EDCONTOURS: HIGH-SPEED PARAMETER-FREE CONTOUR DETECTON USING EDPF

Cuneyt Akinlar   *Cihan Topal*

Anadolu University, Computer Engineering Department, Eskisehir, TURKEY
Phone: +90-222-321 3550, Fax: +90-222-323 9051
{cakinlar, cihant}@anadolu.edu.tr

ABSTRACT

We present a high-speed contour detector, which we name *EDContours,* that works by running our real-time parameter-free edge segment detector, Edge Drawing Parameter Free (EDPF), at different scale-space representations of an image. Combining the edge segments detected by EDPF at different scales, EDContours generates a soft contour map for a given image. EDContours works on gray-scale images, is parameter-free, runs very fast, and results in an F-measure score of 0.62 on the Berkeley Segmentation Dataset (BSDS300).

***Index Terms*—** Contour detection, Scale-Space Analysis, Edge Drawing Parameter Free (EDPF)

1. INTRODUCTION

Contour detection is an important problem and is usually used as a preprocessing step for image segmentation [1-8]. Among many contour detection and image segmentation frameworks, Berkeley Segmentation Dataset and Benchmark (BSDS300) [3, 4] have found wide-spread use for evaluating new contour detection and segmentation algorithms.

Most recent and successful contour detection algorithms have made use of the compass-operator proposed in [5]. This operator works by comparing the gradient distributions of the pixels inside the two halves of a circle of a certain radius centered at a pixel. The contours detected by a combination of brightness, color and texture gradients using this operator have produced good results [1, 2, 6], but at the expense of a high computational cost.

In this paper, we present a high-speed contour detector based on our real-time, parameter-free edge segment detector, Edge Drawing Parameter Free (EDPF) [9-12]. EDPF makes use of the edge segment validation methodology due to the Helmholtz principle presented by Desolneux, Moisan and Morel [13-15], and detects perceptually "*meaningful*" edge segments in a given image. EDPF works by running our real-time edge segment detector, Edge Drawing (ED) algorithm, with all of ED's parameters at their extremes, and validating the detected edge segments by the Helmholtz principle [13-14]. Thus, EDPF is a completely parameter-free edge segment detector.

To convert EDPF into a contour detector, which we name *EDContours*, we make use of the scale space theory [16-18]. The idea is to use EDPF to detect meaningful edge segments of an image at different scale-space representations of an image, and combine the detected edge segments into a soft contour map. We show through experimentations on BSDS300 benchmark that EDContours is a high-speed contour detector that produces good results. Specifically, EDContours takes an average of 100 ms per image in a PC having a Intel 2.20GHz 2670QM CPU on the 100 test images in BSDS300, and results in an F-measure score of 0.62 for gray-scale benchmark.

2. EDGE DRAWING PARAMETER FREE (EDPF)

Edge Drawing (ED) [9, 10] is our recently-proposed, real-time edge/edge segment detector. Unlike traditional edge detectors, e.g., Canny, which work by identifying a set of potential edge pixels in an image and eliminating non-edge pixels through operations such as non-maximal suppression, hysteresis thresholding, erosion etc., ED follows a proactive approach and works by first identifying a set of gradient extrema points in the image, called the anchors, and then links these anchors using a smart routing procedure; that is, ED literally draws edges in an image.

ED has many parameters that must be set by the user, which requires the tuning of ED's parameters for different types of images. Ideally, one would want to have a real-time edge/edge segment detector which runs with a fixed set of internal parameters for all types of images and requires no parameter tuning. To achieve this goal, we have recently incorporated ED with the "*a contrario*" edge validation mechanism due to the Helmholtz principle [13, 14], and obtained a real-time parameter-free edge segment detector, which we name Edge Drawing Parameter Free (EDPF) [11, 12]. EDPF works by running ED with all ED's parameters at their extremes, which detects all possible edge segments in a given image with many false detections. We then validate the extracted edge segments by the Helmholtz principle, which eliminates false detections leaving only perceptually meaningful edge segments.

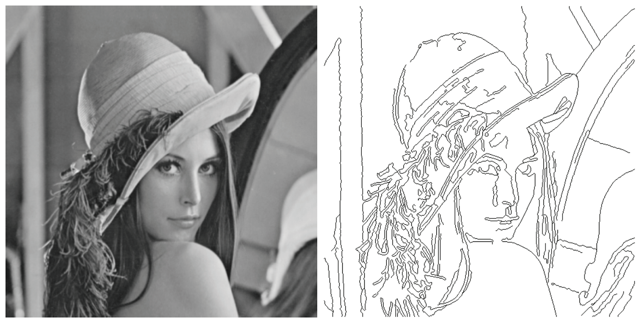

Fig 1. Lena and its edge segments detected by EDPF. The image was first smoothed by a Gaussian kernel with σ = 1.0.

Fig. 1 shows the famous Lena image and the edge segments detected by EDPF. The image was first smoothed by a 5x5 Gaussian kernel with σ = 1.0 (default for EDPF) before being fed into EDPF. Clearly, EDPF detects all perceptually meaningful edge segments in the image with very few false positives. We note that the current version of EDPF only processes gray-scale images and computes the gradient map of an image using any one of the well-known gradient operators, Prewitt, Sobel or Scharr. Our experiments have shown that the choice of the gradient operator does not change the overall quality of the results produced by EDPF, so we use the Prewitt operator by default [11, 12].

3. SCALE-SPACE THEORY

Scale-space theory is a framework for handling the structures in an image at different scales by representing an image as a one-parameter family of smoothed images parametrized by the size of the smoothing kernel used for suppressing fine-scale structures [16-18]. The main type of scale-space is the linear *Gaussian* scale-space, and is formalized as follows: Given an image I[x,y], its *Gaussian* scale-space representation is a family of smoothed images L(I[x,y],t) = I[x,y]*g(x,y;t) obtained by the convolution of the image I[x,y] with the Gaussian kernel g(x,y;t) = $\frac{1}{2\pi t}e^{-(x^2+y^2)/2t}$, where "t" is the scale parameter and has the interpretation that at scale-space level "t", image structures of spatial size smaller than about $\sqrt{t}$ are smoothed away, and are therefore not perceptible [16-18]. To rephrase this, at coarse scales, only the general outlines of the big structures inside the image would be perceptible; while at fine scales, all details would be perceptible.

Fig. 2 shows the scale-space representations of a sample image from the BSDS300 benchmark and the corresponding edge segments detected by EDPF at each representation. As expected, at fine scale representations all details of structures inside the image get detected by EDPF; however, at coarse scales many fine details are lost with only the general outlines of the large structures getting detected. So, depending on your application, one can feed different scale-space representations of an image to EDPF and obtain the amount of detail desired in the final result.

3. EDCONTOURS

To convert EDPF into a contour detector, our idea is to analyze an image at different scale-space representations, detect edge segments by EDPF at each scale and combine the detected edge segments into a soft contour map. We observe from the edge segments produced by EDPF in Fig. 2 is that at coarse scale representations, fine details of the image get lost; therefore, EDPF detects only the general outlines of the large structures in the image. Boundaries detected at coarse scales must therefore have a high probability of being a contour than those detected at finer scales. This is because the boundaries detected at coarse scales are also likely to be detected at finer scales. This can easily be observed in the edge segments shown in Fig. 2.

With this observation, EDContours follows a very simple idea for contour detection: Given an image, generate representations of an image at different scales; feed each representation to EDPF to detect meaningful edge segments, and simply combine edge segments together to compute the final soft contour map. Here is the sketch of the algorithm:

```
EDContours(Image I[x,y]){
I.      ContourMap[x,y] = 0;  // initialize the contour map to 0
II.     for (sigma=1.0; sigma<=4.0; sigma+=0.25){
II.I.       t = sigma*sigma;
II.II.      SSI[x,y] = L(I[x,y],t);   // SS rep. of image I[x,y]@t
II.III.     EdgeSegments = EDPF(SSI[x,y]);
II.IV.      ContourMap[x,y] += EdgeSegments[x,y];
        } //end-for
III. Scale ContourMap[x,y] values to [0-255]
IV. return ContourMap[x,y]
} // end-EDContours
```

This algorithm simply calls EDPF for several scale-space representations of the image and adds the results on top of each other. We observe during experimentations that below t=1.0, no extra details are detected; above t=16.0, the image becomes too coarse, so too much detail gets lost. This is why we use the scale-space representations of the image between t=1 and t=16 for contour detection. The value of the increment amount (0.25) is an experimental parameter. Making this smaller simply increases the running time without improving the quality of the results. Making this bigger, e.g., 0.50, decreases the overall quality of the resulting contour map. Thus, we have settled on a value of 0.25. So, the loop in EDContours iterates for a total of 21 times; that is, we call EDPF 21 times to compute the soft contour map.

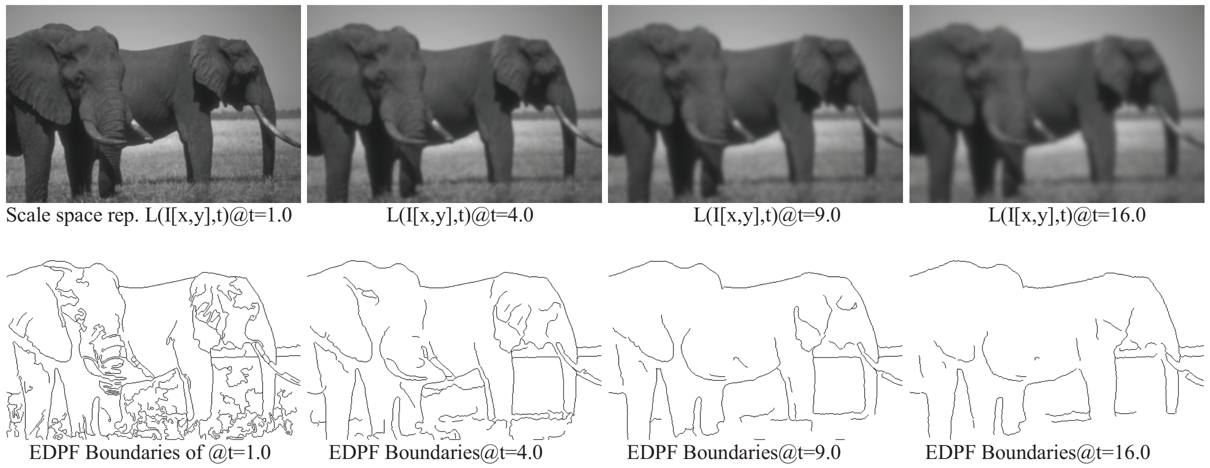

Fig 2. Boundaries detected by EDPF at different scale-space representations of an image from the Berkeley Segmentation Dataset (BSDS300). At small scales, all details present in the image are detected; while at larger scales, finer details get eliminated and only the general outline of the large structures are detected. EDPF takes only 7ms to compute the boundaries for a single image of size 481x321.

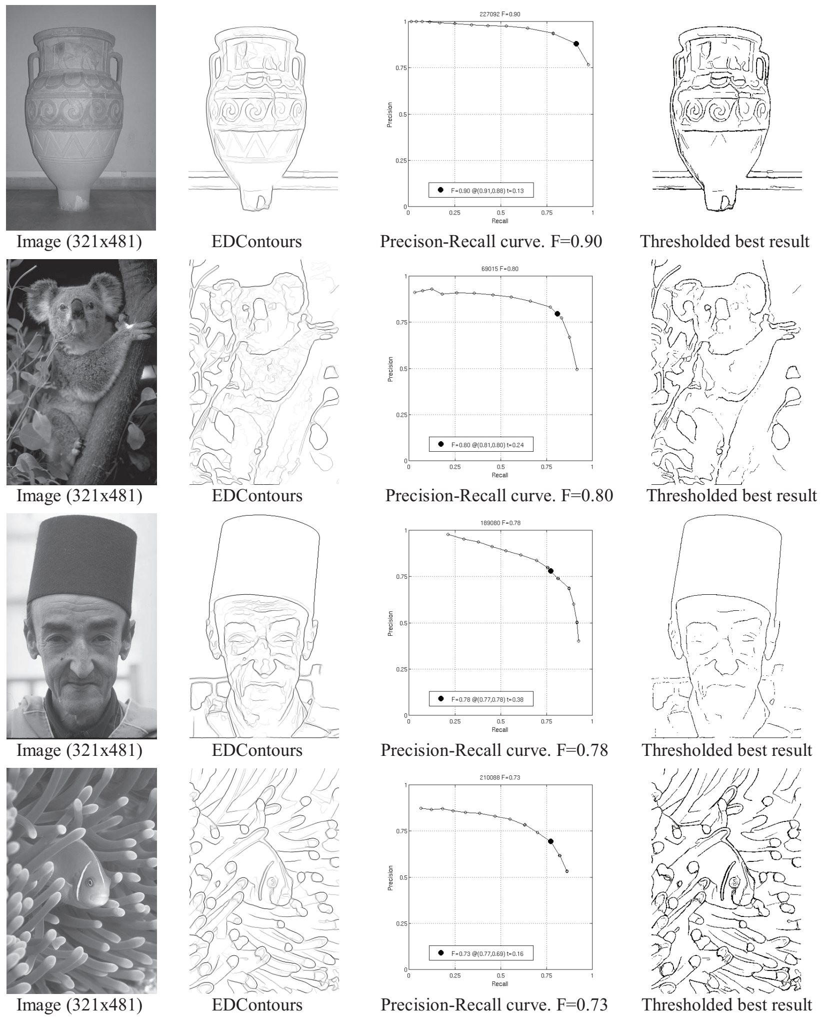

Fig 3. 1st column: Original images from the BSDS300 dataset; 2nd column: Soft contours generated by EDContours; 3rd column: Precision-Recall curves; 4th column: Thresholded EDContours, which correspond to the points of maximal F-measure on the Precision-Recall curves shown on the 3rd column. The reader can get more results online at EDContours's Web site [19].

3. EXPERIMENTAL RESULTS

To test the performance of EDContours on BSDS300, we downloaded the benchmarking dataset and code from BSDS300 Web site [4], and ran EDContours on the 100 test images. Fig. 3 shows the contour maps produced by EDContours on several sample images from the dataset. Clearly, the contour maps are visually good for the given sample images, and produce good F-measure scores as shown by the Prevision-Recall curves. We note that EDContours takes an average of about 100ms per image in a PC having a Intel 2.20GHz 2670QM CPU on the 100 test images in BSDS300 dataset.

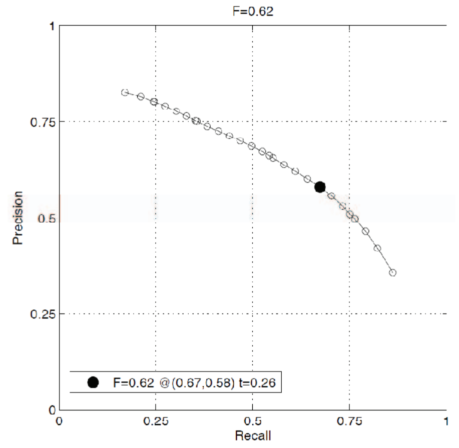

Fig 4. Overall Precision-Recall curve for EDContours.

Fig. 4 shows the overall Precision-Recall curve for EDContours. As seen, EDContours's maximal F-measure score is 0.62 on the 100 test images. Although this score is low compared to the F-measure score of "global Probability of boundary (gPb)" [1, 2] (F=0.68), which uses the compass operator [6] for gradient computation, EDContours is very fast compared to gPb and has better F-measure score compared to Roberts (F=0.47), Prewitt (F=0.48), Sobel (F=0.48), Canny (F=0.58) [2], APD (F=0.61) [7] and many other detectors using traditional gradient operators such as Prewitt, Sobel, Scharr etc. Considering that EDContours uses the Prewitt operator for gradient computation, its performance is remarkable. Our future work would be to extend EDContours with other gradient operators and improve its F-measure score while preserving its speed.

4. CONCLUSIONS

We presented a high-speed, parameter-free contour detector named *EDContours* that results in an F-measure score of 0.62 on the 100 test images in BSDS300 benchmark. EDContours works by combining the edge segments detected by our real-time, parameter-free edge segment detector, Edge Drawing Parameter Free (EDPF), on different scale-space representations of a given image. The current version of EDContours makes use the generic Prewitt gradient operator working on the brightness channel, which is known not to give good response at all perceptually meaningful boundaries of an image. As our future work, we plan to integrate other gradient operators, i.e., color and texture channel gradient operators, into EDPF to get better results on the BSDS300 benchmark. Our goal is to catch and exceed (if possible) the results obtained by state of the art contour detectors while preserving the speed of EDContours. Interested reader may obtain contour maps for other images online at EDContours's Web site [19].

5. REFERENCES

[1] D.R. Martin, C.C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color and texture cues," TPAMI, vol. 26, no. 1, 2004.

[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," TPAMI, 2010.

[3] D. Martin, C. Fowlkes, D. Tal and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in Proc. ICCV, 2001.

[4] BSDS300 Web Site. http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

[5] M.A. Ruzon and C. Tomasi, "Edge, junction, and corner detection using color distributions," TPAMI, 2001.

[6] X. Gong, and J. Liu, "A Daisy-like Compass Operator," in Proc. ICIP 2011.

[7] G. Papari, and N. Petkov, "Adaptive Pseudo Dilation for Gestalt Edge Grouping and Contour Detection," IEEE Trans. On Image Processing, vol.17, no.10, pp. 1950-1962, 2008.

[8] G. Papari, and N. Petkov, "Edge and line oriented contour detection: State of the art," Elsevier Image and Computer Vision, vol. 29, pp. 79-103, 2011.

[9] C. Topal, C. Akinlar, and Y. Genc, "Edge Drawing: A Heuristic Approach to Robust Real-Time Edge Detection," in Proc. ICPR, 2010.

[10] Edge Drawing Web Site. http://ceng.anadolu.edu.tr/CV/EdgeDrawing

[11] C. Akinlar, and C. Topal, "EDPF: A Parameter-Free Edge Segment Detector with a False Detection Control," International Journal of Pattern Recognition and Artificial Intelligence, 2012 (in press).

[12] EDPF Web Site. http://ceng.anadolu.edu.tr/CV/EDPF

[13] A. Desolneux, L. Moisan, and J.M. Morel, From Gestalt Theory to Image Analysis: A Prob. Approach, Springer, 2008.

[14] A. Desolneux, L. Moisan, and J.M. Morel, "Edge Detection by Helmholtz Principle," Journal of Mathematical Imaging and Vision, vol.14, no.3, pp. 271-284, May 2001.

[15] E. Meinhardt-Llopis, "Edge Detection by Selection of Pieces of Level Lines," Proceedings of International Conference on Image Processing (ICIP), pp. 613-616, October 2008.

[16] T. Lindeberg, "Scale-space theory: A basic tool for analyzing structures at different scales," Journal of Applied Statistics, vol. 21, no. 1, pp. 224-270, 1994.

[17] T. Lindeberg, Scale-Space Theory in Computer Vision, Kluwer Academic Publishers, 1994.

[18] Scale space, http://en.wikipedia.org/Scale_space

[19] EDContours Web Site. http://ceng.anadolu.edu.tr/CV/EDContours