Persistence-based resolution-independent meshes of superpixels

Vitaliy Kurlin[a,*], Grzegorz Muszynski[b]

[a] *Department of Computer Science, Ashton street University of Liverpool, Liverpool L69 3BX, UK*
[b] *Lawrence Berkeley National Laboratory, 1 Cyclotron Rd, Berkeley, CA 94720, USA*

ARTICLE INFO

Article history:
Received 15 May 2019
Revised 11 September 2019
Accepted 8 January 2020
Available online 13 January 2020

MSC:
41A05
41A10
65D05
65D17

Keywords:
Edge detection
Polygonal meshes
Persistent homology

ABSTRACT

The over-segmentation problem is to split a pixel-based image into a smaller number of superpixels that can be treated as indecompasable regions to speed up higher level image processing such as segmentation or object detection. A traditional superpixel is a potentially disconnected union of square pixels, which can have complicated topology (with holes) and geometry (highly zigzag boundaries). This paper contributes to new resolution-independent superpixels modeled as convex polygons with straight-line edges and vertices with real coordinates not restricted to a fixed pixel grid. Any such convex polygon can be rendered at any resolution higher than in original images, hence superpixels are resolution-independent.

The key difficulty in obtaining resolution-independent superpixels is to find continuous straight-line edges, while classical edge detection focuses on extracting only discrete edge pixels. The recent Persistent Line Segment Detector (PLSD) avoids intersections and small angles between line segments, which are hard to fix before a proper polygonal mesh can be constructed. The key novelty is an automatic selection of strongest straight-line segments by using the concept of persistence from Topological Data Analysis, which allows to rank segments by their strength. The PLSD performed well in comparison with the only past Line Segment Detector Algorithm (LSDA) on the Berkeley Segmentation Database of 500 real-life images. The PLSD is now extended to the Persistent Resolution-Independent Mesh (PRIM).

1. Introduction: motivations and problem statements

The traditional over-segmentation problem is to group square pixels into superpixels that adhere to object boundaries. The boundaries of these pixel-based superpixels consist of short horizontal and vertical edges restricted to a given grid, see [17].

Since images represent a spatially continuous world, it makes sense to find segmentations over a *continuous image domain*, not over a discrete grid. Due to anti-aliasing, grayscale values across a real image edge change gradually over 2–3 pixels, see (Viola et al. [21], Fig. 1). Hence a real edge is often not along pixel boundaries and should be found in the infinite family of line segments that can have any slope and endpoints with real coordinates.

The above facts motivate the harder version of edge detection to find straight-line segments that approximate object boundaries in pixel-based images at subpixel resolution. The next step is to extend straight-line segments to a full mesh of polygons that can be rendered at any resolution higher than in an original image. Such polygons are called *resolution-independent* superpixels, see [9,20].

The only past algorithm by Grompone von Gioi [10] solving the above problem was the Line Segment Detection Algorithm (LSDA). This “a contrario” approach guarantees at most one false alarm on random data.

The LSDA often outputs line segments that intersect each other near their endpoints, see 48 intersections for the image in Fig. 11 from the Berkeley Segmentation Database, see [1]. Duan and Lafarge [6] 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, see [10, Fig. 1.1 on page 1]. So these close lines should be removed or carefully repaired to avoid very narrow superpixels. Fig. 11 shows how the PLSD avoids all intersections in comparison with the 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 (± 1, 1) and four non-diagonal directions (± 2, 1), (± 1, 2). Fig. 2 shows that these 8 directions are enough to approximate any reasonable shapes in images, e.g., a disk in Fig. 2 is

[*] Corresponding author.
E-mail address: vitaliy.kurlin@gmail.com (V. Kurlin).

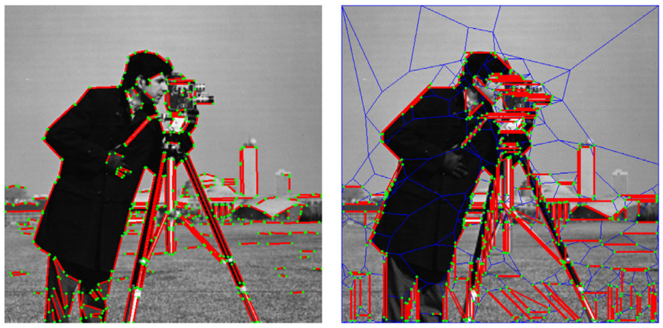
Fig. 1. **Left**: LSDA outputs (an unpredictable number of) 258 edges with 16 intersections. **Right**: the PLSD outputs a given number of 258 edges without any intersections (a key advantage over LSDA).

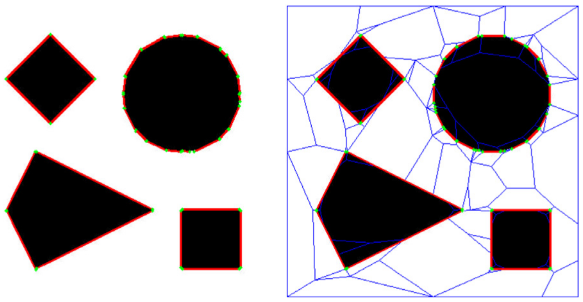
Fig. 2. The new algorithm PLSD outputs line segments in 8 directions, which can well approximate complicated shapes, even a large round disk in the last image above.

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.

2. Main novelty: selection of persistent segments

The main novelty of the recent Persistent Line Segment Detector (PLSD) by Kurlin and Muszynski [18, section 3] and its proposed extension to the Persistent Resolution-Independent Mesh (PRIM) 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 $p_1, \dots, p_k \in \Omega$, 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 straight line $L \subset \Omega$, 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, see [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 point clouds not restricted to a pixel grid as in digital images, see [13-16].

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 4.2.

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 4.3.
- The main innovation of the Persistent Line Segment Detector is a data-driven automatic selection of persistent line segments without manual thresholding.
- The PLSD edges are extended to a Persistent Resolution-Independent Mesh (PRIM) of superpixels, which are convex polygons having straight-line edges and vertices with any real coordinates.
- The PLSD and PRIM algorithms run in a near linear time with respect to the number of pixels and required line segments, see Theorem 5 and Corollary 7.

3. Review of the past closely related work

This section discusses the algorithms for detecting only straight line segments at subpixel precision.

3.1. 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 by Ballard [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. [11] uses only orientations of image gradients, but not their magnitudes. Their algorithm produces well localized edges, but requires carefully chosen thresholds. One of the first algorithms for reconstructing arbitrary planar graphs from point clouds was suggested by Chernov and Kurlin [4].

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 described below.

3.2. The line segment detection algorithm (LSDA)

The LSDA outputs line segments detected in a grayscale image at subpixel resolution, see [10]. The first step is to estimate the image gradient dI as the vector (g_x, g_y) whose components are ob-

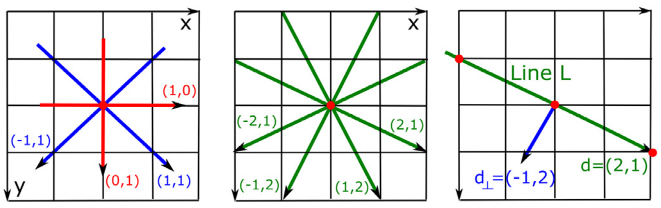

Fig. 3. **Left**: first 4 basic directions of line segments in the current implementation of the PLSD. **Middle**: for more directions. **Right**: the contrast function $f_L : L \to \mathbb{R}$ from Definition 1 is computed at all red points $(x, y) \in L$ with both integer coordinates. (For interpretation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

tained by convolving with these 2×2 masks:

$$g_x = \begin{bmatrix} -1 & +1 \\ -1 & +1 \end{bmatrix}, \qquad g_y = \begin{bmatrix} +1 & +1 \\ -1 & -1 \end{bmatrix}. \tag{3.1}$$

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 $\tau = 22.5^\circ$ 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.

3.3. 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.

Forsythe et al. [9] and Forsythe and Kurlin [8] used a sophisticated refinement of the LSDA and proved that the resulting Convex Constrained Meshes (CCM) have no small angles with LSDA edges as hard constraints.

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

4. PLSD: The persistent line segment detector

This section describes the 3 stages of the PLSD.

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 straight 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.

4.1. Stage 1: 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), (± 1, 1), (± 1, 2), (± 2, 1) in Fig. 3. These 8 directions are chosen to speed up the current implementation. Further steps work for any number of directions.

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 (3.1) to estimate the image gradient as $DI = (g_x * I, g_y * I)$. If I is a colour image, the same linear operators g_x, g_y are applied to every colour 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

$$\text{the directional derivative } f(x, y) = ||DI(x, y) \cdot d_\perp||, \tag{4.1}$$

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 colour images. For every straight line L intersecting the image Ω, formula (4.1) 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 (4.1) 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, compute the image gradient DI using 2×2 masks in (3.1). Any more advanced de-noising is possible. One can consider other estimates of DI instead of 2×2 masks in (3.1).

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 (4.1).

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.

4.2. Stage 2: Segments selected 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.

Stage 1 has reduced the detection problem from dimension 2 to 1, because 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: $f \ge u$. This superlevel set splits into several connected components, which are line segments within the line L. Some of these segments can be filtered out by (say) their short length or compared with segments in other lines.

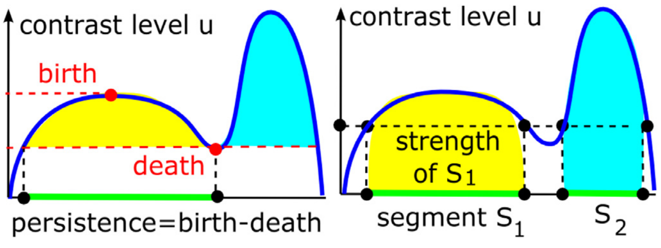

Fig. 4. Segments in superlevel sets $f_L^{-1}[u, +\infty)$ of a contrast function f_L grow and merge when the contrast level u goes down. The *strength* of a segment S is $\int_S f(p)dp$.

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, see [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, see [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 $S_1 > \ldots > S_k$.

To separate strongest segments from the rest, we use the heuristic of the widest gap between these ordered strengths. Find an index i such that $S_i - S_{i+1}$ (the gap between successive strengths) is largest over $i = 1, \ldots, k-1$. The segments with the strengths $S_1, \ldots, S_{i(u)}$ 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.

4.3. Stage 3: A given number of non-intersecting segments

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. 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 2 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 2 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 has a near linear time, see the proof in [18, section 3].

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.*

5. PRIM: Persistent resolution-independent mesh

This section describes a new mesh of convex polygons whose boundaries include all PLSD straight-line edges as hard constraints. The key idea is to extend PLSD segments in both directions until a convex partition is formed. The brute-force extension of k segments can create $O(k^2)$ intersections and polygons in time $O(k^2)$.

The faster approach below uses Shewchuk's Triangle [19], which accepts any planar straight line graph and outputs a Steiner constrained Delaunay Triangulation without small angles in a near linear time.

Theorem 6. *[19, Theorem 12] For a planar straight line graph G having m vertices and no angles smaller than $\varphi \le 60^\circ$, in time $O(m\log m)$ one can build a triangulation $T(G)$ without angles smaller than the minimum angle equal to $\arcsin(\frac{1}{\sqrt{2}}\sin\frac{\varphi}{2})$.*

If $\varphi = 60^\circ$, the minimum angle is $\arcsin(\frac{1}{\sqrt{2}}\sin\frac{\varphi}{2}) \approx 20.7^\circ$. Due to only 8 directions in Fig. 3, the minimum angle between PLSD is at least $\varphi = 22.5^\circ$, hence the minimum angle in polygons of PRIM is about 8°.

A Persistent Resolution-Independent Mesh (PRIM) of convex polygons is built as follows.

PRIM Step 1: add to PLSD edges the four boundary edges of a given image to restrict a final mesh to the image.

PRIM Step 2: run Shewchuk's Triangle code to get a constrained triangulation T containing all PLSD edges.

PRIM Step 3: order unconstrained and non-boundary edges of the triangulation from lower to higher strength.

PRIM Step 4: for each edge in the ordered list, merge the incident polygons if the resulting polygon is convex, which requires computing 4 angles as shown in Fig. 5 (left).

Corollary 7. *For any image consisting of n pixels, if the algorithm PRIM outputs k straight line segments whose extended infinite lines have m intersections, the total complexity is $O(kn\log n)$.*

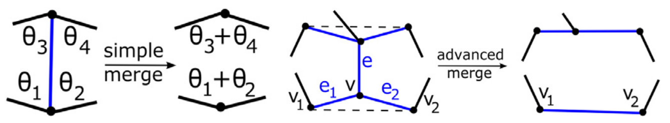

Fig. 5. **Left**: a simple merge of adjacent polygons respects convexity. **Right**: a more advanced merge may perturb constrained edges.

Proof follows from Theorems 5 and 6, because we have $m \leq 2k = O(k)$ vertices. Step 3–4 sort $O(k)$ edges and compute 4 angles per edge to check if merging two polygons respects convexity. The total complexity is dominated by the near linear time in the number n of pixels.

The approach above applied to LSDA edges with intersections led to unsatisfactory because of small angles. LSDA edges are often short and form long “almost connected” zigzag chains along boundaries of curved objects. Extending almost parallel edges creates very small angles, which required a sophisticated refinement by using Shewchuk’s triangulations in [8].

6. Discussion and conclusions

Figures 6, 7, 8, 9, 10 and 11 compare outputs of the LSDA and PLSD algorithms, see more examples in [18, section 5]. Typically, the LSDA outputs many short (often intersecting) segments, while PLSD outputs longer segments without intersections. The novel method of automatic selection in Subsection 4.2 can be used for finding skeletons, where thresholds should be avoided, see [12,14,15].

Table 1
BR is the boundary recall on the Berkeley segmentation database BSD500. The numbers for PLSD and PRIM with 8 directions are offsets (minimum distance between close parallel edges).

8 dir.	PLSD3	PLSD4	PRIM3	PRIM4	PRIM5
BR	0.561	0.541	0.672	0.661	0.639

Table 2
Boundary recall when only 4 directions are used in PLSD.

4 dir.	PLSD3	PLSD4	PRIM3	PRIM4	PRIM5
BR	0.542	0.509	0.657	0.637	0.617

The algorithm LSDA has the Boundary Recall of 0.517, i.e. about 51.7% of human-labeled boundary pixels were within 2 pixels from LSDA segments. Table 1 shows how extending PLSD edges to a full mesh captures more boundary pixels. Table 2 justifies 8 directions, because the Boundary Recall of PLSD (with offset of 4 pixels) for 4 directions drops below BR of LSDA. Since the drop is small, then increasing directions may not add big value.

The experiments in [18, section 5] have demonstrated that the proposed detection of line segments parallel to one of 8 directions performs better than the LSDA and guarantees no intersections and no small angles between final segments. Here is the summary of contributions to the line segment detection and over-segmentation problems.

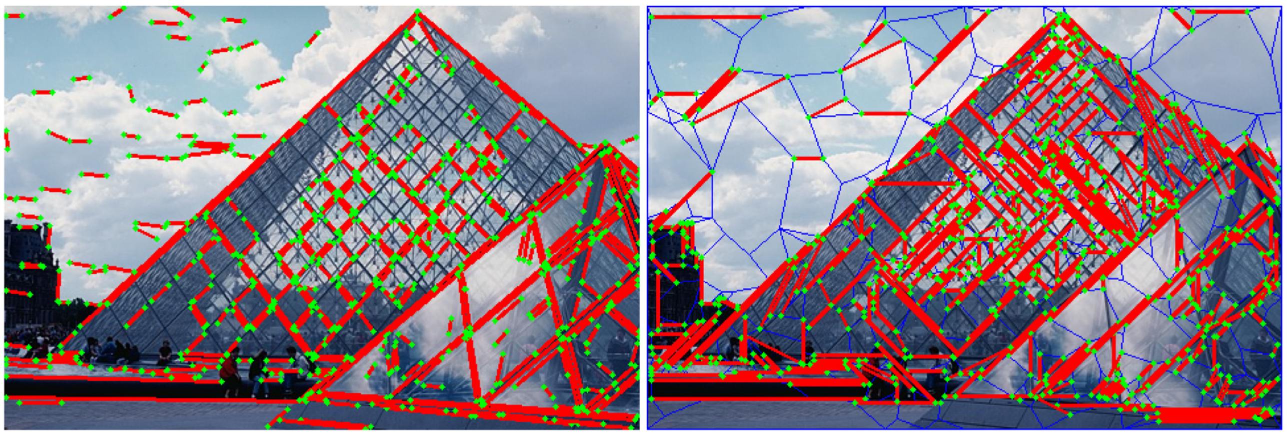

Fig. 6. **Left**: 378 LSDA edges with 28 intersections on image 223,060 in BSD. **Right**: PRIM with 378 PLSD edges without intersections.

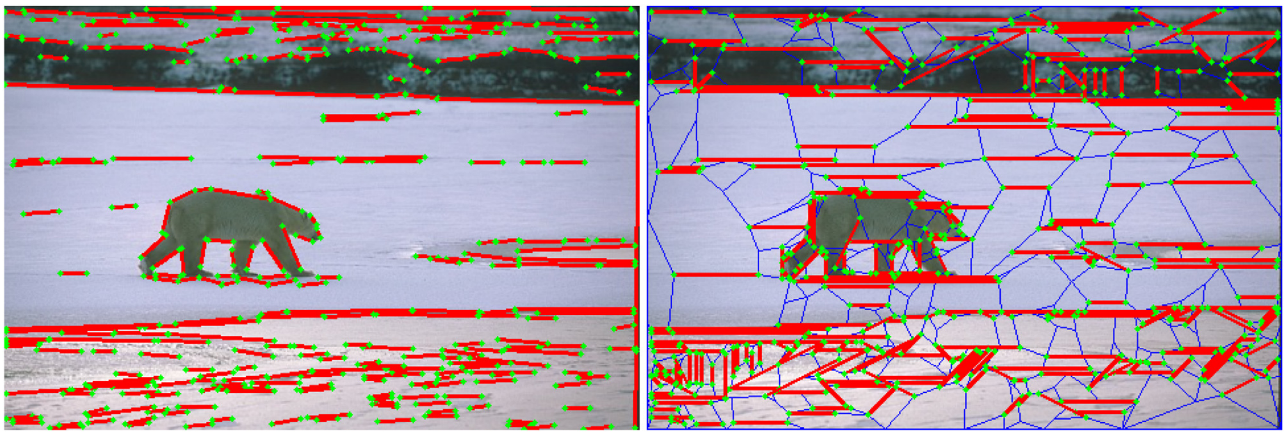

Fig. 7. **Left**: 269 LSDA edges with 32 intersections on image 100,007 in BSD. **Right**: PRIM with 269 PLSD edges without intersections.

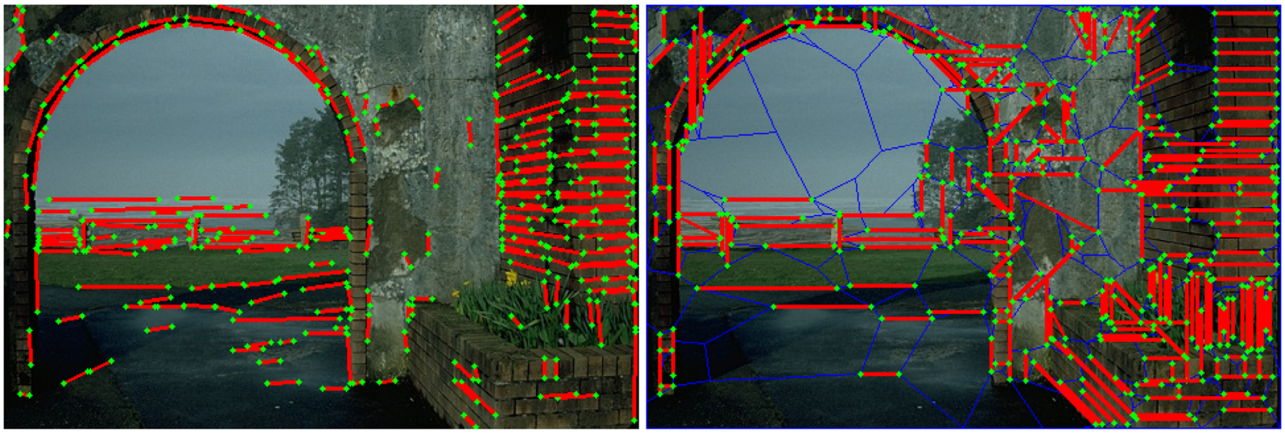

Fig. 8. **Left**: 230 LSDA edges with 1 intersection on image 5096 in BSD. **Right**: PRIM with 230 PLSD edges without intersections.

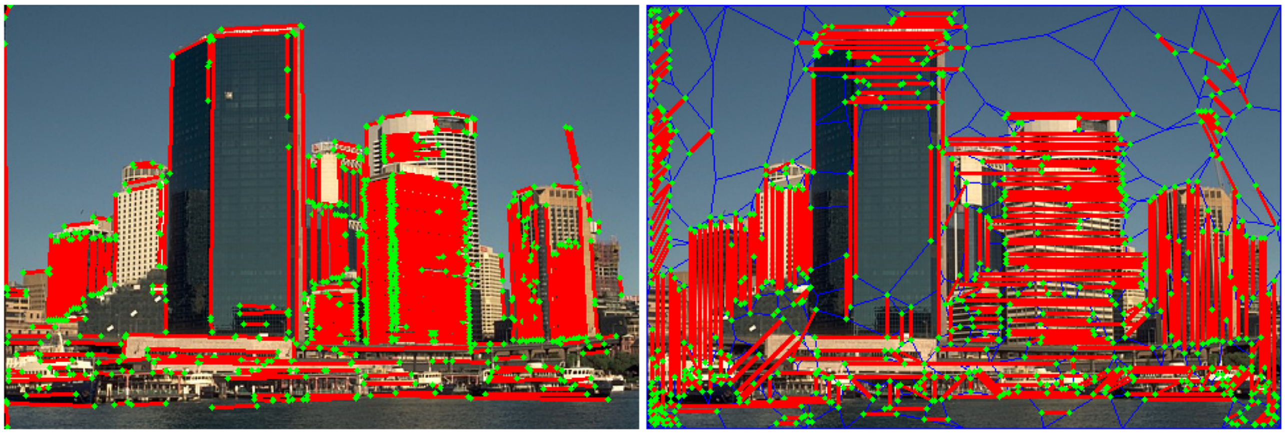

Fig. 9. **Left**: 359 LSDA edges with 86 intersections on image 69,007 in BSD. **Right**: PRIM with 359 PLSD edges without intersections.

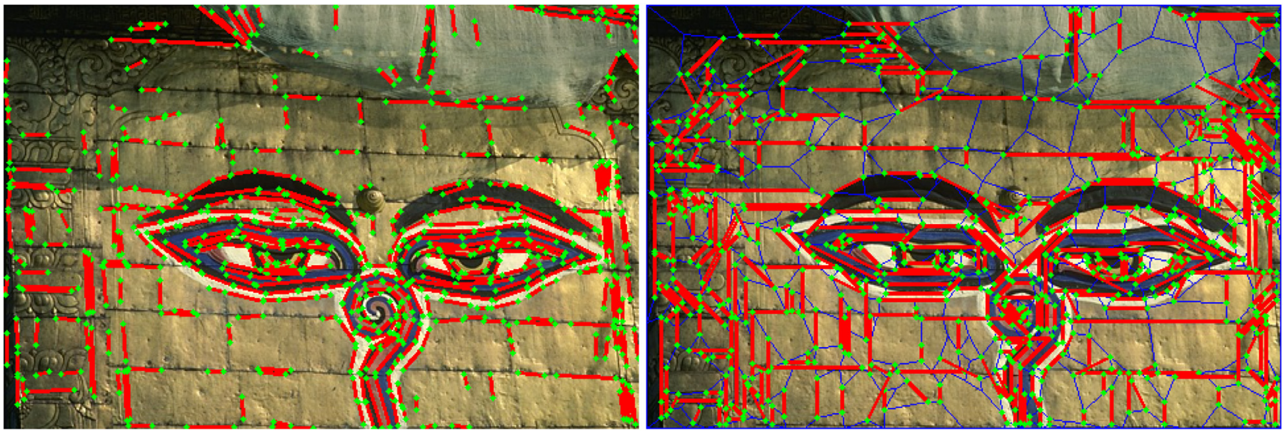

Fig. 10. **Left**: 424 LSDA edges with 5 intersections on image 56,028 in BSD. **Right**: PRIM with 424 PLSD edges without intersections.

- The PLSD allows a user to choose a number of required line segments, which are ranked by their strength.
- A thresholding of contrast values was avoided due to the new data-driven method motivated by a multi-scale approach of Topological Data Analysis.
- All PLSD line segments have no intersections by Definition 4, which has allowed us to extend PLSD edges to the PRIM mesh of convex polygonal superpixels.
- The PLSD and PRIM algorithms have a near linear time in the number of pixels by Theorem 5 and Corollary 7.
- Table 1 shows that the PRIM algorithm has further improved the PLSD performance in comparison with the LSDA on the Boundary Recall for BSD500.

Other possible improvements are better filtering, e.g. optimizing the size and sigma in the Gaussian kernel, and more advanced denoising before Step (4.1a). The current non-optimized code runs for about 1 s per BSD image on a laptop with 8 Gb 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

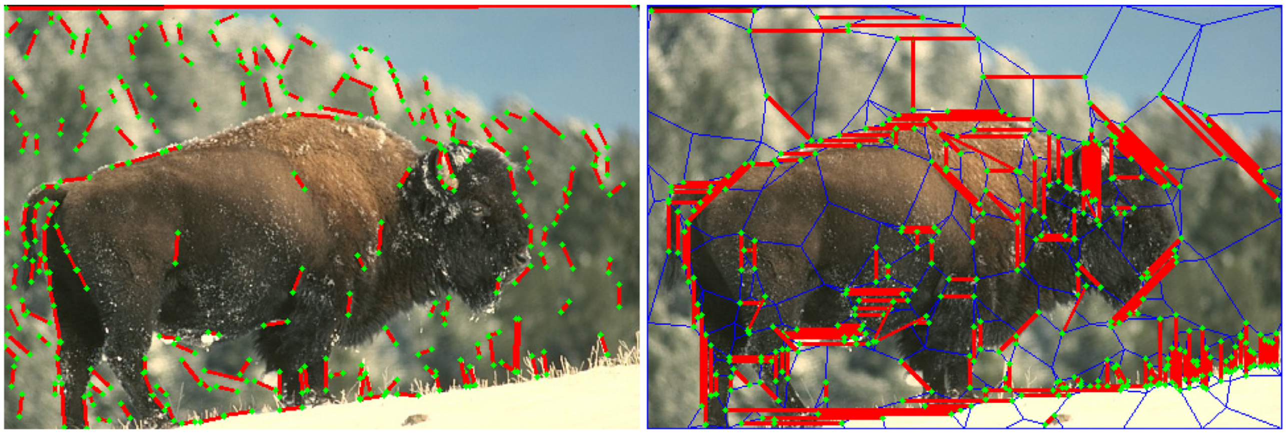

Fig. 11. **Left**: 193 LSDA edges with 48 intersections on image 41,029 in BSD. **Right**: PRIM with 193 PLSD edges without intersections.

neural networks on straight line sketches can be much faster than on original images.

The PLSD/PRIM algorithms have provisional C++ codes at http://kurlin.org/polygonal-meshes/PLSD.zip. This work was supported by the EPSRC grant EP/R018472/1 "Application-driven Topological Data Analysis".

We thank all reviewers for their helpful suggestions.

Declaration of Competing Interest

There is no known conflict of interest.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, J. Malik, Contour detection and hierarchical image segmentation, Trans. PAMI 33 (2011) 898–916.
[2] D. Ballard, Generalizing the hough transform to detect arbitrary shapes, Pattern Recognit. 13 (1981) 111–122.
[3] J. Canny, A computational approach to edge detection, Trans. PAMI 8 (1986) 679–698.
[4] A. Chernov, V. Kurlin, Reconstructing persistent graph structures from noisy images, Image A 3 (2013) 19–22.
[5] D. Cohen-Steiner, H. Edelsbrunner, J. Harer, Stability of persistence diagrams, Discrete Comp. Geome. 37 (2007) 103–130.
[6] L. Duan, F. Lafarge, Image partitioning into convex polygons, in: Proceedings of CVPR, 2015, pp. 3119–3127.
[7] H. Edelsbrunner, J. Harer, Computational Topology. An Introduction, AMS, Providence, 2010.
[8] J. Forsythe, V. Kurlin, Convex constrained meshes for superpixel segmentations of images, J. Electron. Imaging 26(6) (061609) (2017).
[9] J. Forsythe, V. Kurlin, A. Fitzgibbon, Resolution-independent superpixels based on convex constrained meshes, in: Proceedings of ISVC, 2016, pp. 223–233.
[10] R. Grompone von Gioi, A Contrario Line Segment Detector, Briefs in Computer Vision, Springer, 2014.
[11] P. Kahn, L. Kitchen, E. Riseman, A fast line finder for vision-guided robot navigation, Trans. PAMI 12 (1990) 1098–1102.
[12] S. Kalisnik, V. Kurlin, D. Lesnik, A higher-dimensional homologically persistent skeleton, Adv. Appl. Math. 102 (2019) 113–142.
[13] V. Kurlin, 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, IEEE, 2014, pp. 594–601.
[14] V. Kurlin, A homologically persistent skeleton is a fast and robust descriptor of interest points in 2D images, in: LNCS, Proc. CAIP: Comp. Analysis of Images and Patterns, 9256, 2015, pp. 606–617.
[15] V. Kurlin, A one-dimensional homologically persistent skeleton of a point cloud in any metric space, Comput. Graph. Forum 34 (2015) 253–262.
[16] V. Kurlin, A fast persistence-based segmentation of 2D clouds with provable guarantees, Pattern Recognit. Lett. 83 (2016) 3–12.
[17] V. Kurlin, D. Harvey, Superpixels optimized by color and shape, in: Proceedings of EMMCVPR 2017, Lecture Notes in Computer Science, 10746, Springer, 2018, pp. 297–311.
[18] V. Kurlin, G. Muszynski, A persistence-based approach to automatic detection of line segments in images, in: Proc. CTIC: Comp. Topology in Image Context, Springer, 2019, pp. 137–150.
[19] J.R. Shewchuk, Delaunay refinement algorithms for triangular mesh generation, Comp. Geome. 22 (2002) 21–74.
[20] P. Smith, V. Kurlin, Resolution-independent meshes of superpixels, in: Proceedings of International Symposium on Visual Computing, 2019, pp. 194–205.
[21] F. Viola, A. Fitzgibbon, R. Cipolla, A unifying resolution-independent formulation for early vision, in: Proceedings of CVPR, 2012, pp. 494–501.