A persistence-based approach to automatic detection of line segments in images

Vitaliy Kurlin[1][0000−0001−5328−5351] and Grzegorz Muszynski[1,2]

[1] Department of Computer Science, University of Liverpool, Liverpool L69 3BX, UK
[2] Lawrence Berkeley National Laboratory, CA, United States
vkurlin@liverpool.ac.uk, gmuszynski@lbl.gov

Abstract. Edge detection algorithms usually produce a discrete set of edgels (edge pixels) in a given image on a fixed pixel grid. We consider the harder problem of detecting continuous straight line segments at subpixel resolution. The state-of-the art Line Segment Detection Algorithm (LSDA) outputs unordered line segments whose total number cannot be easily controlled. Another motivation to improve the LSDA is to avoid intersections and small angles between line segments, hence difficulties in higher level tasks such as segmentation or contour extraction.

The new Persistent Line Segment Detector (PLSD) outputs only non-intersecting line segments and ranks them by a strength, hence the user can choose a number of segments. The main novelty is an automatic selection of strongest segments along any straight line by using the persistence from Topological Data Analysis. The experiments on the Berkeley Segmentation Database of 500 real-life images show that the new algorithm outperforms the LSDA on the important measure of Boundary Recall.

Keywords: Topological persistence · edge detection · skeletonization

1 Introduction

1.1 The edge detection problem in the continuous setting

Detecting edges in images is a key problem in the low-level vision that aims to identify pixels where the image intensity suddenly changes. The edge detection was usually considered in the discrete setting when an output consists of pixels from a given pixel grid. However, pixel-based images represent a continuous world, where the most basic objects are continuous line segments, which may have arbitrary directions and endpoints with any real coordinates.

The hard version of edge detection is to find straight line segments at subpixel resolution that approximate boundary contours in pixel-based images.

The state-of-the-art algorithm [11] solving the above problem is the Line Segment Detection Algorithm (LSDA). The main advantage of the LSDA over past edge detection algorithms is the "a contrario" approach that theoretically guarantees at most one false alarm on random data, see details in subsection 2.2.

1.2 Motivations to detect line segments without intersections

The LSDA often outputs line segments that intersect each other near their endpoints, see 48 intersections for the image in Fig. 1 from the Berkeley Segmentation Database [1]. Duan and Lafarge [6] have proposed a refinement of the LSDA edge for producing Voronoi superpixels at subpixel resolution. This refinement has revealed that some LSDA edges are too close and almost parallel to each other as clearly illustrated in [10, Fig. 1.1 on page 1]. So these close lines should be removed or carefully repaired to avoid very narrow superpixels. Fig. 1 shows how the PLSD avoids all intersections of edges in comparision with the LSDA.

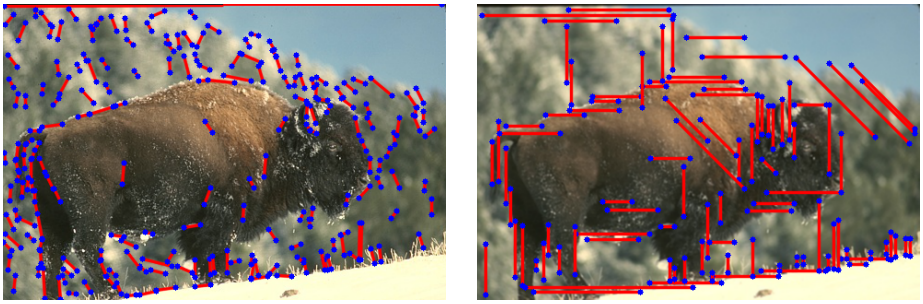

Fig. 1. Left: 193 LSDA edges with 48 intersections. **Right**: the PLSD outputs exactly 100 edges without any intersections of edges, which is the key advantage over LSDA.

The hard difficulties above are understandable taking into account that the LSDA attempts to capture line segments with any possible slope. Since approximate solutions are acceptable in real-life, we simplify the problem and will detect line segments that are parallel to one of 8 directions: horizontal $(1, 0)$, vertical $(0, 1)$, two diagonal $(\pm 1, 1)$ and four non-diagonal directions $(\pm 2, 1)$, $(\pm 1, 2)$.

We believe that 8 directions are enough to approximate any reasonable shapes in images, e.g. a large round disk in Fig. 2 can be well approximated by polygonal curves with 16 edges split into 8 pairs of opposite parallel edges.

One more important motivation to improve the LSDA is to control the number of edges in a final output. When LSDA edges are included into a polygonal mesh, the size of the mesh (number of polygons) may depend on the number of original edges. Hence, it would be great to order detected edges by some sort of strength so that a smaller number of strongest edges can be selected.

1.3 Automatic selection of persistent segments

The main novelty of the proposed algorithm PLSD (Persistent Line Segment Detector) is the automatic selection of strongest segments in any straight line.

A grayscale image on $\Omega = [0, w] \times [0, h]$ is a function $I : \Omega \to [0, 255]$ sampled at pixel positions $p \in \Omega$ with integer coordinates in the image $[0, w] \times [0, h]$. An edge detection algorithm outputs pixels $p_1, \ldots, p_k \in \Omega$, where the function I substantially changes (depending on an algorithm) along some direction. This change at a fixed pixel is measured as the magnitude of the image gradient.

For a function $f : L \to \mathbb{R}$ of contrast values along a fixed straight line L in an image, we analyze the sequence of superlevel sets $f^{-1}[u, +\infty) = \{p \in L \ : f(p) \geq u\}$. For every fixed level u of the contrast, the superlevel set splits into a few continuous segments over which the contrast is at least u.

When the contrast level u goes down, new segments appear around local maxima of f and then merge with each other, see Fig. 4. So each segment S *persists* from its *birth* (at the maximum value of u) to its *death* (at the value when S merges with another segment having a higher birth), see Definition 2.

A segment S is usually characterized by its persistence=birth-death (when the parameter u is decreasing). We suggest another characteristic (the *strength* $|S| = \int_S f(p)dp$), which is more stable under perturbations of contrast values, hence is more suitable for noisy data, see formal details in Definition 3.

Line segments are ranked according to the concept of persistence, which was introduced in Topological Data Analysis [7]. The idea of persistence is to study a nested sequence of shapes parameterized across all potential thresholds.

At every level u the strongest segments are separated from noisy artefacts by a widest gap in strength, which is the maximum difference between successive ordered strength values over all current segments, see Definition 3.

The same widest gap in persistence was successfully used for segmenting clouds of points [14]-[17] that are not restricted to a fixed pixel grid as in digital images. So the strongest segments are independently selected along every straight line L considered in an image. Hence there is no uniform thresholding for the whole image, see details of this new automatic method in subsection 3.2.

Here is the summary of key contributions.

• The edge detection is studied in the continuous setting, which is harder than for discrete square-based pixels.

• The algorithm PLSD can output a desired number of strongest straight line segments that have no intersections guaranteed by Stage 2 in subsection 3.3.

• The main innovation of the Persistent Line Segment Detector is a data-driven automatic selection of persistent line segments without manual thresholding.

• The PLSD runs in a near linear time, see Theorem 5, and outperforms the state-of-the-art Line Segment Detector on the Boundary Recall benchmark from the Berkeley Segmentation Database 500 [1].

2 Review of the past closely related work

This section discusses a few representative algorithms for detecting only straight line segments at subpixel precision.

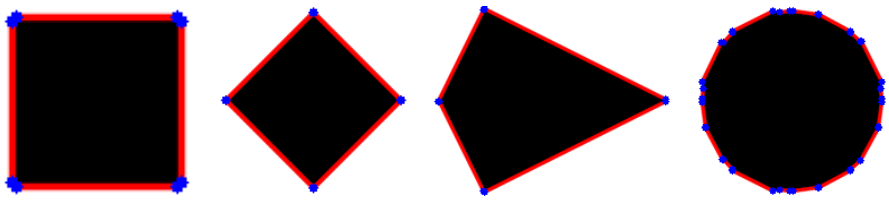

Fig. 2. The new algorithm PLSD outputs line segments in 8 directions, which can well approximate complicated shapes, even a large round disk in the last image above.

2.1 From discrete pixels to continuous arcs

Many past algorithms are based on the famous Canny detector of edge pixels [3], which already requires three parameters. The next usual step is to apply a Hough transform [2] to find lines passing through a certain number of edgels.

The Hough transform often leads to many false positives in textured regions. Another approach by Kahn et al. [12] uses only orientations of image gradients, but not their magnitudes. Their algorithm produces well localized edges, but requires carefully chosen thresholds.

A different "a contrario" (by contraries) approach is to validate potential candidates by setting thresholds on random data as follows. If a parametric algorithm on random data outputs a small number of false positives on average, the corresponding thresholds should be fixed and applied to real data. The only drawback was the exhaustive search through $O(P^4)$ possible straight lines, where P is the perimeter of an image. This method has led to the fast LSDA below.

2.2 The state-of-the-art Line Segment Detection Algorithm (LSDA)

The LSDA outputs line segments detected in a grayscale image at subpixel resolution [11]. The first step is to estimate the image gradient dI as the vector (g_x, g_y) whose components are obtained by convolving with these 2×2 masks:

$$g_x = \begin{bmatrix} -1 & +1 \\ -1 & +1 \end{bmatrix}, \qquad g_y = \begin{bmatrix} +1 & +1 \\ -1 & -1 \end{bmatrix}. \tag{2.2}$$

The operators above estimate the image derivatives in the x, y directions at the corner point shared by 4 pixels $(x, y), (x, y+1), (x+1, y), (x+1, y+1)$. So ideal edges were expected to be along boundaries of square pixels, but the original LSDA code shifted the final edges by $(0.5, 0.5)$. After normalising the gradient by its Euclidean length, the resulting field consists of unit length vectors.

Pixels whose estimated unit vectors are almost parallel (within a default tolerance $\tau = 22.5°$ for angles) are clustered. The resulting clusters are approximated by thin rectangles whose long middle lines are the final line segments. The output is an unordered list of line segments whose total number depends on a given image, so users may struggle to get a specific number of line segments.

2.3 Applications of line segments for superpixels

Since rectangles covering adjacent clusters may overlap, LSDA edges may have intersections close to their endpoints. The LSDA outputs line segments with intersections on about 80% of 500 BSD images without any order. Hence any further application of the LSDA for segmentation or contour extraction requires a careful refinement of LSDA edges. The LSDA output was used for Voronoi superpixels by Duan and Lafarge [6], who designed a multi-step post-processing to repair segments that intersect each other or have very close endpoints.

The main result of Duan and Lafarge [6] is probably the first algorithm splitting an image into convex polygons whose vertices may have any real coordinates. Forsythe and Kurlin [8]-[9] used a more sophisticated refinement of the LSDA output and proved that the resulting Convex Constrained Meshes (CCM) have no small angles and approximate LSDA edges considered as hard constraints.

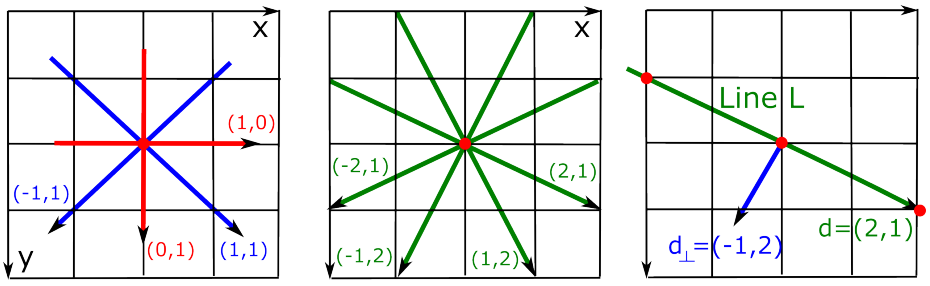

Fig. 3. Left: first 4 basic directions of line segments in the current implementation of the PLSD. **Middle**: for more directions. **Right**: the contrast function $f_L : L \to \mathbb{R}$ from Definition 1 is computed at all red points $(x, y) \in L$ with both integer coordinates.

The new detector PLSD can be used in both methods above without extra refinement, because all final edges have no intersections by construction.

3 PLSD: the new Persistent Line Segment Detector

This section describes the following 3 stages of the PLSD algorithm.

Stage 1: estimating the change of contrast along every straight line $d_x x + d_y y + t = 0$, where (d_x, d_y) is one of the 8 slopes in Fig. 3, the shift t takes all integer values when the resulting line L intersects the image $\Omega = [0, w] \times [0, h]$.

Stage 2: automatic selection of strongest line segments by their persistence using the contrast function along every line $d_x x + d_y y + t = 0$ from Stage 1.

Stage 3: choosing a required number of strongest segments (one by one) so that any weaker segments don't intersect already chosen stronger segments.

3.1 Stage 1: computing the contrast functions f along lines L

The first step convolves a given image I with the Gaussian kernel 3×3 with the default parameter $\sigma = 0.8$ using the GaussianBlur function in OpenCV. The second step considers all straight lines that intersect the image and are parallel to one of the 8 directions: $(1,0)$, $(0,1)$, $(\pm 1,1)$, $(\pm 1,2)$, $(\pm 2,1)$ in Fig. 3. These 8 directions are chosen for simplicity and speed of the current implementation.

Let the image domain Ω be a rectangle $[0,w] \times [0,h]$. Then we consider all points (x,y) with integer coordinates $b \le x \le w-b$, $b \le y \le h-b$. Here b is a small offset (the default value 3 pixels) that allows us to convolve I with gradient masks and avoid boundary effects. For a fixed point (x,y) with integer coordinates, the current implementation uses the simplest 2×2 masks g_x, g_y in formulae (2.2) to estimate the image gradient as $DI = (g_x * I, g_y * I)$. If I is a color image, the same linear operators g_x, g_y are applied to every color channel.

Definition 1 *For each of the 8 directions $d = (d_x, d_y)$ in Fig. 3, the change of contrast at an integer point (x,y) in an image $\Omega = [0,w] \times [0,h]$ is estimated as*

(3.1) *the directional derivative $f(x,y) = ||DI(x,y) \cdot d_\perp||$, where*

$d_\perp$ is the unit vector orthogonal to d. The norm $||\cdot||$ is the absolute value for grayscale images and is $||(R,G,B)||_\infty = \max\{|R|,|G|,|B|\}$ for color images. For every straight line L intersecting the image Ω, formula (3.1) defines the contrast function $f_L : L \to \mathbb{R}$ sampled at points $(x,y) \in \Omega$ with integer coordinates.

Definition 1 may use another norm for RGB images and mentions only 8 directions d for simplicity of the current implementation, The derivatives in (3.1) can be computed for any direction d. For a fixed directional vector d, consider all straight lines L given by $d_x x + d_y y + t = 0$ with the gradient d such that the shift t takes all integer values when the line intersects the image $\Omega = [0,w] \times [0,h]$.

We select segments $S \subset L$ such that the contrast function f_L over S has persistently larger values than over the rest of L. Here are the steps of Stage 1.

Step (1a). After Gaussian filtering an image I, compute the image gradient DI using 2×2 masks in (2.2). Any more advanced de-noising is possible. One can consider more sophisticated estimates of DI instead of 2×2 masks in (2.2).

Step (1b). For every line L parallel to one of 8 directions d and an integer point $(x,y) \in \Omega$ estimate the derivative of I in the direction orthogonal to d by (3.1).

The naive edge detection in the discrete setting can actually stop at this stage and output all points whose gradient magnitudes are above a certain threshold.

3.2 Stage 2: finding strongest segments by their persistence

The aim of this Stage 2 is to automatically select one or several segments within a fixed line L that well approximate contours of an image I within L.

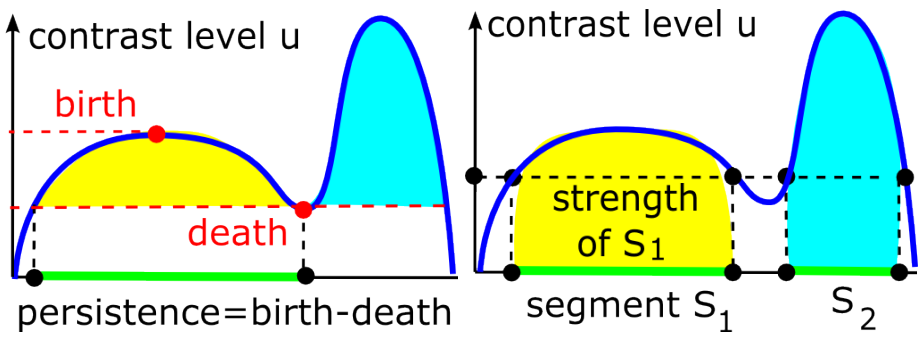

Fig. 4. Segments in superlevel sets $f_L^{-1}[u,+\infty)$ of a contrast function f_L grow and merge when the contrast level u goes down. The *strength* of a segment S is $\int_S f(p)dp$.

Stage 1 has essentially reduced the detection problem from dimension 2 to 1. Indeed, the input for Stage 2 is a graph of the contrast function $f_L : L \to \mathbb{R}$ sampled at integer points in the line L. The output will be segments $S_1, \dots, S_k \subset L$ over which the function f is substantially larger than over the rest of L.

The traditional approach is to manually choose a contrast threshold u and consider line segments where the contrast is sufficiently high: $f \geq u$.

The new approach is very different and has no thresholds at this stage. Following the key idea of Topological Data Analysis, we consider the sequence of all superlevel sets $f_L^{-1}[u,+\infty)$ when the level u goes down from a global maximum to a reasonable minimum. During this evolution, connected components of $f_L^{-1}[u,+\infty)$ appear at local maxima of f, grow and merge into larger components. Fig. 4 shows two segments that merge into a longer one.

Definition 2 *The* birth *of each component (line segment S) is the maximum value of f_L over S. The* death *of S is the level where S merges with another component. By the standard* elder rule *of persistence [7, p. 150], the older component (with a larger birth here) survives and the younger one dies. The whole process can be combinatorially described by a* topological barcode *of intervals $(death, birth]$ or a* persistence diagram *of pairs $(birth, death)$.*

The main advantage of the persistence diagram is the stability under bounded noise. If a function f_L is perturbed up to ϵ (say with respect to the L_∞ norm), the diagram is perturbed also up to ϵ with respect to the so-called bottleneck distance [5]. Since outliers may destroy this stability we suggest a new measure for selecting segments by analyzing the sequence of superlevel sets.

Definition 3 *At every fixed level u, any current segment (a connected component of $f_L^{-1}[u,+\infty)$) has the* strength $|S| = \int_S f_L(p)dp$, *which is approximated for a pixel-based image as the sum of $f_L(p)$ for all points $p \in S$ with integer coordinates. Fig. 4 shows the strength $|S|$ as the area below the graph of f_L.*

Now all segments at the fixed level u can be ranked according to their strengths, say $S_1 > \cdots > S_k$. To separate strongest segments from the rest, below we use the heuristic of the widest gap between these ordered strengths.

Find an index i such that the difference $S_i - S_{i+1}$ (the gap between successive strengths) is largest over all $i = 1, \dots, k-1$. The segments with the strengths $S_1, \dots, S_{i(u)}$ above this widest gap are called strongest *at the current level u.*

Contrast values of real images have wide gaps usually in a high-value range, because low values tend to be densely packed. Hence selecting segments with strengths above the widest gap (in every line L individually) is a better data-driven approach than guessing one threshold for contrast over the whole image.

3.3 Stage 3: a required number of segments without intersections

After Stage 2 above we have one or more strongest segments within every line L parallel to one of 8 directions. So a straight line may continue a few disjoint segments, not necessarily one. Final Stage 3 greedily selects a required number of strongest segments without intersections. In more details, we first take the strongest segment S from those obtained at Stage 2 in all lines L. Then we remove all line segments that *contradict* the strongest segment S as follows.

Definition 4 *A line segment S'* contradicts *another line segment S if either*
(4a) S' is parallel to S and is away from S within 3 pixels (a default value) or
(4b) S' intersects the segment S, endpoints of S can be inside S' and vice versa.

The default value of 3 pixels between line segments is the reasonable minimum, because the accuracy of human-drawn contours in the BSD is 2 pixels. After removing the chosen segment S_1 all segments contradicting S_1, we select the strongest segment S_2 from the remaining ones, again remove all segments contradicting S_2 and so on until we have found a required number of segments or there are no segments left from Stage 2.

To quickly check the conditions of Definition 4, we keep all segments parallel to one of 8 directions d in a binary tree T_d ordered by the following *identifier* of a line parallel to d. This tree is implemented as a multi-map structure of pairs (identifier of a line L, a segment S within L).

For any non-horizontal infinite line L, this identifier is the x-coordinate at the intersection of L with the x-axis. For a horizontal line L parallel to $d = (1, 0)$, the identifier of L is the constant y-coordinate of L.

Since the number k of required segments is usually much smaller than the number n of pixels, Theorem 5 justifies that the PLSD algorithm is near linear.

Theorem 5 *For any image consisting of n pixels, the algorithm PLSD outputs k straight line segments in time $O(kn \log n)$ and requires $O(n)$ space.*

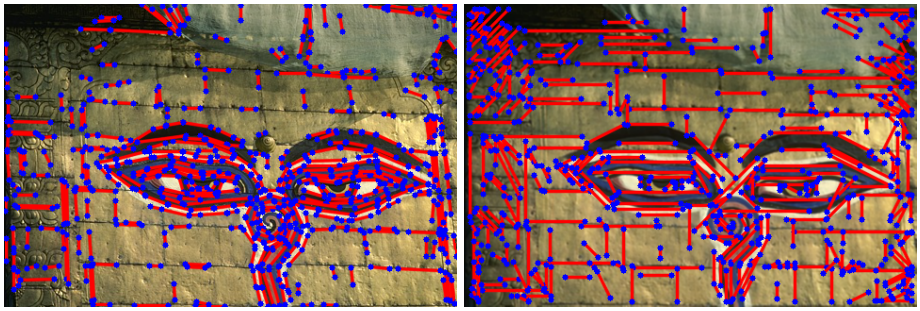

Fig. 5. Left: LSDA output on image 56028 in BSD. **Right**: More and longer straight line segments are found by PLSD.

Proof. For an image of $n = w \times h$ pixels and any of the 8 basic directions d, there at most $w + h = O(\sqrt{n})$ straight lines L parallel to d. Each of these lines contains at most $w + h = O(\sqrt{n})$ points (x, y) with integer coordinates.

For every fixed line L, we use a union-find structure to analyze the evolution of segments $S \subset L$, which are connected components of superlevel sets $f_L^{-1}[u, +\infty)$. We sort the contrast values of $O(\sqrt{n})$ points p within the line L in time $O(\sqrt{n} \log n)$ and process them starting from the largest.

All current segments are kept in a binary tree of size $O(\sqrt{n})$. When a new point p is added to a superlevel set $f_L^{-1}[u, +\infty)$, we have one of three cases:

(5a) p forms a new segment consisting of a single node.

(5b) p joins one of existing straight line segments S.

(5c) p is the merge point of two segments S_1 and S_2.

In case (5a) a new segment is added to the binary tree T_d in time $O(\log n)$. In case (5b) the existing segment is found and its strength is updated in time $O(\log n)$. In case (5c) two segments are removed and a new larger one is inserted in time $O(\log n)$. In general, making $O(\sqrt{n})$ updates, the union-find structure [18] maintains connected components of $O(\sqrt{n})$ points $p \in L$ in time $O(\sqrt{n} \log n)$.

At every step of Stage 2 we update the binary tree of $O(\sqrt{n})$ segments (parallel to a fixed direction d) as we need to know the widest gap between successive strengths. The binary trees T_d ordered by unique identifiers of lines L parallel to d help to remove in time $O(\sqrt{n} \log n)$ all segments contradicting a current strongest segment in the sense of Definition 4. The factor k in the complexity is from the number of strongest segments that are searched in the trees T_d. □

4 Experiments on 500 BSD images

4.1 The Boundary Recall benchmark BR(2) from BSD500

The Berkeley Segmentation Database (BSD) [1] consists of 500 images widely used for evaluating segmentation algorithms due to human-sketched ground truth boundaries. The human-drawn boundaries for each image are discretized and saved as a set G of ground-truth pixels. If E is another set of pixels produced by an edge detection algorithm, the standard *Boundary Recall* is

$$BR(G,E,\epsilon)=\frac{\#\{\text{pixels } p\in G \ : \ \text{distance}(p,E)\le\epsilon\}}{|G|},$$

where $\text{distance}(p,E)$ is the Euclidean distance between (centers of) p and its closest neighbor in E. The standard offset of $\epsilon=2$ pixels for the Boundary Recall is usually chosen, because human drawings cannot be more accurate.

Since there are up to 7 human-drawn boundaries B per image, the convention is to compute $BR(E,\epsilon)$ for a fixed image as the maximum of $BR(G,E,\epsilon)$ over all ground-truths G, hence over the best human drawing. The final Boundary Recall $BR(\epsilon)$ in Fig. 6 is the average of $BR(E,\epsilon)$ over all 500 images.

For any line detector at subpixel resolution, there is little sense to discretize its output set S of line segments. We compute the Euclidean $\text{distance}(p,S)$ from a ground truth pixel $p\in G$ to a closest line segment in the output S.

$$BR(G,S,\epsilon)=\frac{\#\{\text{pixels } p\in G \ : \ \text{distance}(p,S)\le\epsilon\}}{|G|}.$$

4.2 LSDA vs PLSD on the Boundary Recall BR(2)

Since the LSDA was extensively compared with past line segment detections in [10, section 4], this paper quantitatively compares the PLSD only with the LSDA. The LSDA "was designed as an automatic image analysis tool and must work without requiring any parameter tuning" [10, page 2]. We followed the advice of R. Grompone von Gioi [10] to run the LSDA with the default parameters. Hence the LSDA results are represented by a single black dot whose horizontal coordinate is the average number of line segments across BSD500.

For each BSD image, the LSDA produced a number of line segments according to the "a contrario" model. The PLSD algorithm is more flexible and can output a smaller or larger number of segments by users' choice, see Fig. 5.

For a fair comparison, on every image we first ran the LSDA code and then asked PLSD to output the same number of line segments as LSDA. However, in some cases the PLSD algorithm outputs a smaller number of edges because all edges are required to be non-intersecting.

The graph in Fig. 6 shows one black dot for LSDA and the blue polygonal curve with 4 dots corresponding to offset = 2,3,4,5 pixels (from right to left). This offset parameter is used in condition (4) to avoid very close parallel segments.

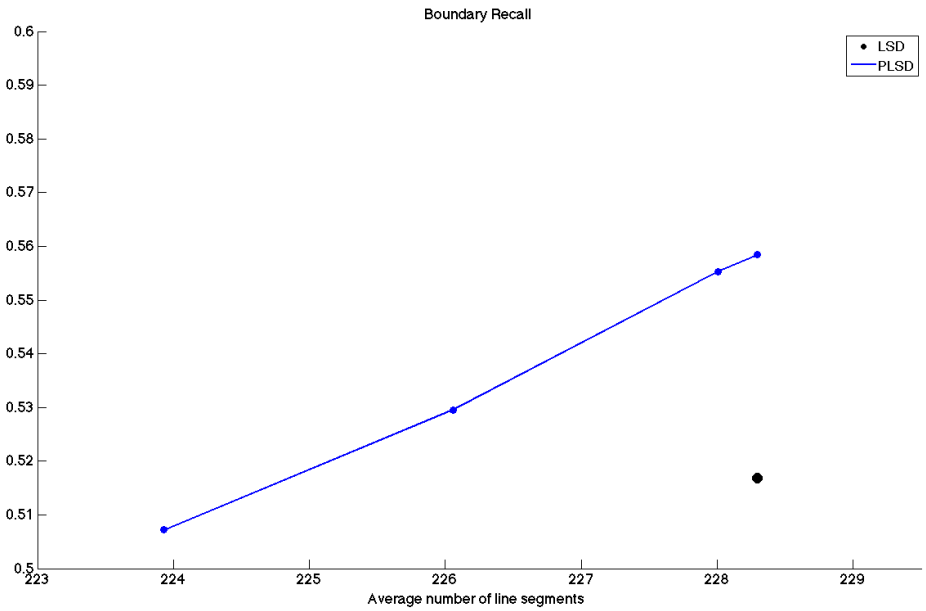

Fig. 6. Boundary Recall BR(2) for PLSD and LSDA on 500 BSD images, the horizontal axis shows the average number of segments.

Any straight line segment found by LSDA or PLSD is discretized by drawing black lines on a white background and extracting resulting black pixels as the discrete output, because the human drawings were discretized in BSD.

Let P be the number of output boundary pixels by an algorithm and TP be the number of those output pixels that are also in a ground truth boundary. The best suitable ground truth human drawing is chosen for every image.

The *sensitivity* or *True Positive Rate* is $TPR = \dfrac{TP}{P}$. The *precision* or *Positive Predictive Value* is $PPV = \dfrac{TP}{TP + FP}$, where FP is the number of false positives (all pixels that are in the output, but not in the ground truth).

Fig. 7 shows that the PLSD has the True Positive Rate (TPR) almost twice better than the LSDA. Fig. 8 shows that the PLSD outperforms the LSDA on the Positive Predictive Value (PPV).

5 Discussion and conclusions

The experiments in section 4 have demonstrated that the proposed detection of line segments parallel to one of 8 directions already outperforms the state-of-the-

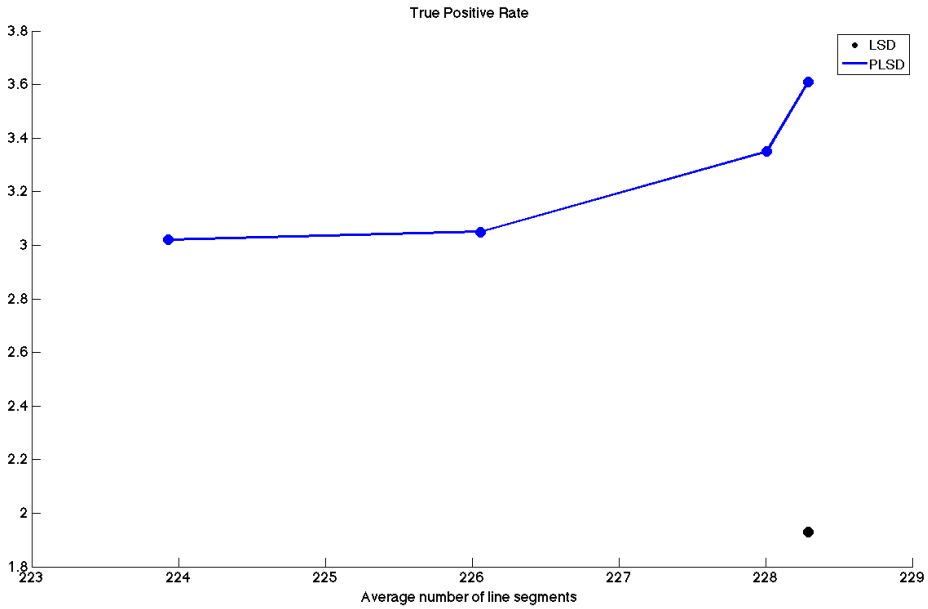

Fig. 7. Average TPR in percents vs the number of segments over BSD500.

art algorithm that allows line segments with any slope. The data-driven approach of selecting strongest segments can be extended to more than 8 directions.

Other possible improvements are better filtering, e.g. optimizing the size and sigma in the Gaussian kernel, and more advanced de-noising before Step (3.1a). The current non-optimized code runs for about 1 sec per BSD image on a laptop with 8Gb Ram, which is a bit slower than the LSDA on the same machine.

The straight line segments can be used as very economical descriptors of complicated scenes. For example, training convolution neural networks on straight line sketches can be much faster than on original images.

The novel method of automatic selection in subsection 3.2 can be used for finding skeletons of objects [4], [13], [15],[16], where thresholds should be avoided. Here is the summary of contributions to the line segment detection problem.

• The PLSD allows a user to fix a desired number of strongest line segments.

• All line segments in the output have no intersections by Definition 4, hence PLSD can be easily extended to planar skeletons and polygonal meshes.

• The PLSD has a near linear computational complexity by Theorem 5.

• A thresholding of contrast values was avoided due to the new data-driven method motivated by a multi-scale approach of Topological Data Analysis.

• The PLSD has outperformed the LSDA on the Boundary Recall, e.g. for the default offset of 3 pixels the measure BR(2) has improved from 0.517 to 0.559.

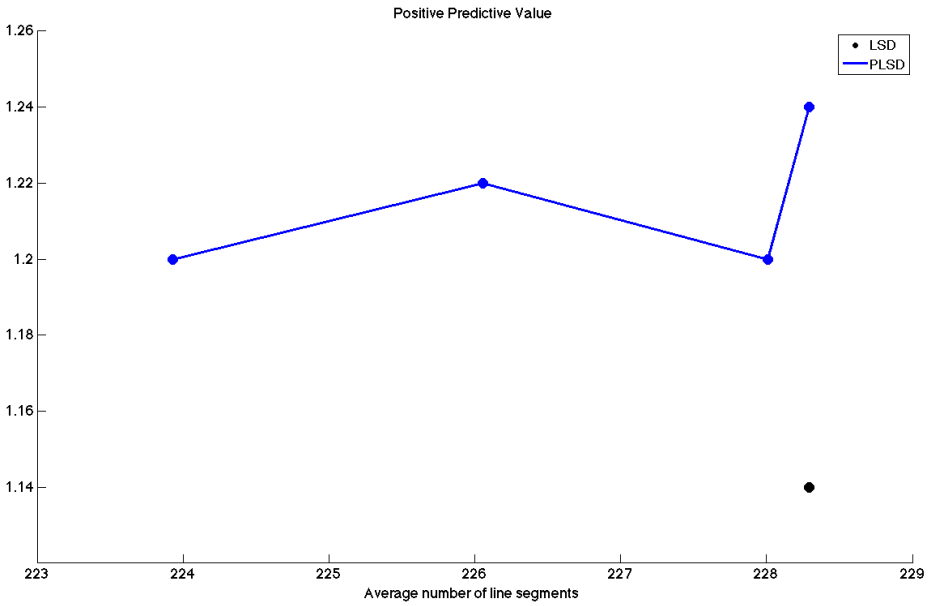

Fig. 8. Average PPV in percents vs the number of segments over BSD500.

We thank all reviewers for helpful suggestions. This work was supported by the EPSRC grant "Application-driven Topological Data Analysis" (EP/R018472/1).

References

1. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. Transactions PAMI **33**, 898–916 (2011)
2. Ballard, D.: Generalizing the hough transform to detect arbitrary shapes. Pattern Recognition **13**, 111–122 (1981)
3. Canny, J.: A computational approach to edge detection. Transactions PAMI **8**, 679–698 (1986)
4. Chernov, A., Kurlin, V.: Reconstructing persistent graph structures from noisy images. Image-A **3**, 19–22 (2013)
5. Cohen-Steiner, D., Edelsbrunner, H., Harer, J.: Stability of persistence diagrams. Discrete and Computational Geometry **37**, 103–130 (2007)
6. Duan, L., Lafarge, F.: Image partitioning into convex polygons. In: Proceedings of CVPR (Computer Vision and Pattern Recognition). pp. 3119–3127 (2015)
7. Edelsbrunner, H., Harer, J.: Computational topology. An introduction. AMS, Providence (2010)
8. Forsythe, J., Kurlin, V.: Convex constrained meshes for superpixel segmentations of images. Journal of Electronic Imaging **26(6)**(061609) (2017)
9. Forsythe, J., Kurlin, V., Fitzgibbon, A.: Resolution-independent superpixels based on convex constrained meshes. In: Proceedings of ISVC. pp. 223–233 (2016)

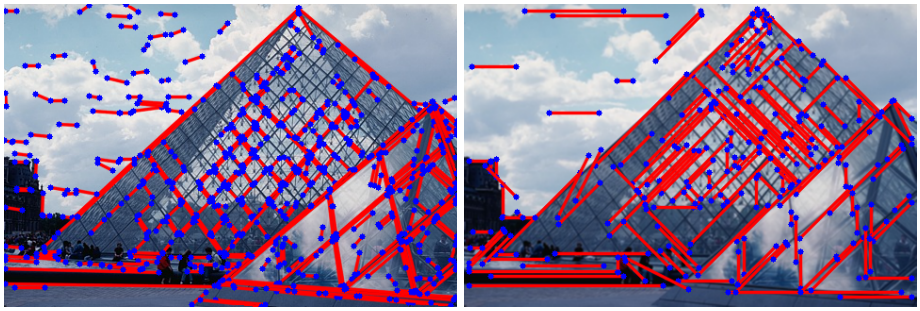

Fig. 9. Left: the LSDA output has too short segments in image 223060 from the BSD500. **Right**: many longer line segments are found by the new PLSD algorithm.

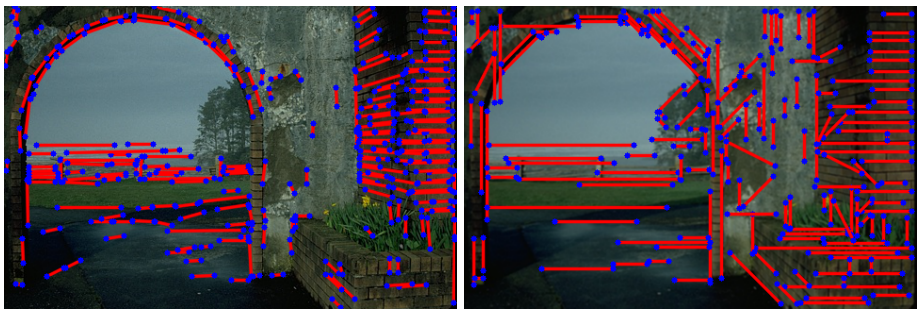

Fig. 10. Left: the LSDA missed some vertical lines in image 5096 from the BSD500. **Right**: many longer line segments are found by the new PLSD algorithm.

10. Grompone von Gioi, R.: A Contrario Line Segment Detector. Briefs in Computer Vision, Springer (2014)
11. Grompone von Gioi, R., Jakubowicz, J., Morel, J.M., Randall, G.: Lsd: a line segment detector. Image Processing On Line **2**, 35–55 (2012)
12. Kahn, P., Kitchen, L., Riseman, E.: A fast line finder for vision-guided robot navigation. Transactions PAMI **12**, 1098–1102 (1990)
13. Kalisnik, S., Kurlin, V., Lesnik, D.: A high-dimensional homologically persistent skeleton. Advances in Applied Mathematics **102**, 113–142 (2019)
14. Kurlin, V.: Auto-completion of contours in sketches, maps and sparse 2d images based on topological persistence. In: Proceedings of SYNASC 2014 workshop CTIC: Computational Topology in Image Context. pp. 594–601. IEEE (2014)
15. Kurlin, V.: A homologically persistent skeleton is a fast and robust descriptor of interest points in 2d images. In: LNCS. vol. 9256, pp. 606 – 617 (2015)
16. Kurlin, V.: A one-dimensional homologically persistent skeleton of a point cloud in any metric space. Computer Graphics Forum **34**, 253–262 (2015)
17. Kurlin, V.: A fast persistence-based segmentation of noisy 2d clouds with provable guarantees. Pattern Recognition Letters **83**, 3–12 (2016)
18. Tarjan, R.: Data structures and network algorithms. SIAM (1983)