Learning High-Level Visual Concepts using Attributed Primitives and Genetic Programming

Krzysztof Krawiec

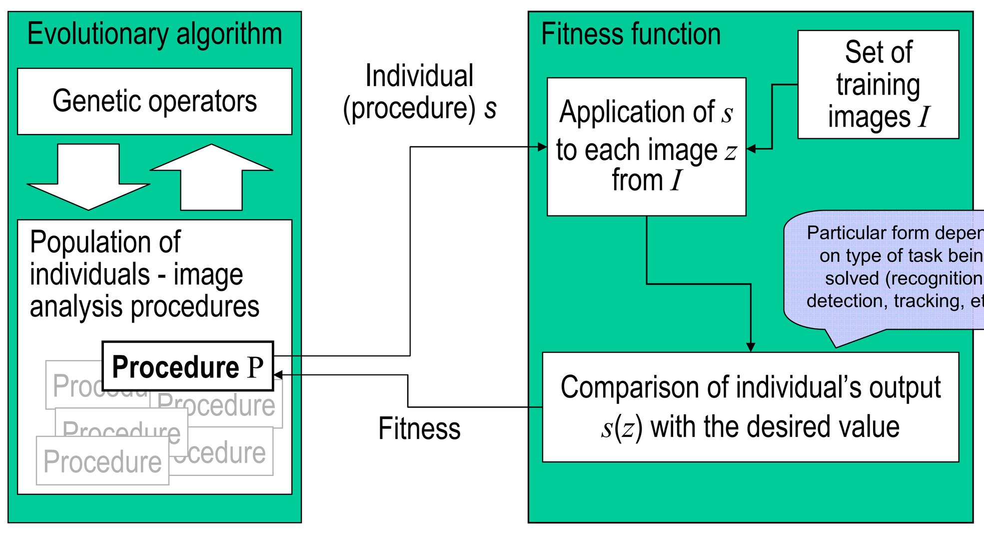
Institute of Computing Science, Poznań University of Technology, Poznań, Poland

Pages in proceedings: 515 – 519

Topic and Motivation

Research topic:

- **Evolutionary synthesis of image analysis programs**.
- An attempt to evolve an [almost] complete data flow starting from the input raster image and ending up with the final decision.



Problems identified in previous work (e.g., [Bhanu, Lin, Krawiec 2005]):

- The evolving programs operated only on raster images and scalar features extracted from them. Intermediate representations were not supported. The set of feature extraction methods was limited.
- Time-consuming raster processing becomes a bottleneck for evolutionary learning (costly fitness calculation).

Aim

To propose a framework for evolutionary visual learning which:

- Provides **high flexibility** of image processing and analysis (rich choice of intermediate representations and processing methods),
- Enables **hierarchical concept building**,
- Relies mostly on general vision knowledge, and is, as far as possible, **application-independent** and **task-independent**,
- Provides **high processing speed** (e.g., does not necessarily rely on raster processing).

Method

The proposed method:

The central idea of this work

- Genetic Programming (GP) with individuals implementing image analysis procedures.
- GP operators process **visual primitives** (precisely: sets of visual primitives, SOPs) rather than raster images.

Visual primitive = an entity that represents a **salient local feature** of the image, described by some elementary **attributes**.

In this work: each primitive corresponds to image location with prominent gradient. A primitive is described by 4 attributes: coordinates x and y, gradient orientation, and intensity.

GP types: Scalars, sets of primitives (SOP), attribute labels (A).

GP operators: Terminals:

- The primitive representation VP(z) of the input raster image z,
- A 'name' (tag) of a single primitive attribute: x coordinate, y coordinate, orientation, or intensity,
- An ephemeral random constant (ERC).

GP operators: Non-terminals:

- Scalar operators (arithmetic, basic scalar functions).
- **Selectors**: Filter the primitives received from child node(s) according to some criterion/condition. Return value of type SOP. There two types of selectors:
  - Non-parametric: Processing does not depend on attributes. Examples: Set union, set intersection, set difference, etc.
  - Parametric: Processing does depend on primitive attributes. Example: The operator *LessThan* applied to child nodes (*P*, 'orientation', 2) filters out all primitives from *P* for which the value of the attribute 'orientation' is less than 2.
- **Aggregators**: Combine the values of a chosen primitive attribute for primitives received from child node(s). Return scalar values. There two types of aggregators:
  - Non-parametric: Mostly statistical descriptors. Example: The operator *Mean* applied to child nodes (*P*, 'x_coordinate') computes and returns the mean value of coordinate *x* of all primitives in *P*.
  - Parametric: Example: The operator *CentralMoment* applied to child nodes (*P*, 'y_coordinate', 3) computes and returns the central moment of order 3 of coordinate *y* of all primitives in *P*.

Operator summary:

Type	*Non-parametric*	*Parametric*
Scalar	$+, -, \times, /, sin, cos, exp, log$	
Selector	$\cup, \cap, \setminus$, *SymmetricDifference*	*Equals, LessThan, GreaterThan*
Aggregator	*Sum, Mean, Product, Median*	*Moment, CentralMoment, Percentile*

Experiment

The task: To locate computer screen in indoor scene.

Data source and preprocessing:

- The MIT-CSAIL Database of Objects and Scenes http://web.mit.edu/torralba/www/database.html
- Database folder: aug1_static_atb_office_bldg400
- Actual screen locations given as image annotation.
- Number of source images containing exactly one computer screen: 38
- Cropped copies created to avoid bias towards the center of the scene.
- Final number of images in the training set I: $38\times$(1(*original*) + 4(*cropped versions*)) = 190

Selected source images from the MIT-CSAIL database:

The cropped versions of images:

Primitive Extraction

Carried out **once**, prior to the evolutionary run.

Input: raster image z. **Output**: z's visual primitive representation VP(z)

1. Filter the input image z using bank of four Gabor filters with orientations: 0, 45, 90, and 135 degrees.
2. Create set C of primitive candidates containing 5% of brightest pixels in filter responses.
3. Find the brightest candidate in C and, based on it, create a primitive p=(x,y,$intensity$,$orientation$) and add it to VP(z). Remove from C all candidates located closer than d_{min} from p ($d_{\min}$= 20 pixels).
4. If $C \neq \emptyset$, go to 3. Otherwise, stop and return VP(z).

Resulting # of primitives per image: 103 to 145 (average: 122.2).

Settings

GP settings:

Parameter	*Setting*
Algorithm type	Generational
Number of generations	50
Mutation probability	0.1
Crossover probability	0.9
Population size	1000
Maximum tree depth	7
Retries	3

Other parameters set to ECJ's defaults [Luke 2002]

Objective function (*minimized*): *Penalizes* the solution s for selecting primitives more distant that d_{min} from the screen to be located, and *rewards* s for selecting primitives less distant than d_{min}

$$f(s) = \exp\left(\frac{1}{|I|}\sum_{z\in I}\sum_{p\in s(z)} d(p,t(z))\right)$$

where:

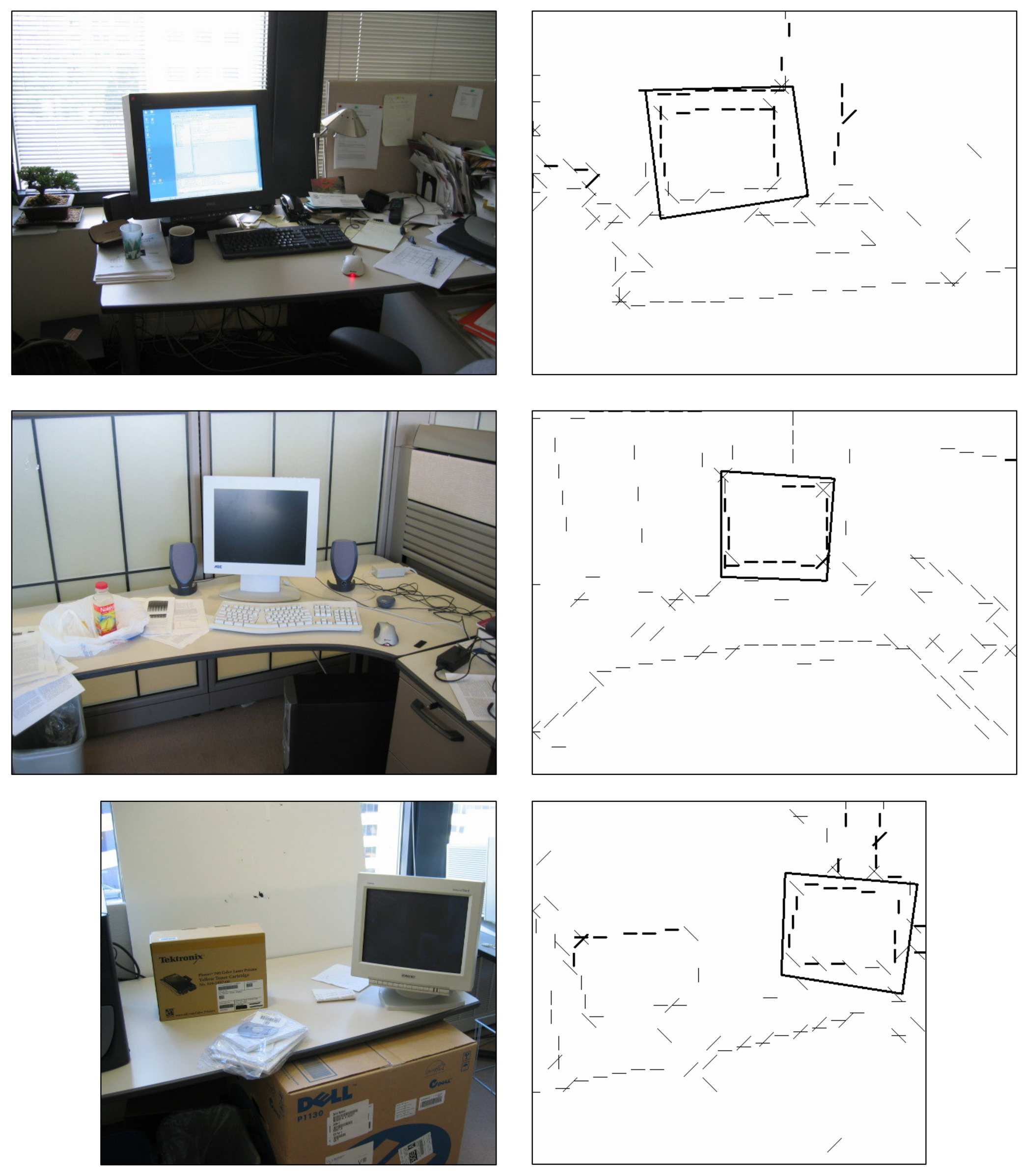
$$d(p,t) = \begin{cases} d_e(p,t), & \text{if } d_e(p,t) > d_{\min} \\ -d_{\min}, & \text{otherwise} \end{cases}$$

I training set
$s(z)$ individual's response to z
p primitive
z training image
d_e Euclidean distance
t target (screen) location

Results

Processing performed by the best evolved individual for selected images: (raw fitness 1.24×10^{-6})

Left: Input image z — *Right: Output s(z) produced by the best individual (in bold)*



Legend *for the images in right-hand column:*

- *Rectangular shape – actual screen location* (*based on database annotation*)
- *Short sections* (*all*) – VP(z), i.e., *primitives as obtained from the input image*
  - *Section location corresponds to primitive location.*
  - *Section slant depicts primitive orientation.*
- ***Bold*** *short sections* – s(z), i.e. *primitives selected by the individual*

The code of one of the best individuals (simplified for presentation):

```
(Selector >
  (Selector <
    (Selector >
      (Selector <
        (Selector >
          VP(z)
          Coordinate_Y
          (exp 0.96))
        Orientation
        (exp 0.97))
      Coordinate_X
      (cos
        (Median
          (Selector >
            VP(z)
            Intensity
            0.19)
          Orientation)))
(continued in right column)
```

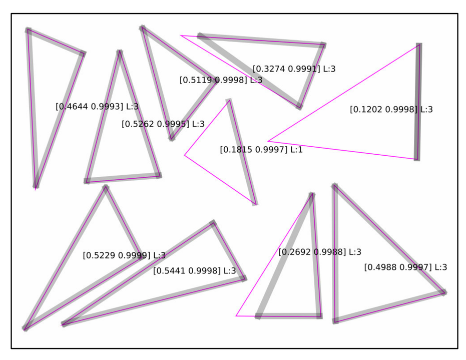
```
  Coordinate_Y
  (+
    (-
      (% (% 0.97 -0.56) 0.87)
        -0.84)
    (Mean
      (Selector >
        VP(z)
          Intensity
        (exp 0.96))
      Coordinate_Y)))
Intensity
 (exp
  (-
    (-
      (cos
        (Median VP(z) Orientation))
      -0.83)
    -0.83)))
```

Conclusion

- The approach enables smooth transition from low-level, local concepts and features to the more general ones.
- Reasonable location accuracy obtained at low design costs.
- Low computational complexity of processing: Average processing time ~0.5 ms per image and individual (implementation in Java).
- Application-independence.
- Extra knowledge, if needed, may be easily incorporated by extending the set of GP operators.

Future and related work:

- Extension to other types of primitives (e.g., regions)
- Application to other types of tasks (e.g., recognition)
- **Extended and unsupervised** version of the method to be presented at CEC'2006: Individuals compete to elaborate an exact, general, and simple description of the input image.
- Long-term goal: incremental acquisition of visual concepts of gradually increasing complexity.



Contact

Krzysztof Krawiec
Institute of Computing Science, Poznań University of Technology
Piotrowo 2, 60965 Poznań, Poland
krawiec@cs.put.poznan.pl, http://idss.cs.put.poznan.pl/~krawiec

This research has been supported by KBN research grant 3 T11C 050 26