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

Vitaliy Kurlin, Grzegorz Muszynski
Materials Innovation Factory (MIF) and Computer Science, University of Liverpool
Problem: line segment detection

By a *segment* we mean not a 2D region in a pixel-based image, but a straight line segment separating different objects in an image.

*Edge detection* usually means detecting ”edge pixels” from boundary contours of objects.

Our problem at subpixel resolution: given a 2D image, detect a required number of straight line segments that do not intersect each other.
State-of-the-art: LSDA

The Line Segment Detection Algorithm by Grompone von Gioi et al. (2014) outputs lines.

Plus: line segments can have any directions.
LSDA: intersected segments

Minuses: can’t choose a number of segments, many intersections of edges and tiny angles.

The LSDA output above has 48 intersections.
Motivations to improve LSDA

Cameras now produce huge images \( \geq 16\text{Mp} \).

Can we *smartly* reduce the number of pixels without losing the original quality?

Line segments can be used to build a skeleton and a polygonal mesh on image.
Over-segmentation problem

Split a large image into a smaller number of superpixels. Past work: a superpixel is a union of square pixels at a given resolution, often disconnected irregular shapes with holes.

If instead of $n = 1M$ pixels we produce 10K (1%) good superpixels, time $O(n^2)$ will be $10^4$ faster.
Polygonal superpixels

New approach: a mesh of convex polygons minimising the exact reconstruction error =

\[ \sum_{\text{pixels}} \left( \text{Intensity} - \sum \text{Area}(\text{Polygon} \cap \text{Pixel}) \text{Colour} \right)^2. \]

The key advantages are an objective evaluation, subpixel straight edges, nice convex superpixels.
Approximation by superpixels

Replace a large grid of pixels by a smaller graph of superpixels, usually unions of square pixels.

The reconstructed image has constant colours over superpixels: VK, D.Harvey. EMMCVPR’17. Superpixels Optimized by Color and Shape.
Past vs polygonal superpixels

Left: past pixel-based superpixels by clustering.

Right: polygonal superpixels with best colors.
Cloud $\rightarrow$ skeleton $\rightarrow$ superpixels

A careful refinement of LSDA output was done in J Electronic Imaging (2017) with J. Forsythe.

**Left**: edge points detected at subpixel resolution.

**Middle**: a simplified initial skeleton of a cloud $C$.

**Right**: Convex Constrained Mesh of superpixels.
Convex Constrained Meshes
Better line segments are needed

We avoid intersections and output a desired number of segments, not possible by LSD.
Only 8 fixed directions

To avoid tiny angles, we consider potential lines only along the 8 directions, which are usually enough to approximate real boundaries.
Stage 1a: image gradient

A given image $I$ is a matrix $w \times h$. The image gradient $DI = (g_x * I, g_y * I)$ is obtained from the masks $g_x = \begin{bmatrix} -1 & +1 \\ -1 & +1 \end{bmatrix}$, $g_y = \begin{bmatrix} +1 & +1 \\ -1 & -1 \end{bmatrix}$ for the image derivatives in the $x, y$-direction.

Fix one of 8 directions $(d_x, d_y)$. For the image domain $\Omega = [0, w] \times [0, h]$, fix a straight line $L$ given by $d_x x + d_y y + t = 0$ that intersects the rectangle $\Omega$ for a variable parameter $t \in \mathbb{Z}$. 
Stage 1b: contrast function $f_L$

The fixed straight line $L$ in the rectangle $\Omega = [0, w] \times [0, h]$ goes through points that have integer coordinates and RGB values of $I$.

After taking the norm of each RGB color vector, $||(R, G, B)|| = \max\{|R|, |G|, |B|\}$, we get a discrete sampled scalar contrast function $f_L : \mathbb{R} \rightarrow \mathbb{R}$ along the fixed line $L \subset \mathbb{R}^2$.

Big changes of contrast along a line $L$ (edge pixels) are local maxima of the contrast function.
Stage 2a: strength of a segments

For the function $f_L : \mathbb{R} \to \mathbb{R}$, a line segment $S \subset L \cap \Omega$ has the strength $|S| = \int_S f_L dp = \text{the sum of contrast values over } S = \text{yellow area below } f_L \text{ (not the persistence = birth–death)}$. 
Stage 2b: strength computation

The strengths are computed from high to lower values of $f_L$. Segments grow around local maxima and merge. After each merger, the ordered strengths $|S_1|$, $|S_2|$, $|S_3|$, $|S_4|$ are updated.
Stage 2c: strongest segments

The strongest segments have strengths $|S_i|$ above the widest gap among ordered strengths.

New strongest segments can be added to the growing set of all strongest segments after a merger when the contrast level $u$ of $f_L$ drops.
Stage 3: avoid intersections

After ordering all strongest segments by strength, output a desired number of segments:

choose a highest strength segment $S$, then remove all other strongest segments that intersect a small offset of the line segment $S$. 
Running time complexity

**Thm:** 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.

**Left:** LSDA segments. **Right:** PLSD segments.
BSD Boundary Recall

BSD is the Berkeley Segmentation Database, where every of 500 images has up to 7 hand-drawn boundaries $G$ (sets of pixels).

To evaluate a set of line segments $S$, compare the ground-truth boundary pixels $G$ by this Boundary Recall benchmark (for $\varepsilon = 2$ pixels)

$$BR(G, S, \varepsilon) = \frac{\# \{ \text{pixels } p \in G : d(p, S) \leq \varepsilon \}}{|G|},$$

where $d(p, S)$ is the Euclidean distance from a pixel $p$ to a closest line segment in the output $S$. 
Evaluation on Boundary Recall

BR measures the ratio of detected ground-truth pixels within $\varepsilon = 2$ pixels from output segments.

Black dot: LSDA mean over all BSD 500 images.

Blue line: our PLSD, segment offset $=3,4,5,6$. 
More comparisons with LSD
Summary: line segment detector

Key guarantee: all line segments are disjoint.

Parameter: a number of line segments (more control for building a future skeleton or mesh).

We conjecture stability of line segments under suitable noise in images due to persistence.

The new PLSD method to detect straight line segments consists of 3 independent stages, which can be separately improved. C++ code: github.com/muszyna25/Image_Experiments.