Evolution of Visual Feature Detectors

Tony Belpaeme
Artificial Intelligence Lab,
Vrije Universiteit Brussel,
Pleinlaan 2, 1050 Brussels
tony@arti.vub.ac.be http://arti.vub.ac.be/~tony/
Tel: +32 2 629 3700, Fax: +32 2 629 3729

ABSTRACT

This paper describes how sets of visual feature detectors are evolved starting from simple primitives. The primitives, of which some are inspired on visual processing observed in mammalian visual pathways, are combined using genetic programming to form a feed-forward feature-extraction hierarchy. Input to the feature detectors consists of a series of real-world images, containing objects or faces. The results show how each set of feature detectors self-organizes into a set which is capable of returning feature vectors for discriminating the input images. We discuss the influence of different settings on the evolution of the feature detectors and explain some phenomena.

1. INTRODUCTION

This paper investigates how visual feature detectors can evolve under selectionistic pressure. It has been demonstrated that visual functionality is affected by the visual, and subsequent neural, activity [Hubel and Wiesel, 1979; Purves, 1988] of the animal. Experiments in which test subjects are deprived from visual stimulation, result in the underdevelopment of the associated visual processing. While extra stimuli effect strengthening of the visual processing. This demonstrates the plasticity of visual pathways, and shows that the way in which visual pathways organize is not entirely specified by the genome of the animal. The mammal genome, having approximately 10^5 genes, can not define the mammal nervous system, which has about 10^{14} connections. During development of the visual system, due to lack of information to prespecify the neural connections, a very important role is played by self-organization.

The results of self-organization depend on a few basic factors.

The sensory input to the processing pathways. Depending on the input, certain functionality may or may not develop. Hubel and Wiesel [Hubel and Wiesel, 1963] demonstrated this by depriving kittens from certain visual patterns, which resulted in the loss of the ability to recognize these particular patterns during adulthood. Depending on the animal species, genetic predispositions can even prevent certain visual stimuli to ever reach cortical areas, thus prohibiting any visual behavior to these particular stimuli to develop.

The task. The task uses results from the visual processing, and has an equally important influence on the selection of visual functionality. If the task relies heavily on specific processing pathways, these pathways will be strengthened during development. If however a pathway is little used or unused, it will disappear and be replaced by more successful pathways.

The primitives. The initially available functionality will affect the resulting architecture. Highly specific visual behaviors have been found at birth in different cortical areas. Primitive behavior reacting to spatial and temporal regularities in the visual input have been found; even between different animal species largely the same basic functionality occurs. We assume that these primitives are readily available to construct more complex feature detectors. This however does not mean that the behaviors are hard-wired, internal pattern generation can be responsible for the development of primitive processing prior to birth [Shatz, 1996].

The learning mechanism. Finally, the learning mechanism will have an important impact on the development of the visual functionality. We use genetic programming [Koza, 1992] to evolve the feature detectors, issues on this approach will be discussed later.

Much in the same way as Darwinian processes are the driving force for the evolution of species, Darwinian-like processes are responsible for functional adaptation of the nervous system and the brain to its possessors environment and tasks. Edelman's neuronal group selection theory [Edelman, 1987] shows how different neural functionality is in constant competition during development and adulthood. Very much like a micro-ecology, in which often used pathways tend to get the upper hand, while unused pathways disappear. This selectionistic process requires two ingredients: a source of diversity and a selectionistic mechanism.

In this work, visual feature detectors are evolved using a genetic program. Each feature detector is formed by a combination of primitives. These primitives perform basic visual processing, some inspired on visual behavior observed in mammals. The input for the feature detectors consists of real-world images, showing objects or faces. The task, needed to evaluate the effectiveness of the feature detectors, can be any task, such as a sensori-motor task or a discrimination task. However, in our experiments no task is used, but instead a measure for the information returned by the feature detectors is used.

Section 2 describes the experimental setup, section 3 shows some results and section 4 discusses particular issues of the approach taken and section 5 concludes.

2. EXPERIMENTS

In the experiment a series of real-world images is shown to each individual in a GP, an individual consists of a set of N feature detectors. For each image s, every feature detector in the set produces a scalar output $o_j \in [0, 1]$, so the set of features detectors produces an output $o_s = \{o_1, o_2, \ldots, o_N\}$. This vector is then used by the task to act accordingly to the presented visual stimulus.

2.1 The architecture

For the experiments a strongly typed genetic program is used [Montana, 1995]. A selectionistic process requires a source of diversity: this is taken care of by the population of individuals, providing random and ample raw material to bootstrap the evolution, and by the genetic operators of the GP. Furthermore, selectionistic pressure is needed to steer the selection, this is provided by the task (see 2.2). The strongly typed GP uses three types. `scalar` is defined as $x \in [0, 1]$, `point` is an ordered pair of scalars, defining a relative image coordinate and `image` is an monochrome image of fixed size. Table 1 shows the terminals used. The ephemeral random constant is a random number of type `scalar`, which keeps its value during the lifetime of the individual.

Table 2 shows the non-terminals. There is one extra non-terminal, not shown in the table, which combines all N feature detectors: it accepts as parameters N scalars and has no return type. Six of the non-terminals represent primitive image processing functionality. Some have been chosen to facilitate the image processing, while other have been selected for their resemblance to phenomena recorded in animal visual pathways. The `spatialFilter` and `spatialResponse` each edge-detect the image with a certain sensitivity, the first returning the edge-detected image and the second returning a measure for the number of edges detected. Both are inspired on the fact that mammals perform spatial filtering at retinal and cortical level [De Valois and De Valois, 1990]. The `orientationResponse` primitive is based on the orientation specifity of the visual cortex. No color sensitive primitives are used, since we have a small set of training images and spectral sensitive primitives would probably bias the feature detectors too much.

2.2 The task

The task provides feedback on the quality of the generated feature detectors. The task can be anything, such as a sensori-motor task or a discrimination task. However, in the experiments described here no task is used for the simple reason that evaluating a task is rather time consuming. Instead a measure is used to calculate the information content returned by each feature detector. This is not only faster, but it also does not commit the evolution of the feature detectors to one specific task, thus leaving the

Terminal	Type
leftTop	`point`
rightBottom	`point`
zero	`scalar`
one	`scalar`
ephemeral random constant	`scalar`

Table 1: Terminals for the strongly typed GP.

possibility to later connect any task to the feature detectors.

The information content can be seen as the spread of data in the N-dimensional space (with N being the number of feature detectors). The fitness of each set of feature detectors is measured as the entropy of the output vector o_s for each image s in the discretized N-dimensional space. Each dimension is split into M equal divisions, giving a total of $P = N^M$ partitions. Figure 1 shows an example. If I is the number of images presented to each set of feature detectors, the entropy is calculated as in eq. 2.1, with x_i being the number of output vectors in partition i.

$$\epsilon = \sum_{i=1}^{M^N} -\frac{x_i}{I} \ln \frac{x_i}{I} \qquad \text{for } x_i > 0 \tag{2.1}$$

The maximal possibly entropy is reached when every partition in N-dimensional space contains one or no output vectors (eq. 2.2).

$$\epsilon_{max} = I(-\frac{1}{I} \ln \frac{1}{I}) = \ln I \tag{2.2}$$

The fitness of a set of feature detectors is defined as

$$f = \frac{1}{1 + (\epsilon_{max} - \epsilon)} \tag{2.3}$$

If the outputs are well spread, meaning that the feature detectors return useful information, the fitness will be high. On the other hand, clustered outputs will result in a low fitness. This way of calculating the fitness ensures that all feature detectors are different enough to provide ample information. If the fitness would be evaluated for each single feature detector and then be summed to obtain the fitness for the set of feature detectors, this would be exploited by the GP to evolve good -but often identical- feature detectors, which would be produce all similar feature vectors. (for an example

Function	Functionality	Input Types	Output type
averageIntensity	Computes average intensity of an image region, delimited with left-top and bottom-right coordinate	`image, point, point`	`scalar`
thresholdImage	Filters out all pixels not in range $[a, b]$	`image, scalar, scalar`	`image`
spatialFilter	Filters the image using a Laplacian of Gaussian filter with size σ, as defined by [Marr, 1982]	`image, scalar`	`image`
spatialResponse	Gives the response of a Laplacian of Gaussian filter with size σ	`image, scalar`	`scalar`
orientationResponse	Gives the response to edges in a particular direction α, $\alpha = 1$ is $\pi/2$	`image, scalar`	`scalar`
combineImage	Makes an image by taking the average of two input images	`image, image`	`image`
cons	Combines two scalars in an ordered pair	`scalar, scalar`	`point`
divide	Returns protected division, the second parameter acts as an amplifier of the first parameter.	`scalar, scalar`	`scalar`
multiply	Returns multiplication, acts an attenuation of both parameters.	`scalar, scalar`	`scalar`

Table 2: Non-terminals for the strongly typed GP. The last three non-terminals do not perform image processing.

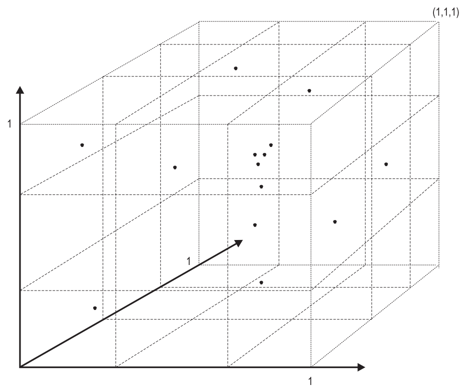

Figure 1: Illustrating a partitioned output space, there are $N = 3$ feature detectors, and each dimension is split into $M = 3$ divisions, giving a total of $P = N^M = 27$ partitions. The dots represent outputs; ideally every partition should contain one output or no outputs.

see [de Jong and Steels, 1999]).
Note that if there are more partitions (M being bigger), it will be easier to satisfy the fitness criterium. There should be at least I partitions, with I being the number of images. This means that every dimension should be divided into ceil($\log_N I$) divisions.

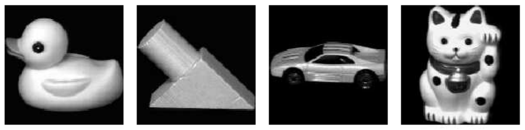

Figure 2: Sample of images of the first series.

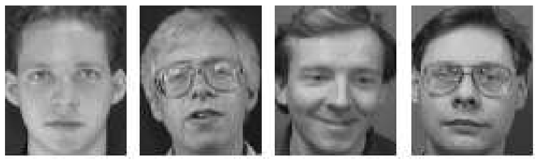

Figure 3: Sample of images of the second series.

3. RESULTS

Two series of monochromatic images were used as test images. Series 1 contains 20 rigid objects[1], while series 2 contains 39 monochromatic frontal face images. All images

[1]The images for series 1 were taken from the the Columbia Object Image Library (COIL-20) at `http://www.cs.columbia.edu/CAVE`, while the series 2 was taken from the Olivetti face database at `http://www.cam-orl.co.uk/facedatabase.html`.

are shown in only one view. Images of series 1 have size 50 by 50 pixels, images of series 2 are 65 by 80 pixels.

The GP has a crossover rate of 0.8, a mutation rate of 0.1 and a reproduction rate of 0.1. Note that the reproduction rate is actually higher due to failed crossover operations. The system runs for 500 generations, with a population of 100 individuals. Candidates for operators are chosen using fitness proportionate selection. In each run, 5 feature detectors are evolved. This gives a five-dimensional feature space, of which each dimension is split up in M equal divisions.

3.1 Discriminating objects

Figure 4 shows the fitness for 500 generations. The system evolved feature detectors, which for every input image produce a distinct feature vector.

Table 3 shows the best individual after 500 generations. Two of five feature detectors do not participate in discriminating the images. One relies on the orientation of intensity changes in the image, while two rely on the average intensity of a region in the image. If the fitness evaluation is made more demanding (e.g. giving the feature detectors less partitions in the feature space), more feature detectors tend to be used to discriminate all inputs.

3.2 Discrimating faces

Table 4 shows the best individual for the face discrimination task. Feature detectors for discriminating objects often rely on primitives that use intensity values (e.g. `averageInt`), but in the case of face discrimination the feature detectors only use spatial information and information on the orientation of edges. Figure 5 shows a 500 generation run with 100 individuals, crossover rate is 0.8, mutation is 0.1, reproduction 0.1 and fitness proportionate selection is used. Evaluating one individual takes on average 4.5 seconds on a 350MHz AMD K6-2 PC, due to the quite computational expensive orientation operators. One run (500 generations, 100 individuals) typically takes 50 hours.

3.3 Tree growth

Parsimony pressure is used to limit the size of the feature detectors, when a feature detector has a depth higher than δ, a penalty term is added to the fitness (eq. 3.1).

`(div 0.02266 (orientResponse x 0.43329))`
`1.0`
`(averageInt x (cons 0.07613 0.97146) rightBottom)`
`(averageInt x (cons 0.0 0.65232) (cons 1.0 0.48591))`
`0.23671`

Table 3: Best individual at generation 500, for run described in fig. 4, it has maximal fitness.

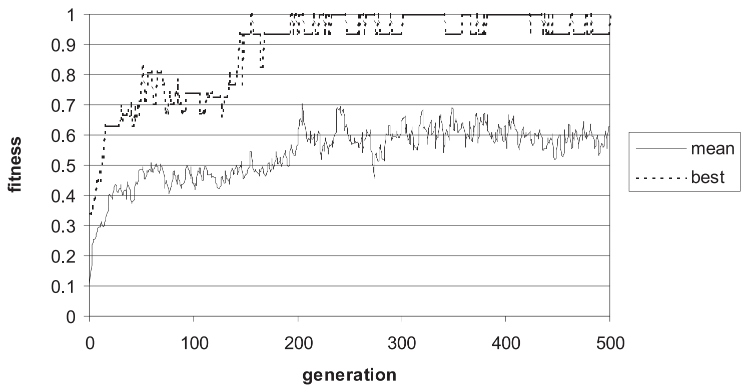

Figure 4: Results of run for series 1 (images of objects) with individuals consisting of a set of 5 feature detectors. Each dimension has $M = 5$ divisions, thus creating $P = M^N = 5^5$ partitions. The graph shows the mean fitness and the fitness of the best-of-generation individual.

$$f_{\text{raw parsimony}} = f_{\text{raw}} + \frac{\text{depth}(ind)}{\delta} \tag{3.1}$$

This deters the GP from code growth. However, it soon seems that no parsimony pressure is needed: the feature detectors are prevented from growing too large by another phenomena. This can be explained by the *attenuating effect* of large trees. A feature detector consists of serially connected primitives, each primitive receiving the output of another primitive. Primitives not only extract information, they also filter information; and several connected primitives can eventually filter out too much information, which is the case in large feature detectors. A run with no fitness information, produced individuals with an average depth of 7.3 and on average 130 nodes (the par-

```
(div (OrientResponse (SpatialFilter x 0.05069) (OrientResponse x 0.79191))
     (OrientResponse (Threshold x 1.0 0.72663) 1.0)))
0.16157
(div (OrientResponse x 0.0) (OrientResponse x 1.0))
(div (OrientResponse x 0.0)
     (OrientResponse (CombineImage (Threshold x 1.0 0.72663)
                                   (Threshold x 1.0 0.0))
                          (div (OrientResponse x 0.0) 0.0)))
(div 0.07276 1.0)
```

Table 4: Example of best individual after 500 generations for face discrimation.

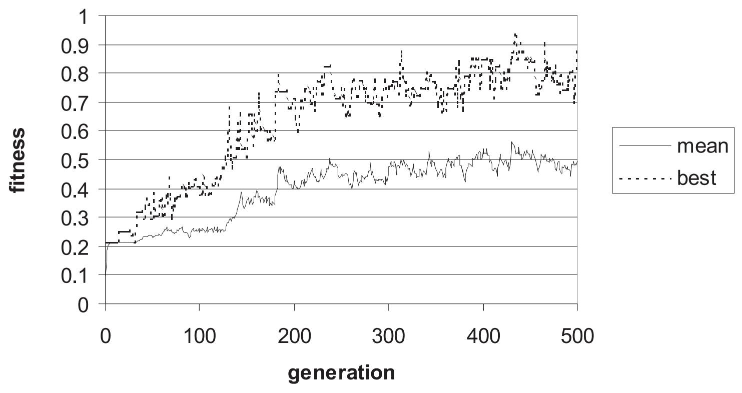

Figure 5: Results of run for series 2 (images of faces) with individuals consisting of a set of 5 feature detectors. Each dimension has $M = 5$ divisions, thus creating $P = M^N = 5^5$ partitions. The graph shows the mean fitness and the fitness of the best-of-generation individual.

simony pressure was set to $\delta = 10$). While the run shown in figure 4 has individuals with average depth 3.2 and node size 25.

4. DISCUSSION

The system uses a GP to evolve feature detectors, this however is not in accord with how visual pathways develop. Instead of exploring the search space using hundreds of individuals, the nervous system only has one individual, itself, which it can adapt to selectionistic pressure. A genetic program is not a good metaphor for the nervous system, instead a dynamical system (i.e. one individual) might be more representative for the development of parts of the nervous system, for example the visual system.
Also the number of available primitives is low and the functionality they present is limited. Though color proves to be meaningful in recognition and categorization tasks [Swain and Ballard, 1991], primitives sensitive to spectral information are deliberately not used, because color information would probably prevail over other kinds of information, and this would be exploited by the GP. In future experiments new primitives should be added, such as Gabor filters and line-end sensitive primitives. Also the number of feature detectors should be increased, instead of having a fixed number of feature detectors, the system should be able to create new feature detectors when needed and suppress feature detectors which provide little or no information.

5. CONCLUSIONS

The possibility of evolving feature detectors starting from primitive image processing functions is investigated, and the results show that it is possible to construct visual functionality under selectionistic pressure. Furthermore the nature of the input images influences the evolution of the feature detectors; images of objects produce feature detectors which rely more on intensity information, while images of faces push the feature detectors towards using spatial and orientation information. The feature detectors are not tested on their ability to generalize or categorize visual inputs.

ACKNOWLEDGMENTS
For the genetic programming system an adapted version of LilGP by Douglas Zongker and Bill Punch was used. The author is a Research Assistant of the Fund of Scientific Research - Flanders (Belgium) (F.W.O.)

REFERENCES

[de Jong and Steels, 1999] Edwin D. de Jong and Luc Steels. Generation and selection of sensory channels. In *Proceedings of the First European Workshop on Evolutionary Computation in Image Analysis and Signal Processing EvoIASP'99*. Springer-Verlag, 1999.

[De Valois and De Valois, 1990] R.L. De Valois and K.K. De Valois. *Spatial vision*. Oxford University Press, Oxford, 1990.

[Edelman, 1987] Gerald M. Edelman. *Neural Darwinism: The Theory of Neuronal Group Selection*. Basic Books, 1987.

[Hubel and Wiesel, 1963] D.N. Hubel and T.N. Wiesel. Receptive fields of cells in striate cortex of very young, visually inexperienced kittens. *Journal of Neurophysiology*, 26:994–1002, 1963.

[Hubel and Wiesel, 1979] D.N. Hubel and T.N. Wiesel. Brain mechanisms of vision. *Scientific American*, 241(3):150–162, 1979.

[Koza, 1992] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[Marr, 1982] David Marr. *Vision*. Freeman, 1982.

[Montana, 1995] David J. Montana. Strongly typed genetic programming. *Evolutionary Computation*, 3(2):199–230, 1995.

[Purves, 1988] Dale Purves. *Body and Brain: A Trophic Theory of Neural Connections*. Harvard University Press, 1988.

[Shatz, 1996] C.J. Shatz. Emergence of order in visual system development. In *Proceedings of the National Academy of Sciences, USA*, volume 93, pages 602–608, 1996.

[Swain and Ballard, 1991] M. J. Swain and D. Ballard. Colour indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.