EDLINES: REAL-TIME LINE SEGMENT DETECTION BY EDGE DRAWING (ED)

Cuneyt Akinlar and Cihan Topal
{cakinlar, cihant}@anadolu.edu.tr

Computer Engineering Department, Anadolu University, Eskisehir, Turkey

ABSTRACT

We propose a linear time line segment detector that gives accurate results, requires no parameter tuning, and runs up to 11 times faster than the fastest known line segment detection algorithm in the literature; namely, the LSD by Gioi et al. The proposed algorithm also includes a line validation step due to the Helmholtz principle, which lets it control the number of false detections. Our detector makes use of the clean, contiguous (connected) chain of edge pixels produced by our novel edge detector, the Edge Drawing (ED) algorithm; hence the name EDLines. With its accurate results and blazing speed, EDLines will be very suitable for the next generation real-time computer vision applications.

Index Terms— Real-Time Line Segment Detection, Edge Drawing Algorithm, LSD, Helmholtz Principle, NFA

1. INTRODUCTION

Line segment detection is an important problem in image processing and computer vision with many applications including image compression [1], crack detection in materials [2], stereo-matching [3], robot-navigation [4, 5], roadbed detection, among many others.

Traditional line segment detection algorithms first compute an edge map (typically by the famous Canny edge detector [6]), then apply the Hough transform [7, 8], and finally extract all lines that contain a certain number of edge points. Lines are broken into segments using gap and length thresholds. These methods are not only too slow, but they also combine non-contiguous line segments together, and usually produce a lot of false detections.

Progressive Probabilistic Hough Transform [9, 10] has been proposed to solve some of the shortcomings of the traditional Hough transform. Although it manages controlling the number of false detections, it fails to detect a lot of true positives, and it is extremely slow to be of any real use.

Etemadi [11] proposed a parameterless line detector that detects both line segments and arcs. Similarly, Burns et al. [14, 15] introduced a line segment detector that works by combining pixels having the same orientation. Although both of these algorithms produce well-localized line segments, they also generate a lot of false positives, especially in images with noisy backgrounds or those containing complex structures such as trees, grass, cloudy sky, or similar anisotropic structures.

Desolneux et al. [12, 13] proposed a parameterless line detector that controls the number of false positives. Their idea is to count the number of aligned pixels in a certain orientation and accept the set of pixels as a line segment if the observed structure is perceptually meaningful. This is called the Helmholtz principle from Gestalt theory [13], and is used as the line validation method. The problem with Desolneux's method is that it generates long lines which in fact should be broken down into several line segments, and that it is very computationally intensive, thus very slow.

By extending Burns's work for line segment generation and combining it with Desolneux's line validation method using the Helmholtz principle [13], von Gioi et al. [16, 17, 18] have recently proposed a parameterless line detection algorithm, called the Line Segment Detector (LSD), that produces accurate line segments, and also controls the number of false detections. Although LSD produces good results for most types of images, it fails especially in images where the background contains a lot of white noise, and its running time is still prohibitive; which makes it unsuitable for the next-generation real-time applications.

In this paper, we propose a fast, parameterless line segment detector that produces accurate results, and runs up to 11 times faster than the fastest known edge detector; namely, the Line Segment Detector (LSD) by von Gioi et al. [16, 17, 18]. Our detector also includes a line validation step due to the Helmholtz principle [13], which lets it control the number of false detections.

The proposed line detector is comprised of 3 steps: (1) Given a grayscale image, we first run our fast, novel edge detector, the Edge Drawing (ED) algorithm [19, 20, 23], which produces a set of clean, contiguous chains of pixels, which we call *edge segments*. Edge segments intuitively correspond to object boundaries. (2) Next, we extract line

segments from the generated pixel chains by means of a straightness criterion, i.e., by the least squares line fitting method [21]. (3) Finally, a line validation step due to the Helmholtz principle [13, 18] is used to eliminate false line segment detections.

The rest of the paper is organized as follows: In section 2, we briefly describe our edge detector, the Edge Drawing (ED) algorithm [19, 20]. Section 3 describes how we fit lines to pixel chains produced by ED. Section 4 describes our line validation step, and section 5 compares the performance of EDLines to others in the literature through experimentation.

2. EDGE DETECTION BY EDGE DRAWING

Edge Drawing (ED) is our recently-proposed, novel, fast edge detection algorithm [19, 20]. What makes ED stand out from the existing edge detectors, e.g., Canny [6], is the following: While the other edge detectors give out a binary edge image as output, where the detected edge pixels are usually independent, discontinuous entities; ED produces a set of edge segments, which are clean, contiguous, i.e., connected, chains of edge pixels. Thus, while the output of the other edge detectors requires further processing to generate potential object boundaries, which may not even be possible or result in inaccuracies; ED not only produces perfectly connected object boundaries by default, but it also achieves this in blazing speed compared to other edge detector [19].

Given a grayscale image, ED performs edge detection in 4 steps:

(1) The image is first passed through a filter, e.g., Gauss, to suppress noise and smooth out the image.
(2) The next step is to compute the gradient magnitude and direction at each pixel of the smoothed image. Any of the known gradient operators, e.g., Prewitt, Sobel, Scharr, etc., can be used at this step.
(3) In the third step, we compute a set of pixels, called the *anchors*, which are pixels with a very high probability of being edgels. Intuitively, these points correspond to pixels where the gradient operator produces maximal values, i.e., the peaks of the gradient map.
(4) Finally, we connect the anchors computed in the third step by drawing edges between them; hence the name *Edge Drawing (ED)*. The whole process is similar to children's boundary completion puzzles, where a child is given a dotted boundary of an object, and s/he is asked to complete boundary by connecting the dots. Starting from an anchor (dot), ED makes use of the neighboring pixels' gradient magnitudes and directions, and walks to the next anchor by going over the gradient maximas. If you visualize the gradient map as a mountain in 3D, this is very much like walking over the mountain top from one peak to the other.

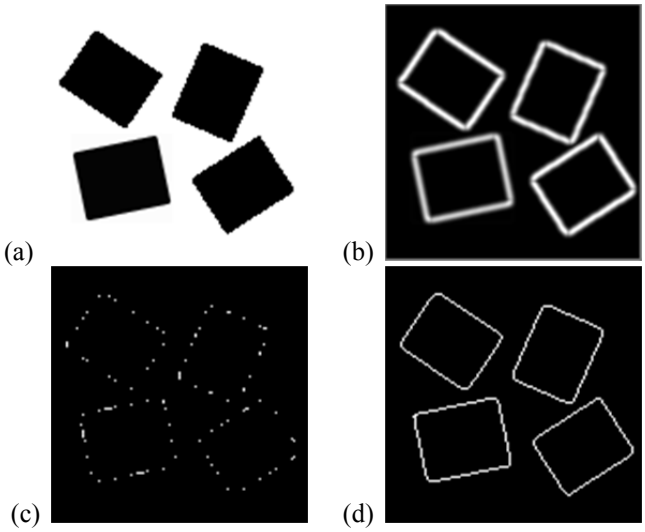

Figure 1: (a) A grayscale image containing 4 rectangles, (b) Gradient map, (c) Anchor points, (d) Final edge map.

Figure 1 shows ED in action on a 128x128 pixels grayscale image. Figure 1(b) shows the gradient map, where white pixels correspond to high gradient values (mountain tops). Figure 1(c) shows an example set of anchors, which correspond to peaks of the gradient map, and clearly depict the boundaries of the rectangles in the image. The final edge map, shown in Figure 1(d), is obtained by linking the anchors (dots) (refer to [19] for a detailed description of how ED links the anchors to obtain the final edge map). As mentioned before, ED not only produces a binary edge map similar to other edge detectors, but it also produces a set of edge segments, which are connected chain of pixels corresponding to object boundaries. In the given example, ED generates 4 edge segments, one for the boundary of each rectangle. Given these edge segments, all that remains for line segment extraction is to go over these pixel chains, and fit lines to the pixels. Next section describes how this is done.

3. LINE SEGMENT EXTRACTION

Given an edge segment comprised of a contiguous chain of edge pixels, the goal of this step is to split this chain into one or more straight line segments. The basic idea is to walk over the pixels in sequence, and fit lines to the pixels using the least squares line fitting method [21] until the error exceeds a certain threshold, e.g., 1 pixel error. When the error exceeds this threshold, we generate a new line segment. The algorithm then recursively processes the remaining pixels of the chain until all pixels are processed.

Figure 3: A comparison of EDLines to LSD on three challanging images. EDLines is 10 times or more faster than LSD while producing similar or better line segments. Both algorithms are run on a PC with an Intel 2.2 GHz CPU and 2 GB RAM.

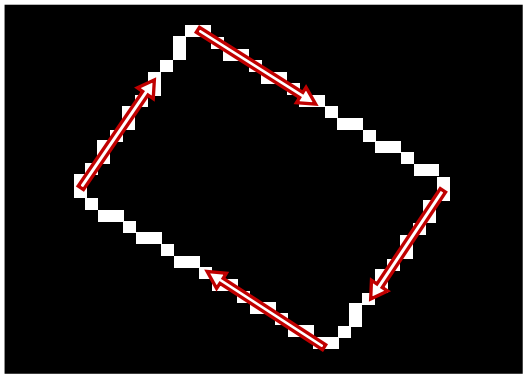

Figure 2: Illustration of line segment extraction from a contiguous chain of pixels. This chain corresponds to the boundary of the top-left rectangle in Figure 1.

Figure 2 illustrates the idea: Here, we see the pixels of the edge segment corresponding to the boundary of top-left rectangle in Figure 1. Starting from the first pixel of the chain, we take a certain number of pixels, e.g., 10, and fit a line to these pixels by the least squares line fitting method. This initial line determines the current direction of the line segment and is illustrated by an arrow in Figure 2. We then walk over the remaining pixels of the chain and compute the distance of each pixel to the currently fitted line. We would add pixels to the current line as long as the mean square error stays within a certain bound, e.g., 1 pixel. Intuitively, we would continue adding pixels to the current line segment until we turn a corner and the direction of the line changes. At that point, we output the current line segment and start a new one. With this simple algorithm, 4 line segments would be extracted from the given chain of pixels as illustrated in Figure 2.

4. LINE VALIDATION

Similar to Desolneux [12] and LSD [18], our line validation method is based on the Helmholtz principle, which basically

states that for a structure to be perceptually meaningful, the expectation of this structure (grouping) by chance must be very low [13]. This is an "*a contrario"* approach, where the objects are detected as outliers of the background model. As shown by Desolneux [12], a suitable background model is one in which all pixels (thus the gradient angles) are independent. They show that the simplest such model is the Gaussian white noise. To make validation by Helmholtz principle concrete, Desolneux defines what is called the "Number of False Alarms (NFA)" of a line segment as follows [12]: Let A be a segment of length "n" with at least "k" points having their directions aligned with the direction of A in an image of size NxN pixels. Define NFA of A as:

$$NFA(n,k) = N^4 . \sum_{i=k}^{n} p^i (1-p)^{n-i}$$

An event is called ε-meaningful if its $NFA(n,k) \leq \varepsilon$. Desolneux [12] advises setting ε to 1, which corresponds to one false detection per image. Given these definitions, we validate our line segments as follows: For a line segment of length "n", we compute the gradient angle of each pixel along the line segment and count the number of aligned pixels "k". We then compute $NFA(n,k)$, and accept the line segment as valid if $NFA(n,k) \leq 1$. Otherwise the line is rejected. We would also like to note that line validation is an optional last step in EDLines, and can be omitted if deemed unnecessary. This would further speed up EDLines.

5. EXPERIMENTS

We compared the performance of EDLines [24] to other line extraction algorithms in the literature. In this paper, however, we compare EDLines only to LSD [22]. This is due to the lack of space and LSD being the best line segment detector (to the best of our knowledge) in terms of speed and performance. Figure 3 shows the results produced by EDLines and LSD on three challenging images. In all cases, both EDLines and LSD were run with default parameters, i.e., without any parameter tuning. It is clear from the results that EDLines produces similar or better line segments than LSD while running up to 11 times faster. We should also note that LSD by default scales the image width and height by 0.8; that is, it reduces the original image to 64% of its size before processing. The results for LSD given in Figure 3 are based on this 0.8 scaling parameter. Without any scaling, LSD takes much longer to execute, and produces many more line segments.

6. REFERENCES

[1] P. Franti, E.I. Ageenko, H. Kalviainen, and S. Kukkonen, "Compression of Line Drawing Images Using Hough Transform for Exploiting Global Dependencies," Int. Conf. on Inf. Sci., 1998.

[2] S. Mahadevan, and D.P. Casasent, "Detection of Triple Junction Parameters in Microscope Images," Proc. of SPIE, pp. 204-214, 2001.

[3] C.X. Ji and Z.P. Zhang, "Stereo Match Based On Linear Feature," International Conference on Pattern Recognition, pp. 875-878, 1988.

[4] P. Kahn, L. Kitchen, and E.M. Rieseman, "A Fast Line Finder for Vision-Guided Robot Navigation," *IEEE Transactions* on Pattern Analysis and Machine Intelligence, vol. 12, no. 11, pp. 1098-1102, November 1990.

[5] P. Kahn, L. Kitchen, and E.M. Rieseman, "Real-Time Feature Extraction: A Fast Line Finder for Vision-Guided Robot Navigation," *Technical Report 87-57*, COINS, 1987.

[6] J. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 4, pp. 679-698, November 1986.

[7] R.O. Duda and P.E. Hart, "Use of Hough Transformation to Detect Lines and Curves in Pictures," Communications of the ACM, vol. 15, pp. 11-15, January 1972.

[8] H. Kalviainen, P. Hirvonen, and E. Oja, "Houghtool – A Software Package for the use of the Hough Transform," *Pattern Recognition Letters*, vol. 17, no. 8, pp. 889-897, 1996.

[9] J. Matas, C. Galambos, and J. Kittler, "Robust Detection of Lines Using the Progressive Probabilistic Hough Transform," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 119-137, 2000.

[10] C. Galambos, J. Kittler, and J. Matas, "Gradient Based Progressive Probabilistic Hough Transform," *IEEE Proc. Vision, Image and Signal Processing*, vol. 148, no. 3, pp. 158-165, 2001.

[11] A. Etemadi, "Robust Segmentation of Edge Data," *Int. Conf. on Image Processing and its Applications*, pp. 311-314, 1992.

[12] A. Desolneux, L. Moisan, and J.M. Morel, "Meaningful Alignments," *International Journal of Computer Vision*, vol. 40, no. 1, pp. 7-23, 2000.

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

[14] J.B. Burns, A.R. Hanson, and E.M. Riseman, "Extracting Straight Lines," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 4, pp. 425-455, July 1986.

[15] J.R. Beveridge, C. Graves, and C. Lesher, "Some Lessons Learned from Coding the Burns Line Extraction Algorithm in the Darpa Image Understanding Environment," *Technical Report CS-96-125,* Comp. Science Dept., Colorado State University, 1996.

[16] R.G. von Gioi, J. Jakubowicz, J.M. Morel, and G. Randall, "LSD: A Line Segment," *technical report*, Centre de Mathematiques et de leurs Applications (CMLA), Ecole Normale Superieure de Cachan (ENS-CACHAN), 2008.

[17] R.G. von Gioi, J. Jakubowicz, J.M. Morel, and G. Randall, "On Straight Line Segment Detection," *Journal of Mathematics Imaging and Vision*, vol. 32, no. 3, pp. 313-347, November 2008.

[18] R.G. von Gioi, J. Jakubowicz, J.M. Morel, and G. Randall, "LSD: A Fast Line Segment Detector with a False Detection Control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 722-732, April 2010.

[19] C. Topal, C. Akinlar, and Y. Genc, "Edge Drawing: A Heuristic Approach to Robust Real-Time Edge Detection," *Proceedings of the ICPR*, pp. 2424-2427, August 2010.

[20] C. Topal, and C. Akinlar, "Edge Drawing: A Real-Time Edge Segment Detector with a False Detection Control," submitted for publication, 2011.

[21] Least Squares Line Fitting, http://en.wikipedia.org/wiki/Least_Squares

[22] LSD: a Line Segment Detector, http://www.ipol.im/pub/algo/gjmr_line_segment_detector/

[23] Edge Drawing, http://ceng.anadolu.edu.tr/cv/EdgeDrawing

[24] EDLines, http://ceng.anadolu.edu.tr/cv/EDLines