to their strength. Specifically, we compute the response of 5×5 low-pass filter to the image, and set the mask in the ith image register, $i \in [1, k]$, to the location corresponding to ith maximum of filter response. As a result, mask locations mark bright "blobs" in input image. Scalar registers are set to the midpoint coordinates of corresponding masks, in particular, scalar registers $2i$ and $2i+1$ store the x and y coordinates of the ith image mask.
2) *Execution*: The n_{op} operations encoded by S are carried out one by one. For each operation, the stored output value(s) override the former register contents (see example in Fig. 3).
3) *Interpretation*: The values obtained in scalar registers after execution of the entire procedure S are interpreted as the features computed by S for image I. Let $S_i(I)$ denote the value computed by S and stored in the ith scalar register, $i \in \langle 1, k \rangle$, after processing the image I. The overall result of processing carried out by S for an image I is, therefore, a vector of k scalar features

$$\mathbf{Y} = S(I) = \langle S_1(I), S_2(I), \ldots, S_k(I) \rangle .$$

The resulting vectors of features $\mathbf{Y}$, computed for all images from the training set D, are the basis for estimating the utility of S for recognizing the objects from the training data D. The details of this process are provided at the end of Section III-C.

Fig. 3 shows the processing and data flow for execution of an exemplary procedure S of length $n_{op} = 3$ for a single input image I in a simple setting ($k = 2$, two image registers and two scalar registers). The first four bytes of the procedure encode the operation of image norm (city-block distance), computed pixelwise:

$$r_3 = \mathrm{norm}(I_1, I_2) = \sum_{y=1}^{\mathrm{height}(I_1)} \sum_{x=1}^{\mathrm{width}(I_1)} |I_1(x,y) - I_2(x,y)|$$

where $I(x,y)$ denotes brightness of pixel (x,y) in image I, and $\mathrm{height}(I)$ and $\mathrm{width}(I)$ denote, respectively, the height and width (in pixels) of image I. This operation requires two image arguments, I_1 and I_2, which are taken from image registers #1 and #2. The result of the operation, r_3, is stored in scalar register #1. The next operation in this example is image thresholding that reads the input image I_2 from image register #2, the threshold value r_3 from the scalar register #2, and stores the image I_3 (in image register #2), processed according to the following equation:

$$I_3(x,y) = \begin{cases} 1, & \text{if } I_2(x,y) \geq r_3 \\ 0, & \text{otherwise} \end{cases}$$
$$\forall x \in \langle 1, \mathrm{width}(I_2) \rangle\,,\ \forall y \in \langle 1, \mathrm{height}(I_2) \rangle\,.$$

As a result of thresholding, the contents of this image register changes and it does not contain the input image I anymore. The third operation adds a scalar to all pixels in the image, according to the equation

$$I_4(x,y) = I_3(x,y) + r_2,$$
$$\forall x \in \langle 1, \mathrm{width}(I_3) \rangle\,,\ \forall y \in \langle 1, \mathrm{height}(I_3) \rangle\,.$$

It uses the image I_3 fetched from the image register #2 (previously modified by the second operation) and the scalar value r_3 fetched from scalar register #1 as input arguments, and stores the resulting image I_4 in the image register #1.

The overall result of execution of S is here a two-element vector of features $\mathbf{Y} = \langle S_1(I), S_2(I) \rangle = \langle r_3, r_2 \rangle$, reflecting the final (postexecution) state of scalar registers. Note that, in fact, $S_1(I) = r_3$ has already been computed by the first operation of S, and the remaining two operations do not influence the result. Such dead code, often found in experimentally evolved solutions, though apparently superfluous, could act as *introns* in the genetic code and may improve the individual's resistance to destruction in genetic modifications [15]. Thus, procedure length constitutes a kind of *upper limit* of solution complexity, but it does not force a particular level of complexity, as not all instructions have to be effective.

Analogously, although the number of scalar registers k is a fixed parameter of the learning process, it only imposes an upper limit on the number of computed scalar features. Experiments show that some code fragments often produce and store in registers *constant* scalar values that in fact do not depend on the actual input image (similar to the dead code branches in tree-like GP individuals [11]). In the example shown in Fig. 3, the first of the computed features $S_1(I) = r_3$ is constant and does not depend on input image, as the distance $\mathrm{norm}(.)$ between an image and itself is 0 for any input image. As constant scalar features (S_i values) do not differentiate any images, they do not contribute to fitness function f. This phenomenon gives the learning process the possibility of dynamically adjusting the number of evolved scalar features, and allows the effective number of features to be less than k.

B. Cooperative Coevolution (CC)

According to Wolpert's "no free lunch" theorem [33], the search for a universal, best-of-all metaheuristic algorithm is futile, for both optimization and learning problems. In other words, the average performance of any metaheuristic over a set of all possible fitness functions is the same. In the real world, however, not *all* fitness functions are equally probable or meaningful. Most real 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 that specificity. In particular, the complexity of the problem and the way it may be *decomposed* are such characteristics.

To cope with the inherent complexity of the visual learning task posed above, we should find a way to *decompose the problem* into subproblems rather than trying to solve it in one step. For that purpose, we use CC [25], a variation of standard EA, which in the last few years has been reported as a promising approach to handle the increasing complexity of problems posed in artificial intelligence and related disciplines. There are two important factors that make CC different from standard EA. First, instead of having just one population P of individuals, CC maintains $n\,(n > 1)$ populations P_i, $i = 1, \ldots, n$. Second, individuals in a particular population encode only *part* of the solution to the problem, as opposed to EA, where each individual encodes complete solution to the problem.

Fig. 4 presents the CC algorithm in detail. It is essentially equivalent to the standard EA (compare to Fig. 2), except for the evaluation phase (lines 8 to 15 of the algorithm). To evaluate an individual X from ith population P_i, it is temporarily combined (line 10) with *representatives* R_j, i.e., selected individuals from the remaining populations P_j, $j = 1, \ldots, n$, $j \neq i$, to form the solution S. Then, the entire solution is evaluated by means of the fitness function f (line 11) and X gets the resulting fitness value (line 12). Evaluation of an individual from ith population does not affect the remaining populations. The remaining part of the iteration (lines 17–22) follows the standard EA and it is executed independently for each population.

The joint evaluation scheme forces the individuals from particular populations to cooperate. As a result, the evolutionary search in a population is driven by the context built up by the representatives of remaining populations. The choice of representatives R_j from other populations is, therefore, critical for the convergence of the evolutionary process. Although many

```
1  procedure CooperativeCoevolution(D)
2  for i := 1,K ,n do                          // loop over all n populations
3    P_i := random_initialization()
4    R_i := randomly chosen X ∈ P_i            // initialization of representatives
5  end for
6  S*.fit := 0                                 // initialize fitness of the best solution
7  repeat                                      // loop over generations
8    for i := 1,K ,n do                        // loop over all n populations
9      for each X ∈ P_i do
10       S := < R_1,K , R_{i-1}, X, R_{i+1},K , R_n >   // compose the solution
11       S.fit := f(S, D)                      // fitness calculation and assignment
12       X.fit := S.fit                        // the individual receives the solution's fitness
13       if S.fit > S*.fit then S* := S
14     end for
15   end for
16   for i := 1,K ,n do                        // loop over all n populations
17     P' := ∅                                 // next population
18     while card(P') < card(P_i) do
19       M := selection(P_i); P' := P' ∪ crossover(M)      // M − mating pool
20     end while
21     P_i := mutation(P')
22     R_i := arg max_{X∈P_i} X.fit            // best individual becomes representative
23   end for
24 until S*.fit = 1 or stopping_conditions()
25 return S*
```

Fig. 4. CC algorithm ($card(.)$—set cardinality).

different variants are possible here, it has been shown experimentally that the so-called CCA-1 scheme works best [31]. In CCA-1, in the first generation a representative of ith population is an individual drawn randomly from it (line 4 in Fig. 4). In the following generations, a representative R_i of population P_i is its best individual with respect to the previous generation (line 22). Therefore, only one representative per population is used for evaluation, as opposed to CCA-2 (not used here), where there are two such individuals (the best and a randomly chosen one [31]).

The major advantage of CC is that it provides the possibility of breaking up a complex problem into components *without specifying explicitly the objectives for them*. The way the individuals from populations cooperate *emerges* as the evolution proceeds. In [3], we provided experimental evidence for the usefulness of CC in feature construction for standard ML problems. In [13], and here we show that CC (phenotypic in [13] or genotypic in this paper) is also especially appealing to the problem of visual learning, where the overall objective 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.

C. Using CC to Decompose Feature Extraction Procedures

In the proposed methodology, we use CC to scale down the task of synthesis of feature extraction procedures. One can consider here different approaches depending on the *way* the task is to be decomposed. Conceptually, three basic levels may be considered (see Fig. 5). In each of them, the individuals in populations are represented as strings of bytes; however, these methods differ in the level at which the cooperation takes place.

(a) *Cooperation at chromosome level*: Each population P_i, $i = 1, \ldots, n$, is evolving a *fragment* (substring of bytes) of a single procedure S. To build a procedure, all the n substrings are concatenated and interpreted as a single procedure.

(b) *Cooperation at feature level*: Each population P_i is evolving a *complete* procedure $S^{(i)}$. Each $S^{(i)}$ is executed independently and the features $S^{(i)}(I)$ computed by them are joined to form a compound feature vector $\mathbf{Y}$.

(c) *Cooperation at classifier level*: Proceeds like (b), however, the feature vectors $S^{(i)}$ resulting from execution of particular procedures are used independently for induction of separate classifiers instead of being combined. The cooperation takes place during classifiers' voting.

In our previous work [13], we examined the cooperation at feature (b) and classifier (c) levels, focusing on the pattern recognition perspective. These two variants may be jointly termed as *phenotypic*, as the composition of solutions takes place *after* the genotype-phenotype mapping. Such setting, though effective in terms of recognition accuracy, has some drawbacks outlined in Section II-C. Thus, in this contribution, we follow the variant (a), which is a special case of *genotypic* CC, with solution composition taking place *prior* to genotype-phenotype mapping. In particular, we break up the task at *chromosome level*, with each population being responsible for optimizing a predefined, fixed length fragment (substring) of procedure code.

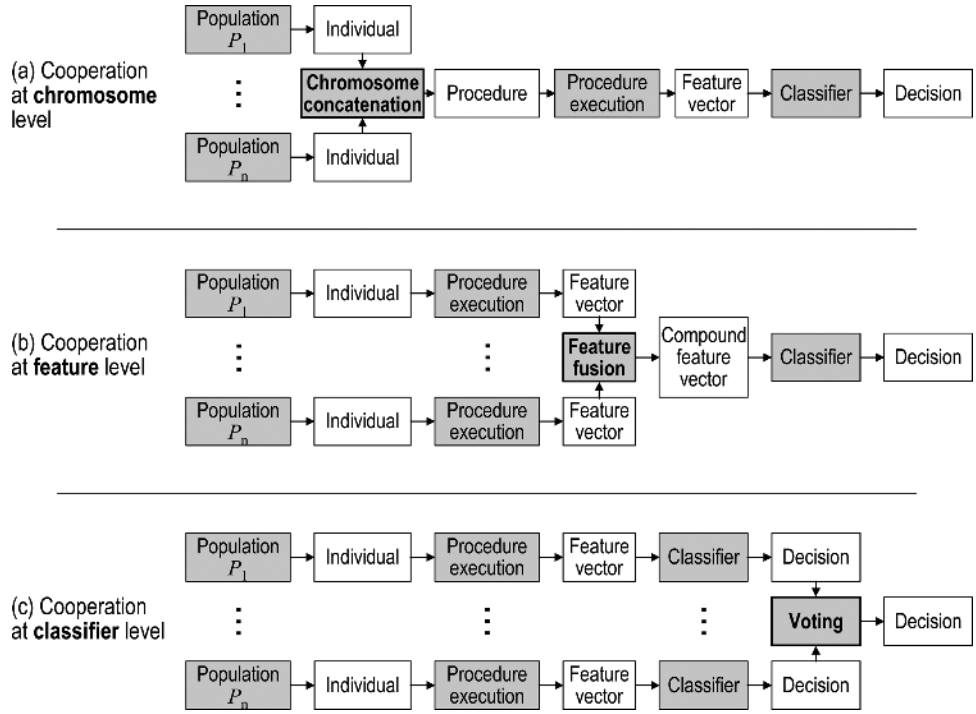


Fig. 5. Possible levels of cooperation in the proposed approach.

The system architecture outlined earlier in Fig. 1 may now be detailed with respect to the CC algorithm presented in Fig. 4. The evaluation of an individual X from ith population P_i starts with creating temporary procedure S composed of X and of representatives R_j, $j \neq i$, of the remaining populations (see line 10 in Fig. 4). This process is straightforward and consists in concatenating X's chromosome with the chromosomes of all R_js, in the order consistent with the values of indices i and j.

The procedure S is then executed for all images from the training set (see Section III-B). The values $\mathbf{Y}$ computed by S for all training images, together with the images' class labels c, constitute the learning dataset T that is the basis for evaluation of an individual

$$T = \{\langle S(I), c\rangle : \forall I \in D_c, \forall D_c \subset D\}.$$

The next step of fitness computation consists in estimating the utility of T (and, indirectly, of S and X) for recognizing/classifying images from D. For this purpose, we perform a multiple train-and-test cross validation using a fast ML inducer, with the training data D temporarily subpartitioned into n_f disjoint subsets $D^{(i)}$, $i = 1, \ldots, n_f$; $D^{(i)} \cap D^{(j)} = \emptyset$, $i \neq j$. The internal partitioning of D into $D^{(i)}$ sets is fixed prior to evolutionary run to provide comparability of fitness values across different generations. In each iteration of cross validation, the inducer is trained on the temporary training subset $D^{(i)}$ and produces a classifier $C^{(i)}$ that is subsequently tested on the temporary testing subset $D \setminus D^{(i)}$. The resulting estimate of recognition ratio becomes the evaluation of fitness for the solution-procedure S

$$f(S, D) = \frac{1}{\mathrm{card}(T)} \sum_{i=1}^{n_f} \mathrm{card}\left(\left\{\langle S(I), c\rangle \in T : I \in D \setminus D^{(i)} \wedge C^{(i)}\left(S(I)\right) = c\right\}\right)$$

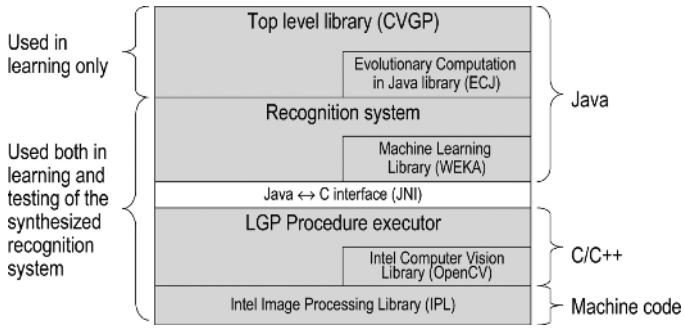


Fig. 6. Software implementation of the evolutionary feature synthesis system.

where $\mathrm{card}(.)$ denotes set cardinality. This value is subsequently assigned to the individual X (see line 12 of CC algorithm in Fig. 4). Note that it does not affect the representatives R_j of the remaining populations.

IV. Experiments

The objective of the computational experiments is to explore the idea of LGP-based synthesis of recognition procedures using genotypic CC for search, in the context of demanding real-world object recognition task that involves images of three-dimensional objects. The experiments have been carried out in software environment integrating image processing and computer vision libraries written in C (Image Processing Library [9] and Open Computer Vision Library [21]), and higher level soft-computing libraries written in Java (WEKA Machine Learning Library [32] and ECJ Evolutionary Computation Library in Java [17]). Fig. 6 shows the overall software architecture of the system.

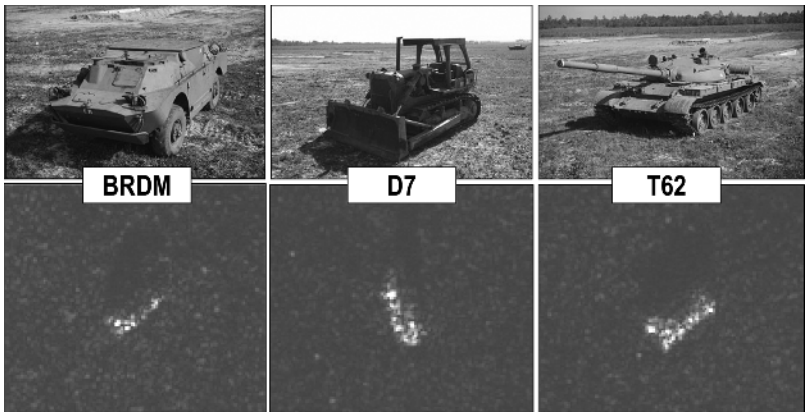


Fig. 7. The representatives of three decision classes. Top row—visual photographs, bottom row—corresponding 48 × 48 pixel SAR images.

TABLE III
DATASET STATISTICS

Class	*Total images*	*Training set*		*Testing set*	
		Number of images	*Aspect interval*	*Number of images*	*Aspect interval*
BRDM	188	64	5.62°	124	2.90°
D7	188	64	5.62°	124	2.90°
T62	131	64	5.62°	67	5.37°
Total	507	192		315	

TABLE IV
CONDITIONS IMPOSED ON PARAMETER SETTINGS TO PROVIDE COMPARABILITY OF RESULTS

Condition	*Compare EC and CC runs which*:
1	Both use the same *total chromosome length* n_{op}.
2	Both use the same *total number of individuals* evolving in all n populations.
3	Both have been allowed to run for the same amount of *time*.

A. The Data

Similarly to our previous work [13], the proposed approach has been tested on the demanding task of object recognition in synthetic aperture radar (SAR) imagery. The MSTAR public database [28] of SAR images taken at one foot resolution has been used as the data source. The task posed to the system was to recognize images representing three different vehicles (decision classes D_c): BRDM, D7, and T62 (see Fig. 7), at 15° depression angle and any azimuth $(0°, \ldots, 359°)$.

Let us emphasize that recognizing objects in SAR images is hard in general. Some of the difficulties associated with the object recognition task in real SAR images are [2] the following.

- Nonliteral nature of the data. Radar images appear different than the images acquired in the visible spectrum. Bright spots on the images, called scattering centers, correspond to those parts of the object which reflect radar signal strongly. No line features are present for vehicle images at this resolution.
- Low persistence of features under rotation (high rotation-variance).
- High levels of noise and low resolution.

From the MSTAR database, 507 images of three objects classes (see Fig. 7) have been selected. The resulting set of images has been split into disjoint training and testing parts (see Table III) to provide a reliable estimate of the recognition ratio for the learned recognition system. This selection was aimed at providing uniform azimuth coverage: for each class, there is a training image for approximately every 5.62° of azimuth, and a testing image for every 2.9°–5.37° of azimuth, on the average.

This training data D is used for evolutionary feature synthesis. The testing images are used only for verifying selected (usually the best ones) evolved procedures after training comes to an end. The original images have different sizes, so they are cropped to 48 × 48 pixels. The original images are also complex (two-channel), with real and imaginary parts corresponding to signal magnitude and phase, respectively; however, only the magnitude image part is used in the experiments. No other form of image preprocessing (e.g., speckle removal) is applied. As opposed to the contribution [13], where we considered different training sets, here the training task and the data described above are fixed and are used in *all* the experiments, while we consider different settings of the evolutionary parameters.

B. Parameter Setting

The major objective of the experiment was to compare genotypic CC to EA using different parameter settings, particularly, the number of registers k and the number of coevolving populations n. As EA and CC approaches differ significantly, some parameters have been set as to provide comparability of their results. We treat EA as CC running with $n = 1$ population, and compare EA and CC runs using the conditions for parameter settings, as given in the following Table IV.

Condition (1) provides the same flexibility in synthesizing feature extraction procedures for both approaches. Condition (2) ensures that the same number of complete solutions is evaluated in each generation. Condition (3) makes the comparison objective from practical viewpoint, and, in conjunction with condition (2), it ensures that EA and CC may run approximately for the same number of generations. Note that in terms of the number of generations, minor differences in favor of EA are possible due to the extra time needed by CC to maintain selection and recombination in multiple populations.

When applying conditions (1) and (2), we used a *symmetric* model of decomposition, i.e., each population receives $1/n$ fraction of EA individuals and $1/n$ fraction of the total code length

TABLE V
PARAMETER SETTINGS

Parameter	*Setting*
Selection operator	tournament selection with tournament pool size = 5
Crossover operator	one-point, prob. 1.0
Mutation operator	one-point, prob. 0.1
Selection of representatives	CCA-1 (see Section 3.3)
Number of cross-validation folds (subsets) n_f used in f	3
Classifier	J4.8 [26][32]

n_{op} used in the corresponding EA experiment. It is to be observed that the small number of individuals in EA may lead to small CC populations, for which it may be difficult for CC to maintain sufficient diversity among individuals in populations, and the search may become ineffective. In our experiments, we found that using less than 1000 individuals in total does not allow the evolutionary dynamics of CC to exploit the search space properly. As a result, in the following experiments, the total number of individuals is set to a rather large value of 2000. This large number of evolved individuals and time-consuming fitness function required us to set the time limit of evolutionary run to 8000 seconds to allow for a sufficient number of generations. The results are obtained using a PC with Pentium 1.8 GHz processor.

The remaining parameters, grouped in Table V, have been fixed for all the runs. In particular, we used generational GA [20], and the tournament selection with tournament pool size 5. Tournament size has been set to 5 as a compromise between 2 (which seems to be most popular, and also the simplest possible, setting for GA), and 7, used commonly in GP. Some preliminary experiments have shown that this setting provides satisfactory selective pressure on the one hand, and prevents premature convergence to low-quality local optima on the other hand. The approach did not exhibit statistically significant sensitivity to small changes (4...6) of this parameter.

The one-point crossover operator used here works on pairs of parent individuals by selecting a randomly chosen cutting point in their chromosomes and exchanging the "tails." The chromosome cutting is allowed between any pair of consecutive bytes (three possibilities) of LGP encoding of a single operation, i.e., not only between LGP operations (quadruples of bytes), but also within them. Note that, in this way, a particular operation (a quadruple of chromosome bytes) may be disrupted. Despite this, the resulting operation is always syntactically correct and has valid interpretation, as it is the opcode that determines how the remaining elements of operation encoding are interpreted, and, in particular, what kind of modulo mapping (see Section III-B) should be applied to each of them. For instance, assume the procedure shown in Fig. 3 undergoes crossover and its second operation becomes disrupted between bytes 3 and 4, and byte 4 is being replaced by value 12 coming from another individual. The opcode of the resulting new operation does not change (22) and encodes morphological opening. This opcode determines that byte 4 (argument #3 of the operation) is still interpreted as the number of image register used as an output for this operation. Given 2 image registers, argument #3 will now refer to the first image register (as $12 \bmod 2 = 0$), in contrast to the situation before crossover, when argument #3 referred to the second image register. Thus, in this particular example, crossover changes the address of register that is used by the operation to store its result. We found out that this nonrestrictive crossover improves the convergence speed of the algorithm by allowing more flexibility.

The experiments described here aim at reaching one good solution per run. Thus, we do not have to maintain *elitism*, and we keep track of the best solution found so far. This is why we decided to make *all* the individuals subject to crossover with $\mathrm{probability} = 1.0$, which ensures a thorough search.

The mutation operator used randomly selects a byte in individual's chromosome and replaces it with a randomly generated number. Similar to crossover, this genetic operator always leads to a syntactically correct and interpretable feature extraction procedure thanks to opcode-dependent interpretation and modulo mapping. Individuals resulting from crossover undergo such mutation with a probability 0.1. This relatively high mutation probability is due to the presence of the "modulo mapping" in the procedure execution (see Section III-B), which causes many mutations to be ineffective (the chromosome changes, but its interpretation/execution does not).

To induce classifiers within the fitness function (see Fig. 1) we used the J4.8 induction algorithm, a WEKA [32] implementation of the last public release of popular decision tree inducer C4.5 [26]. The wrapper within fitness function works with $n_f = 2$ folds of cross validation to speed up the computation. All the remaining parameters are set to default values used in the software packages ECJ [17] and WEKA [32].

C. Results

1) Experiment I. Number of Populations: The first of the experiments described here concerns the sensitivity of the approach to the number of populations. For this purpose, we set the total procedure length $n_{\mathrm{op}} = 9$ instructions, i.e., $9 \times 4 = 36$ bytes, as such a value allows for relatively many symmetric decompositions into $n = 2$, 3, 4, 6, and 9 populations ($n > 9$ has not been considered due to the issue of population size discussed earlier).

The leftmost four columns of Table VI illustrate the settings imposed by conditions (1) and (2) for this experiment (see Table IV). For instance, for $n = 4$, each CC population contains $2000/n = 500$ individuals and works on code fragment composed of $36/n = 9$ bytes. The right part of Table VI and Fig. 8 show the fitness of the best individual found in particular evolutionary runs. Three series of runs have been carried out, for different number of registers $k = 3$, 4, 5. To provide for statistical evidence, all evolutionary runs have been repeated ten times, starting with different initial populations, so Table VI and Fig. 8 present the average performance of the best individuals together with their 0.95 confidence intervals.

Results presented in Table VI and Fig. 8 clearly indicate that CC outperforms EA for $n = 2$ and 3 populations, CC outperforms EA working with the same number of registers as k. For larger values of n, CC's advantage declines—in extreme case, $n = 9$, cooperation ceases to be effective. This may be explained as follows. In CC, each evaluated solution is, in fact, composed of an individual that undergoes evaluation and $n - 1$ representatives of the remaining populations (see Fig. 4). In the CCA-1 scheme used here, the representatives are fixed during the entire

TABLE VI
THE AVERAGE FITNESS OF BEST INDIVIDUALS EVOLVED IN TEN INDEPENDENT RUNS FOR DIFFERENT NUMBER OF POPULATIONS n (TOTAL CODE LENGTH 36 BYTES, 2000 INDIVIDUALS, 8000 s)

Method	# of populations n	Bytes of code per population	Individuals per population	Fitness of the best solution		
				$k = 3$ registers	$k = 4$ registers	$k = 5$ registers
EA	1	36	2000	0.898 ± .007	0.886 ± .005	0.891 ± .011
CC	2	18	1000	0.908 ± .009	0.895 ± .012	0.904 ± .012
CC	3	12	667	0.911 ± .009	0.893 ± .012	0.902 ± .011
CC	4	9	500	0.888 ± .007	0.893 ± .011	0.903 ± .009
CC	6	6	333	0.864 ± .019	0.888 ± .009	0.874 ± .009
CC	9	4	222	0.833 ± .015	0.852 ± .009	0.857 ± .011

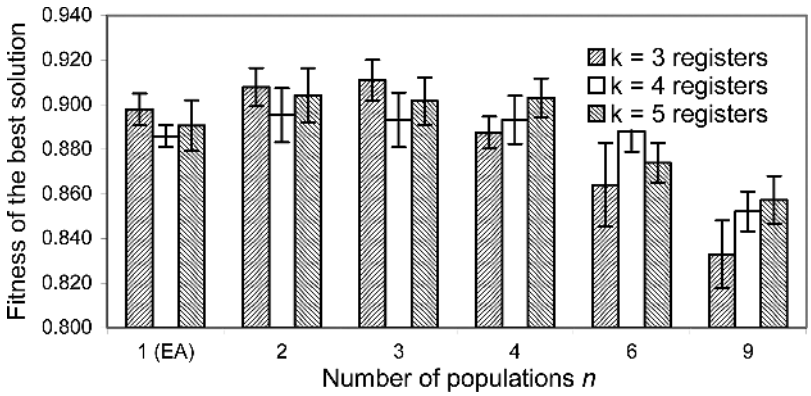


Fig. 8. The average fitness of best individuals evolved in ten independent runs with 0.95 confidence intervals (total code length 36 bytes, 2000 individuals, 8000 s).

evaluation phase within a particular CC generation (lines 8–15 in Fig. 4). When a new best individual is found, it becomes the representative of the population it belongs to. If, in a particular generation, such an improvement happens in m of n populations, the set of representatives changes in m/n fraction. Obviously, m tends to decrease as evolution proceeds, and when the search starts to saturate, m usually drops to 0 or 1 per generation. Thus, at further stages of evolutionary process, the representatives usually do not change at all or change only in $1/n$ fraction. Therefore, the CC search is technically less thorough than EA search, especially for large n.

The importance of n also seems to be dependant on k. A low value of k leads to relatively large differences in performance depending on n, especially for $n > 3$, whereas for $k = 5$ and $k = 4$ the differences are less prominent. It seems that, for larger values of k, there is more chance for the evolutionary process to synthesize useful (though worse than for $k = 3$ and $n = 2$ or 3) features, even if the cooperation is difficult due to the large number of populations n. For low k, the quality of cooperation is more critical. In absolute terms, the differences in favor of CC are not large, but in most prominent cases they are statistically significant. For $k = 3$, the false positive probability related to t-Student test amounts to 0.057, 0.026, 0.036, when comparing fitness of the best CC solution evolved with $n = 2, 3, 4$ populations, respectively, to the fitness of corresponding EA solution. For $k = 4$ and 5, these figures are larger, but do not exceed 0.15.

This result seems to be remarkably good, keeping in mind that the recognition problem under consideration is difficult, and that CC devotes some extra time to maintain multiple populations P_i, perform selection and mating in each of them independently, keep track of the best representatives R_i, and assemble temporary solutions S from individuals and population representatives. In our experimental setting, this overhead amounts to about 5% of the total computation time for chosen values of k, but may grow as k increases. Nevertheless, as Table VI shows, CC performs better despite this extra burden. We can, therefore, conclude that our feature synthesis method may indeed benefit from CC-based chromosome-level (genotypic) problem decomposition.

The evolutionary runs lasted for 133.8, 123.1, and 116.4 generations on the average, for $k = 3$, 4, and 5, respectively. This tendency was expected, as, though the number of registers k (and the number of features at the same time) does not influence the execution time of LGP procedure alone, the classifier training that takes place within computation of fitness function becomes more time-consuming as k increases. Within each group (i.e., for fixed k), the EA and CC runs lasted approximately for the same number of generations, which confirms our expectation formulated with respect to conditions (1)–(3) (Table IV). Note also that execution time of our fitness function may vary depending on the actual code implemented by the evaluated solution. For instance, for MSTAR images, extraction of spatial moments from an image takes about two orders of magnitude more time than scalar multiplication of two numeric registers.

Fig. 9 shows fitness graphs (learning curves) for particular number of populations n, for the experiments with $k = 3$. Each

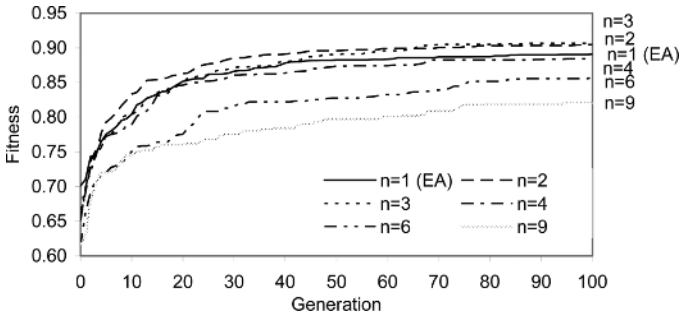


Fig. 9. Fitness graphs (mean over ten runs of best solution found so far) for runs with $k = 3$ registers (cf. Table VI and Fig. 11).

TABLE VII
THE AVERAGE FITNESS OF BEST INDIVIDUALS EVOLVED IN TEN INDEPENDENT RUNS FOR DIFFERENT NUMBER OF REGISTERS k (TOTAL CODE LENGTH 32 BYTES, 2000 INDIVIDUALS, 8000 s)

Number of registers k	Fitness of the best solution			
	Total # individuals = 2000		Total # individuals = 300	
	EA	CC	EA	CC
2	.809 ± .015	.811 ± .017	.753 ± .012	.750 ± .011
3	.888 ± .007	.906 ± .008	.877 ± .012	.880 ± .018
4	.876 ± .013	.890 ± .012	.865 ± .016	.868 ± .018
5	.885 ± .008	.903 ± .012	.891 ± .015	.868 ± .022
6	.891 ± .005	.890 ± .017	.832 ± .021	.858 ± .015
7	.881 ± .018	.907 ± .009	.858 ± .012	.830 ± .024
8	.862 ± .013	.871 ± .017	.849 ± .022	.846 ± .017
9	.865 ± .013	.860 ± .022	.861 ± .024	.811 ± .011
10	.847 ± .013	.853 ± .018	.823 ± .007	.854 ± .026

data point represents a mean of ten runs of the best solution found so far in a particular generation. As the termination condition was time-based, some runs ended earlier than the others, so the graph has been limited to 100 generations only, to avoid fluctuations resulting from lack of statistical support (for each series of runs, all ten runs lasted for at least 100 generations). It may be observed that most graphs exhibit saturation, so further performance improvement by runtime extension is highly improbable. However, the runs for large number of population ($n = 6$ and $n = 9$), which generally performed worse, show an opportunity for further improvement. Confidence intervals are not shown in Fig. 9, as all of them were below 0.001.

2) Experiment II. Number of Registers: In this experiment, we investigate the sensitivity of the genotypic CC to the number of registers k, which also determines the number of features. To focus on this aspect, in CC runs we consider only $n = 2$ populations, the simplest form of decomposition possible in this framework, which also provided good results in Experiment I. At this time, there is no need to divide the LGP chromosome into a different number of parts, we use shorter feature extraction procedures composed of 32 bytes, i.e., $32/4 = 8$ instructions. Thus, in case of CC, we coevolve two populations, each working on four consecutive instructions of LGP code. All runs have been repeated ten times, starting with different initial populations.

Table VII and Fig. 10 present mean fitness for corresponding EA and CC runs for a different number of registers (and thus features) $k = 2, \ldots, 10$. The general conclusion is that both EA and CC seem to work best for $k = 3, \ldots, 7$. The deterioration for $k \geq 8$ may seem surprising, as, from ML perspective, having more distinct and well-discriminating features is usually better than having less of them. However, increasing the number of features increases the computational cost of classifier induction in wrapper (see Section III-B). For instance, for CC and $k = 2$, the mean run length amounted to 184.2 generations, but for $k = 10$ that figure reduces to 74.5 generations only. Thus, the potential gain resulting from extra features is compensated by the increased processing time. On the other hand, $k = 2$ features are definitely not enough to perform recognition of three different classes of objects at a satisfactory performance level.

For $k = 3$, 4, 5, and 7 registers, CC significantly outperforms EA (t-Student's p amounts to 0.002, 0.08, 0.017, and 0.01, respectively). For the remaining values of k, none of the methods is better. Thus, we conclude, that moderate number of registers enables effective cooperation between populations, by co-building common feature extraction procedures, or evolving mutually complementary features.

The rightmost two columns of Table VII also present reference results of experiments ran for a much smaller total number of individuals (300). It may be easily observed, that in this case there is no clear winner: EA and CC seem to outperform each other quite randomly for different values of k. This confirms our hypothesis, that a low number of individuals divided into separate populations P_i does not allow the CC to exhibit enough evolutionary dynamics in each separate population. CC needs larger populations to attain qualitative progress in comparison to EA.

3) Experiment III. Test Set Performance: The results presented so far offer a good description of the performance of the approach on the training set, but do not tell us much about the predictive ability of the synthesized recognition systems. Fig. 11 shows the receiver operating characteristics (ROCs) curves, for the best CC individuals found in runs reported in the second row of Table VII, i.e., for $n = 2$, $k = 3$, and the total number of individuals to be 2000. The ROC curves are computed for each decision class separately, with the selected decision class playing the role of the positive class, and the remaining two classes treated as a negative class. Each curve shows the true positive ratio (TP, the share of correctly recognized objects), as a function of false positive ratio (FP, the share of incorrectly classified objects), for the best recognition systems obtained in ten independent runs.

These parametric curves have been obtained from the testing set, by varying the confidence threshold that controls the interpretation of the output of the trained classifier. Usually, 3...7 different threshold values have been sampled to build up a single graph in Fig. 11. The confidence threshold imposes a lower limit on the ratio of *a posteriori* probabilities of positive and negative decision classes produced by the classifier for a particular example. If, for a particular test example, the ratio is lower than a threshold, no recognition decision is made and the example remains unclassified. The ROC curves clearly show that the approach maintains good recognition ability on the test set. This applies mostly to the D7 decision class, for which ROC curves approach the ideal point ($\mathrm{FP} = 0$, $\mathrm{TP} = 1$). For instance, the point marked by cross corresponds to recognition system having

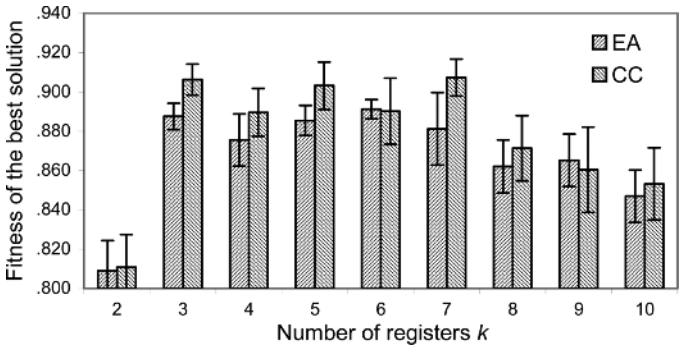


Fig. 10. The average fitness of best individuals evolved in ten independent runs with 0.95 confidence intervals for different number of registers k (total code length 32 bytes, 2000 individuals, 8000 s).

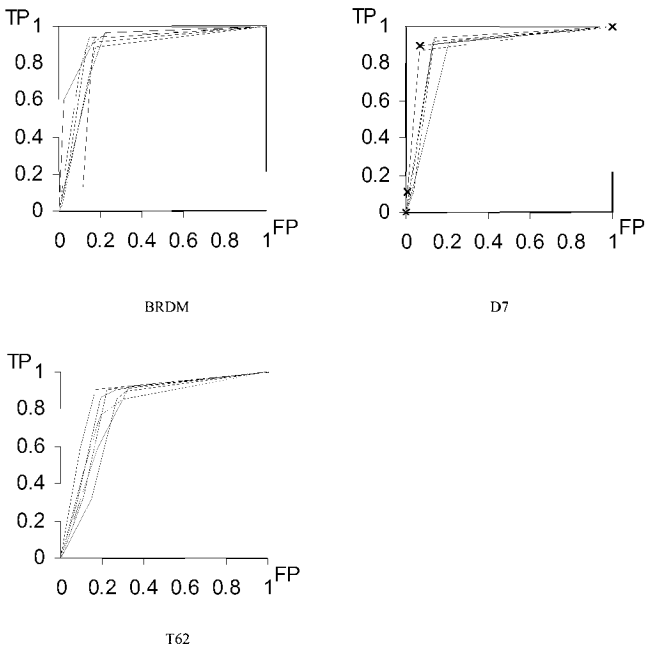


Fig. 11. ROC curves for particular decision classes, obtained for the testing set using the best CC individuals found for $n = 2$ and $k = 3$ (one curve per run (ten runs); see second row of Table VII).

$\mathrm{FP} = 0.072$ and $\mathrm{TP} = 0.901$, i.e., such that it correctly recognizes over 90% images of D7 vehicle, while mistaking only 7.2% of BRDM and T62 images for D7. Our previous research results obtained for phenotypic CC [13], [14] indicate that further improvements are possible by combining many recognition systems (preferably those that produce significantly different ROCs), and aggregating their outputs by voting.

4) Examples of Synthesized Feature Extraction Procedures: One of the important virtues of the proposed approach is the readability of the synthesized feature extraction procedures. Fig. 12 presents the processing carried out by the best procedure found in one of the evolutionary runs. This procedure has been elaborated by CC working with $n = 2$ populations and $k = 4$. It effectively computes two features and stores them in two (of a total of four) scalar registers. In this example, we used the naïve Bayesian classifier [32] to demonstrate that the evolved recognition systems are also able to produce a continuous response (contrary to C4.5's discrete decision making). The processing is shown for test-set representatives of all three decision classes used in experiments: BDRM2 image taken at 342.3° aspect, D7 image taken at 102.3° aspect, and T62 image taken at 346.5° aspect.

For each of the input images, Fig. 12 shows execution of the procedure in the form of a data-flow graph. Each column of images in the figure shows how the contents of a particular image register changes with procedure execution; for better readability, images have been enhanced (by means of *iplHistoEqualize* function from IPL [9]), but this does not affect the actual LGP processing. The execution of the procedure starts from the top and proceeds downwards through several intermediate image processing steps. Rounded boxes denote global operations (working on the entire image), while slanted boxes correspond to local operations (working on the marked rectangular mask). Transparent and gray boxes correspond to operations encoded in LGP fragments elaborated by populations P_1 and P_2, respectively. The operations used in this particular example are: *AbsDiff*—pixelwise absolute difference of a pair of images, $HiPass3\times3$—high-pass filtering using a 3×3 mask, *CrossCorrel*—cross correlation of a pair of images, *PushROIX*—(local) shifts horizontally the current image's mask to the closest local maximum of brightness, *Gaussian*—(local) image smoothing using 3×3 Gaussian mask, *MorphClose*—morphological closing operation (for simplicity, all morphological operations work with 3×3 square as a structural element), *LogicalOr*—pixelwise logical "OR" operation (computed bitwise from the arguments). Note that, as it is common for GP, not all input data (initial register contents) and not all intermediate results are utilized for the final decision making (e.g., the result of *CrossCorrel* is not further processed).

Eventually, two of the executed operations yield scalar features: the x coordinate of the shifted mask $S_1(I)$, and the normalized difference of two processed images $S_2(I)$. The overall processing ends with the final recognition decision made by the classifier (previously trained on the training set). The numbers in the "Output" box denote *a posteriori* probabilities yielded by the classifier. For all three input images, the recognition system makes a correct decision, yielding maximum *a posteriori* probability for the decision class the presented image belongs to.

In case the of genotypic decomposition discussed in this paper, particular populations can specialize in different stages of the recognition task. It may be hypothesized, that populations

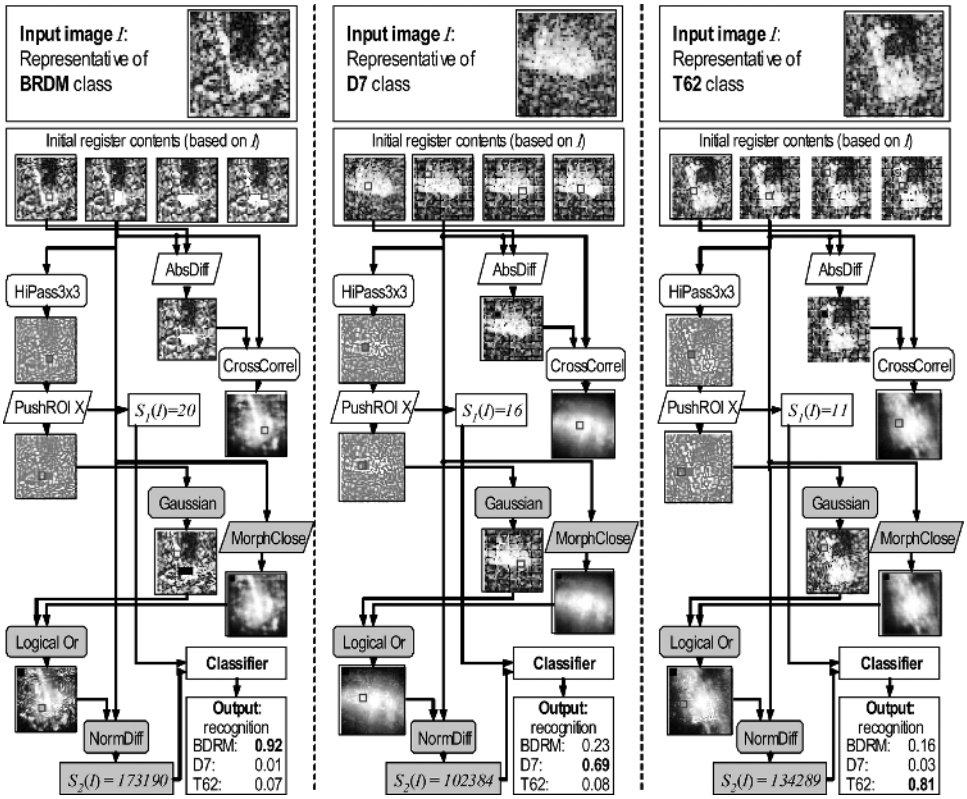


Fig. 12. A synthesized processing graph of a selected best-of-run procedure evolved by means of CC, processing exemplary images from particular decision classes.

delegated to the development of the early parts of procedure would tend to specialize in image processing, whereas populations working on the final parts of the procedure would focus on feature extraction and aggregation. This supposition is partially confirmed in the example presented in Fig. 12. In a sense, two modes of cooperation between elaborated solution parts may be observed here. Technically, each population evolved a separate scalar feature: populations P_1 and P_2 evolved features S_1 and S_2, respectively. Nevertheless, S_2 is actually not only due to P_2: its value depends on the result produced by the *PushROIX* operator call, which takes place in the code chunk evolved by P_1.

V. Conclusion

In this paper, we proposed a coevolutionary learning method that performs automatic problem decomposition at the genotype (chromosome) level. The approach enables the learner to acquire knowledge from complex examples by autonomously transforming their representation. The proposed approach and the results of its experimental evaluation allow us to formulate the following conclusions and observations.

1) The paradigm of coevolution allows us to decompose the task of representation synthesis 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 the particular stages of information processing. In particular, the experimental analysis indicates the conditions that allow the genotypic CC to outperform the EA; these conditions are: (i) a large population (order of thousands of individuals); (ii) a moderate number of populations $n\,(2,\ldots,4)$; and (iii) a small number of registers $k\,(3,\ldots,7)$.
2) By incorporating feature synthesis into the learning loop, the evolutionary learner searches for performance improvement by modifying representation of the training data. The approach manipulates feature extraction *procedures*, as opposed to most methods, which are usually limited to learning meant as parameter optimization. This allows for discovering well-discriminating features, which are often sophisticated and novel for human experts.
3) The procedural approach to feature synthesis gives the learner access to complex, structural input data that otherwise could not be directly used. As a result, the learning process requires only raw training data that are usually easy to acquire, i.e., images and their class labels. It does not rely on domain-specific knowledge, using only general vision-related knowledge encoded in basic operations. The approach is well-scalable: its computational complexity (mostly the cost of fitness function call) grows linearly with the size of the training set. If the problem is difficult and requires using more features, the approach also scales linearly with it.
4) We provide evidence for the possibility of solving, using the proposed approach, a demanding real-world task of visual learning. The encouraging experimental results for SAR object recognition are obtained without recurring to means that are commonly used in conventional approaches to the design of recognition systems, such as comparing

input images to the database of object models, explicit estimation of object pose, hand tuning of basic operations for a specific application, and, in particular, introducing SAR-specific concepts or features like "scattering center." On this task, the CC-based genotype-level decomposition leads to statistically significant performance improvement, as compared with standard EA paradigm. This result may generalize to other visual and nonvisual learning applications. It is an important argument in favor of CC for tackling complex learning problems and offers an interesting research direction.

5) The proposed method may be characterized as feature-based. Compared to model-based recognition, there is no need for the possibly expensive matching of an image with models from the database. Thus, our synthesized recognition systems attain high recognition speed during the runtime (i.e., testing). The average time required by the entire recognition process, starting from the raw image and ending up with the final recognition result, totaled 4.9 ms on the average on Pentium 1.8 GHz processor, for a single 48×48 image and a procedure composed of 18 operations. This recognition speed makes our approach suitable for real-time application.

In its canonical form our approach is close to but does not always outperform human-designed recognition systems working in SAR modality. Nevertheless, more sophisticated solutions based on this idea are able to reach that frontier. In particular, in [13], we show how further improvement may be obtained by aggregating outputs of many recognition systems obtained using the proposed approach.

As mentioned in Section II-C, the approach discussed here may be characterized as working on the genotypic level, as opposed to phenotypic-level CC described previously in [13] (cf. Fig. 5). Due to the differences in learning tasks used in both papers, we cannot exactly compare these two approaches. Nevertheless, as the overall problem domain (vehicle recognition in SAR images) is the same, and the computer implementations share a relatively large portion of the code, in the following we consider both of these approaches.

When testing the phenotypic approach in [13] we considered, among others, a multiclass learning task concerning three classes of vehicles: BRDM, ZSU, and T62 (see [13, Sec. IV.E, p. 422]). On that task, the test-set accuracy of classification varied from 94.6% to 96.1%, depending on the classifier used (C4.5 and SVM; [13, Fig. 15(a)]). In particular, the near-optimum test-set performance in terms of false positives and true positives amounted to 0.0345 and 0.9590, respectively [13, Fig. 15(b)].

The BRDM-ZSU-T62 learning task used in [13] exhibits some similarity to the learning task BRDM-D7-T62 considered in this paper, where the ZSU decision class has been replaced by the D7 class (see Table III). The genotypic approach applied to BRDM-D7-T62 problem reaches per-class test-set false-positive ratios ranging from 0.1 to 0.18, and true-positive ratios ranging from 0.91 to 0.94, depending on the decision class (Fig. 11). Though this might indicate some superiority of the phenotypic coevolution, one should note that phenotypic results have been obtained at a much greater computational expense, with a powerful committee of ten classifiers working in parallel, each of them using a different set of features evolved in an independent evolutionary process. The genotypic coevolution, on the contrary, used three registers and, hence, only three features to recognize the targets.

In terms of recognition performance, there is not enough evidence to claim the superiority of any of the two approaches. However, genotypic CC considered here outperforms the standard EA in a systematic way (see Figs. 8 and 10) and allows a better understanding of the dynamics of the evolutionary process. In phenotypic CC [13], on the contrary, the mutual interactions between features encoded in coevolving individuals are very complex, due to the memetic adaptation that takes place during classifier training, and make it difficult to predict the outcome of fitness calculation. For instance, an individual may be mistakenly rewarded for evolving a poor-discriminating feature if the classifier cleverly uses only features from the remaining populations. In genotypic CC presented in this paper, such mistakes are impossible, as *each* of the evolved features results from the cooperation of *all* populations.

From the more general perspective of the NFL theorem [33], we cannot expect the genotypic or phenotypic CC to beat each other, or other metaheuristics. To draw definitive conclusions concerning hypothetical superiority of genotypic or phenotypic coevolution, a separate study would be required, preferably using a less sophisticated learning task. Such study could aim at identifying the class of learning tasks that are effectively solved by CC in its different variants, as shown in Fig. 5.

REFERENCES

[1] W. Banzhaf, P. Nordin, R. Keller, and F. Francone, *Genetic Programming. An Introduction. on the Automatic Evolution of Computer Programs and Its Application*. San Mateo, CA: Morgan Kaufmann, 1998.

[2] B. Bhanu, D. E. Dudgeon, E. G. Zelnio, A. Rosenfeld, D. Casasent, and I. S. Reed, Eds., "Introduction to the special issue on automatic target detection and recognition," *IEEE Trans. Image Process.*, vol. 6, no. 1, pp. 1–6, Jan. 1997, (Guest Editors).

[3] B. Bhanu and K. Krawiec, "Coevolutionary construction of features for transformation of representation in machine learning," in *Proc. Genetic and Evol. Comput. Conf.*, 2002, pp. 249–254.

[4] B. Bhanu and S. Lee, *Genetic Learning for Adaptive Image Segmentation*. Norwell, MA: Kluwer, 1994.

[5] B. Bhanu and J. Peng, "Adaptive integrated image segmentation and object recognition," *IEEE Trans. Syst., Man, Cybern.-Part C*, vol. 30, pp. 427–441, Nov. 2000.

[6] B. Bhanu, Y. Lin, and K. Krawiec, *Evolutionary Synthesis of Pattern Recognition Systems*. New York: Springer-Verlag, 2005.

[7] B. Draper, A. Hanson, and E. Riseman, "Learning blackboard-based scheduling algorithms for computer vision," *Int. J. Pattern Recogn. Artif. Intell.*, vol. 7, pp. 309–328, Mar. 1993.

[8] J. Y. Goulermas and P. Liatsis, "A collective-based symbiotic model for surface reconstruction in area-based stereo," *IEEE Trans. Evol. Comput.*, vol. 7, no. 5, pp. 482–502, Oct. 2003.

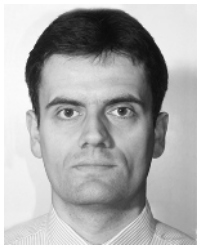
[9] *"Intel Image Processing Library,"* Intel Corporation, 2000, Reference manual.

[10] M. P. Johnson, P. Maes, and T. Darrell, "Evolving visual routines," in *Proc. 4th Int. Workshop Synthesis and Simulation of Living Systems: Artificial Life IV*, R. A. Brooks and P. Maes, Eds., 1994, pp. 373–390.

[11] J. R. Koza, D. Andre, F. H. Bennett, III, and M. A. Keane, *Genetic Programming III: Darwinian Invention and Problem Solving*. San Mateo, CA: Morgan Kaufmann, 1999.

[12] K. Krawiec, "Pairwise comparison of hypotheses in evolutionary learning," in *Proc. 18th Int. Conf. Mach. Learn.*, C. E. Brodley and A. P. Danyluk, Eds., 2001, pp. 266–273.

[13] K. Krawiec and B. Bhanu, "Visual learning by coevolutionary feature synthesis," *IEEE Trans. Syst., Man, Cybern.-Part B*, vol. 35, pp. 409–425, Jun. 2005.
[14] K. Krawiec and B. Bhanu, "Coevolution and linear genetic programming for visual learning," in *Lecture Notes in Computer Science*, E. Cantú-Paz *et al.*, Ed. New York: Springer-Verlag, 2003, vol. 2723, Proc. Genetic Evol. Comput., pt. I, pp. 332–343.
[15] J. R. Levenick, "Inserting introns improves genetic algorithm success rate: Taking a cue from biology," in *Proc. 4th Int. Conf. Genetic Algorithms*, 1991, pp. 123–127.
[16] Y. Lin and B. Bhanu, "Evolutionary feature synthesis for object recognition," *IEEE Trans. Syst., Man, Cybern., Part C*, vol. 35, no. 2, pp. 156–171, May 2005, Special issue on knowledge extraction and incorporation in evolutionary computation.
[17] S. Luke, "ECJ evolutionary computation system," 2002. [Online]. Available: http://www.cs.umd.edu/projects/plus/ec/ecj/
[18] M. A. Maloof, P. Langley, T. O. Binford, R. Nevatia, and S. Sage, "Improved rooftop detection in aerial images with machine learning," *Mach. Learn.*, vol. 53, pp. 157–191, 2003.
[19] C. J. Matheus, "A constructive induction framework," in *Proc. 6th Int. Workshop Mach. Learn.*, 1989, pp. 474–475.
[20] Z. Michalewicz, *Genetic Algorithms+Data Structures=Evolution Programs*. Berlin, Germany: Springer-Verlag, 1996.
[21] *"Open Source Computer Vision Library,"* Intel Corporation, 2001, Reference manual.
[22] J. Peng and B. Bhanu, "Closed-loop object recognition using reinforcement learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 2, pp. 139–154, Feb. 1998.
[23] ——, "Delayed reinforcement learning for adaptive image segmentation and feature extraction," *IEEE Trans. Syst., Man, Cybern.*, vol. 28, no. 3, pp. 482–488, Aug. 1998.
[24] R. Poli, "Genetic programming for image analysis," in *Proc. 1st Int. Conf. Genetic Program.*, J. R. Koza, Ed., Jul. 1996, pp. 363–368.
[25] M. A. Potter and K. A. De Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evol. Comput.*, vol. 8, pp. 1–29, 2000.
[26] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1999.
[27] M. Rizki, M. Zmuda, and L. Tamburino, "Evolving pattern recognition systems," *IEEE Trans. Evol. Comput.*, vol. 6, no. 6, pp. 594–609, Dec. 2002.
[28] 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 for Synthetic Aperture Radar Imagery V*, Orlando, FL, 1998, vol. 3370, pp. 566–573.
[29] 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 Mateo, CA: Morgan Kaufmann, 1994, pp. 621–634.
[30] A. Teller and M. M. Veloso, "PADO: A new learning architecture for object recognition," in *Symbolic Visual Learning*, K. Ikeuchi and M. Veloso, Eds. Oxford, U.K.: Oxford Univ. Press, 1997, pp. 77–112.
[31] R. P. Wiegand, W. C. Liles, and K. A. De Jong, "An empirical analysis of collaboration methods in cooperative coevolutionary algorithms," in *Proc. Genetic and Evol. Comput. Conf.*, 2001, pp. 1235–1242.
[32] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques With Java Implementations*. San Mateo, CA: Morgan Kaufmann, 1999.
[33] 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.



Krzysztof Krawiec (M'07) received the M.S., Ph.D., and Habilitation degrees in computer science from the Poznań University of Technology, Poznań, Poland, in 1993, 2000, and 2005, respectively.

Since 1993, he has been with the University of Computing Science, Poznan University of Technology, as an Assistant Professor. From 2002 to 2003, he worked as a Visiting Researcher at the Center for Research in Intelligent Systems, University of California, Riverside, CA. He is the coauthor 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 the Association for Image Processing, the 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. In 2001 and in 2005, he received the Minister of National Education Award.

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.

He has been 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. Currently, he also serves as the founding Director of an interdisciplinary Center for Research in Intelligent Systems (CRIS), UCR. Previously, he was a Senior Honeywell Fellow at Honeywell Inc., Minneapolis, MN. He has been on the faculty of the Department of Computer Science, University of Utah, Salt Lake City, and has worked at Ford Aerospace and Communications Corporation, CA, INRIA-France, and IBM San Jose Research Laboratory, CA. He has been the principal investigator of various programs for DARPA, NASA, NSF, AFOSR, ARO, 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 coauthor of *Evolutionary Synthesis of Pattern Recognition Systems* (New York: Springer-Verlag, 2005), *Computational Algorithms for Fingerprint Recognition* (Norwell, MA: Kluwer, 2004), *Genetic Learning for Adaptive Image Segmentation* (Norwell, MA: Kluwer, 1994), and *Qualitative Motion Understanding* (Norwell, MA; Kluwer, 1992), and the co-editor of *Computer Vision Beyond the Visible Spectrum*, (New York: Springer-Verlag, 2004). He holds 11 U.S. and international patents and over 250 reviewed technical publications, including 90 plus journal papers in the areas of his interest.

Dr. Bhanu has 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 (PAMI, IP, SMC, R&A, IFS) and other journals. He has been General Chair for the IEEE Conference on Computer Vision and Pattern Recognition, the IEEE Workshops on Applications of Computer Vision, the IEEE Workshops on Learning in Computer Vision and Pattern Recognition, Multimodal Biometrics; Chair for the DARPA Image Understanding Workshop, 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 (AAAS), the International Association of Pattern Recognition (IAPR), and the International Society for Optical Engineering (SPIE).