Evolving Boundary Detectors for Natural Images via Genetic Programming

Ilan Kadar, Ohad Ben-Shahar, Moshe Sipper
Department of Computer Science, Ben-Gurion University of the Negev, Israel
{*ilankad,ben-shahar,sipper*}*@cs.bgu.ac.il*

Abstract

Boundary detection constitutes a crucial step in many computer vision tasks. We present a novel learning approach to automatically construct a boundary detector for natural images via Genetic Programming (GP). Our approach aims to use GP as a learning framework for evolving computer programs that are evaluated against human-marked boundary maps, in order to accurately detect and localize boundaries in natural images. Our GP system is unique in that it combines filter kernels that were inspired by models of processing in the early stages of the primate visual system, but makes no assumption about what constitutes a boundary, thus avoiding the need to make ad-hoc intuitive definitions. By testing the evolved boundary detectors on a benchmark set of natural images with associated human-marked boundaries, we show performance to be quantitatively competitive with existing computer-vision approaches. Moreover, we show that our evolved detector provides insights into the mechanisms underlying boundary detection in the human visual system.

1. Introduction

Boundary detection in images is a central problem in computer vision. The performance of many high-level computer vision tasks, such as segmentation and object recognition, is highly dependent upon the boundary map of an image. Figure 1 shows an image and its associated boundary map, as marked by human observers. Automatically extracting such boundary maps is a fundamental goal of low-level vision.

A boundary is a contour in the image plane that represents a change in the pixel's "ownership" from one object or surface to another. In general, there are different types of boundaries: for example, those formed between two regions with an abrupt change in the image brightness, and those formed between two regions with a change in the texture. Clearly, boundaries in natural images are marked by changes in both brightness and texture. There are some attempts in computer vision to address both brightness and texture cues using complex and computationally intensive schemes [3] [5] [15]. In contrast, humans have an outstanding ability to detect boundaries pre-attentively, and hence very fast. Correspondingly, evidence from behavioral science and neuroscience strongly suggest that this process occurs in early stages of visual processing.

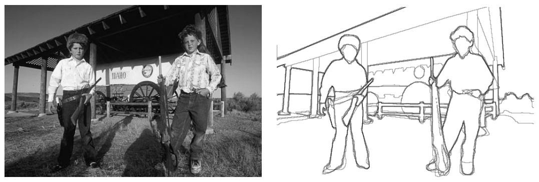

Figure 1. Example image and human-marked boundaries taken from the Berkeley dataset [4]. Boundary map shows boundaries marked by 5 observers. The pixels are darker where more observers marked a boundary.

This paper presents a novel approach that aims to use genetic programming—a form of evolutionary algorithm—as a learning framework for evolving computer programs; the latter are evaluated against human marked boundary maps in order to accurately detect and localize boundaries in grayscale natural images. The evolving programs use both linear and non-linear operators to combine multiple cues from the early stages in the visual cortex. Our results show that this approach is effective at automatically generating boundary detectors. By testing the evolutionary algorithm on a benchmark set of natural images with associated human-marked boundaries, we show performance to be quantitatively human-competitive [8]. Then, by analyzing the evolved detector, insights into the visual mechanism underlying boundary detection are developed.

2. Related Work

There is a wide variety of algorithms for boundary detection, but none has come close to human proficiency. The most common approach to local boundary detection is to look for discontinuities in image brightness. The Canny edge detector [2], for example, which is based only on local gradient and has scale parameters to tune, responds strongly inside textured regions where high-contrast edges are present, but no boundary exists. In addition, it is unable to detect a boundary between textured regions when there is only subtle change in average image brightness. The significant problems with simple brightness edge models have led researchers to develop more complex detectors that look for changes in texture , e.g., [14]. While these models work well on pure texture-texture boundaries, they have problems in the vicinity of simple brightness boundaries. Just as a brightness edge model does not detect texture boundaries, a pure texture model does not detect brightness boundaries effectively. Clearly, boundaries can be marked by joint changes in several cues, including brightness and texture. Evidence from psychophysics [13] suggests that humans make combined use of multiple cues to improve their detection and localization of boundaries. Malik et al. [5] associate a measure of texturedness with each point in the image, in order to suppress contour processing in textured regions, and vice versa. However, their solution is full of ad-hoc design decisions and hand chosen parameters. In another research Malik et al. [3] provide a more principled approach to cue combination, by framing the task as a supervised learning problem. They use learning to perform a cue combination on six carefully designed local features (texture gradient and brightness gradient at three scales each). Learning is found to improve performance over setting the weights by hands. Another supervised algorithm for boundary detection—boosted edge learning [11]— which uses a computationally intensive scheme with tens of thousands of low-, mid- and high- level features achieved the highest score up to date on the Berkeley benchmark [1].

3. Boundary Detector Performance Evaluation

The most common method for evaluation of boundary detectors for natural images is to use human-marked boundaries from a large dataset as ground-truth data. We use the *Berkeley Segmentation DataSet and Benchmark (BSDB)* [1], which contains 300 natural images, each of which was manually segmented by human subjects [4]. The dataset is divided into two independent sets of images: A training set of 200 images and a test set of 100 images. In order to ensure the integrity of the evaluation, only the images and segmentations from the training set can be accessed during the learning process. A methodology for evaluating the performance of the boundary detector with this dataset is the precision-recall framework, a standard evaluation technique in the information retrieval community [16]. Two quality measurements are considered: *Precision* (P), defined as the fraction of detections which are true positives, and *Recall* (R), given by the fraction of true boundaries that are detected. Thus, *Precision* quantifies the amount of noise in the output of the detector, while *Recall* quantifies the amount of ground-truth detected. Measuring these descriptors over a set of images for different thresholds of the detector provides a parametric *Precision-Recall curve*. The two quantities are then combined in a single quality measure, the *F-measure*, defined as their harmonic mean:

$$F(P,R) = \frac{2PR}{P+R} \tag{1}$$

Finally, the maximal *F-measure* on the curve is used as a summary statistic for the quality of the detector on the set of images.

4. Genetic Programming

Genetic Programming (GP) is an automated process that uses simulated evolution to generate computer programs that represent candidate solutions to a given problem [7]. These computer programs are represented by individuals in a GP population, where an individual is composed of LISP sub-expressions encoded using a tree representation, each sub-expression being a LISP program constructed from *functions* and *terminals*. The functions are usually arithmetic and logical operators that receive a number of arguments as input and compute a result as output. The terminals are simply zero-argument functions. Fitness is assigned to individual x according to a performance measure $f(x)$ defined by the user. GP uses standard genetic operators, including crossover and mutation, to generate new individuals in the simulated evolutionary process. When a predefined termination criterion is met the evolutionary process stops and GP returns the fittest individual found.

5. Outline of our Approach

We present a visual learning approach to automatically construct a boundary detector via *Genetic Programming*. Each individual in the GP population represents a candidate boundary detector. *Fitness* assignment is the *F-measure* computed for a set $T = \{I_i\}$ of n images taken from the training set of the Berkeley data set. The *Terminal set* is image independent, such that the terminals for image I_i , given in matrix form, are the convolution of the image I_i with filter kernels tuned to various orientations and spatial frequencies. These filter kernels are inspired by models of processing in the early stages of the primate visual system which model three types of receptive fields in the visual cortex (see Figure 2):

1. Circularly symmetric receptive fields, e.g., On-cells.

2. Odd-symmetric simple cells at various orientations.
3. Even-symmetric simple cells at various orientations.

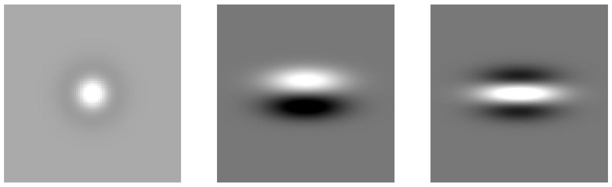

Figure 2. **(a)** Circularly symmetric on-cell. **(b)** Odd-symmetric cell with preferred orientation of $0°$. **(c)** Even-symmetric cell with preferred orientation of $0°$.

Several models have been proposed for the point-spread function of simple cells. We have chosen to use Differences of Offset of Gaussians (DOOG) kernel filters, given their good fit with the physiological measurements and their computational simplicity. As noted in the past [10], we too believe that this specific choice is not critical. Table 1 summarizes the terminals:

Table 1. Terminal set for a given image I_i

$M = I_i * K_{odd}$	Image I_i convolved with oriented odd symmetric kernel
$M = I_i * K_{even}$	Image I_i convolved with oriented even symmetric kernel
$M = I_i * K_{onCell}$	Image I_i convolved with on-cell kernel

The *Terminal set* contains oriented odd and even symmetric kernels at three scales (e.g. odd1_0, odd2_0, odd3_0) and four orientations (e.g. odd1_0, odd1_45, odd1_90, odd1_135) and on cell kernels at three scales (e.g. oncell1, oncell2, oncell3) which gives us 27 terminals for a given image I_i. The *Function set* contains both unary and binary functions. All functions, input and output, are data matrices with the same size as images in T and all are summarized in Table 2.

6. Implementation Details

The implementation of the previously described approach was programmed in Java with the genetic programming package *ECJ* [9] along with matlab-based code [1] for computing the *F-measure* of the fitness function. All experiments were performed on a PC with an Intel Pentium 4 processor and 512MB of RAM running Linux OS. We used a population size of 100 individuals, initialized with the ramped half-and-half method [7] with a depth limit of 5. The evolved individual trees were limited to a maximum depth 13. The crossover probability was set to $p_c = 0.85$ and mutation probability was set to $p_m = 0.05$. We used tournament selection [7] with a tournament size of 7. Due to the limited available computational resources, the *F-measure* was computed for a set T of 4 different images chosen randomly from the training set in each one of the 50 generations.

Table 2. Function Set

M = Add(M_1,M_2)	Matrix addition
M = Sub(M_1,M_2)	Matrix subtraction
M = Max(M_1,M_2)	Largest elements taken from M_1 or M_2
M = Min(M_1,M_2)	Smallest elements taken from M_1 or M_2
M = Pow2(M_1)	Elements are 2 raised to the power M_1
M = Sqrt(M_1)	Square root of each element of M_1
M = 2 * M_1	Multiply the elements by the scalar 2
M = 3 * M_1	Multiply the elements by the scalar 3
M = M_1 * K_{odd}	M_1 convolved with oriented odd symmetric kernel

7. Experimental Results

The output of each individual for a given image I_i is a soft boundary map, which provides the probability of a boundary at each image location. We present the fittest individual generated with our approach:

```
(sub (2* (2* (2* (sqrt (add (pow2 (add (pow2 even1_0)
(pow2 odd1_0)))(pow2 (add (pow2 odd1_90)(pow2 even1_90)
)))))))(max (max (max (max (max (odd1_90 (add (add
(pow2 oncell2)(pow2 oncell1))(pow2 oncell3)))(odd1_45
(add(add (pow2 oncell2)(pow2oncell1))(pow2 oncell3))))
(odd1_135 (add (add (pow2 oncell2)(pow2 oncell1))(pow2
oncell3)))))(odd1_0 (add (add (pow2 oncell2)(pow2
oncell1))(pow2 oncell3))))(odd1_45 (add (add (pow2
oncell2) (pow2 oncell1)) (pow2 oncell3)))))
```

Studying this individual reveals the interesting evolved strategy. The two major parts of this computer program are the *Sub* arguments. The first argument has a structure similar to the oriented energy operator, also known as the "quadrature energy" [6]. This operator was suggested by [12] as a well-suited model to detect real image edges that are not step functions but more typically a combination of steps, peak, and roof profiles. The second argument seems highly responsive inside textured regions, hence lowering the probability for a boundary.

The presented individual was tested on the Berkeley test set of 100 images [4], and the overall performance was computed using the Berkeley benchmark algorithm [1], which computes a score based on the *F-measure*. Table 3 shows the obtained score of our approach compared with other existing approaches (Note that like us, most of these approaches use training). One of the obtained soft boundary map is shown in Figure 3.

While still a bit short of the best reported algorithm, we believe our results are highly promising. Indeed, so far we have only lightly touched upon the vast repertoire of visual system operators available, using but the most basic and elementary ones. In the future, we will include many more biologically plausible operators and incorporate additional insights already available from the primate visual system.

Table 3. Performance Summary Table [1]

Method	Performance
Boosted edge learning	0.64
Brightness and texture gradient	0.63
Learning of the brightness distribution (brightness Gradient)	0.60
Our Approach	0.59
Texture gradient	0.58
Multiscale gradient magnitude	0.58
Second moment matrix	0.57
Gradient magnitude	0.56
Segmentation induced by scale invariance	0.48

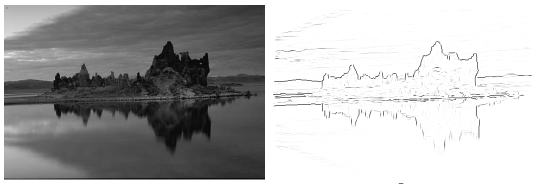

Figure 3. **(a)** Sample test image. **(b)** Extracted boundary map, using our evolved detector, with $F = 0.77$, which is the highest score up to date for this image.

8. Conclusion and Future Work

We have presented a learning framework, based on genetic programming, for evolving boundary detectors. Our GP implementation uses filter kernels, which were inspired by models of processing in the early stages of the primate visual system, but makes no assumptions about what constitutes a boundary, thus avoiding the need to make ad-hoc intuitive definitions. We note that although the terminal and function sets are inspired by our present understanding of the primary visual cortex and the visual mechanism underlying boundary detection, we do not claim that both sets are necessary nor sufficient and further analysis of the best evolved individuals will try to determine optimal sets for the problem. Experiments showed that the proposed approach generates boundary detectors that automatically extract boundaries with no parameter tuning and with a performance level competitive with existing computer vision approaches on a very challenging benchmark. By studying the top individual, we discovered that the evolutionary process has generated a computer program composed of oriented energy filters, and an operator that responds strongly inside textured regions.

Obviously, analyzing the top individuals can provide additional insights which can improve the selection of function and terminal sets. Similarly, adding mid- and high-level cues that have been shown [11] to improve overall performance on the Berkeley benchmark [1] could also contribute to the evolutionary process. These, and the informed use of additional computational resources to improve performance, all constitute our short-term future work.

9. Acknowledgments

We thank the generous support of Frankel Fund and Paul Ivanier Robotics Center at Ben-Gurion University.

References

[1] The berkeley segmentation dataset and benchmark http://www.cs.berkeley.edu/projects/vision/grouping/segbench/.

[2] J. F. Canny. A computational approach to edge detection. *PAMI*, 1986.

[3] C. F. D. Martin and J. Malik. Learning to detect natural image boundaries using local brightness, color and texture cues. *PAMI04*, 2004.

[4] D. T. D. Matin, C. Fowlkes and J. Malik. A database of human segmented natural images and its applications to evaluating segmentations algorithms and measuring ecological statistics. In *Proc. of international Conference on Computer Vision*, 2001.

[5] T. L. J. Malik, S. Belongie and J. Shi. Contour and texture analysis for image segmentation. *Intl J. Computer Vision, vol. 43, no. 1,pp. 7-27*, 2001.

[6] H. Knutsson and G. Granlund. Texture analysis using two-dimensional quadrature filters. In *Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, pages 206–213, 1983.

[7] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, USA, 1992.

[8] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence.* Kluwer Academic Publishers, 2003.

[9] S. Luke. *ECJ: A Java-based Evolutionary Computation and Genetic Programming Research System*, 2000. http://www.cs.umd.edu/projects/plus/ec/ecj/.

[10] J. Malik and P. Perona. *Preattentive texture discrimination with early vision mechanism.* J. Optical Society of America 923-32, 1990.

[11] Z. T. P. Dollar and S. Belongie. Supervised learning of edges and object boundaries. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[12] P. Perona and J. Malik. Detecting and localizing edges composed of steps, peaks and roofs. In *Proc. 3rd Int. Conf. Computer Vision*, pages 52–57, Osaka , Japan, 1990.

[13] J. Rivest and P. Cavanagh. Localizing contours defined by more than one attribute. *Vision Research, vol. 36, no. 1, pp.53-66*, 1996.

[14] J. B. S. Will, L. Hermes and J. Puzicha. On learning texture edge detectors. *Proc. Intl Conf, Image Processing, pp.887-880*, 2000.

[15] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and machine Intelligence pp. 888-905*, 2000.

[16] C. van Rijsbergen. *Information Retrieval, 2nd ed.* Dept. of Comp. Sci., UNIV. of Glasgow, 1979.