Learning and Recognition of Hand-Drawn Shapes Using Generative Genetic Programming

Wojciech Jaśkowski, Krzysztof Krawiec, and Bartosz Wieloch

Institute of Computing Science, Poznań University of Technology
Piotrowo 2, 60965 Poznań, Poland
{wjaskowski|kkrawiec|bwieloch}@cs.put.poznan.pl

Abstract. We describe a novel method of evolutionary visual learning that uses generative approach for assessing learner's ability to recognize image contents. Each learner, implemented as a genetic programming individual, processes visual primitives that represent local salient features derived from a raw input raster image. In response to that input, the learner produces partial reproduction of the input image, and is evaluated according to the quality of that reproduction. We present the method in detail and verify it experimentally on the real-world task of recognition of hand-drawn shapes.

1 Introduction

In supervised learning applied to object recognition, the search in the space of hypotheses (learners) is usually guided by some measure of quality of discrimination of training examples from different object classes. This requires defining, somehow arbitrarily, learner's desired response for objects from particular classes (e.g., defining desired combinations of output layer excitations in case of an artificial neural network). Though proven effective in several applications, such an approach suffers from relatively high risk of overfitting, especially when the number of image features is large. For instance, in our past experience with evolutionary synthesis of object recognition systems [1,2], many evolved learners tended to use irrelevant image features, coincidentally correlated with the partitioning of training examples into concepts. This happens because learners are rewarded exclusively for decisions they make, and not for the actual 'understanding' of the recognized objects.

Moreover, applicability of supervised feature-based learning methods is restricted to simple recognition tasks with a limited number of object classes. For more complex objects and for large numbers of object classes, one usually has to rely on model-based approach and explicitly specify the models of objects to be recognized, which is often tedious and time-consuming.

To avoid overfitting on one hand and model-based approach on the other, in this paper we make the learning process unsupervised in the sense that the learner is not explicitly told how it should discriminate the positive examples from the negative ones. Rather than that, it is encouraged to reproduce a selected aspect of the object being recognized, which, in turn, enables more thorough evaluation.

In experimental part, we tackle the problem of interpretation of hand-drawn sketches. In real-world scenarios, such recognition systems may be helpful for direct digitization of hand-sketched diagrams, for instance, block diagrams, UML diagrams, etc., saving the time required for tedious manual re-entry of paper notes. Such drawings are typically acquired using input devices like TabletPC computers, PDAs, or graphics tablets (digitizers). Most such devices produce on-line data, i.e., provide both spatial and temporal information about pen (stylus) position. As our approach requires spatial information only, its applicability spans also off-line interpretation of sketches stored as ordinary raster images (e.g., acquired from paper drawing using a scanner).

According to Krishnapuram *et al.* [3], the complete task of sketch interpretation may be subdivided into three subtasks: (i) segmentation of drawing into disjoint shapes that are recognized independently, (ii) fitting of shapes (models) to drawings, and (iii) recognition of particular shapes. The approach described in this paper tackles two latter tasks. However, we discuss the possibility of tackling the segmentation task as well.

The primary contribution of this paper may be summarized as development and practical verification of a novel method to object recognition that (i) uses genetic programming to evolve visual learners, (ii) estimates learner's fitness by assessing its ability to restore essential features of the input image, and (iii) uses visual primitives as basic 'granules' of visual information.

2 Generative Visual Learning

The proposed approach may be shortly characterized as *generative visual learning*, as our evolving learners aim at reproducing the input image and are rewarded according to the quality of that reproduction. The reproduction is partial, i.e., the learner restores only a particular *aspect* of the image contents. In this paper, the aspect of interest is shape, whereas other factors, like color, texture, shading, are ignored.

The reproduction takes place on a virtual canvas spanned over the input image. On that canvas, the agent is allowed to perform some elementary *drawing actions* (DAs for short). To enable successful reproduction, DAs should be compatible with the image aspect that is to be reconstructed. In this paper, we consider hand-drawn polygons and, to enable the learner to restore their shape, we make our DAs draw sections.

As an example, let us consider reconstruction of an empty triangular shape. It requires from the learner performing the following steps: (i) detection of conspicuous features — triangle corners, (ii) pairing of the detected triangle corners, and (iii) performing DAs that connect the paired corners. However, within the proposed approach, the learner is not given *a priori* information about the concept of the corner nor about the expected number of them. We expect the learner to discover these on its own.

To reduce the amount of data that has to be processed and to bias the learning towards the image aspect of interest, our approach abstracts from raster data and relies only on selected salient features in the input image s. For each locally detected feature, we build an independent *visual primitive* (VP for short). The complete set of VPs derived from s is denoted in following by P.

The learning algorithm itself does not make any assumptions about the particular salient feature used for VP creation. Reasonable instances of VPs include, but are not limited to, edge fragments, regions, texems, or blobs. However, the type of detected feature determines the image aspect that is reconstructed. As in this paper we focus on shape, we use VPs representing prominent local luminance gradients derived from s using a straightforward procedure. Each VP is described by three scalars called hereafter *attributes*; these include two spatial coordinates of the edge fragment and the local gradient orientation.

3 GP-Based Learners

On the top level, the proposed method uses evolutionary algorithm that maintains a population of visual learners (individuals, solutions), each of them implemented as genetic programming (GP, [4]) expression. Each visual learner L is a procedure written in a form of a tree, with nodes representing *elementary operators* that process sets of VPs. The terminal nodes (named *ImageNode*s) fetch the set of primitives P derived from the input image s, and the consecutive internal nodes process the primitives, all the way up to the root node. A particular tree node may (i) group primitives, (ii) perform selection of primitives using constraints imposed on VP attributes or their other properties, or (iii) add new attributes to primitives.

Table 1 presents the complete list of GP operators that may reside in particular tree nodes. We use strongly-typed GP (cf. [4]), which implies that two operators may be connected to each other only if their input/output types match. The following types are used: numerical scalars ($\Re$ for short), sets of VPs (Ω, potentially nested), attribute labels (A), binary arithmetic relations (R), and aggregators (G).

The non-terminal GP operators may be divided into following categories:

1) Scalar operators (as in standard GP applied to symbolic regression; see [4]). Scalar operators accept arguments of type $\Re$ and return result of type $\Re$.

2) Selectors. The role of a selector is to filter out some of the VPs it receives from its child node(s) according to some objectives or condition. Selectors accept at least one argument of type Ω and return result of type Ω. *Non-parametric selectors* expect two child nodes of type Ω and produce an output of type Ω. Operators that implement basic set algebra, like set union, intersection, or difference, belong to this category. *Parametric selectors* expect three child nodes of types Ω, A, and $\Re$, respectively, and produce output of type Ω. For instance, operator *LessThan* applied to child nodes (P, p_o, 0.3) filters out all VPs from P for which the value of the attribute p_o (orientation) is less than 0.3.

Table 1. The GP operators

Type	Operator
$\Re$	Ephemeral random constant
Ω	*ImageNode* – the VP representation P of the input image s
A	p_x, p_y, p_o and custom attributes added by *AddAttribute*
R	*Equals*, *Equals5Percent*, *Equals10Percent*, *Equals20Percent*, *LessThan*, *GreaterThan*
G	Sum, Mean, Product, Median, Min, Max, Range
$\Re$	$+(\Re,\Re)$, $-(\Re,\Re)$, $*(\Re,\Re)$, $/(\Re,\Re)$, $\sin(\Re)$, $\cos(\Re)$, abs($\Re$), sqrt($\Re$), sgn($\Re$), $\ln(\Re)$
Ω	*SetIntersection*(Ω,Ω), *SetUnion*(Ω,Ω), *SetMinus*(Ω,Ω), *SetMinusSym*(Ω,Ω), *SelectorMax*(Ω,A), *SelectorMin*(Ω,A), *SelectorCompare*(Ω,A,R,$\Re$), *CreatePair*(Ω,Ω), *CreatePairD*(Ω,Ω), *ForEach*(Ω,Ω), *ForEachCreatePair*(Ω,Ω,Ω), *ForEachCreatePairD*(Ω,Ω,Ω), *AddAttribute*(Ω,$\Re$), *AddAttributeForEach*(Ω,$\Re$), *GroupHierarchyCount*(Ω,$\Re$), *GroupHierarchyDistance*(Ω, $\Re$), *GroupProximity*(Ω, $\Re$), *GroupOrientationMulti*(Ω, $\Re$), *Ungroup*(Ω), *Draw*(Ω)

3) Iterators. The role of an iterator is to process one by one the VPs it receives from one of its children. For instance, operator *ForEach* iterates over all the VPs from its left child and processes each of them using the GP code specified by its right child. The VPs resulting from all iterations are grouped into one VP and returned.

4) Grouping operators. The role of those operators is to group primitives into a certain number of sets according to some objectives or conditions. For instance, *GroupHierarchyCount* uses agglomerative hierarchical clustering, where euclidean distance of primitives serves as the distance metric.

5) Attribute constructors. An attribute constructor defines and assigns a new attribute to the VP it processes. The definition of a new attribute, which must be based on the values of existing VP attributes, is given by the GP code contained in the right child subtree. To compute the value of a new attribute, attribute constructor passes the VP (operator *AddAttribute*) or the sub-primitives of the VP (operator *AddAtributeToEach*) through that subtree. Attribute constructors accept one argument of type Ω and one of type $\Re$, and return a result of type Ω.

The detailed description of all implemented operators may be found in [5,6].

Given the elementary operators, the learner L applied to an input image s builds gradually a hierarchy of VP sets derived from s. Each application of selector, iterator, or grouping operator creates a new set of VPs that includes other elements of the hierarchy. In the end, the root node returns a nested VP hierarchy built atop of P, which reflects the processing performed by L for s. Some of the elements of the hierarchy may be tagged by new attributes created by attribute constructors.

Figure 1 illustrates an example of VP hierarchy built by the learner in response to input image/stimulus s. In the left part of the figure, the short edge fragments labeled by single lower-case letters represent the original VPs derived from the

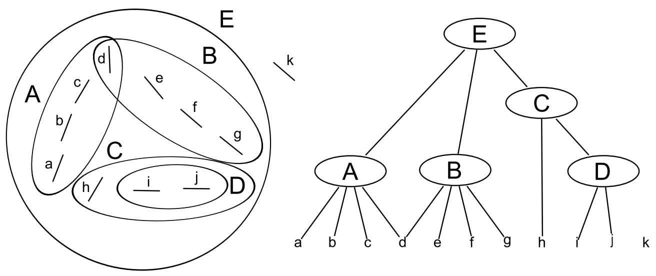

Fig. 1. The primitive hierarchy built by the learner from VPs, imposed on the image (left) and shown in an abstract form (right); VP attributes not shown for clarity

input image s, which together build up P. In the right part, the VP hierarchy is shown in an abstract way, without referring to the actual placement of particular visual primitives in the input image. Note that the hierarchy does not have to contain all VPs from P, and that a particular VP from P may occur in more than one branch of the hierarchy.

Individual's fitness is based on DAs (drawing actions) that it performs in response to visual primitives P derived from training images $s \in S$. To reconstruct the essential features of the input image s, the learner is allowed to perform DAs that boil down to drawing sections on the output canvas c. To implement that within the GP framework, an extra GP operator called *Draw* is included in the set of operators presented in Table 1. It expects as an argument one VP set T and returns it unchanged, drawing on canvas c sections connecting each pair of VPs contained in T.

The drawing created on the canvas c by the learner L for an input image s is then evaluated to provide feedback for L and enable its potential improvement. This evaluation consists in comparing the contents of c to s. For this purpose, a simple and efficient approach was designed. In general, this approach assumes that the difference between c and s is proportional to the minimal total cost of bijective assignment of lit pixels of c to lit pixels of s. The total cost is a sum of costs for each pixel assignment. The cost of assignment depends on the distance between pixels in the following way. When the distance is less than 5, the cost is 0; maximum cost equals 1 when the distance is greater than 15; between 5 and 15 the cost is a linear function of the distance. For pixels that cannot be assigned (e.g., because there are more lit pixels in c than in s), an additional penalty of value 1 is added to the total cost. In order to compute the minimal total cost of assignment a greedy heuristic was applied.

The (minimized) fitness of L is defined as the total cost of the assignment normalized by the number of lit pixels in $s \in S$, averaged over the entire training set of images S. The ideal learner perfectly restores shapes in all training images and its fitness amounts to 0. The more the canvas c produced by a learner differs from s, the greater its fitness value.

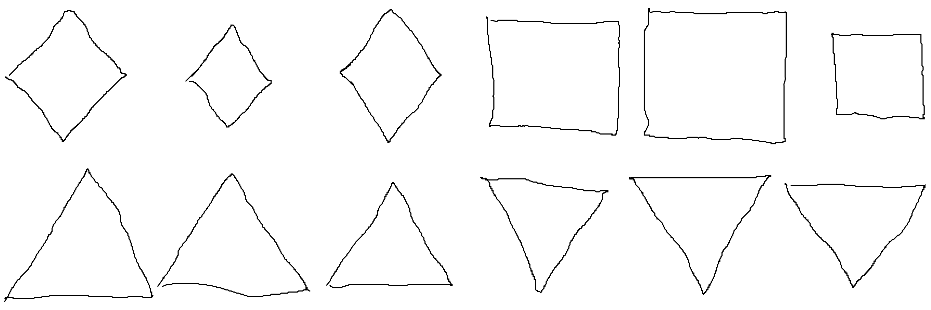

Fig. 2. Selected training examples

4 Related Research in Visual Learning

In most approaches to visual learning reported in literature, learning is limited to parameter optimization that usually concerns only a particular processing step, such as image segmentation, feature extraction, etc. A limited number of learning methods concerns more or less complete recognition systems [7,8,9,2,10,11]. In [1,2] we proposed a methodology that evolved feature extraction procedures encoded either as genetic programming or linear genetic programming individuals. The idea of GP-based processing of attributed visual primitives was explored for the first time in [12], and was further developed in [13,5,6].

The approach presented in this paper may be considered as a variant of generative pattern recognition. In a typical paper on that topic [14], Revow *et al.* used a predefined set of deformable models encoded as B-splines and an elastic matching algorithm based on expectation maximization for the task of handwritten character recognition. In [3], an analogous approach has been proved useful for recognition of hand-drawn shapes. However, the approach presented here goes significantly further, as it does not require *a priori* database of object models. And, last but not least, the recognition (restoration) algorithm has to restore the input image using *multiple* drawing actions, which implies the ability to decompose the analyzed shape into elementary components.

5 Experimental Evaluation

In this section we demonstrate how to apply the approach to recognize and classify hand-drawn sketches (shapes). Using a TabletPC computer we prepared a training set containing 48 images of four elementary shapes: diamonds (D), rectangles (R), triangles pointing upwards (TU), and triangles pointing downwards (TD), each shape represented by 12 examples. The shapes were of different dimensions, orientations and placed at random locations on a raster image of 640×480 pixels. Figure 2 illustrates selected training examples, shown for brevity in one image; note however, that each shape is a separate training example.

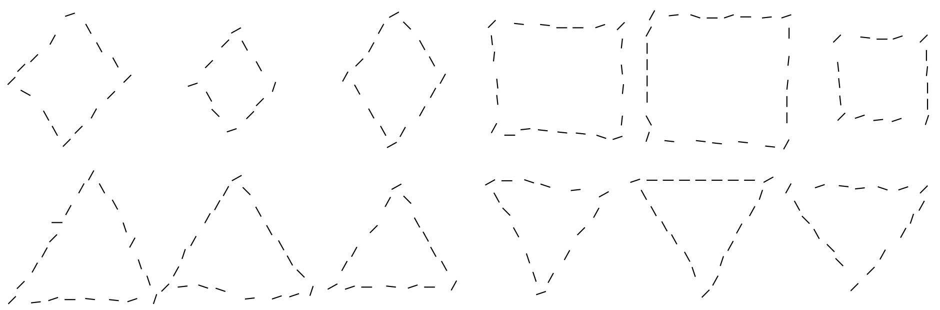

Fig. 3. Visualization of primitives derived from objects depicted in Fig. 2

Figure 3 shows visualization of primitives obtained from objects from Fig. 2. Each segment depicts a single VP, with its spatial coordinates located in the middle of the segment and the orientation depicted by slant.

Technically, we used generative evolutionary algorithm maintaining the population of 25,000 GP individuals for 300 generations. Koza's ramped half-and-half operator with ramp from 2 to 6 [4] was used to produce the initial population. We applied tournament selection with tournament of size 5, using individuals' sizes for tie breaking and thus promoting smaller GP trees. Offspring were created by crossing over selected parent solutions from previous generation (with probability 0.8), or mutating selected solutions (with probability 0.2). The GP tree depth limit was set to 10; the mutation and crossover operations might be repeated up to 5 times if the resulting individuals do not meet this constraint; otherwise, the parent solutions are copied into the subsequent generation. Except for the fitness function implemented for efficiency in C++, the algorithm has been implemented in Java with help of the ECJ package [15]. For evolutionary parameters not mentioned here explicitly, ECJ's defaults have been used.

The experiment was conducted according to the following procedure. First, for each class of shape (D, R, TU and TD), 5 evolutionary runs were performed using the training set for fitness computation. From each run, the best individual (learner) was chosen. These 20 individuals constituted the pool of individuals, which we used to build a recognition system that was subsequently evaluated on a separate test set containing 124 shapes. We present results for two recognition systems:

1) Simple recognition system consists of 4 best-in-class individuals, selected from the pool according to fitness value (based on the training data). This straightforward system performs recognition of the test example t by computing fitnesses of all four individuals for t and indicating the class associated with the fittest individual. The rationale behind such a procedure is following. The learner was taught only to perform well on images from one class and its fitness should be near 0 only for images of this class. For example, it is unlikely that an individual that learned the concept of triangle could recognize squares, thus it will get a high fitness value. Simple recognition system achieves test-set accuracy of classification of 88.71%; the detailed confusion matrix is presented in Table 2a.

Table 2. Test-set confusion matrices for simple recognition system (a) and voting recognition system (b) (rows: actual object classes, columns: system's decisions)

(a)

	D	R	TU	TD
D	**33**	0	3	0
R	1	**25**	3	2
TU	0	0	**28**	0
TD	5	0	0	**24**

(b)

	D	R	TU	TD
D	**34**	1	1	0
R	2	**27**	1	1
TU	0	0	**28**	0
TD	0	0	0	**29**

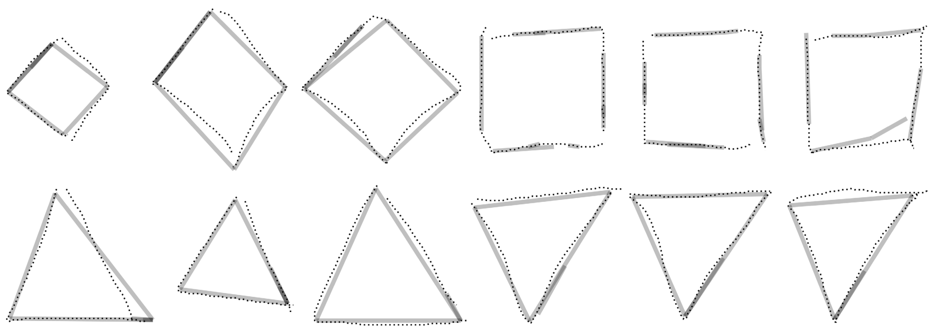

Fig. 4. The generative restoration process performed by trained learners

2) Voting recognizer uses all 20 individuals from the pool and runs multiple voting procedures to utilize the diversity of individuals obtained from different evolutionary runs. Technically, all 5^4 possible combinations of individuals-voters are considered, always using one voter per class. Each voting produces one decision, using the same procedure as the simple recognition system. The class indicated most frequently in all votings is the final decision of the recognition system. This approach performs significantly better than the simple recognition system and attains test-set recognition accuracy of 95.16% (confusion matrix presented in Table 2b).

In Fig. 4, we illustrate the process of generative shape restoration performed by the well-performing individuals for randomly selected test shapes. Thin dotted lines mark the shapes drawn by a human, whereas thick continuous lines depict drawing actions performed by the individual (sections). Darker colors reflect overlapping of multiple DAs. It may be easily observer that, in most cases, the evolved individuals successfully reproduce the overall shape of the recognized object. Reproduction seems to be robust despite various forms of imperfectness of the hand-drawn figures.

In Fig. 5, we present the GP code of selected individual trained to recognize objects from the TU class. It should be emphasized that this individual uses several nodes to issue drawing actions. In this way, the evolved individual exploits the inherent ability of our approach that allows gradual composition of recognized object from primitive components (sections).

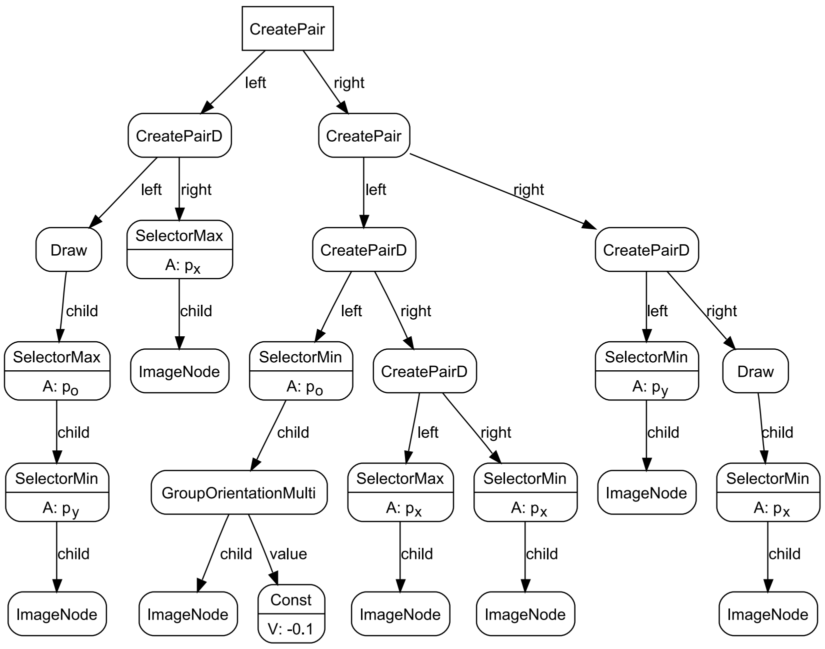

Fig. 5. The GP code of selected well-performing individual trained on TU class

6 Conclusions

The obtained results demonstrate the ability of the approach to evolve generative object recognition systems that successfully classify real-world sketches. This result has been obtained using very limited background knowledge, encoded in GP operators. Learners are not provided with negative examples (each learner is trained only on examples from its own class), so new object classes may be added without the need of modifying the existing recognizers. The method offers low time complexity, resulting mostly from primitive-based processing. Recognition of an object takes on average only 0.1 ms for implementation running on 2.4 GHz AMD Opteron processor.

In this preliminary study we focused on recognition of basic geometrical shapes. Interpretation of more complex images may require more sophisticated and substantially larger GP individuals. To alleviate scalability issues that may arise in such a case, we devised an extended approach that performs automatic decomposition of image processing into multiple GP trees and enables sharing of decomposed trees between multiple learning tasks. Preliminary results indicate that such approach is profitable in terms of convergence speed [5,6].

Future research could concern other aspects of visual information, like color or texture, and other input representation spaces, like region adjacency graphs. It would be also interesting to investigate the possibility of some integration of different aspects of visual stimuli.

References

1. Bhanu, B., Lin, Y., Krawiec, K.: Evolutionary Synthesis of Pattern Recognition Systems. Springer-Verlag, New York (2005)
2. Krawiec, K., Bhanu, B.: Visual learning by coevolutionary feature synthesis. IEEE Transactions on System, Man, and Cybernetics – Part B **35** (2005) 409–425
3. Krishnapuram, B., Bishop, C.M., Szummer, M.: Generative models and bayesian model comparison for shape recognition. In: IWFHR '04: Proceedings of the Ninth International Workshop on Frontiers in Handwriting Recognition (IWFHR'04), Washington, DC, USA, IEEE Computer Society (2004) 20–25
4. Koza, J.: Genetic programming – 2. MIT Press, Cambridge, MA (1994)
5. Jaskowski, W.: Genetic programming with cross-task knowledge sharing for learning of visual concepts. Master's thesis, Poznan University of Technology, Poznań, Poland (2006)
6. Wieloch, B.: Genetic programming with knowledge modularization for learning of visual concepts. Master's thesis, Poznan University of Technology, Poznań, Poland (2006)
7. Teller, A., Veloso, M.: PADO: A new learning architecture for object recognition. In Ikeuchi, K., Veloso, M., eds.: Symbolic Visual Learning. Oxford Press, New York (1997) 77–112
8. Rizki, M., Zmuda, M., Tamburino, L.: Evolving pattern recognition systems. IEEE Transactions on Evolutionary Computation **6** (2002) 594–609
9. Maloof, M., Langley, P., Binford, T., Nevatia, R., Sage, S.: Improved rooftop detection in aerial images with machine learning. Machine Learning **53** (2003) 157–191
10. Olague, G., Puente, C.: The honeybee search algorithm for three-dimensional reconstruction. In Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H., eds.: EvoWorkshops. Volume 3907 of Lecture Notes in Computer Science., Springer (2006) 427–437
11. Howard, D., Roberts, S.C., Ryan, C.: Pragmatic genetic programming strategy for the problem of vehicle detection in airborne reconnaissance. Pattern Recognition Letters **27** (2006) 1275–1288
12. Krawiec, K.: Learning high-level visual concepts using attributed primitives and genetic programming. In Rothlauf, F., ed.: EvoWorkshops 2006. LNCS 3907, Berlin Heidelberg, Springer-Verlag (2006) 515–519
13. Krawiec, K.: Evolutionary learning of primitive-based visual concepts. In: Proc. IEEE Congress on Evolutionary Computation, Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada July 16-21. (2006) 4451–4458
14. Revow, M., Williams, C.K.I., Hinton, G.E.: Using generative models for handwritten digit recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **18** (1996) 592–606
15. Luke, S.: ECJ evolutionary computation system (2002) (http://cs.gmu.edu/eclab/projects/ecj/).