Genetic Programming for Primitive-Based Acquisition of Visual Concepts

Krzysztof Krawiec[1]

[1] Poznań University of Technology, Institute of Computing Science, Poznań, Poland,
e-mail: krawiec@cs.put.poznan.pl

Abstract. We describe a novel method for acquisition of higher-level visual concepts using GP-based learners that process attributed visual primitives derived from raw raster images. The approach uses an original evaluation scheme: individuals-learners are rewarded for being able to restore the essential features (here: shape) of the visual stimulus. The approach is general and does not require any a priori knowledge about the particular application or target concept to be learned; the only prerequisite is universal knowledge related to interpretation of visual information, encoded in nodes of GP trees. The paper demonstrates the performance of the method on a specific visual task of acquiring the concept of a triangle from examples given in a form of raw raster images.

1 Introduction

The primary motivation for research described in this paper is the lack of a general and widely applicable methodology for automated design of pattern recognition (PR) and computer vision (PR) systems. Manual development of such systems is for most real-world tasks tedious, time-consuming and expensive. The handcrafted solutions are usually limited in their scope of applicability and have poor ability to adapt, i.e., to perform *visual learning*.

In authors' opinion, visual learning is the only way to build application-independent systems with the ability to understand a broad class of images. However, 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. Reports on methods that synthesize complete object recognition systems starting from raw image data are rather scant. Most of them are also application-specific that makes the acquired knowledge difficult or, in most cases, impossible to transfer to other recognition tasks and applications.

Moreover, contemporary research on learning in CV/PR is focused mainly around numerical or subsymbolic paradigms. For instance, in the area of cognitive science, neural methods dominate the topic. In other approaches, it is common to extract from the image a predefined (and often large) set of numerical features and feed them into standard machine learning algorithms. Such features are usually tailored to a specific application. Attempts to approach visual learning in a more symbolic way are rather scant – the most prominent example would be here syntactic

[1] This work has been supported by KBN research grant 3 T11C 050 26.

pattern recognition, which is, however, known for some limitations (difficult learning, high sensitivity to noise, and imprecision in image data).

We claim that visual learning needs stronger reference to symbolic approaches and effective search heuristics like evolutionary computation (EC); in particular, the possibility of the encapsulation of earlier acquired concepts (like module acquisition in Genetic Programming (GP)) seems to especially appealing for gradual acquisition of visual concepts (*developmental learning*). By 'symbolic' we mean here recognition systems that analyze, process, and interpret visual information using some elementary operators that refer to basic common-sense and mathematical concepts. To meet this objective, in this paper we propose a visual learning paradigm where both the processing (image recognition system) as well as input/training data (image representation) are defined in a symbolic way. In particular, each learner, implemented as a GP individual, works with visual primitives (VPs) that represent local salient features derived from the raw input raster image. It processes such visual information using a sequence of predefined yet general operators which represent background knowledge. Most importantly, the final result of the learner's computation is also given in a symbolic form.

The proposed method allows for building general, higher-level visual concepts by grouping, selecting, aggregating the visual primitives, and defining new attributes that describe them. It provides a gradual transition from subsymbolic raster image data to higher-level visual concepts. By working directly with VPs rather than with some predefined scalar features derived from the image, the learners conform also the principle of least commitment [8]. Moreover, the symbolic setting significantly speeds up the processing of visual information, as the volume of visual primitives data is usually only a fraction of that of the raster image from which they have been derived. This feature is extremely important from an EC viewpoint, as the learning process maintains a large population of visual learners which are evaluated on a set of training images.

Nevertheless, the major benefit from symbolic representation and the main novelty of the method described here is that learner's response to the input image may be interpreted in a meaningful manner. It allows us to construct learner's *interpretation* of the input stimulus and evaluate it with respect to *simplicity* and *accuracy* (conformance) when compared to the input image. We define appropriate criteria that measure these features of learner's response and build up a multiobjective fitness, which in turn drives the evolutionary search. Such interpretation enables us to close the feedback loop of the learning process, i.e., to assign an appropriate fitness to each learning individual, despite the fact that the learning is unsupervised in the sense that there is no extra information on desired output produced by the learner. Thus, to some extent, the method described here may be attributed as *generative*.

Thanks to generic representation of visual data and to the unsupervised nature of the learning process, the proposed approach is application-independent and abstracts from any specific task like recognition, identification, or tracking. Though CV/PR achievements in selected applications like face recognition, fingerprint classification, or aerial imagery, are unquestionable, they are extremely application-oriented and require lots of domain-specific knowledge. It is only thanks to immense human effort in designing/choosing the appropriate image processing and analysis procedures, that such recognition systems are able to solve the recognition tasks they have been designed for. It is difficult to see how the knowledge encoded in such systems could contribute to the actual understanding of generic visual information.

The original contributions of this paper may be summarized as development of a novel variant of typed genetic programming that (i) is customized to process visual primitives (VPs) by as

basic 'granules' of visual information, and (ii) uses the ability to restore essential features of the input image to guide the evolutionary learning.

The following Section 2 briefly summarizes the related work on the topics of visual learning and visual evolutionary learning. Then, in Section 3 we thoroughly describe the proposed approach. Section 4 demonstrates the performance of the approach on a visual task of acquiring a simple yet nontrivial higher-level visual concept. In Section 5, we provide a summary and draw conclusions for further research.

2 Related Work

In most approaches to visual learning reported in the literature, learning is limited to parameter optimization that usually concerns a particular processing step, such as image segmentation, feature extraction, etc. Only limited number of methods close the feedback loop of the learning process at the highest (e.g., recognition) level [1][2][4][6][7][9][10][12]. Also, reports on approaches that learn using raw images as training data, and, therefore, produce the entire object recognition system, are rather scant. Moreover, some of the proposed methods make use of domain-specific knowledge and are highly specialized towards a particular application.

In our previous work, we applied the idea of symbolic processing of attributed visual primitives to the specific *supervised* learning task of recognizing objects (computer screens) in office scenes. Though the approach produced interesting results, it was obvious that learning from such specific task cannot lead to elaboration of more general visual concepts, which, in turn, would be useful in solving potentially wider range of visual learning tasks.

3 Symbolic Primitive Learning

3.1 Processing Visual Primitives

In the proposed method, the learning takes place in an evolving population of *visual learners* encoded as GP individuals (expression trees). In general terms, we define a visual learner as an entity which is able to process visual data represented as a set P of attributed *visual primitives* (VPs for short); let Ω denote the space of all such representations. Each individual is evaluated based on the results it produces when applied to a set of training images (examples) S. In machine learning terms, S represents the *target concept* to be learned (or, precisely, a finite sample of such a concept). The learner is stimulated by each training image $s \in S$ independently. In response to s, it produces its representation, expressed also in terms of VPs.

The processing carried out for each learner (GP individual) L and for each training image $s \in S$ is outlined in Fig. 1. In the diagram, P_s, $P_s \in \Omega$, denotes the set of VPs derived from the original training raster image $s \in S$ (visual stimulus). The algorithm does not make any assumptions about the particular form of visual primitives. Reasonable instances of VPs include edge fragments, regions, or blobs. In the variant of our approach presented in this paper, VPs represent short edge fragments detected in s by straightforward image processing (see Section 4).

Each VP $p \in P_s$ is described by a vector of scalars called hereafter *primitive attributes*; in this paper, this vector encompasses the coordinates of the edge fragment (p_x and p_y), and edge orientation p_o. Attribute values are derived from local features of the input raster image s. As this step is fixed and common for all learners, it is carried out prior to the evolutionary run and the P_s's are

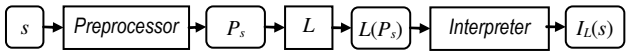


Figure 1. The processing of visual information for a particular learner/individual *L*.

cached for all images $s \in S$ to reduce time complexity of evolutionary run. The technical details on this process are provided in Section 4.

In Fig. 1, $L(P_s)$ denotes the representation of image *s* produced by the learner *L*. This representation is also defined in terms of VPs. In this paper, we assume $L(P_s)$ to be a hierarchy of sets of primitives derived from P_s. In other words, $L(P_s)$ is a hierarchical set of VPs (a tree of nested sets of VPs) built by *L* atop of P_s. The nodes of the hierarchy encapsulate (group) other nodes from $L(P_s)$ and VPs from P_s. The learner *L* may also assign some new attributes to any nodes of hierarchy, in addition to the original primitive attributes of VPs. The particular structure and contents of that hierarchy determines the fitness assigned to *L* in the way described in the following subsection.

Figure 2 illustrates an example of VP $L(P_s)$ representation produced by a learner *L* in response to input image/stimulus *s*. In the left part of the figure, the short edge fragments denoted by capitals represent the original VPs derived from the input image *s*, which in total build up P_s. The dashed-line shapes depict $L(P_s)$; in particular, each of the shapes denoted by p_1, p_2, p_3, and p_4, corresponds to a single VP created by the learner. In the right part of Fig. 2, the VP hierarchy is shown in an abstract way, without referring to the actual placement of particular visual primitives in the input image. In both parts of the figure, the primitive attributes of VPs are not shown for clarity. Note that the hierarchy does not have to contain all VPs from P_s, and that a particular VP from P_s may occur in more than one branch of the hierarchy tree.

3.2 Fitness Estimation

The representations $L(P_s)$ produced by the learner for all training examples $s \in S$ constitute learner's *L acquired concept*. To estimate *L*'s fitness, its acquired concept has to be confronted with the target visual concept. In supervised learning, this is usually done by assessing how well the learner discriminates the training examples representing different *classes* of concepts from each other. Such an approach has, however, some drawbacks. First of all, it requires *a priori* labeling of visual stimuli, which is tedious and time-consuming. Existing labeled image collections are usually tailored towards recognition of very complex objects like humans or cars (see, e.g., the MIT-CSAIL database [13]). The knowledge acquired while learning such specific tasks is difficult to generalize to/apply in other visual learning tasks.

Secondly, the immense number of degrees of freedom available to the learners, resulting from the vast space of GP solutions that may be considered during search, increases significantly the risk of overfitting. In our past experience with evolutionary design of pattern recognition systems synthesized in a supervised way, an evolved recognition system would often found its decision on an irrelevant feature of the input image, which was coincidentally correlated with the actual partitioning of training examples into concepts. Such overfitting can be fought by constraining the search space and/or GP procedure representation, however, such an intervention requires background knowledge and is somehow arbitrary.

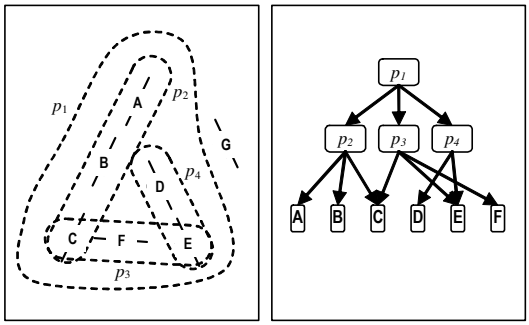


Figure 2. The primitive hierarchy built by the learner from the visual primitives, imposed in the image (left) and shown in an abstract form (right); VP attributes not shown for clarity.

But most importantly, the supervised setting of the learning process requires making, in our opinion a very arbitrary, decision about learner's *desired output/response* for particular input examples or target concepts. When the learner implements a machine learning classifier, this requirement is in a natural way imposed by a particular type of knowledge representation (e.g., desired combination of output layer excitations for an artificial neural network). However, applicability of such learning methods is limited to simple recognition tasks with a limited number of concepts to be learned. For more complex objects and for large numbers of concepts to be acquired, special means have to be applies (e.g., *explicit* building of object models and model-based approach).

The supervised setting imposes therefore undesired biases on the learning process, which in turn impact the generality of the evolved solutions. To avoid bias toward particular recognition tasks and to evolve learners who are able to recognize more general visual concepts, the learning process proceeds here in an unsupervised mode in the sense that the *learner is not explicitly told what is the particular contents of output it should produce*. Rather we reward the learners for elaborating acquired concepts that are both *simple* and *accurate*.

The *simplicity* criterion, denoted in following by $f_{sim}()$, is motivated by Occam's razor principle and by the obvious observation that simple concepts usually generalize better. The way we determine the *accuracy* $f_{acc}()$ of the acquired concept with respect to the target concept is the central idea of the proposed approach: $f_{acc}(L)$ measures to what extent it is possible to *reconstruct/restore the essential features of the training example* $s \in S$ using the image representation $L(P_s)$ produced by the learner L. To this aim, we introduce an *Interpreter* (see Fig. 1), which produces a *reconstructed image* $I_L(s)$ based on $L(P_s)$. The reconstructed image $I_L(s)$ is then compared to the original image s.

Even without providing technical details, one can easily conclude that criteria $f_{sim}()$ and $f_{acc}()$ are conflicting. To compete with other learners in the population on $f_{sim}()$, the learner tends to elaborate such representations $L(P_s)$ that make the Interpreter produce output that is simplified in comparison to P_s, as far as information contents is concerned (e.g., the Interpreter reproduces only the most salient features of P_s). On the other hand, to be able to restore the essential features of the input stimuli, L has to provide the resulting representation $L(P_s)$ in enough information (hierarchy nodes and attributes), which inevitably deteriorates its evaluation on $f_{sim}()$.

When understanding the accuracy criterion literally and without providing any extra means, these criteria must be contradictory: any attempt to reduce the size of representation results in

loss of accuracy. However, the tightness of this trade-off is significantly alleviated by the following two assumptions.

1) Firstly, we do not expect the learner to produce an acquired concept that enables restoration of *all* information contained in the original training image *s*. Rather than that, we limit our interest to only one aspect of visual stimuli, *shape* in the current variant of our approach. Other features, like color or texture, are ignored. We also limit our interest to 2D visual stimuli, thus other aspects related to perception of 3D scenes, like depth, shading, etc., are ignored. $f_{acc}()$ measures only how well does the Interpreter restore the *shape* of the input stimulus *s* given $L(P_s)$.

2) Quite obviously, the Interpreter has to be equipped with some means of producing the reconstructed image $I_L(s)$. For this purpose, we assume that the Interpreter may literally *draw* simple graphical primitives on a 'canvas' and that the result of that drawing is returned as $I_L(s)$. As in this paper we focus on shape, we let the Interpreter draw only the simplest possible graphical primitives: sections.

The coordinates of section ends drawn by Interpreter are retrieved from $L(P_s)$. The problem which arises at this point is which nodes of the $L(P_s)$ hierarchy should be interpreted as section ends. Though one could come up with various arbitrary methods for retrieving coordinate data from $L(P_s)$, any such choice would introduce unnecessary bias into the learning process. To avoid that, we *let the learner decide* what parts of the output it produces should be drawn by the Interpreter. Technically, we introduce an extra GP node (*Draw*) which does no processing but 'tags' the selected part of the produced $L(P_s)$. Interpreter then traverses $L(P_s)$ and draws sections spanned over tagged nodes.

A very important advantage of such an approach is that the individual/learner *L* together with the Interpreter produces a 'rich' output $I_L(s)$ in response to the input stimulus, as opposed to, for instance, the case of supervised learning, where individuals produce decision class labels or scalar features. In such a way, the individual undergoes a more thorough evaluation, which is, in a sense, based more on the actual *contents* of $L(P_s)$ rather than on its quality.

To search the space of GP solutions in a more thorough manner, we split the $f_{acc}()$ criterion into two maximized criteria: $f_{tp}()$ and $f_{fp}()$, which separately reflect the two types of errors that may be committed by Interpreter when trying to restore the shape of the input stimulus. The criterion $f_{tp}()$ measures the true positive ratio that depends on the cumulative brightness of pixels that belong to the target object *and* have been lit by the Interpreter. The criterion $f_{fp}()$ computes the false positive ratio. Formally,

$$f_{tp}(L) = \frac{1}{|S|}\sum_{s\in S}\frac{1}{|I_L(s)|}\sum_{(x,y)\in s, I_L(s)[x,y]>0}\max(s[x,y]-I_L(s)[x,y],0), \tag{1}$$

and

$$f_{fp}(L) = 1-\frac{1}{|S|}\sum_{s\in S}\frac{1}{|s|-|I_L(s)|}\sum_{(x,y)\in s, I_L(s)[x,y]>0}\max(I_L(s)[x,y]-s[x,y],0), \tag{2}$$

where $s[x,y]$ denotes the brightness of pixel (x,y) in the training image *s*, and $I_L(s)[x,y]$ denotes the brightness of the corresponding pixel in the Interpreter's output. The terms preceding summations serve for normalization: $|S|$ is the number of training images, and $|I_L(s)|$ and $|s|$ denote the total cumulative brightness (sum) of all pixels lit by the Interpreter and lit in the input image *s*, respectively.

The following properties of $f_{tp}()$ and $f_{fp}()$ need emphasizing:

1) Computing of both $f_{tp}()$ and $f_{fp}()$ requires considering only those pixels which have been drawn by the Interpreter (condition $I_L(s)[x,y]>0$ in formulas (1) and (2)). These pixels constitute in most cases only a tiny fraction of the image. This reduces significantly the time of fitness computation.

2) Rather than on the *number* of pixels being lit, the criteria rely on pixel *brightness*. This makes them more continuous and improves the convergence of the evolutionary run, especially when the input image is not binary but gray-scale (what is the case in the experiment described in the following).

3) The $f_{tp}()$ criterion is designed in such a way that it rewards an individual for 'drawing' sections that restore the shape the original image s. However, when Interpreter draws *multiple* sections that traverse *the same* pixel $[x,y]$, its brightness $I_L(s)[x,y]$ increases (technically, we use the 'alpha' transparency channel for that purpose). Thus, drawing sections that overlap (partially or completely) does not pay, as it decreases the value of the $I_L(s)[x,y]$-$s[x,y]$ term. The individual is therefore penalized for superfluous sections. The $f_{fp}()$ criterion has an analogous property.

The fact that $f_{tp}()$ and $f_{fp}()$ criteria penalize an individual for commanding the Interpreter to draw superfluous sections inclined us to give up using an *explicit* simplicity criterion in the experiment described in Section 4. In a sense, the particular criteria definitions in formulas (1) and (2) cause the simplicity to be *implicitly* comprised by $f_{tp}()$ and $f_{fp}()$ [2].

3.3 Representation of Individuals

For representation of solutions, we choose a variant of GP expressions [3]. Each learner is a GP tree with nodes representing elementary operators that are able to process sets of visual primitives. The tree fetches the VP representation P_s of the input image s using some of its terminal nodes and processes that representation, building consecutive levels of primitive hierarchy $L(P_s)$ on top of it. During that process, particular tree nodes introduce new primitives by aggregation (encapsulation) of VPs, perform selection of primitives using constraints imposed on attributes or other properties of primitives, and add new attributes to visual primitives on the arbitrary level of hierarchy. And, last but not least, the GP procedure may mark selected VPs in the created hierarchy for the Interpreter, to be drawn during interpretation process.

Technically, we use strongly-typed GP [3] with following elementary types: numerical scalars ($\Re$ for short), nested sets of VPs (Ω), attribute labels (A), binary arithmetic relations (R), and aggregators (G). We also define appropriate types of GP operators (nodes) that are able to handle particular types of data. The current specification of the approach includes two non-terminal operators: one producing an ephemeral random constant (type: $\Re$), and one producing the VP representation P_s of the input image s (type: Ω). The non-terminal GP operators used in this approach may be divided into the following categories:

1) *Scalar operators* (as in standard GP applied to symbolic regression; see [3]). Scalar operators accept argument(s) of type $\Re$ and return result of type $\Re$. The following scalar operators are currently implemented: +, -, *, /, *sin*, *cos*, *abs*, *sqrt*, and *sgn*.

---

[2] One should note, however, that there exist theoretical 'malign' solutions that lead to complex interpretation yet have relatively high values of $f_{tp}()$ and $f_{fp}()$. For instance, a straight line in the original input image s may be represented in $I_L(s)$ by a chain of sections spanned over consecutive VPs that have been derived from that line. Nevertheless, due to partial overlap of such sections, the evaluation of such a solution would be still inferior to a simpler interpretation composed a single section.

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 criteria 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, the operator *LessThan* applied to child nodes ($\{p_1, \ldots, p_n\}$, p_o, 0.3) filters out all VPs from $\{p_1, \ldots, p_n\}$ for which the value of the attribute p_o (orientation) is less than 0.3. When an inner node of the VP hierarchy is queried by a GP operator for attribute value, it computes that value by averaging the values of attributes of the VP primitives it contains. The current set of operators includes the following selectors: *SetIntersection*, *SetUnion*, *SetMinus*, *SetMinusSym*, *GroupProximity*, *Ungroup*, *SelectorMax*, *SelectorMin*, *SelectorCompareConstant*, and *Draw*. The set of binary relations that may be used by selectors includes: *Equals*, *EqualsPercent*, *Equals10Percent*, *Equals20Percent*, *LessThan*, and *GreaterThan*. The aggregators include: *Sum*, *Mean*, *Product*, *Median*, *Min*, *Max*, and *Range*.

3) *Iterators*. The role of an iterator is to process the VPs it receives from one of its children separately, one after another. Currently, there are two iterators implemented: *ForEach* and *ForEachCreatePair*.

4) *Attribute constructors*. The task of an attribute constructor is to assign a new attribute to the VPs it receives from its left child subtree. The new attribute that is to be added to VP is defined by the right child subtree of attribute constructor. To compute the value of a new attribute, an attribute constructor passes the VP through that subtree. That computation must be based on the values of the existing attributes; this is ensured by the leaves of the attribute definition subtree, which return the values of the defined attributes. Currently, two types of attribute constructors are included in the implementation of the method: *AddAttribute* and *AddAtributeForEach*. The former one operates on the top level (root node) of VP hierarchy and adds the new attribute to that node only. The latter one, on the contrary, performs the attribute creation for all VPs. Attribute constructors return result of type Ω.

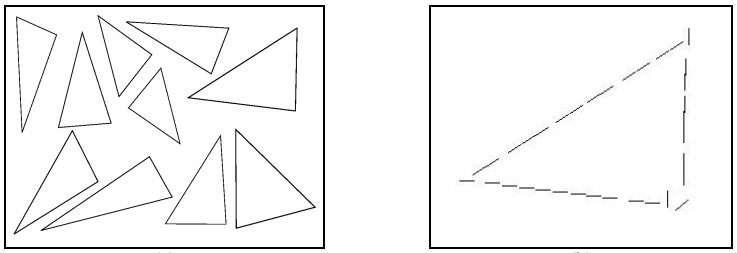
The detailed description of all operators implemented in our method is beyond the scope of this paper and will be presented in a separate research report.

4 The Experiment

4.1 The Task and Training Data

The objective of the experiment is to demonstrate our method on a task of acquiring a target visual concept of a triangle. The triangle concept is nontrivial and requires sophisticated design from the evolutionary learner, but at the same time the evolved solutions may be still relatively easily interpreted and verified by humans.

To attain the learning goal, the evolutionary process has to evolve a learner that is able to identify (e.g., select or create) the VPs representing triangle vertices and span appropriate sections on them by introducing marking tags for the Interpreter. This task, though straightforward for humans, is not that simple: remember that the only input the learner receives is P_s, a 'flat' set of a few dozens of VPs described by locations p_x, p_y and local gradient orientation p_o. The learner has no *a priori* information on, e.g., spatial proximity of VPs, their collinear alignment, etc. The



(a) (b)

Figure 3. The training set (a) and the VP representation P_s of one of the examples (b, magnified).

VPs located at triangle vertices are not marked as 'special' in any way; the learner has to 'discover' them by building an appropriate GP program.

We prepared a training set S containing 20 triangles of different sizes, shapes, and orientations, each placed in a random location on raster image of 640×480 pixels. Figure 3(a) illustrates all the training examples, shown for brevity in one image; note however, that each triangle constitutes a separate training example $s \in S$. Though these examples have been created in an artificial way, a real-world picture of such object would produce similar result.

In the preprocessing phase that transforms an input image s into its representation P_s, we extract *candidate VPs* from s based on local magnitude. Then, the obtained candidate locations are sorted with respect to decreasing magnitude, and at most 80% of most prominent of them are included into the primitive representation. Also, to filter out the less prominent candidates, we impose a lower limit d_{min} on the mutual proximity of VPs. The VP candidates are processed sequentially with respect to the decreasing magnitude, and a new primitive p may be added to P_s only if there is no other primitive already in P_s closer than d_{min}, i.e., there no $p' \in P_s$ such that $||p,p'|| < d_{min}$.

The resulting image representation P_s is usually several orders of magnitude more compact than the original image s. On the other hand, the essential sketch of the input image z is well preserved. Figure 3(b) shows, the VP representation P_s derived from one of objects from Fig. 3(a). Each segment depicts a single VP, with its (x,y) coordinates located in the middle of the segment and the orientation depicted by slant.

As we aim at acquiring a *general* concept of triangular shape, the issue of precision in interpretation is of secondary importance. Also, original VPs in P_s have discrete coordinates, which are not always perfectly consistent with the actual location of triangle pixels due to rounding errors in the preprocessing phase. Some preliminary evolutionary runs have shown that this issue may severely impact the convergence of the algorithm: individuals that restore the overall shape pretty well would receive low fitness due to minor imprecision of the Interpreter's drawing with respect to the input image. For these reasons, for the purpose of $f_{tp}()$ and $f_{fp}()$, we do two things: we 'fuzzyfy' the input images $s \in S$ by passing them through a lowpass filter. On the other hand, the sections drawn by the Interpreter are also drawn with stroke width equal to 3.

4.2 Evolutionary Algorithm Settings

With the solutions encoded as GP individuals and fitness function as described in Section 3.2 we performed an extensive series of preliminary experiments to investigate the characteristics of the evolutionary learning. In this contribution, we presented selected results obtained from a typical

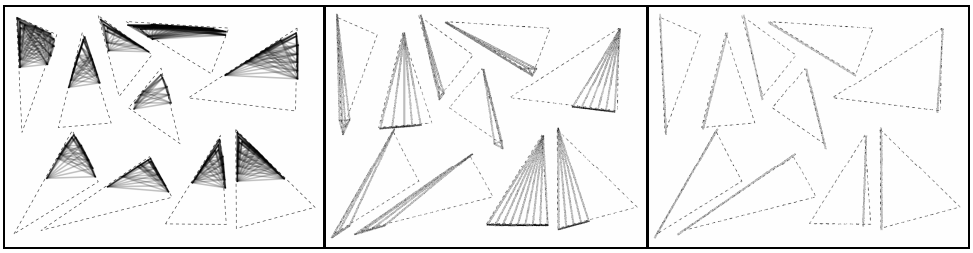


Figure 4. The output of three non-dominated solutions from generation 3 (left, fitness [0.118,0.981]) and generation 10 (middle, fitness [0.246,0.997], right, fitness [0.134,0.9998]).

run of the method. We use a generative evolutionary algorithm, for 100 generations with population of 2000 individuals. As any aggregation of $f_{tp}()$ and $f_{fp}()$ criteria would be arbitrary and involve undesired compensation, we rely on multiobjective evaluation of individuals/learners in the GP run. We use selection method based on Pareto-ranking: after the evaluation phase, the individuals are being ranked using dominance relation from the best to the worst rank. Then, the selection operator randomly selects an individual from r^{th} rank (r=1,2,…) with probability $\gamma/2^{(r-1)}$ (we used γ=0.6 in experiments). The initial population is populated using Koza's standard ramped half-and-half operator with ramp from 2 to 6 [3]. Following generations of individuals are created by crossing over the selected parent solutions from previous generation (with probability 0.5), mutating selected solutions (with probability 0.4), or copying individuals from the previous generation (probability 0.1). The GP tree depth limit is set to 8; the mutation and crossover operations may be repeated up to 5 times if the resulting individuals do not meet this constraint.

The framework for processing of visual primitives and the learning algorithm have been implemented in Java programming language, with help of the ECJ package [5] and Java Advanced Imaging library [11]. For evolutionary parameters not explicitly mentioned here we used their default values as specified in ECJ.

As estimation of $f_{tp}()$ and $f_{fp}()$ requires considering only those pixels which have been drawn by the Interpreter, the fitness estimation is fast and takes approx. 12ms per individual for our Java-based implementation running on 3.0 GHz Pentium processor. This is also due to the low number of primitives in the input representation P_s, which, for the learning task considered here, amounted to 30.3 on the average.

4.3 The Results

Figure 4 shows examples of output produced (drawn by the Interpreter) for three selected Pareto non-dominated individuals evolved in preliminary generations of the evolutionary run. Similarly to Fig. 3(a), this and following figures show the result for the entire training set superimposed in one picture. Dashed lines represent the input examples $s \in S$, whereas continuous lines show the output $I_L(s)$ drawn by the Interpreter when applied to the representation $L(P_s)$ produced by the individual. The pixel intensity (darkness) of the latter lines reflects overlapping of multiple sections, and influences the fitness value (see Section 3.2).

Though, in theory, both criteria range from 0.0 to 1.0 inclusive, the $f_{tp}()$ criterion usually does not draw near 1.0 due to the fact that the primitives are seldom placed *precisely* in the vertices of the recognized objects. It is then difficult for the evolutionary process to elaborate solutions that

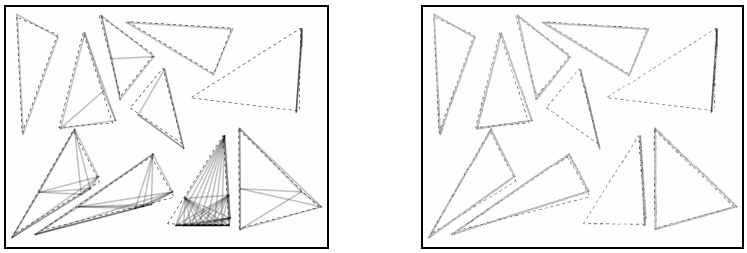


Figure 5. The output of two non-dominated solutions from generation 40 (left, fitness [0.326,0.997]) (right, fitness [0.242,0.9997])

span the recognized object precisely at the vertices. Nevertheless, thanks to the fuzzyfication of criteria described in Section 4.1, and thanks to the fact that the selection method takes into account only the *ordering* of the solutions on particular criteria, this does not seem to affect the convergence of the algorithm.

Figure 5 depicts the outputs obtained from selected well-performing non-dominated individuals found in the 40^{th} generation of the evolutionary run. The bottom part of the figure represents one of the best individuals evolved in the run, which correctly identifies 7 out of 10 triangles. Note that all three incorrectly identified triangles share common feature: an almost vertical right-end edge. As a matter of fact, in 2 of these cases this edge is drawn thrice by the Interpreter.

The figures demonstrate well that the solutions elaborated in preliminary generations produce representations that use too many (Fig. 4 left) or too few (Fig. 4 right) sections. During the evolutionary run, the selective pressure promotes solutions that are almost optimal (Fig. 5 right). Obviously, the figures depict only a fraction of a wide range of non-dominated solutions; for instance, in each generation there is usually a significant fraction of solutions that do not produce any drawing (i.e., do not mark any nodes in the representation hierarchy $L(P_s)$ to be drawn by the Interpreter).

5 Conclusions

We demonstrated the possibility of evolving an image understanding system that is able to reasonably interpret compound patterns present in observed images. This result has been obtained using very limited background knowledge, contained in GP operators and in the ability of the Interpreter to restore the shape of the analyzed object using the a simple primitive (section). The method elaborates symbolic concepts of the objects to be recognized, which potentially enables further re-use in more advanced visual learning tasks.

The approach seems to be general enough to work similarly well for other aspects of visual information, like regions or texture. It would be quite straightforward to apply the method to such data; essentially, only primitive definition would change in such a case. In perspective, some integration of different aspects of visual stimuli is also possible.

The recognition systems (i.e., selected non-dominated GP individuals) $L_1, L_2, \ldots, L_m$, evolved for various target concepts $T_1, T_2, \ldots, T_m$, may be used for interpreting an unknown contents of a new image x, including recognition (classification). For this purpose, we propose the following classification procedure. The image x is first processed by recognition systems $L_1, L_2, \ldots, L_m$ as shown in Fig. 1. Then, the resulting interpretations $I_{L1}(x), I_{L2}(x), \ldots, I_{Lm}(x)$, undergo fitness estimation described in Section 3.2. Finally, the obtained fitness values $[f_{tp}(L_1), f_{fp}(L_1)]$,

$[f_{tp}(L_2), f_{fp}(L_2)]$, …, $[f_{tp}(L_m), f_{fp}(L_m)]$, are mutually compared. If any evaluation $[f_{tp}(L_i), f_{tp}(L_i)]$ strictly dominates all other evaluations, this would unambiguously indicate target T_i as the 'class' that x most probably belongs to. In other words, L_i provides the most accurate and the simplest 'theory' for explaining the actual contents of x. Other cases, with multiple non-dominated evaluation, would require an extra disambiguation procedure.

The visual concepts elaborated in the learning process could be used as an initial knowledge base for visual learners applied to other visual learning tasks. The ultimate objective of our research is to elaborate a methodology for *developmental learning* of visual concepts of gradually increasing complexity. In such a case, the learning process would face a *sequence* of learning tasks, with each learning task devoted to a particular visual concept to be learned/acquired, as described in this paper. In such a setting, the learners make use of visual concepts acquired in previous learning tasks; we plan to adapt a variant of GP-based module acquisition technique for that purpose.

References

[1] Draper, B., Hanson, A., Riseman, E. (1993) "Learning blackboard-based scheduling algorithms for computer vision," International Journal of Pattern Recognition and Artificial In-telligence, vol. 7, 309–328, March.

[2] Johnson, M.P., Maes, P., Darrell, T. (1994) "Evolving visual routines," in: R.A. Brooks, P. Maes (red.) Artificial Life IV: proceedings of the fourth international workshop on the syn-thesis and simulation of living systems, Cambridge, MA: MIT Press, 373–390.

[3] Koza, J.R., Andre, D., Bennett III, F.H., Keane, M.A. (1999) Genetic Programming III: Darwinian Invention and Problem Solving. San Francisco, CA: Morgan Kaufman.

[4] Krawiec, K., Bhanu, B.: Visual Learning by Coevolutionary Feature Synthesis. IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics 35 (2005) 409–425

[5] Luke, S. (2002) "ECJ Evolutionary Computation System," http://www.cs.umd.edu/projects/plus/ec/ecj/.

[6] Maloof, M.A., Langley, P., Binford, T.O., Nevatia, R., Sage, S. (2003) "Improved rooftop detection in aerial images with machine learning," Machine Learning, vol. 53, 157–191.

[7] Marek, A., Smart, W.D., and Martin, M.C. (2003) "Learning Visual Feature Detectors for Obstacle Avoidance using Genetic Programming," In Proceedings of the IEEE Workshop on Learning in Computer Vision and Pattern Recognition, Madison, WI.

[8] Marr, D. (1982) Vision. W.H. Freeman, San Francisco, CA.

[9] Rizki, M., Zmuda, M., Tamburino, L. (2002) "Evolving pattern recognition systems," IEEE Transactions on Evolutionary Computation., vol. 6, 594–609.

[10] Segen, J. (1994) "GEST: A learning computer vision system that recognizes hand gestures," In: R.S. Michalski and G. Tecuci (ed.) Machine learning. A Multistrategy Approach. Volume IV, San Francisco, CA: Morgan Kaufmann, 621–634.

[11] Sun Microsystems, Inc., (2001) "Java Advanced Imaging API Specification", Version 1.2.

[12] Teller, A., Veloso, M.M. (1997) "PADO: A new learning architecture for object recognition," In: K. Ikeuchi and M. Veloso (ed.) Symbolic Visual Learning, Oxford Press, 77–112.

[13] Torralba, A., Murphy, K.M., Freeman, W.T. (2004) "MIT-CSAIL Computer vision annotated image library", http://web.mit.edu/torralba/www/database.html.