LineFit: A Geometric Approach for Fitting Line Segments in Images

Marion Boyer, David Youssefi, Florent Lafarge

LineFit: A Geometric Approach for Fitting Line Segments in Images

Marion Boyer[1,2], David Youssefi[2], and Florent Lafarge[1]

[1] Inria Centre Université Côte d'Azur, France
[2] Centre National d'Etudes Spatiales (CNES), France
Firstname.Lastname@{inria,cnes}.fr

Abstract. We present LineFit, an algorithm that fits line segments from a predicted image gradient map. While existing detectors aim at capturing line segments on line-like structures, our algorithm also seeks to approximate curved shapes. This particularity is interesting for addressing vectorization problems with edge-based representations, after connecting the detected line segments. Our algorithm measures and optimizes the quality of a line segment configuration globally as a point-to-line fitting problem. The quality of configurations is measured through the local fitting error, the completeness over the image gradient map and the capacity to preserve geometric regularities. A key ingredient of our work is an efficient and scalable exploration mechanism that refines an initial configuration by ordered sequences of geometric operations. We show the potential of our algorithm when combined with recent deep image gradient predictions and its competitiveness against existing detectors on different datasets, especially when scenes contain curved objects. We also demonstrate the benefit of our algorithm for polygonalizing objects.

Keywords: line segment detection · shape approximation · Object polygonalization

1 Introduction

Line segments are commonly used to capture image discontinuities with a compact, resolution-independent representation. They constitute a powerful local descriptor to detect vanishing points [41,43], create scene abstractions [18], polygonalize objects [25,39] or address 3D vision tasks such as pose estimation [13,45], Structure-from-Motion [29,31] or SLAM reconstruction [16,35].

Existing detectors are mostly designed to capture line segments on the line-like structures typically found in man-made scenes. This specialization allows, for instance, the reconstruction of indoor environments with wireframes [19,30, 48,52,53] or the stable matching of line segments repeated in multiview images for 3D vision tasks [2,34]. However, by discarding line segments to also capture curved contours, the use of these detectors for vectorization tasks is limited to a small range of objects, mostly buildings. Designing a more general detector that describes both line-like structures and freeform shapes requires us to redefine

the detection problem into an approximation one. While accuracy is the main aim when detecting line segments on line-like structures, complexity, via the number of line segments, becomes equally important on a curved shape as its approximation relies upon a complexity-distortion tradeoff.

To address this issue, we propose LineFit, an algorithm that fits line segments from a predicted image gradient map. Formulated as a point-to-line fitting problem, we use an energy minimization framework to define and search for *good* configurations of line segments in the large solution space. Inspired by planar shape detection techniques from 3D point clouds, we introduce a fitting tolerance parameter that both absorbs the noise in the image gradient map, and controls the level of approximation on curved shapes. The quality of configurations is measured through the local fitting error, but not only; completeness of line segments over the image gradient map and regularity of the configurations are also involved. The latter refers to a low number of line segments with high geometric regularities between them, i.e. co-linearity, concurrence to vanishing points for perspective projection images, or parallelism and orthogonality for orthographic projection images. A key ingredient of our work is an efficient and scalable exploration mechanism that refines an initial configuration by ordered sequences of geometric operations. Inspired by geometry processing techniques, this mechanism alternates global refinements that improve precision and regularity, and local modifications ordered to favor rapid energy decreases.

We show the potential of our algorithm when combined with recent deep image gradient predictions and its competitiveness against existing detectors on different types of datasets. In particular, our algorithm outperforms competitors on the *BSDS500* dataset [6] containing curved objects while remaining competitive on datasets composed of line-like structures only, including on *RoofSat* [1], a new dataset for representing roofs as wireframes in satellite images. We also demonstrate the benefit of our algorithm for polygonalizing objects.

2 Related work

Traditional mechanisms. They typically group alignments of steep gradients in images through iterative procedures. LSD [15] uses region growing to group pixels before approximating each region by a rectangle and testing whether a line segment is inside. Several variants of this popular algorithm were proposed by enforcing regions to grow towards the direction orthogonal to the image gradients [4], relaxing the connectivity condition to avoid sequences of small co-linear line segments [40], adding aggregation rules on line geometry to reduce false positive detections [51] or operating the aggregation from small atomic line segments [9]. MCMLSD [5] generates line hypotheses through the Hough transform and then selects some of them as line segments using a Markov Chain formulation. These algorithms are usually fast but rely upon local heuristics where accuracy of line segments is the main aim. As a consequence, results often lack global consistency and are imprecise in the presence of curved shapes.

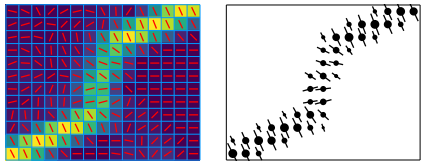
Neural networks. Numerous architectures have been proposed during the last years using for instance attraction fields [47] and transformers [46]. Many of them target a good trade-off between accuracy and processing time for realtime applications [11, 17, 20, 50]. These networks often do not generalize well as datasets with line segment Ground Truths are rare and specific to certain types of scenes, e.g. indoor scenes [19]. SOLD2 [34] and L2D2 [2] propose self-supervised training schemes to address this issue by generating ground truths with homographic transformations between pairs of images and Lidar scans respectively. Recently, hybrid methods propose to replace the often imprecise regression of line segments in end-to-end architectures with traditional fitting mechanisms. DeepLSD [33] predicts a surrogate image gradient map computed from attraction fields and fits line segments on it with LSD [15]. LSDNet [42] uses a similar fitting strategy but from a lightweight CNN, which predicts a line occupancy map and tangent field. Their competitiveness inspired us to follow the same strategy, but with the objective of fitting line segments not just on line-like structures. We depart from the best deep image gradient maps and fit line segments to them, targeting globally consistent approximations of curved objects also.

Wireframe and regularity-aware methods Some works aim to connect line segments to form wireframes, a relevant representation of indoor scenes with a good 3D interpretation potential [24]. Deep models typically predict both junctions and lines before matching them [19, 48, 52, 53]. LGNN [30] detects lines using CNNs and computes the connectivity using GNNs. DPP [14] exploits a dynamic Delaunay triangulation with a costly point sampling strategy. Another class of methods enforces geometric regularities between line segments. Hough transform-based algorithms [27] can capture co-linearity by construction. Post-processing based on energy minimization also allows the enforcement of concurrence to pre-detected vanishing points for perspective projection images [33] and parallelism and orthogonality for orthographic projection images [8]. Our algorithm also enforces these regularities, but during detection.

3 Algorithm

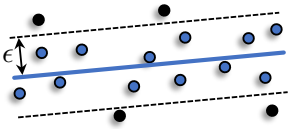
3.1 Problem formulation.

The algorithm takes as input an image gradient map that encodes both for magnitude and orientation. Such maps can be computed by traditional methods or, more interestingly by recent deep learning models. Instead of operating on the grid-based domain of the image, we convert the gradient map into a sparse 2D-point cloud in which only points centered on high magnitude pixels are retained. As illustrated in the inset, each point (right) is weighted by the gradient magnitude of its associated pixel (left), where the bigger the point, the greater the gradient magnitude. Point normals (black lines, right) are aligned to the gradient orientation (red lines, left). This conversion

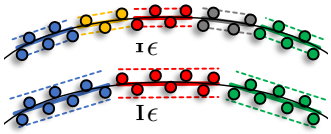


allows us to (i) exploit the sparsity of image discontinuities efficiently and (ii) formulate the problem as a point-to-line fitting problem in a similar way to planar primitive detection algorithms for 3D point clouds [21, 32, 38, 49].

The latter are known to be robust to noise and outliers thanks to a fitting tolerance ϵ that specifies whether a point is close enough to the planar primitive to be fitted to it. As illustrated in the inset, this tolerance defines a zone around the primitive that absorbs the noise contained in the point cloud. We exploit this formalism in our problem. Similarly to Ransac-based model fitting techniques [36, 37], we call an *inlier*, represented as blue dot in the inset, (respectively *outlier*, black dot), an input point which falls in the ϵ-fitting tolerance zone of a line segment (respectively falls in no fitting tolerance zone of the line segments).



Interestingly, the fitting tolerance ϵ can be used to control the level of approximation on curved shapes. This is illustrated in the inset where a small tolerance leads to a finer approximation (top), each color refering to a line segment. We also define a minimal line segment size σ that discards line segments with too few inliers. The pair (ϵ, σ) are the two user-specified parameters defining the level of detail expected in the output. As illustrated in Figure 1, thin elements will be captured by one centered line segment if the value of parameter ϵ is at less half of the element width.



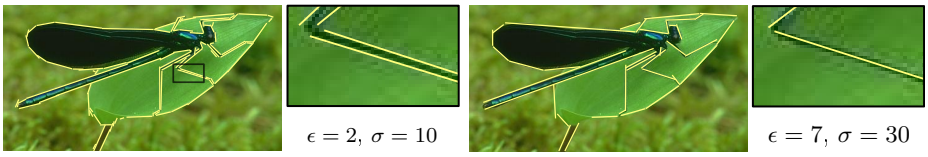


Fig. 1: Line segment detection with different levels of details. Choosing low values for ϵ and σ produces a detailed configuration of 95 line segments where the curved parts are approximated by many small, accurate line segments (left). High values give a coarser configuration of 37 line segments (right).

The output line segments are represented as clusters of inlier points each associated with a supporting line, i.e. the best line fitted to its inlier points in a weighted least squares sense. The two extremities of a line segment are then the two furthest projections of inliers on the supporting line. We denote a configuration of line segments by $\mathbf{x} = (\mathbf{s}, \mathbf{l})$ where $\mathbf{s}$ is a set of 2D lines parametrized in the continuous domain, and $\mathbf{l}$, a set of labels that indicates whether input points are outliers or inliers to one supporting line of $\mathbf{s}$.

3.2 Quality of a line segment configuration

In contrast to existing traditional mechanisms, we explicitly define and measure the quality of a line segment configuration. We argue that the notion of *good* configuration is a tradeoff between three objectives: **(i) fidelity** (the inlier

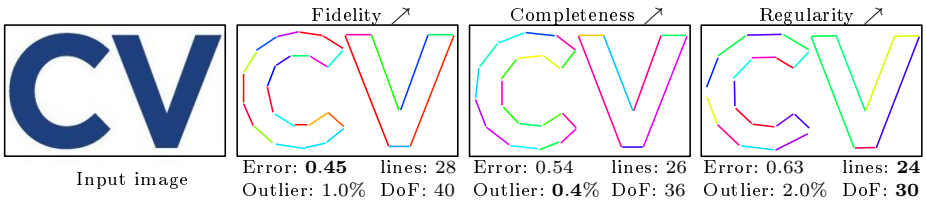


Fig. 2: Impact of energy terms. Giving more importance to the fidelity term favors a low fitting error, the completeness term a low number of outliers, and the regularity term a low number of line segments and degrees of freedom. The impact is visible on the curved parts, i.e. on the letter "C".

points must be close to their associated supporting line), **(ii) completeness** (the number of outliers must be low), and **(iii) regularity** (the number of line segments must be low and line segments must preserve geometric regularities, if any). We measure the quality of a line segment configuration $\mathbf{x}$ by the energy U that encodes these three objectives:

$$U(\mathbf{x}) = \lambda_f U_f(\mathbf{x}) + \lambda_c U_c(\mathbf{x}) + \lambda_r U_r(\mathbf{x}) \tag{1}$$

where terms for fidelity U_f, completeness U_c and regularity U_r are defined in the interval $[0, 1]$. λ_f, λ_c and λ_r are weights in the interval $[0, 1]$ balancing the three terms so that $\lambda_f + \lambda_c + \lambda_r = 1$.

Fidelity term U_f corresponds to the local fitting error and is defined as the mean weighted distance between inliers and their associated supporting line:

$$U_f(\mathbf{x}) = \frac{1}{w_{\mathbf{x}}} \sum_{s \in \mathbf{s}} \sum_{i \in s} w_i D(i, s) \tag{2}$$

where w_i is the magnitude weight of the inlier point i associated to the line s, $w_{\mathbf{x}}$, the accumulated weight from all inlier points of the configuration $\mathbf{x}$, and $D(i, s)$, a normalized distance between the inlier point i and its associated supporting line s. Several choices are possible for the latter, e.g. the orthogonal distance of point i to line s normalized by ϵ. This choice is discussed later in Section 4.

Completeness term U_c encourages a high ratio of weighted inliers by

$$U_c(\mathbf{x}) = 1 - \frac{w_{\mathbf{x}}}{w_T} \tag{3}$$

where w_T is the accumulated weight from all input points.

Regularity term U_r favors configurations with few, regularized line segments. Regularity is measured by counting the degrees of freedom between lines. A 2D line has two degrees of freedom, e.g. an orientation angle and an orthogonal distance to origin. Two parallel, orthogonal or concurrent lines have three degrees of freedom in total. Two co-linear lines have two only. We formulate this as:

$$U_r(\mathbf{x}) = k_{\mathbf{x}} \times \left(\frac{2n}{\sigma} \right)^{-1} \tag{4}$$

where $k_{\mathbf{x}}$ denotes the degrees of freedom in the configuration $\mathbf{x}$. This number is divided by the maximal number of degrees of freedom computed as the ratio of twice the number of input points n to the minimal number of inliers required per line segment σ. Regularities include co-linearity and either parallelism and orthogonality for orthographic projection images, or concurrence to vanishing points for perspective projection images. For the latter, vanishing points are detected using [7]. Note that this term also favors a low number of line segments.

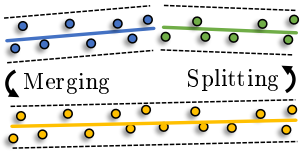
Figure 2 illustrates the impact of the three energy terms. None of the three presented configurations can be considered as better than the others as they each perform best on one of the objectives. In our experiments, we give the same importance to each objective with $\lambda_f = \lambda_c = \lambda_r = 1/3$.

3.3 Optimization

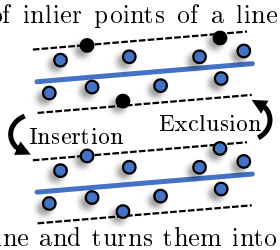
While energy U is a simple and natural expression of the three objectives of quality, its minimization is a difficult task as i) solutions live in a large mixed discrete-and-continuous space and ii) U is non-convex. We propose an exploration mechanism that iteratively operates geometric modifications on the current configuration. It alternates between sequences of local modifications that are ordered to favor rapid energy decreases first and global modifications that refine the precision of line segments and regularities existing between them. A pseudo-code of our exploration mechanism is presented in Algorithm 1.

Local operators. They aim to improve locally the configuration by modifying one or two line segments only. Two types of local operators are considered: one for modifying the number of line segments, and the other, the proportion of outliers.

Merging and splitting operators either combine two adjacent line segments into one, or conversely, divide one line segment into two. Two line segments are considered as adjacent if at least a pair of their respective inlier points are connected in the k-nearest neighbor graph of the input point cloud. The merging operator consists in grouping inlier points of the two adjacent line segments, assigning them to the new line segment and computing the new supporting line as the best weighted least square fitting line of these inlier points. The splitting operator performs a K-means clustering (with K=2) of the joint set of inlier points using the metrics of the fidelity term, i.e. D in Equation 2, and by computing centroids as the optimal weighted least square line of each subset of inlier points.



Insertion and exclusion operators modify the set of inlier points of a line segment with the insertion of outlier points as new inliers and conversely, the exclusion of inliers as new outliers. As shown in the inset, the exclusion operator simply rejects as outliers the k farthest inlier points from the supporting line and recomputes the latter by weighed least square fitting. The insertion operator selects the k closest outlier points from the supporting line and turns them into



inlier points if their distance to the new optimal supporting line is lower than ϵ. These two operators are effective only when no other inliers are rejected from the tolerance zone. In practice, k is fixed to 3.

Global operators. Their role consists in improving either the precision or the regularity of the line segments at the scale of the entire configuration.

The transfer operator exchanges inlier points between the adjacent line segments with the objective of reducing the fitting error without altering regularity and completeness. Similarly to [10], we alternate between clustering of inliers and line fitting on clusters of inliers in an Expectation-Minimization manner. The clustering is performed by K-Means with K chosen as the number of line segments in the current configuration and as metrics, the fitting error defined in Equation 2. Line fitting is done by weighted least square fitting of lines to the clusters of inlier points. To speed-up the clustering, inlier exchanges are allowed where two line segments meet only. In practice, only inliers connected to inliers of a different line segment in the k-nearest neighbor graph are exchangeable.

The regularity operator detects geometric regularities in the current configuration, and enforces the supporting lines to preserve them. Two cases are considered. For perspective projection images, supporting lines that are near-concurrent to a pre-detected vanishing point, i.e. in practice when the orthogonal distance from the vanishing point to the supporting line is below ϵ, are reoriented around its center of mass to pass through the vanishing point. Line segments concurrent to a similar vanishing point are then adjusted to be exactly co-linear if their relative orientation differs by less than 5 degrees, and the mean orthogonal distance from their center of mass to other supporting line, by less than $\epsilon/2$. This adjustement consists in replacing their respective orientation by the mean orientation while imposing they still pass through the vanishing point. Processed in a similar way, the case for orthographic projection images is detailed in supplementary material.

Algorithm 1 Pseudo-code of LineFit

```
Initialize the line segment configuration x
repeat
    Initialize the priority queue Q
    while top operation i of Q decreases energy U do
        Update x by operation i
        Update Q
    end while
    Update x by the global transfer operator
    Update x by the global regularity operator
until stopping condition is valid
```

Operation scheduling. Each geometric operator has a specific role in the exploration of the solution space. Local operators propose variations regarding the

number of line segments and the ratio inliers to outliers. They are called sequentially within a priority queue that sorts the energy variations of all possible local operations in ascending order, i.e. operations that best improve the quality of the configuration first. The queue is updated after performing the operation at its top. Because operators are local and only affect one or two line segments, each update is fast to compute in practice. The sequence of operations stops when the energy variation on top of the queue becomes positive. After such a sequence of local operators, the two global operators are called to refine the precision and regularity of the configuration. The regularity operator is called after the transfer one to not break the regularized line segments. This complete cycle is repeated until reaching convergence.

Initialization and stopping condition. Because the exploration mechanism finds a local minimum, a good initial configuration is expected in practice. We typically depart from configurations returned by LSD [15] for images with line-like structures, and from denser configurations returned by the aggregation-free region growing of the CGAL library [32] for images with curved shapes so that initial line-segments are more homogeneously distributed on high gradients. The exploration stops when the energy does not evolve after two successive cycles, or when 10 cycles have been completed.

4 Experiments

Implementation details. Our algorithm has been implemented in C++ using the CGAL library for the manipulation of geometric data structures and least squares fitting operations. In our experiments, distance $D(i, s)$ of Equation 2 was defined as the orthogonal distance of point i to line s if the point normal and the orthogonal vector of the line deviate by less than 45 degrees, and $+\infty$ otherwise. This choice gives more importance to the gradient magnitude, gradient orientation only avoiding the fitting when the angle deviation is too large.

Datasets and evaluation metrics. We evaluated our algorithm on three datasets: *BSDS500* [6], *YorkUrban* [12] and the new dataset *RoofSat* [1]. They are respectively composed of 100 test images mostly containing curved objects, 102 test images from both indoor and outdoor scenes in a perspective projection geometry, and 550 satellite images of urban scenes at nadir in an orthographic projection geometry. Line-like structures in the latter correspond to both contours and roof skeletons of buildings.

To measure fitting accuracy and completeness, we adopt a strict Average Precision (AP) and Average Recall (AR). They are computed between the pixelized output line segments and the pixelized Ground Truth line segments of *YorkUrban* and *RoofSat*, or the Ground Truth object boundaries of *BSDS500*. We also consider the Average F-score (AF) to represent the tradeoff between the two. Complexity and regularity are measured by the average number of line segments (#lines) and the Degree of Freedom score (DoF) defined as the average ratio

	YorkUrban						*RoofSat*					
	ELSED	HAWP	LETR	DeepLSD	Ours	Ours*	ELSED	HAWP	LETR	DeepLSD	Ours	Ours*
AP(↑)	48.2	39.2	41.3	51.0	**52.0**	<u>51.3</u>	39.1	<u>45.4</u>	21.3	**48.9**	42.1	41.8
AR(↑)	39.8	17.6	<u>40.4</u>	40.2	**41.5**	40.2	<u>38.2</u>	31.2	30.0	34.4	**39.0**	36.8
AF(↑)	42.7	23.5	39.7	<u>44.6</u>	**45.7**	44.5	35.2	35.4	18.2	36.3	**37.9**	<u>36.4</u>
DoF(↓)	82.8	90.5	83.9	<u>79.9</u>	81.1	**47.1**	<u>95.6</u>	96.8	96.1	96.6	96.2	**57.8**
#lines	374	227	357	354	388	414	282	227	228	196	297	288

Table 1: Quantitative comparisons on *YorkUrban* and *RoofSat*. The scores are expressed in percent, except for the average number of line segments. Ours and Ours* correspond to our algorithm without and with regularization respectively (i.e. without or with the activation of the global regularity operator of the exploration mechanism). Bold and underlined values indicate the best and second best scores respectively.

of the degree of freedom to twice the number of line segments where the lower is better. Detailed formulas of these scores and information about the *RoofSat* dataset are provided in supplementary material.

Methods. We compared our algorithm to the traditional mechanism ELSED [40], the learned wireframe detector HAWP [48], the transformer-based method LETR [46] and the hybrid approach DeepLSD [33]. We used the code released by the authors for ELSED and produced variations with the validation threshold ranging from 0.1 to 0.7. For deep learning architectures, we used version 3 of HAWP and the R101 version of LETR both pre-trained on the *Wireframe* dataset [19], and the author version of DeepLSD pre-trained on MegaDepth. We produced variations by thresholding on the confidence score between 0.05 and 0.4 for HAWP and between 0.2 and 0.6 for LETR. For DeepLSD, we varied the magnitude threshold between 1 and 4. These variation ranges were chosen to obtain a similar average number of line segments between the methods while respecting the setting recommendations of the authors.

Results on line-like structures. For experiments on *YorkUrban* and *RoofSat*, we used as image gradient map, the surrogate gradient image of DeepLSD [33] that offers a good robustness to challenging image conditions and generalizes well to datasets with line-like structures. We selected the high gradient pixels by thresholding the magnitude at 25% of the maximal magnitude of the image. This threshold is moderately low to include weak discontinuities without giving them more importance than the strong ones thanks to the magnitude weight of each point. The parameter pair (ϵ, σ) was fixed to 2 pixels and 20 inlier points respectively. To create a variation range similar to competitors, we computed a confidence score on each line segment as one minus the local fitting distance D of Equation 2 and varied it between 0.5 and 0.7. This allows us to produce results at a similar average level of complexity for all the methods, i.e. around 375 line segments on average per image for *YorkUrban* and 250 for *RoofSat*.

Table 1 presents the quantitative results on these two datasets. Considering completeness as a quality objective allows our method to achieve the best recall. Our method also offers a good precision, but not necessarily the best as

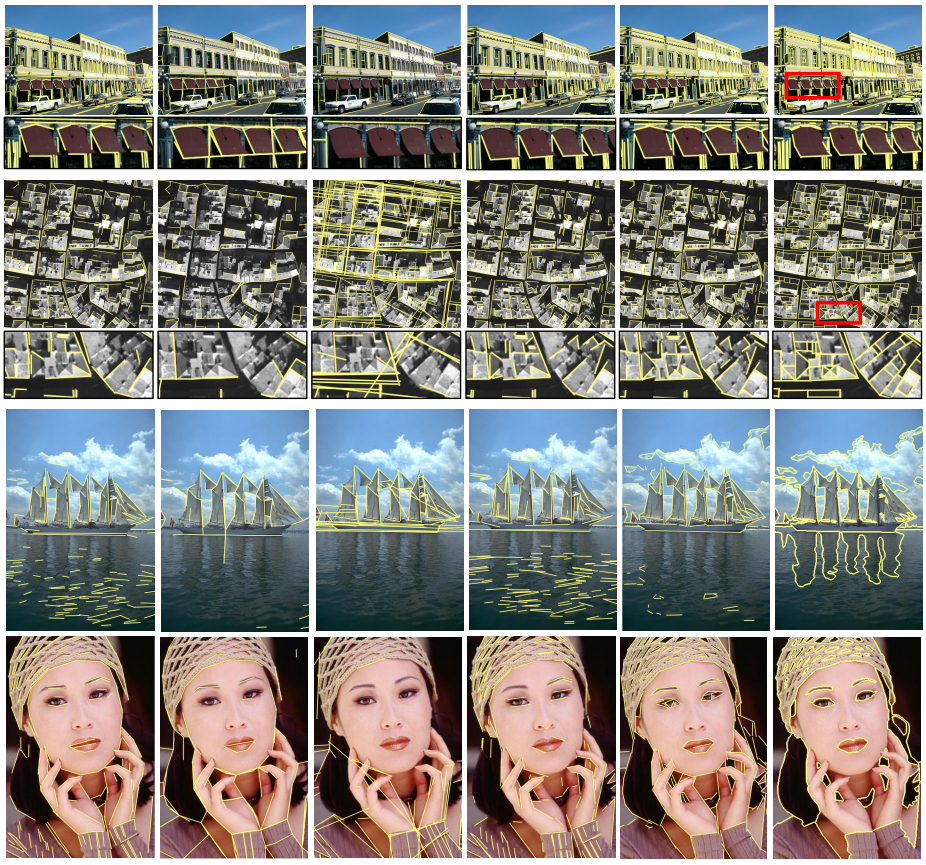


Fig. 3: Visual comparisons. ELSED, DeepLSD, and, to a lesser extent, HAWP and LETR, detect the line-like structures of the *YorkUrban* image (first row) and the *RoofSat* image (second row) with a good precision, but often miss relevant line segments (see close-ups). Our method produces more complete configurations. We perform best on the two *BSDS500* images (bottom rows) by more accurately and densely approximating the ground truth boundaries of curved objects with line segments.

DeepLSD and, to a lesser extent, HAWP perform better on *RoofSat*. Considering the tradeoff between precision and recall, our algorithm remains competitive with the best F-score on the two datasets. Activating regularities sightly degrades the precision and recall, but allows us to reduce the degree of freedom by around 60%. Overall, our algorithm offers the best compromise between the different quality metrics on line-like structures, but is not necessarily the best choice for applications requiring high precision only.

Results on curved objects. For experiments on *BSDS500*, we used as image gradient map, the contour probability map of SAM [22] that we threshold at 0.5

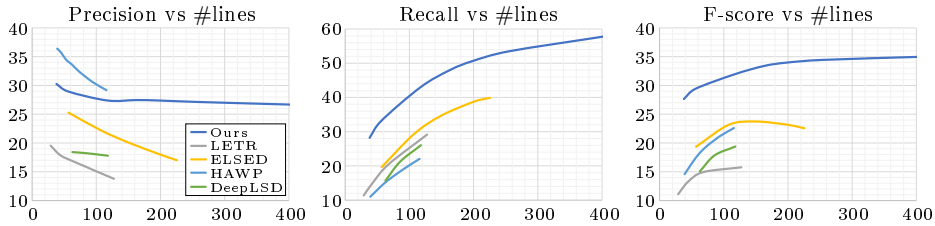


Fig. 4: Quantitative comparisons on *BSDS500*. Precision, Recall and F-score (vertical axis, expressed in percent) are plotted as a function of the number of line segments (horizontal axis). Note that, thanks to the fitting tolerance that controls the level of approximation on curved objects, only our method allows a large variation of the number of line segments.

to select our set of input points. Such a map captures quite well the silhouette of curved objects. We produced variations in the level of approximation of the output configurations through the parameter pair (ϵ, σ) that we made varied from $(3, 5)$ to $(10, 100)$. We also deactivated the regularity operator as enforcing orthogonality or concurrence to vanishing points is not relevant on curved shapes. The regularity term thus only favors a low number of line segments here.

Figure 4 shows how precision, recall and F-score evolve as a function of the number of line segments. While our method can produce configurations of line segments varying greatly from few to many, competitors (and especially the learning approaches) offer less flexibility as they seek fidelity and do not consider complexity as a quality objective. Our algorithm outclasses competitors on the three metrics by a significant margin. Only HAWP exhibits a better precision than ours at low complexity, but obtains the lowest recall among the methods. Figure 3 illustrates this gap where our method approximates more accurately and densely the silhouette of the curved objects. In particular, the sails of the boat and the hair of the lady are smoothly captured by sequences of line segments.

	Vanilla	(a) Basic gradient map	(b) No gradient orientation	(c) Coarser fitting	(d) Dense initialization	(e) Late stop	(f) Early stop
AP(↑)	41.8	40.2	40.5	45.2	38.3	41.9	42.0
AR(↑)	36.8	34.7	35.8	35.5	39.5	36.8	35.3
AF(↑)	36.4	34.7	35.2	37.0	36.4	36.4	35.6
DoF(↓)	57.8	58.6	57.9	54.7	56.0	57.7	60.9
#lines	288	292	285	233	367	288	300

Table 2: Ablation study. Alternative image gradient maps, fitting parametrization, initialization and optimization strategies are evaluated on the *RoofSat* dataset.

Ablation study. Table 2 shows how evaluation scores evolve on the *RoofSat* dataset when we alter the design of our algorithm. Replacing the deep image gradient map by one traditionally computed by finite differences (a) or not using the information on gradient orientation but only magnitude (b) degrade the

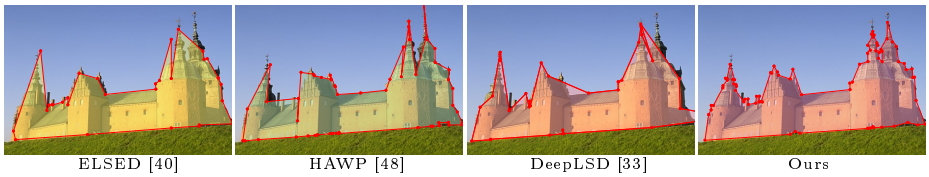
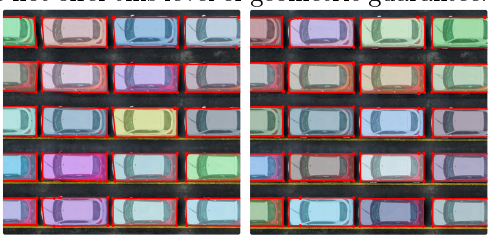


Fig. 5: Polygonalization from various line segment detectors. Combined with ASIP [25], our line-segments lead to a more accurate polygon than from the other detectors. These detectors allow the piecewise linear parts of the castle to be well reconstructed, but only ours allows a fine approximation of the curved roof details.

different evaluation scores by approximately one to two points. In contrast, fitting line segments at a coarser level of details by setting ϵ to 5 and σ to 40 (c) is relevant on this dataset for which image gradient maps are dense and noisy as a higher fitting tolerance allows us to better absorb noise. Initializing our algorithm by a denser configuration of line segments as done for *BSDS500* (d) gives a higher recall as the line segments are more homogeneously distributed on high gradients but degrades precision and complexity. We can also see that continuing the exploration mechanism for 20 cycles (e) does not bring any particular benefits to the quality of the results, while stopping after 2 cycles only (f) is too early. Finally, we replaced our exploration mechanism with a simulated annealing approach calling our geometric operators randomly for a sufficient number of runs. While the two optimization techniques converge towards a similar energy, i.e. with an average difference less than 10^{-3}, our exploration mechanism requires two orders of magnitude less time.

Application to object polygonalization. The ability of our algorithm to well capture curved shapes is particularly interesting for object polygonalization, by connecting detected line segments into polygons through either adjacency graph analysis [39] or polygonal partitioning [25]. We evaluated its potential on this task by combining it to ASIP [25]. The latter extends the line segments at constant speed in a kinetic framework to decompose the image domain into convex polygons that are then merged into larger polygons based on a semantic probability map. By construction, these ouput polygons are guaranteed to be closed, intersection-free and potentially nested, i.e. with polygonal holes inside. Interestingly, neural polygonizers such as PolyFormer [28], BoundaryTransform [23] or PolyTransform [26] or traditional vectorization pipelines that simplify pixel chains such as Douglas-Peucker [44] do not offer this level of geometric guarantee. In our experiments, we used the setting for the images with curved objects, i.e. a dense initialization, an image gradient map computed by SAM [22] and no regularization. We also chose the segmentation map of SAM as the input semantic probability map of ASIP, in coherence with



the image gradient map. As illustrated in the inset, activating regularization is mostly relevant on structured scenes to capture objects with simpler, regular polygons. Without regularization (left), each individual car is approximated by unspecified polygons with 4 to 6 edges. With regularization (right), cars are mostly captured by rectangles that align with the grid-based layout of the parking lot. Note that combining ASIP [25] with other detectors produces polygons that fail to correctly approximate the curved shapes, as illustrated in Figure 5.

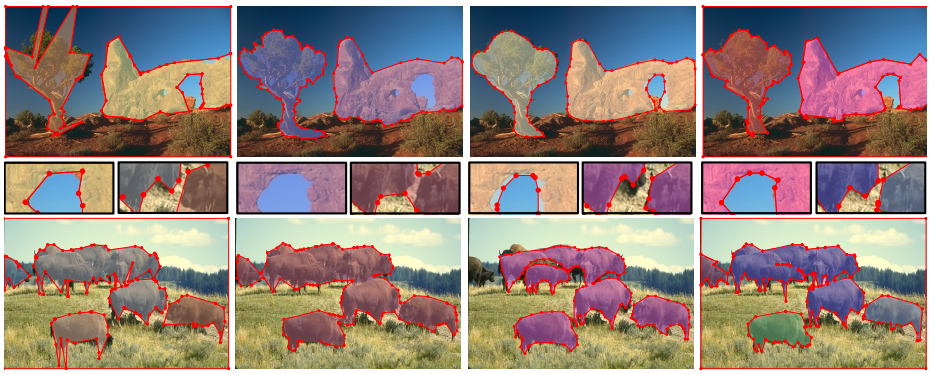


Fig. 6: Visual comparisons of object polygonalizers. For a similar level of complexity (around 50 edges per image), our polygons more accurately capture the object silhouettes than PolyFormer and better separate them than Poly-RNN++ (see closeups).

We evaluated this polygonalization pipeline on the *BSDS500* dataset. We used the standard pixel-based segmentation accuracy metrics, i.e. precision, recall and Intersection-over-Union (IoU), between the pixelized output polygons and the ground truth object masks, and measured how these metrics evolve with polygon complexity, i.e. through the number of edges. We compared it to the original ASIP method [25], which is based on the LSD detector [15], using the same semantic probability maps as ours, and the neural polygonizers Poly-RNN++ [3] and Polyformer [28]. To vary the complexity of the latter two methods, we simplified their output polygons using Douglas-Peucker [44].

Figure 7 shows how precision, recall and IoU evolve as a function of the number of edges. Our pipeline offers the best precision and IoU, except at a very low complexity (i.e. 25 edges on average per image) where it becomes too inaccurate to fit a few line segments on multiple complex shapes. Neural polygonizers followed by a Douglas-Peucker simplification are a more appropriate choice in this case. Our method outclasses the original ASIP by a large margin, as a consequence of the LSD detector [15] not performing well on curved shapes. Poly-RNN++ exhibits a high recall, mostly resulting from close objects that are often glued in a single, large polygon. This also gives a low precision, similarly to PolyFormer that tends to over-smooth object silhouettes and fails to capture fine details. Figure 6 presents some visual results obtained at a similar complexity.

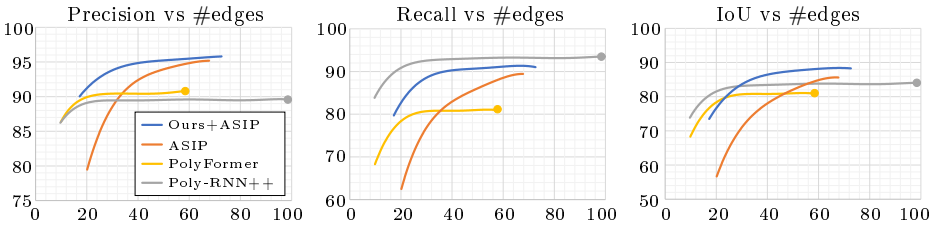


Fig. 7: Object polygonalization on *BSDS500*. Precision, Recall and IoU, which are expressed in percent, are plotted as a function of the number of polygon edges (horizontal axis). The Polygon-RNN++ [3] and Polyformer [28] results are represented by the grey and yellow dots respectively. They have been simplified by Douglas-Peucker [44] to lower polygon complexities (grey and yellow curves).

Limitations. Our fitting algorithm has a few shortcomings. First, the exploration mechanism is not globally optimal and requires a good initial configuration in practice. Fortunately, the initialization strategies used in our experiments rarely produce irrelevant initial configurations. Our algorithm is also not real time, even though images of 512*512 can be processed in less than one second. Standard images of a few millions pixels typically require a few seconds. Then, our algorithm cannot be embedded into end-to-end architectures. It can however be used to refine their results. Finally, for detecting specific line-like structures where a relevant training set exists, our algorithm is not as competitive as the deep learning methods trained on it. This is particularly true on the indoor images of the *Wireframe* dataset [19] where LETR produce accurate results.

5 Conclusion

We proposed a geometric reformulation of the line segment detection problem in which a line segment detected in an image does not necessarily refer to line-like structures, but can also approximate the local geometry of curved objects within a fitting tolerance. Our approach simultaneously seeks high fidelity, completeness and regularity of line segment configurations with an energy minimization framework. We showed its competitiveness against existing detectors on images with curved objects and, to a lesser extent, with line-like structures. We also showed its applicability to the object polygonalization task. Operating from an image gradient map, our method can be combined with recent deep learning approaches for the fitting step or simply used as post-processing for refining a result. In future work, we will generalize the approach to the fitting of parametric curves, typically Bézier curves which are popular representations to address Vector Graphics problems.

Acknowledgements. This work was supported by Airbus DS and the French Space Agency (CNES). The authors thank Laurent Gabet, Emmanuel Garcia and Roberto Dyke for the technical discussions.

References

1. The roofsat dataset. https://project.inria.fr/roofsat/ .
2. Abdellali, H., Frohlich, R., Vilagos, V., Kato, Z.: L2d2: Learnable line detector and descriptor. In: 3DV (2021)
3. Acuna, D., Ling, H., Kar, A., Fidler, S.: Efficient interactive annotation of segmentation datasets with polygon-rnn++. In: CVPR (2018)
4. Akinlar, C., Topal, C.: Edlines: Real-time line segment detection by edge drawing. In: ICIP (2011)
5. Almazan, E.J., Tal, R., Qian, Y., Elder, J.H.: Mcmlsd: A dynamic programming approach to line segment detection. In: CVPR (2017)
6. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. PAMI **33**(5) (2011)
7. Barath, D., Matas, J.: Progressive-x: Efficient, anytime, multi-model fitting algorithm. In: ICCV (2019)
8. Bauchet, J.P., Lafarge, F.: Kippi: Kinetic polygonal partitioning of images. In: CVPR (2018)
9. Cho, N.G., Yuille, A., Lee, S.W.: A novel linelet-based representation for line segment detection. PAMI **40**(5) (2018)
10. Cohen-Steiner, D., Alliez, P., Desbrun, M.: Variational shape approximation. In: SIGGRAPH (2004)
11. Dai, X., Gong, H., Wu, S., Yuan, X., Ma, Y.: Fully convolutional line parsing. Neurocomputing **506** (2022)
12. Denis, P., Elder, J.H., Estrada, F.J.: Efficient edge-based methods for estimating manhattan frames in urban imagery. In: ECCV (2008)
13. Fabbri, R., Duff, T., Fan, H., Regan, M., da Costa de Pinho, D., Tsigaridas, E., Wampler, C., Hauenstein, J., Giblin, P.J., Kimia, B.B., Leykin, A., Pajdla, T.: Trifocal relative pose from lines at points. PAMI **45**(6) (2023)
14. Favreau, J.D., Lafarge, F., Bousseau, A., Auvolat, A.: Extracting Geometric Structures in Images with Delaunay Point Processes. PAMI **42**(4) (2020)
15. Gioi, R.V., Jakubowicz, J., Morel, J.M., Randall, G.: Lsd: A fast line segment detector with a false detection control. PAMI **32**(4) (2010)
16. Gomez-Ojeda, R., Moreno, F.A., Zuniga-Noel, D., Scaramuzza, D., Gonzalez-Jimenez, J.: Pl-slam: A stereo slam system through the combination of points and line segments. IEEE Transactions on Robotics **35**(3) (2019)
17. Gu, G., Ko, B., Go, S., Lee, S.H., Lee, J., Shin, M.: Towards light-weight and real-time line segment detection. In: AAAI (2022)
18. Hofer, M., Maurer, M., Bischof, H.: Efficient 3d scene abstraction using line segments. CVIU **157** (2017)
19. Huang, K., Wang, Y., Zhou, Z., Ding, T., Gao, S., Ma, Y.: Learning to parse wireframes in images of man-made environments. In: CVPR (2018)
20. Huang, S., Qin, F., Xiong, P., Ding, N., He, Y., Liu, X.: Tp-lsd: Tri-points based line segment detector. In: ECCV (2020)
21. Kaiser, A., Ybanez Zepeda, J.A., Boubekeur, T.: A survey of simple geometric primitives detection methods for captured 3d data. Computer Graphics Forum **37** (2018)
22. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., Dollár, P., Girshick, R.: Segment anything. In: ICCV (2023)

23. Lazarow, J., Xu, W., Tu, Z.: Instance segmentation with mask-supervised polygonal boundary transformers. In: CVPR (2022)
24. Lee, D., Hebert, M., Kanade, T.: Geometric reasoning for single image structure recovery. In: CVPR (2009)
25. Li, M., Lafarge, F., Marlet, R.: Approximating shapes in images with low-complexity polygons. In: CVPR (2020)
26. Liang, J., Homayounfar, N., Ma, W.C., Xiong, Y., Hu, R., Urtasun, R.: Polytransform: Deep polygon transformer for instance segmentation. In: CVPR (2020)
27. Lin, Y., Pintea, S.L., van Gemert, J.C.: Deep hough-transform line priors. In: ECCV (2020)
28. Liu, J., Ding, H., Cai, Z., Zhang, Y., Satzoda, R.K., Mahadevan, V., Manmatha, R.: Polyformer: Referring image segmentation as sequential polygon generation. In: CVPR (2023)
29. Mateus, A., Tahri, O., Aguiar, P., Lima, P.U., Miraldo, P.: On incremental structure from motion using lines. IEEE Transactions on Robotics **38**(1) (2022)
30. Meng, Q., Zhang, J., Hu, Q., He, X., Yu, J.: Lgnn: A context-aware line segment detector. In: ACM International Conference on Multimedia (2020)
31. Micusik, B., Wildenauer, H.: Structure from motion with line segments under relaxed endpoint constraints. IJCV **124** (2017)
32. Oesau, S.: Point set shape detection. In: CGAL User and Reference Manual. CGAL Editorial Board, 5.3 edn. (2021)
33. Pautrat, R., Barath, D., Larsson, V., Oswald, M.R., Pollefeys, M.: Deeplsd: Line segment detection and refinement with deep image gradients. In: CVPR (2023)
34. Pautrat, R., Lin, J.T., Larsson, V., Oswald, M., Pollefeys, M.: Sold2: Self-supervised occlusion-aware line description and detection. In: CVPR (2021)
35. Pumarola, A., Vakhitov, A., Agudo, A., Sanfeliu, A., Moreno-Noguer, F.: Pl-slam: Real-time monocular visual slam with points and lines. In: ICRA (2017)
36. Raguram, R., Chum, O., Pollefeys, M., Matas, J., Frahm, J.M.: Usac: A universal framework for random sample consensus. PAMI **35**(8) (2013)
37. Raguram, R., Frahm, J.M., Pollefeys, M.: A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus. In: ECCV (2008)
38. Schnabel, R., Wahl, R., Klein, R.: Efficient ransac for point-cloud shape detection. Computer graphics forum **26**(2) (2007)
39. Sun, X., Christoudias, M., Fua, P.: Free-shape polygonal object localization. In: ECCV (2014)
40. Suárez, I., Buenaposada, J.M., Baumela, L.: Elsed: Enhanced line segment drawing. PR **127** (2022)
41. Tardif, J.P.: Non-iterative approach for fast and accurate vanishing point detection. In: ICCV (2009)
42. Teplyakov, L., Erlygin, L., Shvets, E.: Lsdnet: Trainable modification of lsd algorithm for real-time line segment detection. IEEE Access **10** (2022)
43. Tong, X., Ying, X., Shi, Y., Wang, R., Yang, J.: Transformer based line segment classifier with image context for real-time vanishing point detection in manhattan world. In: CVPR (2022)
44. Wu, S.T., Marquez, M.R.G.: A non-self-intersection Douglas-Peucker algorithm. In: IEEE Symposium on Computer Graphics and Image Processing (2003)
45. Xu, C., Zhang, L., Cheng, L., Koch, R.: Pose estimation from line correspondences: A complete analysis and a series of solutions. PAMI **39**(6) (2017)
46. Xu, Y., Xu, W., Cheung, D., Tu, Z.: Line segment detection using transformers without edges. In: CVPR (2021)

47. Xue, N., Bai, S., Wang, F., Xia, G.S., Wu, T., , Zhang, L.: Learning attraction field representation for robust line segment detection. In: CVPR (2019)
48. Xue, N., Wu, T., Bai, S., Wang, F.D., Xia, G.S., Zhang, L., Torr, P.H.: Holistically-attracted wireframe parsing. In: CVPR (2020)
49. Yu, M., Lafarge, F.: Finding good configurations of planar primitives in unorganized point clouds. In: CVPR (2022)
50. Zhang, H., Luo, Y., Qin, F., He, Y., Liu, X.: Elsd: Efficient line segment detector and descriptor. In: ICCV (2021)
51. Zhang, Y., Wei, D., Li, Y.: Ag3line: Active grouping and geometry-gradient combined validation for fast line segment extraction. PR **113** (2021)
52. Zhang, Z., Li, Z., Bi, N., Zheng, J., Wang, J., Huang, K., Luo, W., Xu, Y., Gao, S.: Ppgnet: Learning point-pair graph for line segment detection. In: CVPR (2019)
53. Zhou, Y., Qi, H., Ma, Y.: End-to-end wireframe parsing. In: ICCV (2019)