Published in Image Processing On Line on 2024–02–28.
Submitted on 2023–05–06, accepted on 2023–12–06.
ISSN 2105–1232 © 2024 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2024.481

Line Segment Detection: a Review of the 2022 State of the Art

Thibaud Ehret, Jean-Michel Morel

Universit Paris-Saclay, CNRS, ENS Paris-Saclay, Centre Borelli, Gif-sur-Yvette, France
thibaud.ehret@ens-paris-saclay.fr

Communicated by Rafael Grompone von Gioi *Demo edited by* Thibaud Ehret

Abstract

We compare nine line segment detectors. The two more ancient ones are based on classical edge growing followed by a statistically founded validation. The next six are very recent and based on supervised deep learning. These six deep learning methods train and validate their neural network on two datasets (*YorkUrban*, *Wireframe*); most of them compared their results with the now classic LSD (Line Segment Detector) and EDlines, and get a better performance than them on these datasets. The ninth paper combines deep learning and classical edge growing to achieve a purely non-supervised method. The seven machine learning based detectors and EDlines are described here. LSD and EDlines are parameter-free, fixed to allow for one false alarm on average. Our experiments show that the six purely ML based line segment detectors show a significant variability to their end-parameters, leading to apparent missed or irrelevant detection. We also compared all nine detectors on two images: one clearly "in domain" for the *Wireframe* dataset, and the other one slightly out of domain. A quantitative comparison would be fallacious. Indeed, while differing in their search strategy, the statistical detectors share a very similar definition and decision threshold for line segments. The purely ML-based detectors have learned from human annotators that were directed at reconstructing architectures as wireframes. Hence, these algorithms aim at a different goal, the architectural interpretation of the scene. Yet, several of them have more complete goals than just line segment detection. Indeed, several of them also associate to each segment a descriptor, and aim at making the pair segment+descriptor fit for image matching. The readers are invited to judge by themselves about the advantages and drawbacks of all methods by submitting their own images to the online demos associated with the present paper.

Source Code

The source codes and documentation for the algorithms presented in this paper are available from the web page of this article[1]. Usage instructions are included in the `README.md` file of each archive.

The acronyms of the methods paper titles and demo links are:

[1] https://doi.org/10.5201/ipol.2024.481

- LETR: Line Segment Detection Using Transformers without Edges, https://ipolcore.ipol.im/demo/clientApp/demo.html?id=48101
- TP-LSD: Tri-Points Based Line Segment Detector, https://ipolcore.ipol.im/demo/clientApp/demo.html?id=48102
- M-LSD: Line Segment Detection with Mobile LSD, https://ipolcore.ipol.im/demo/clientApp/demo.html?id=48103
- SOLD2: Self-supervised Occlusion-aware Line Description and Detection, https://ipolcore.ipol.im/demo/clientApp/demo.html?id=48104
- ULSD: Unified Line Segment Detection across Pinhole, Fisheye, and Spherical Cameras, https://ipolcore.ipol.im/demo/clientApp/demo.html?id=48105
- AFM: Learning Attraction Field Representation for Robust Line Segment Detection, https://ipolcore.ipol.im/demo/clientApp/demo.html?id=48106
- EDlines (classic procedure), https://ipolcore.ipol.im/demo/clientApp/demo.html?id=48107
- DeepLSD: Line Segment Detection and Refinement with Deep Image Gradients (combine classic and deep learning), https://ipolcore.ipol.im/demo/clientApp/demo.html?id=48108

The original implementations of the methods are available at the following links: LETR[2], TP-LSD[3], M-LSD[4], SOLD2[5], ULSD[6], AFM[7], EDlines[8], DeepLSD[9].
This is an MLBriefs article, the source codes have not been reviewed!

Keywords: line segment; detection; learning; edge; a contrario ; NFA ; wireframe; supervised learning; supervised learning; comparison; LSD; EDlines; AFM; TP-LSD; ULSD; LETR; M-LSD; SOLD2; DeepLSD

1 Introduction

Line segment detection in images has innumerable applications. Indeed, most human made objects contain straight 3D edges that cause straight image edges. Along these line segments, the image intensity changes drastically, due to different plane orientations toward the light [14, 16]. In this paper, we compare nine line segment detectors. The two elder ones, LSD (2008) and EDlines (2011) are based on classical edge growing followed by a statistically founded validation. We shall not describe LSD, which is already the object of an extensive IPOL description [9]. The next six line segment detectors where chosen because they are very recent, are based on supervised deep learning, and were trained on the same wireframe dataset. The last one combines deep learning and classical edge growing to achieve a purely non-supervised method. Furthermore, their experiments are reproducible from public code. The six methods purely based on machine learning train and validate their neural network on a dataset (*Wireframe*) and test on the testing part of the same dataset and on another dataset, *YorkUrban*. The *Wireframe* dataset consists of 5,000 training and 462 test images of man-made environments with manually annotated line segments. The *YorkUrban* dataset compiles

[2] https://github.com/mlpc-ucsd/LETR
[3] https://github.com/Siyuada7/TP-LSD
[4] https://github.com/navervision/mlsd
[5] https://github.com/cvg/SOLD2
[6] https://github.com/lh9171338/ULSD-ISPRS
[7] https://github.com/cherubicXN/afm_cvpr2019
[8] https://github.com/CihanTopal/ED_Lib
[9] https://github.com/cvg/DeepLSD

102 images of urban environments (45 indoor images and 57 outdoor images) also manually labeled. The original papers all compare their results with the now classic LSD (Line Segment Detector) or with EDlines, and claim a better performance than them on these datasets.

In Sections 2 to 8, we describe in sufficient detail EDlines (2011) and the six purely machine learning based detectors AFM (2019), TP-LSD (2020), ULSD (2021), LETR (2021), M-LSD (2021), SOLD2 (2021). In Section 9 we compare visually the results of the methods. LSD and EDlines are parameter-free when allowing for one false alarm on average. We find that the six ML based line segment detectors show a significant variability when changing their end-parameters. We also compare all nine detectors on two images: one clearly "in domain" for the *Wireframe* dataset, and the other one slightly out of domain. Since the goals of these methods and their very definition of line segments is different, no quantitative performance seems adequate. Finally, we present the mixed approach DeepLSD (2023) in Section 10.

2 EDlines [1, 2]

Like LSD [21, 8], EDlines [1, 2] is based on a statistical principle, often called Helmholtz principle or *a contrario* method. The simple guiding idea is to count the number of aligned pixels on any possible segment and accept the set of pixels as a line segment if the observed alignment is perceptually meaningful, namely could not occur in a fully disordered (white noise) image. This principle originates in Gestalt theory [7] and is used as a line validation method. The difference between LSD and EDlines resides in the way candidate segments are extracted from the image. Since it would be inconvenient to test all possible segments, LSD finds connected components of pixels sharing the same gradient orientation up to some precision. If the shape of the component is approximately a rectangle, then the medial axis of the rectangle is the detected segment, provided it satisfies the Helmholtz principle. The EDlines proceeds differently and generates candidates by joining seed points that are candidates to be tips of the segments. These candidates are once again selected by Helmholtz principle.

EDLines is comprised of three steps:

- **Edge drawing.** It applies the *edge drawing* algorithm to the input grayscale image to produce edge segments. This algorithm first regularizes the image, computes the gradients for each pixel, then estimates the anchor points (a set of pixels with a high probability of being part of an edge, and finally connects the anchors by following gradients maxima.

- **Line segment extraction.** Using the *least squares line fitting* method, this step splits the edge segments estimated in the first step into multiple lines segments. For a given edge segment, it iteratively fits the longest line segment that follows the remaining pixels in the edge segment until all pixels have been processed.

- **Line validation.** Finally, an *a contrario* validation step similar to LSD [21] is performed. This means that a segment is kept if and only it verifies the NFA condition

$$N^4 \sum_{i=k} n \binom{n}{i} p^i (1-p)^{n-i} < \varepsilon \tag{1}$$

with N the size of the image, n the length of the segment, k the number of points in the segment with a local gradient direction orthogonal to the direction of the segment, p the alignment precision, interpreted as the probability of a point having a given direction (in practice it is uniform in the set of possible directions) and ε the maximum number of false alarms.

3 AFM [24]

This paper poses the problem of line segment detection (LSD) as a problem of region coloring. The most attractive line segment for every pixel p is simply defined as the nearest line segment of pixel p. By applying this criterion, the image is partitioned into as many regions of attraction as the number of segments. The vector field associating each pixel to its closest segment is called attraction field. In that way, the line segment map is transformed into an attraction field map (AFM). Conversely, an AFM can be remapped to a set of line segments without loss of information by a "squeeze module". The squeeze module consists in grouping local points whose AFM are aligned. The final segment candidates are then estimated as the segments that are orthogonal to the AFM of each group of points.

The authors point out that attraction fields are actually two-dimensional feature maps, therefore compatible with convolutional neural networks. In that way, regional attraction transforms the line segment detection problem into a region coloring problem. The attraction field (and the subsequent image partition) are learned from the *Wireframe* synthetic dataset [12]. Before training, the ground truth AFM is computed for each image of the dataset using the ground truth annotations. For that, the partition-region is first computed by associating to each point p its nearest line segment in the image. Given p' the projection of p on its nearest line segment, the AFM corresponds then to $p - p'$. During training, the L_1 loss between the predicted AFM and the ground truth AFM is minimized. In the testing phase, the attraction field map computed by the network is squeezed to output line segments.

Inspired from deep learning based semantic segmentation methods, the method learns the attraction field map end-to-end. Figure 1 illustrates the architecture of the proposed LSD framework based on an encoder-decoder neural network. By the squeeze module, the obtained segments are inserted into the input image. For the encoder-decoder architecture, the authors tested both a simple U-Net [19] and a modified U-Net termed *A-trous* Residual U-Net, that takes advantage of residual blocks [11] and the ASPP module from DeepLab v3+ [5]. Validation is performed on the *YorkUrban* dataset [6].

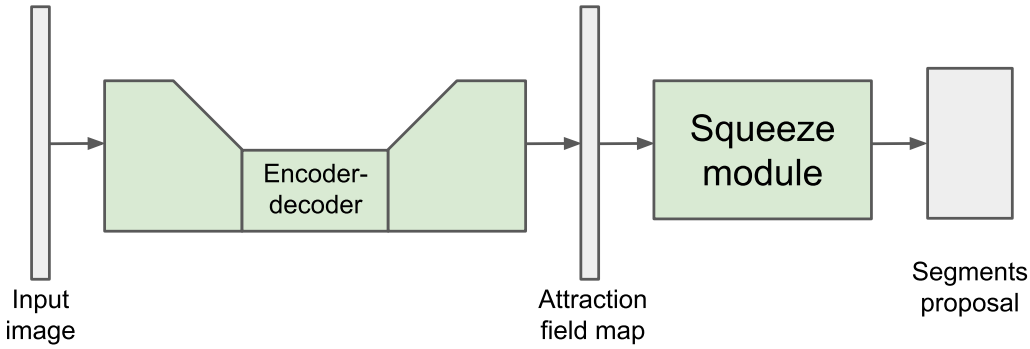


Figure 1: Architecture of AFM.

4 TP-LSD [13]

TP-LSD is based on the concept of the tri-points representation of line segments. Contrary to regular line segments representations based on two points, the start and end points, the tri-points representation considers a root point, representing the "center"of the segment, as well as the two displacements vectors to the start and the end points respectively.

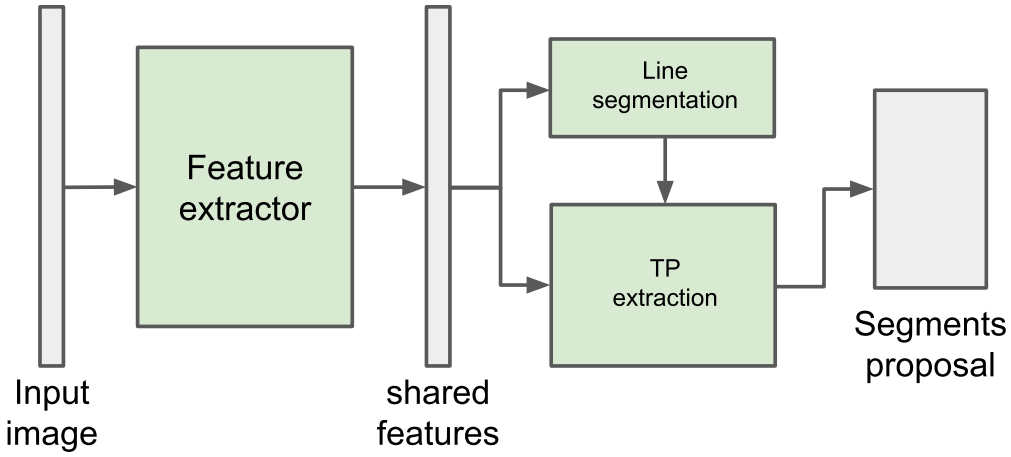


Figure 2: Architecture of TP-LSD.

The method is based on a U-Net style network that outputs a first set of shared features. These shared features are then provided to two branches, the TP extraction branch and the line segmentation branch (see Figure 2):

1. The TP extraction branch outputs the probability of each pixel of the image being a root point or not. In addition, it also outputs the displacements corresponding to the associated start and end points;

2. The line segmentation branch outputs the probability of a given pixel to be on a line. This probability is then injected into the TP extraction branch (using feature aggregation modules and a point filter module) to improve the quality of the TP detections.

The network is trained using the *Wireframe* dataset. From the annotation, ground truth binary line segments map are derived and used to train the line segmentation branch using a binary cross entropy loss $\mathcal{L}_{line}$. The ground truth root points are extracted as all the middle points of the ground truth segments and the root points predictions are also trained using a binary cross entropy loss $\mathcal{L}_{root}$. Similarly, displacements vectors are derived from the root points and the start and end points. A smoothed L_1 loss $\mathcal{L}_{disp}$ is used to learn the displacement map predictions. The complete loss used for training is then $\mathcal{L} = w_{root}\mathcal{L}_{root} + w_{disp}\mathcal{L}_{disp} + w_{line}\mathcal{L}_{line}$ where $(w_{root}, w_{disp}, w_{line}) = (50, 1, 20)$.

The authors made available two pre-trained models. One trained on 320×320 images (TP320) and the other on 512×512 images (TP512).

5 ULSD [15]

Contrary to other methods presented in this review, ULSD [15] proposes to detect line segments that might have been distorted by the optical system like it happens for fisheye or spherical cameras. The model used for the detection is made of three main blocks: an encoding backbone network that creates an intermediate feature map, a line segment proposal network and finally a classification network (see Figure 3). These blocks work as follows:

- The input image is first given to an encoding network, a stacked hourglass network in practice, to produce an intermediate feature map.

- The feature map is then provided to a line proposal network that creates line segment candidates:

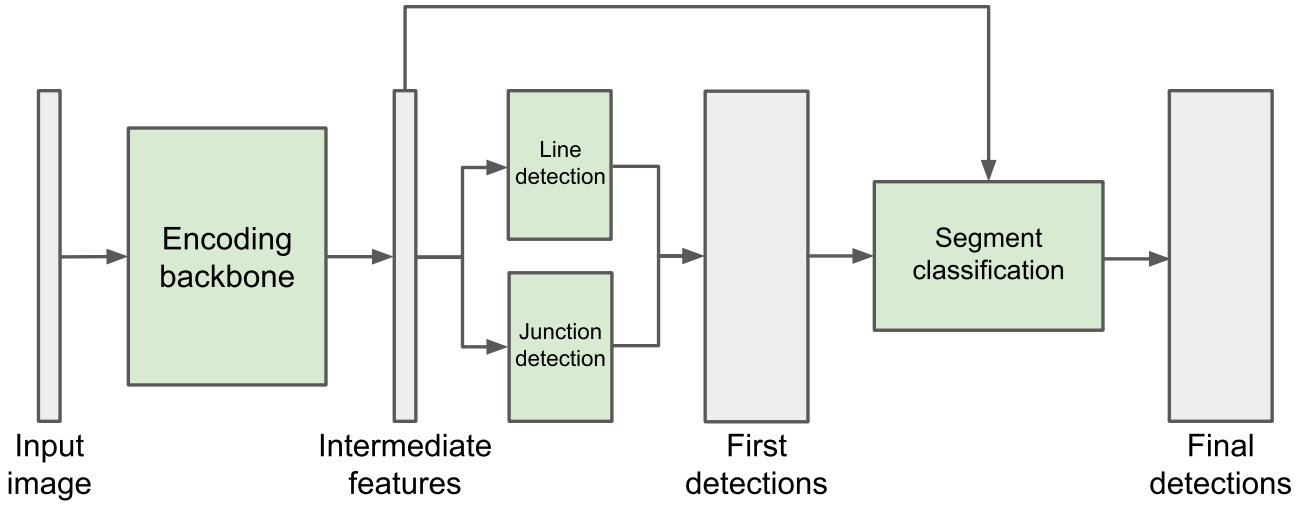


Figure 3: Architecture of ULSD.

1. This feature map is then given to a first detection network whose job is to detect the pixels corresponding to the end of segments, called junctions in the paper. Given that the feature map size is smaller than the input image, this block also outputs a sub-pixel position corresponding to the offset to exact position in the input image.
2. The second detection network predicts the location of lines. A line is comprised of a center that is detected similarly to the end points (i.e. as a per pixel detection map and an offset to map it to the exact sub-pixel position) as well as a series of n points (represented by their offset to the center) that are uniformly located on the segment. The authors compare these points to Bézier control points except these are located on the line and as such are easier to estimate for the network since they are all located inside the image.
3. The junctions and lines are then combined to create the proposed line segment candidates. A line candidate is said to be a line segment candidate if and only if it is possible to match junctions to each of its endpoints.

- Finally, the line segment candidates and the feature maps are given to a final classification network that decides whether the line segment candidate is valid:

  1. For each proposed segment, line features are extracted. For this, a set of n_p points is sampled uniformly on the segment. This is done by estimating a Bézier curve going through the set of points defining the segment. The line features correspond then to a concatenation of the interpolated features associated to these points.
  2. The line features are given to a classification network that predicts whether the proposed segment is valid.

For the pinhole model, training is done on the *Wireframe* dataset [12] and *YorkUrban* dataset [6]. For that, a loss taking into account the intermediate prediction is used. In particular, a combination of three losses is used. The first loss is for the junction detection module. This loss is comprised of a binary cross-entropy loss for the detection part and an L_1 loss for the subpixel refinement offset part. A similar loss is associated to the line detection network. Finally, a binary cross-entropy loss is associated to the classification network. For that, a label (positive or negative) is given to each detected segment based on whether it is possible to match the detection to a segment in the ground truth. Unmatched segments from the ground truth are also added to the positive label set. Segments from both labels are drawn so that both classes are balanced.

Since there are no datasets available for fisheye and spherical, the authors also proposed their own dataset based on a distorded version of the *Wireframe* dataset and the *YorkUrban* dataset using fisheye distortion model and the manually annotated SUN360 dataset [22]. The same loss as in the pinhole case is used for these models.

6 LETR [23]

Object detection based on CNNs works by predicting a bounding box for the detected object. However, the concepts of anchors, necessary for the bounding box detection, is difficult to use for line segments detection. Indeed, the anchors need to be inside the bounding box found by the model, which is difficult for small segments and segments in the image grid directions. It is for this reason that it is not possible to directly use object detection networks to learn line segment detections. Since the recent *transformer* networks for object detection, such as [4], don't use anchors, LETR proposes to use such an architecture instead, and predicts directly a line segment without the intermediate bounding box step.

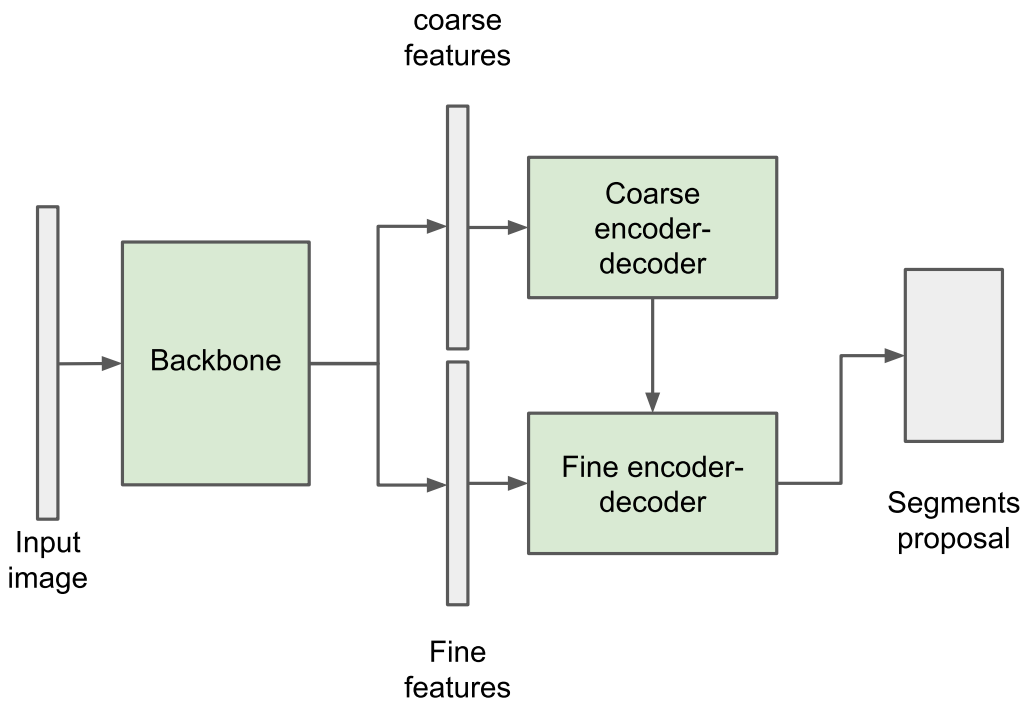


Figure 4: Architecture of LETR.

The architecture of LETR, represented in Figure 4, is the following:

1. The input image is provided to a first backbone network, usually a ResNet-50 or a ResNet-101. From this backbone, two feature layers are extracted at different levels. These two feature levels represent two different scales. Similarly to all transformer architectures, a positional encoding is concatenated to each feature layer;

2. The coarse features are provided to a coarse transformer encoder-decoder, this outputs a first coarse guess of the location of line segments in the image;

3. A fine transformer encoder-decoder is then provided with the initial guess as well as the fine features. This block then provides the probabilities that the prediction is a line segment and the two corresponding endpoints.

During training, the predicted line segments are associated, if possible, to ground truth line segments using the Hungarian algorithm. Once the matching between prediction and ground truth is available, a focal loss is applied to the predicted probabilities and an L_1 loss is applied between the predicted and ground truth endpoints.

7 M-LSD [10]

One major limitation of deep learning based methods is their computation time compared to LSD [9] and ED-Lines [1, 2], see Table 1. This is why M-LSD [10] has been proposed. The idea is to accelerate TP-LSD [13], presented in Section 4, by replacing the U-Net and the two branches with a single lightweight network adapted from MobileNet-v2 [20]. This lightweight U-Net outputs directly the root points prediction as well as the displacement vectors (see Figure 5). The authors suggest to

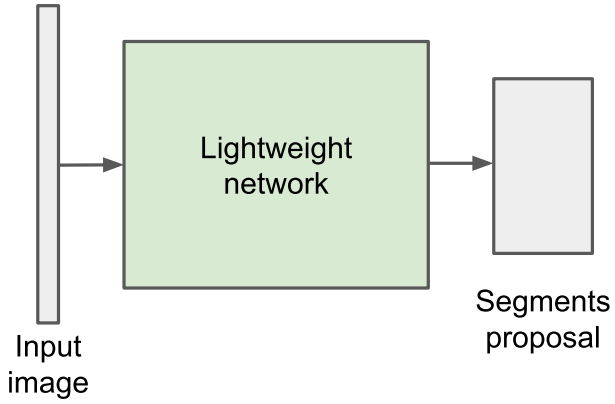


Figure 5: Architecture of M-LSD.

add three extra terms to the loss during training compared to TP-LSD:

- A matching loss term $\mathcal{L}_{match}$: TP-LSD penalizes only the value of the displacement vectors. In M-LSD, the distance to the control points (end points and root point) is also penalized using a L_1 distance;

- A "segments of line" loss $\mathcal{L}_{sol}$: Some segments are too long for the receptive field of the network leading to missed detection. Similarly, segments that are too close can also interact. The authors suggest that forcing the network to also learn how to predict segments of line (SoL) improves the quality of segment predictions in these situations. Instead of using the same output for both complete and incomplete segments, the network is modified (only during training) to output two predictions: one that models the segments of lines and the other for the complete line segments. In order to create the ground truth for the SoL prediction, the ground truth segments are split into multiple smaller ones of same size using the same tri-points representation. The SoL loss is the same as the one for normal detections. During evaluation, the SoL output is discarded to only keep the prediction of the complete segments.

- A geometric loss $\mathcal{L}_{geo}$: The idea is to force the network to learn additional visual geometric cues. Indeed, end points are localized on junctions points and the root point is located on the

line itself. The ground truth for both is derived from the annotation and a binary cross entropy is used. An L_1 loss on the length and the angle of the displacement vector is also used.

The authors made multiple pre-trained models available. Two backbones are available, while both are based on MobileNet-v2, the *large* version uses a 96 channels bottleneck and the *tiny* version uses a 64 channels bottleneck. Additionally, the authors tested both architectures with training images of sizes 512×512 (512) or 320×320 (320) and weights with numeric floating point precision of 32 bits (fp32) or 16 bits (fp16).

8 SOLD2 [18]

SOLD2 [18] proposes to learn a descriptor at the same time as line detection. The goal is to be able to use line segments together with their descriptors for image matching. Contrary to the

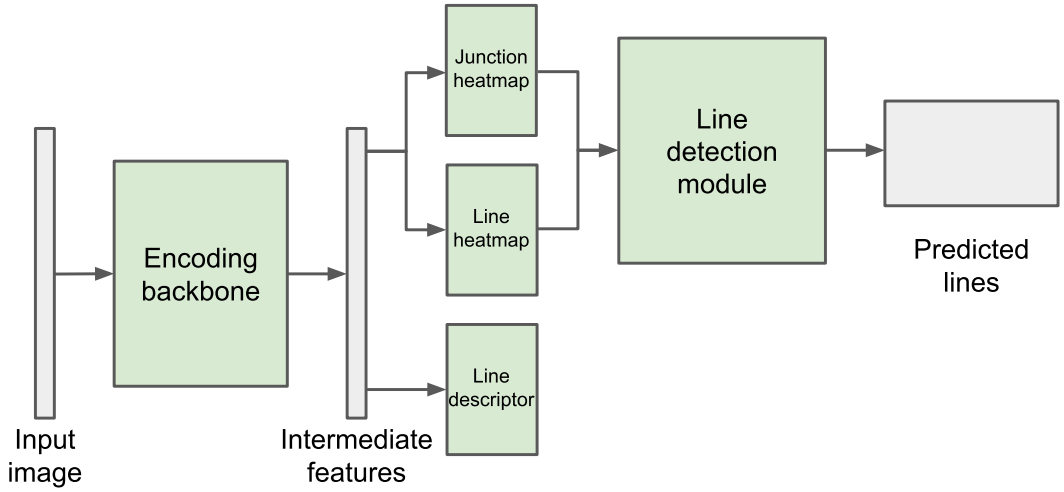


Figure 6: Architecture of SOLD2.

previous methods, lines in SOLD2 are represented using only their two endpoints. The architecture, represented in Figure 6, is the following:

1. The input image is first given to an encoding backbone producing a set of intermediate features.
2. The intermediate features are then given to the junction branch that outputs the probability that a pixel is a junction.
3. At the same time, the heatmap branch produces the probability that a given pixel is in a line.
4. The junction and line probabilities are fed into the line detection module. It first applies a threshold on the line probabilities and keeps the local maxima to define the line junctions. The line candidates consist of all the segments defined by two junction points. In order to extract the valid candidates, a set of N_s equally spaced points is sampled along the candidate. A candidate is said to be valid when the average line probabilities for these points is above a given threshold and if enough points are above this threshold.

Both the junction prediction and the line heatmap are learned using binary cross-entropy losses using the annotated images from the *Wireframe* dataset.

Once the line detector has been learned, a dense line descriptor is learned using self-supervision. Since the *Wireframe* dataset doesn't contain pairs of images with matching lines, the authors propose

to generate training data using data augmentation. Since a line segment in an image still is a line segment after a homography, it generates training pairs to learn descriptors for line matching by applying random homographies to reference images. In the end, the line segments considered during the training are the ones that are detected, using the detector trained in the first part, in the reference image, as well as in each transformed image generated from random homographies. The descriptor is learned using a triplet loss between the descriptors of corresponding points of matching lines and the descriptor of a point in another line segment.

9 Comparative Experiments

Parameters of the machine learning methods. In Figure 7 we compare the results of seven line segment detectors on a photograph. The experiment displays the original image "Le Pirée", followed by results of LSD (2008), EDlines (2011), TP-LSD (2020), ULSD (2021), LETR (2021), M-LSD (2021), SOLD2 (2021). As we commented, the first two detectors are handcrafted and based on edge growing followed by an *a contrario* detection threshold. A detailed comparison of both images shows that most segments present in one image are present in a similar position in the other. Sometimes a segment detected by EDlines is actually split in two with a gap in the LSD result. This corresponds to a slightly different heuristic exploration of the image gradient field by both methods: LSD requires connectedness for a segment to be detected, while EDlines allows for some gap in a segment and therefore sometimes presents a single segment where LSD has two. All in all, the detection maps are very similar because they obey the same statistical definition of meaningful segment. No machine learning is involved in this definition and therefore no dataset dependence: these detectors are agnostic. Something radically different is at stake with the last five detectors obtained by sophisticated deep learning methods learning primarily from the *Wireframe* dataset. This dataset is actually *not* a dataset of images with annotated line segments. It is a dataset of photographs of interior and exterior architectures, and its annotation concerns "wire frames", namely line segments depicting 3D edges of the architecture and mostly meeting each other at corners or apparent T-junctions. Unsurprisingly, the line segment interpretations given by machine learning taught on wireframes are widely different from those proposed by the agnostic statistical definition. This is confirmed by taking a look at an image from the *wireframe* dataset [12], see Figure 8. The middle row shows two images from the *wireframe* dataset and their "ground truth" which is a human annotation. This annotation is actually a mental reconstruction of the architectural sketch of the scene. It presents segments that are actually not visible and misses many that are conspicuous. The agnostic LSD detections in the bottom row show, for example, hundreds of line segments following the ground planks on the left image. On the right hand image, the ground truth interprets the windows's muntins as single line segments while these are obviously thick and endowed with two straight sides. This experiment illustrates perhaps the frailty of human annotation: many obvious line segments are missing. The five neural networks where also evaluated (and compared to LSD and EDlines) on the *YorkUrban* dataset [6]. In Figure 9 we display the same format as for the wireframe. On the middle row, two images from this dataset. On top of them their "wire frame" interpretation, which seems extraordinarily incomplete and arbitrary. On the bottom row the line segments found by LSD.

Returning to Figure 7, one can nevertheless notice some agreement between the results of the four first mentioned networks trained on *wireframe*, TP-LSD,ULSD, LETR, and MLSD, while the result of SOLD2 is puzzling. This last method extends unduly many segments and misses many, so that the building's architecture is not even respected. These results give an incomplete but rather coherent view of the building visible on the foreground. The input image belongs to the "domain" on which the neural network is competent. Yet, on a slightly "out of domain" image, the results

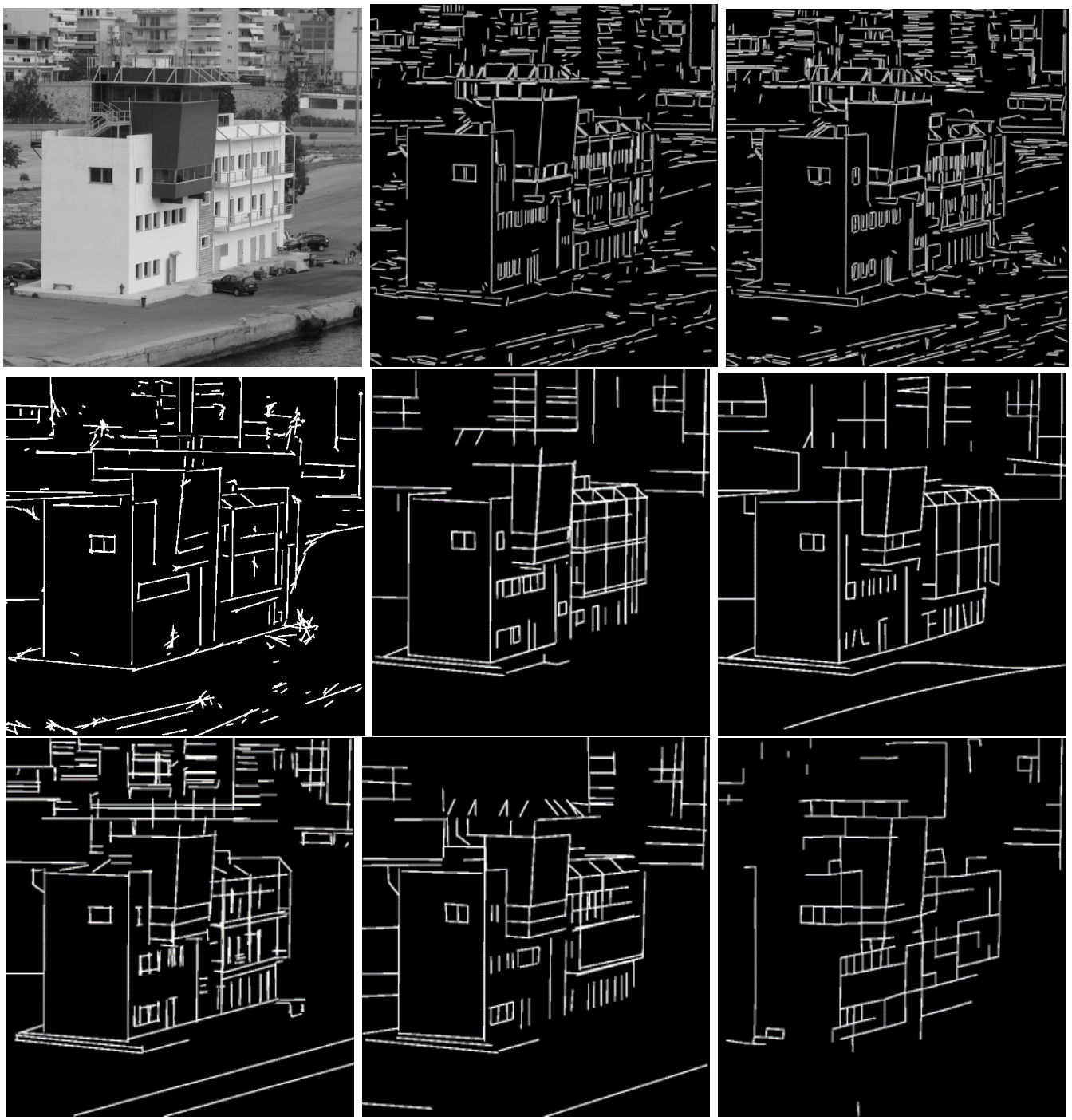


Figure 7: Comparison of eight line segment detectors on a photograph of "Le Pirée": original image "Le Pirée", results of LSD (2008), EDlines (2011), AFM (2019), TP-LSD (2020), ULSD (2021), LETR (2021), M-LSD (2021), SOLD2 (2021). The first two detectors are handcrafted and based on edge growing followed by an *a contrario* detection threshold. The last six are obtained by sophisticated, mainly supervised, deep learning methods.

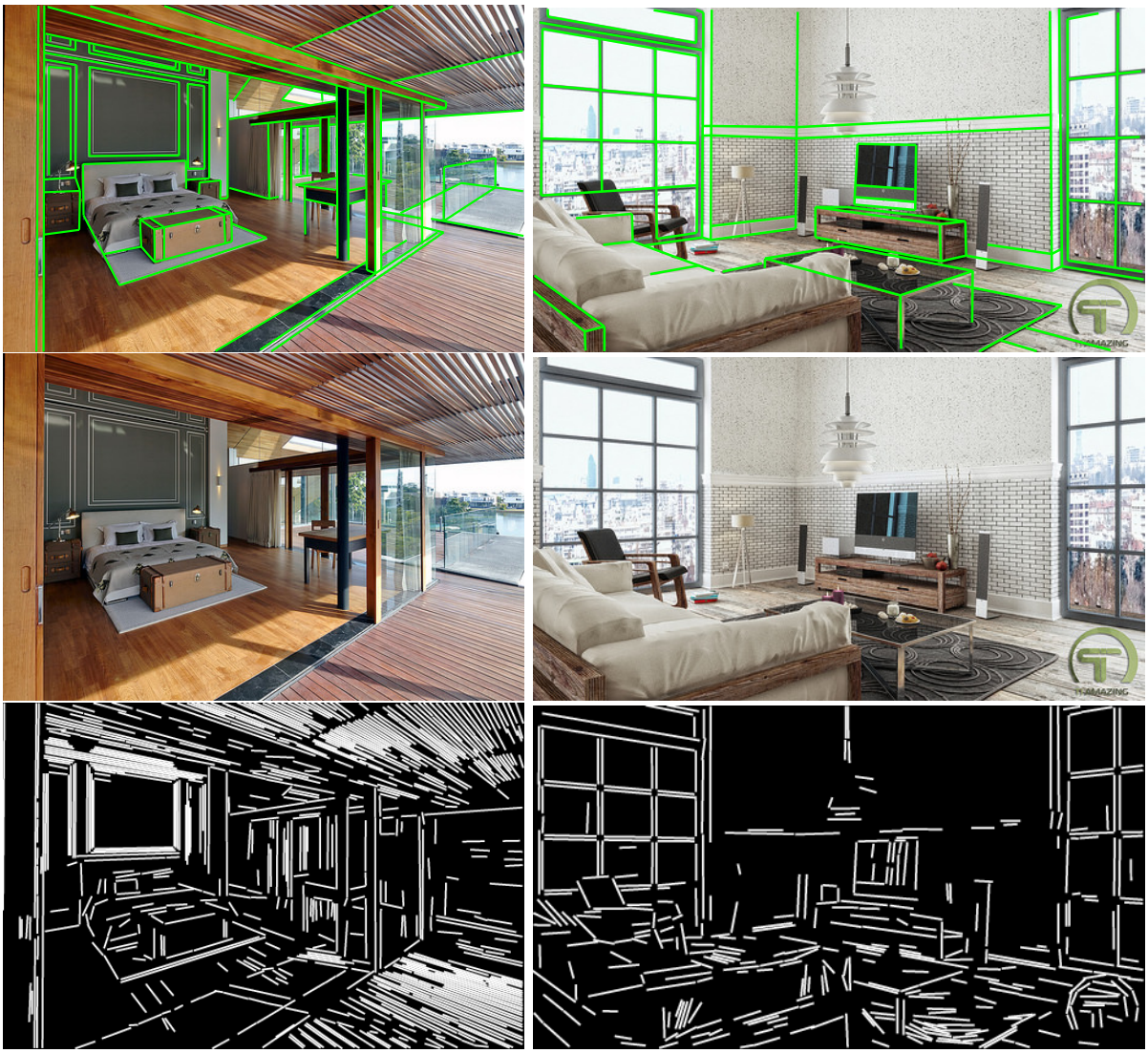


Figure 8: Middle: two images from the *wireframe* dataset. Top: their line segment interpretation in the *wireframe* dataset. Bottom: their interpretation by LSD. This experiment illustrates the frailty of human annotation: many obvious line segments are missing.

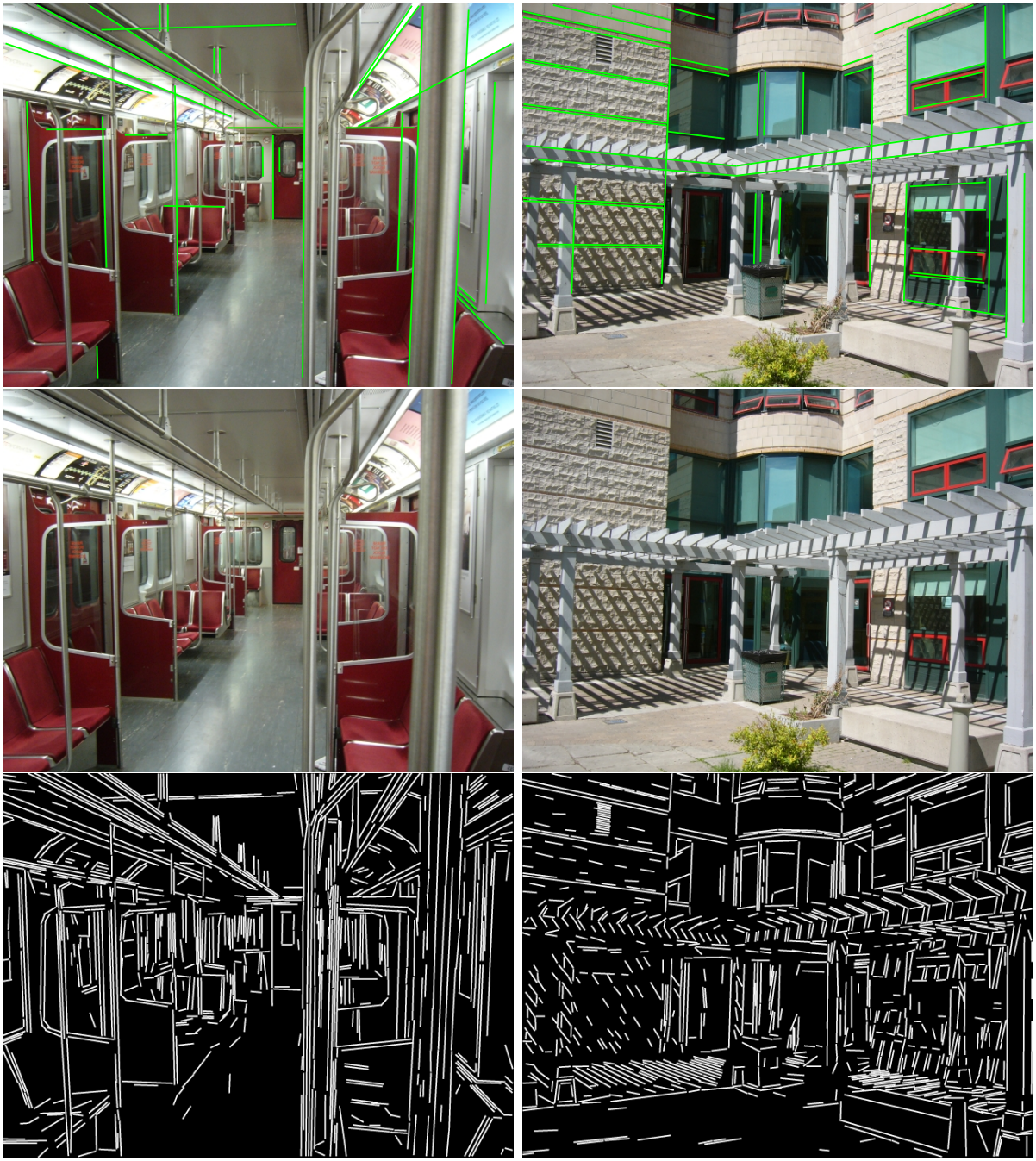


Figure 9: Middle: two images from the *YorkUrban* dataset. Top: their line segment interpretation in the *YorkUrban* dataset. Bottom: their interpretation by LSD. This experiment illustrates the frailty of human annotation: many obvious line segments are missing.

obtained by neural networks trained on *wireframe* diverge considerably more, as becomes evident in the results compared in Figure 10. The scene contains just and only aligned chairs in perspective. The networks perform a "wireframe" analysis. Hence the rounded angles of the chairs are replaced by corners ; more strikingly, most chair borders are visually ridges, not edges, thus delimited by two parallel straight lines. These are found by LSD and EDlines, but systematically replaced by a single line in the wire frame interpretation. Additional results are shown in Figure 11 and Figure 12. Figure 12 verifies if the methods make detections in noise. None does except AFM. Figure 11 tests the methods on very simple graphical images with obvious straight black strokes and uppercase letters containing segments. All methods except LSD and EDlines underdetect. More than that, the black strokes are mostly interpreted as single segments by methods trained on *wireframe* while they are in fact rectangles with two sides.

In the experiments of Figures 7 and 10 the algorithms depend on parameters. For LSD and EDlines, by a classic principle of *a contrario* detection methods, one false alarm (in expectation) is allowed per image. This principle makes sense in detection tasks where multiple detections are expected. Unfortunately for neural networks no such principle is available or accessible to learning.

The considered machine learning methods depend on one or two parameters. One is common to all: the score or detection threshold ranging from 0 to 1. But an additional threshold may be used to enforce accuracy like in ML-LSD, which is endowed with a distance threshold, and SOLD2, which has an inlier threshold ranging in $[0, 1]$. LSD and EDlines have both a single threshold, the NFA, which is fixed to 1, thus actually not an active user parameter. Here is the list of parameters for these methods

- SOLD2: detection threshold from 0 to 1, inlier threshold from 0 to 1
- M-LSD: score threshold from 0 to 1, distance threshold from 0 to 20
- U-LSD: detection threshold from 0 to 1. This network has three structural options depending on the kind of optical camera: mixed, pinhole, spherical, or fisheye
- LETR and TP-LSD: score threshold from 0 to 1.

In Figure 13 we display the results obtained when varying the parameters of each method, particularly the score threshold. Although the median score 0.5 would be expected to give the best result on the learning dataset, this threshold is clearly no longer valid on an "out of domain" image.

We also compare the computation times of the different methods on a Intel(R) Core(TM) i7-7820HQ cpu in Table 1. Unsurprisingly, the two methods that are not based on deep learning are much faster than the methods based on deep learning. However, contrary to what ED-Lines [2] claims, LSD [9] is actually faster by almost a factor of two.

LSD [9]	ED-Lines [2]	SOLD2 [18]	M-LSD [10]	TP-LSD [13]	ULSD [15]	LETR [23]
0.065	0.093	15.108	4.342	31.421	8.132	9.089

Table 1: Computation time for the different methods presented. Computation times (in seconds) were estimated as an average over ten runs for the image used in Figure 10.

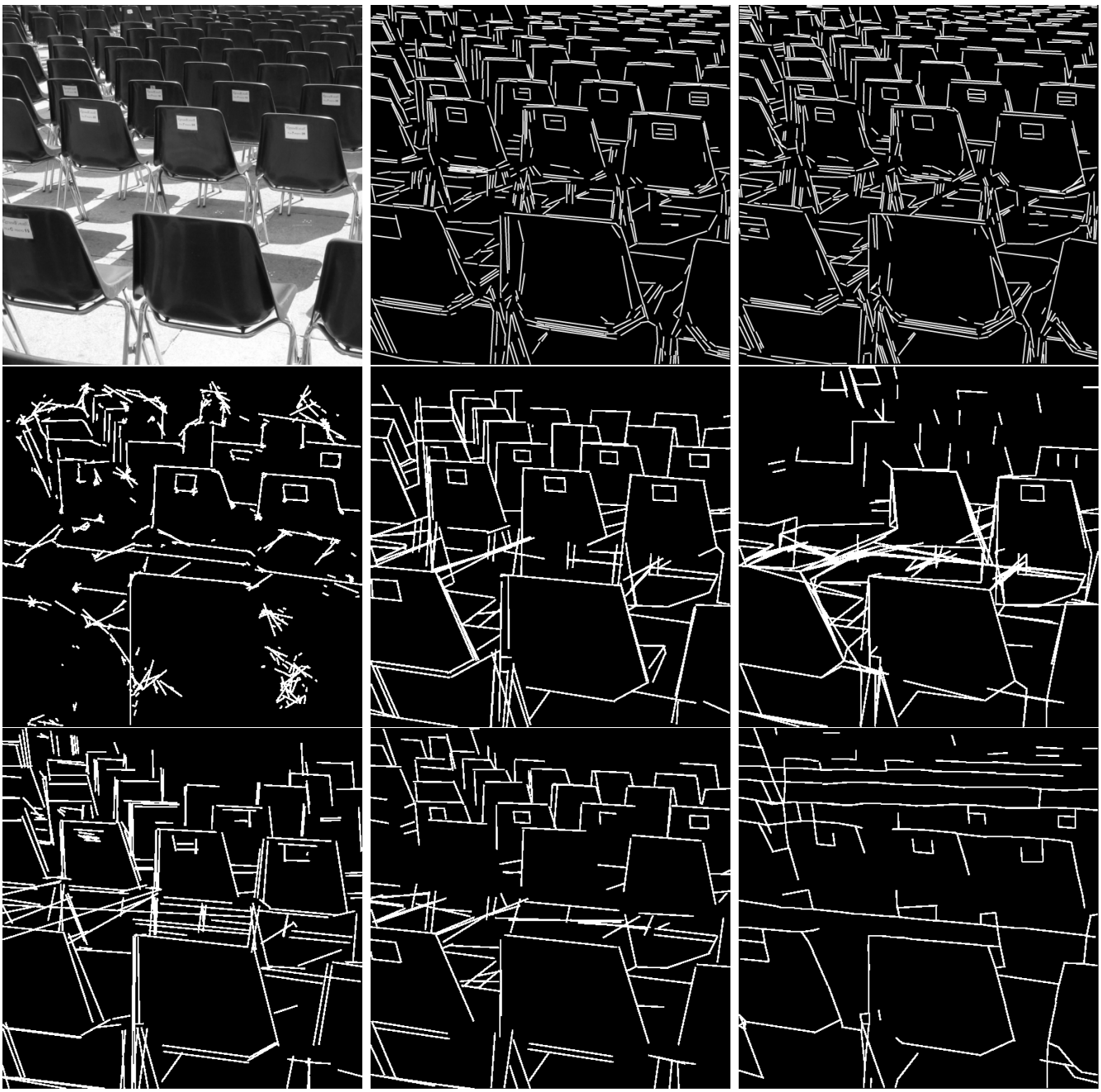


Figure 10: Comparison of eight line segment detectors on an "out of domain" image, a photograph of chairs: original image "chairs" results of LSD (2008), EDlines (2011), AFM (2019), TP-LSD (2020), ULSD (2021), LETR (2021), M-LSD (2021), SOLD2 (2021). The first two detectors are handcrafted and based on edge growing followed by an *a contrario* detection threshold. The last six are obtained by sophisticated, mainly supervised, deep learning methods.

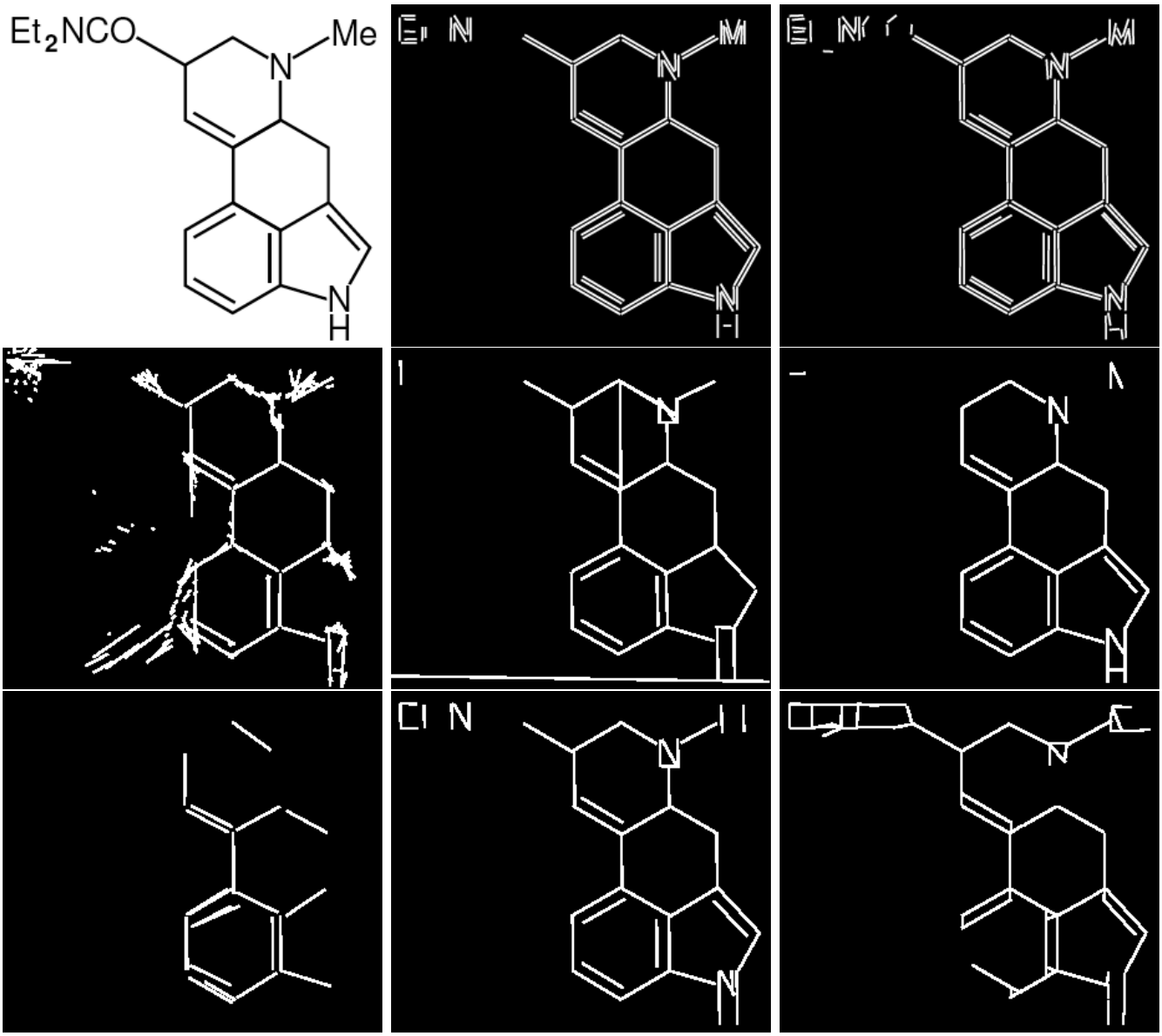


Figure 11: Comparison of eight line segment detectors on a "wireframe" image, the model of the LSD molecule: original image "LSD", results of LSD (2008), EDlines (2011), AFM (2019), TP-LSD (2020), ULSD (2021), LETR (2021), M-LSD (2021), SOLD2 (2021). The first two detectors are handcrafted and based on edge growing followed by an *a contrario* detection threshold. The last six are obtained by sophisticated mainly supervised deep learning methods.

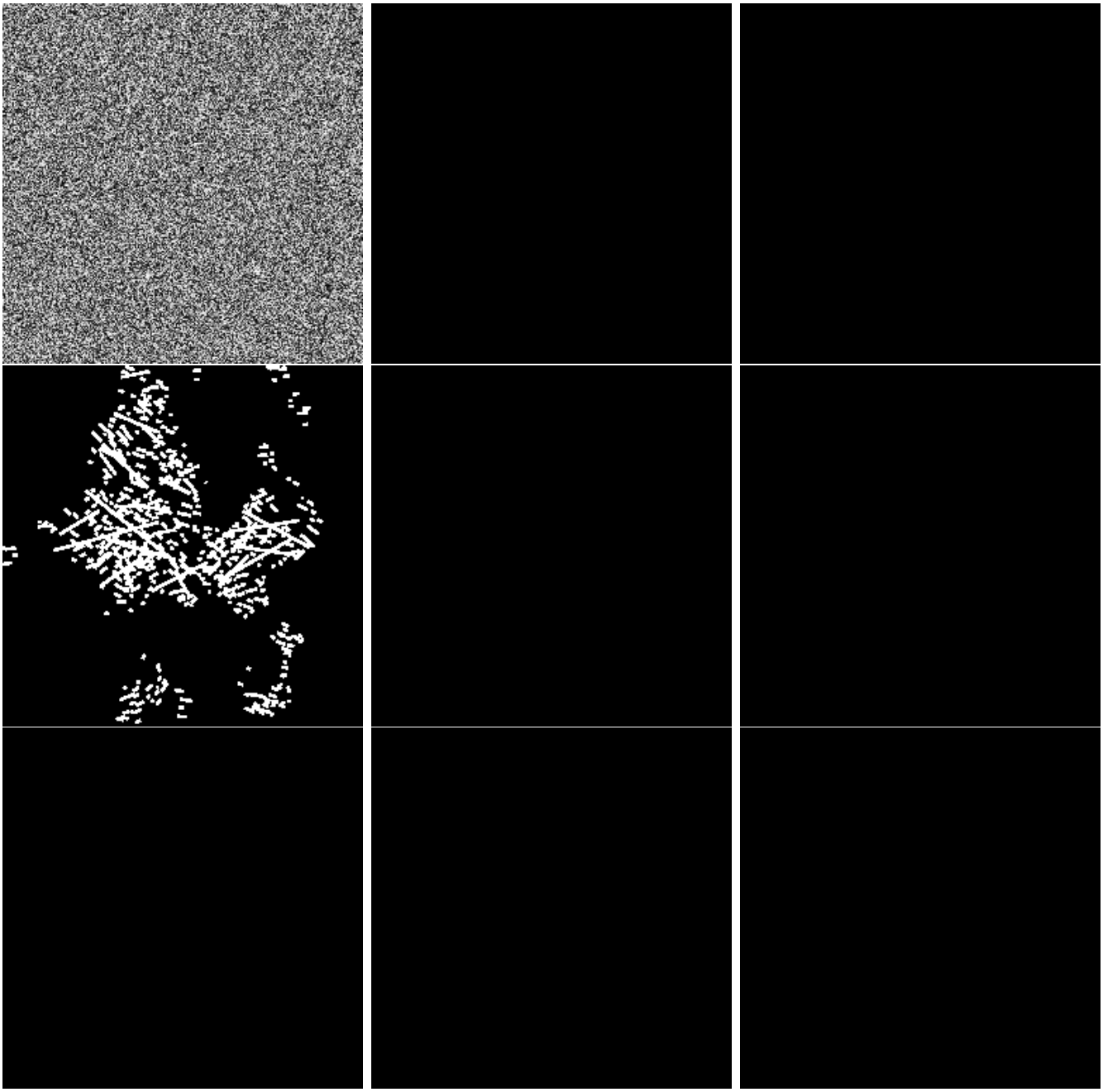


Figure 12: Comparison of eight line segment detectors on a noise image: original noise image, results of LSD (2008), EDlines (2011), AFM (2019), TP-LSD (2020), ULSD (2021), LETR (2021), M-LSD (2021), SOLD2 (2021). The first two detectors are handcrafted and based on edge growing followed by an *a contrario* detection threshold. The last six are obtained by sophisticated mainly supervised deep learning methods.

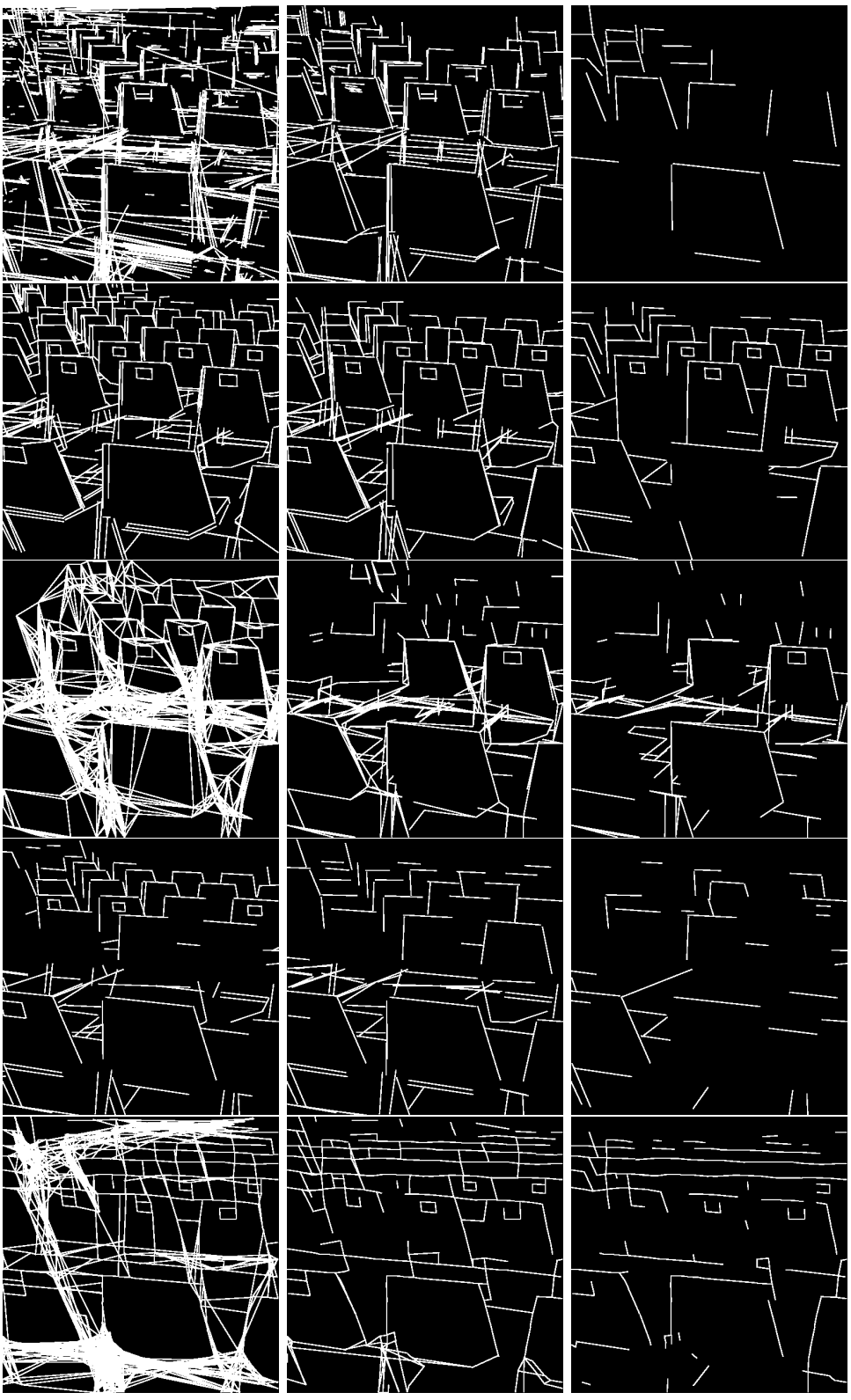


Figure 13: Parameter dependence of five line segment detectors obtained by deep learning on an "out of domain" image, a photograph of chairs. First row: results of LETR with score thresholds 0.7, 0.3 and 0.0 respectively. Second row results of TP-LSD with score thresholds 0.25, 0.1, and an alternative version (TP-LSD-320) with score threshold 0.1. Third row: ULSD with score thresholds 0, 0.1 and 0.2 respectively. Fourth row M-LSD with parameter pairs (0.2, 1), (0, 20) and (0.5, 10) respectively. Fifth row SOLD2 with parameter pairs (0.1, 0.5), (0.1, 0.99), (0.5, 0.99) respectively. The parameters are chosen so the number of segments decreases from over-detection to under-detection, the parameters in the middle column are the more plausible ones.

10 Combining Deep Learning with LSD: DeepLSD [17]

Given the limitations of pure deep learning methods shown in Section 9, Pautrat et al. proposed in [17] to combine LSD with deep learning for better line segment detection. The idea is to train a network to produce a replacement to the gradient field used by LSD [21] and EDLines [1] for the line segment prediction. For that, the authors use the same concept as AFM [24], where the network predicts a distance field (distance to the nearest line) and an angle field (orientation of the nearest line). This prediction is then transformed into a pseudo gradient field that can be used with LSD. The final predicted line segments are the ones predicted by LSD. This results in more accurate detection and closer to what one would expect, as shown in Figure 15. This process is summarized in Figure 14.

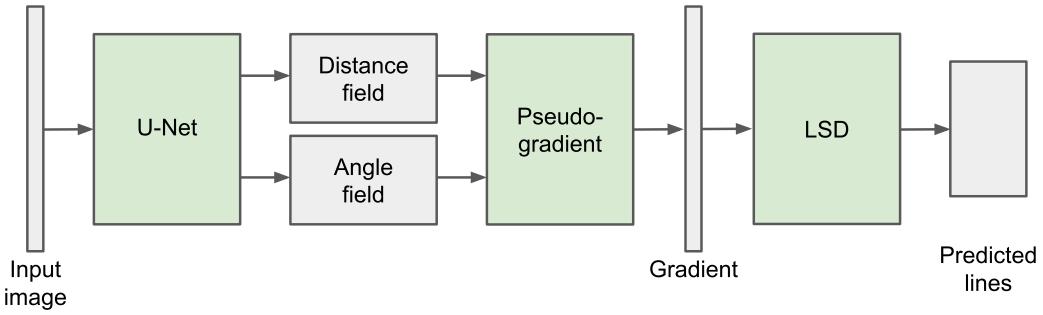


Figure 14: Architecture of DeepLSD.

In order to render the learning process unsupervised, the authors also propose to generate pseudo ground truth from LSD predictions. The reference image is first warped using a set of random homographies, then the corresponding distance and angle fields are estimated using LSD predictions on the warped images, which are then warped back onto the reference image and combined using the median. This has for effect to filter out the detections that are unstable from one warping to another. The network is then trained using the distance and angle fields thus produced.

An additional refinement step is also proposed. To that aim, the authors suggest to first estimate the vantage points using a method such as Progressive-X [3]. These vantage points are then used to refine the predicted line segments so that they match the 3D structure induced by these vantage points. This step can be used for all the other detectors.

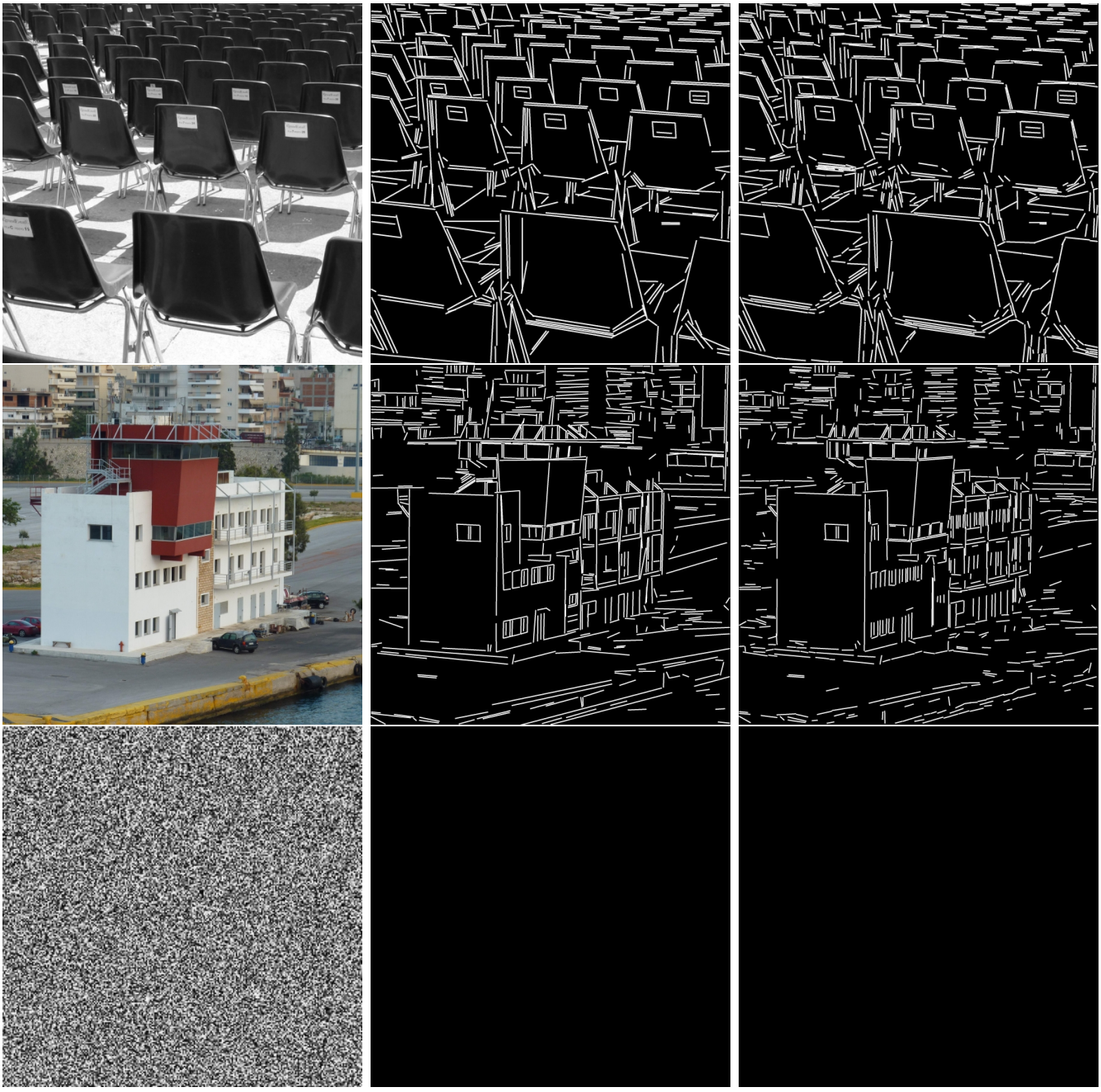
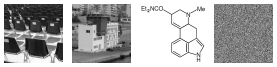


Figure 15: Comparison of DeepLSD [17] (middle column) and LSD [21] (right column).

Image Credits


Wireframe dataset [12]


YorkUrban dataset [12]


From [9]

References

[1] C. Akinlar and C. Topal, *EDLines: A Real-Time Line Segment Detector with a False Detection Control*, Pattern Recognition Letters, 32 (2011), pp. 1633–1642. https://doi.org/10.1016/j.patrec.2011.06.001.

[2] ——, *Edlines: Real-Time Line Segment Detection by Edge Drawing (Ed)*, in IEEE International Conference on Image Processing (ICIP), IEEE, 2011, pp. 2837–2840. https://doi.org/10.1109/ICIP.2011.6116138.

[3] D. Barath and J. Matas, *Progressive-X: Efficient, Anytime, Multi-Model Fitting Algorithm*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 3780–3788. https://doi.org/10.1109/ICCV.2019.00388.

[4] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, *End-To-End Object Detection with Transformers*, in European Conference on Computer Vision (ECCV), Springer, 2020, pp. 213–229. https://doi.org/10.1007/978-3-030-58452-8_13.

[5] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, *Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation*, in European Conference on Computer Vision (ECCV), 2018, pp. 801–818. https://doi.org/10.1007/978-3-030-01234-2_49.

[6] P. Denis, J. H. Elder, and F. J. Estrada, *Efficient Edge-Based Methods for Estimating Manhattan Frames in Urban Imagery*, in European Conference on Computer Vision (ECCV), Springer, 2008, pp. 197–210. https://doi.org/10.1007/978-3-540-88688-4_15.

[7] A. Desolneux, L. Moisan, and J.-M. Morel, *From Gestalt Theory to Image Analysis: a Probabilistic Approach*, vol. 34, Springer Science & Business Media, 2007. https://doi.org/10.1007/978-0-387-74378-3.

[8] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, *On Straight Line Segment Detection*, Journal of Mathematical Imaging and Vision, 32 (2008), pp. 313–347. https://doi.org/10.1007/s10851-008-0102-5.

[9] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, *LSD: a Line Segment Detector*, Image Processing On Line, 2 (2012), pp. 35–55. https://doi.org/10.5201/ipol.2012.gjmr-lsd.

[10] G. Gu, B. Ko, S. Go, S.-H. Lee, J. Lee, and M. Shin, *Towards Light-Weight and Real-Time Line Segment Detection*, in AAAI Conference on Artificial Intelligence, 2022. https://doi.org/10.1609/aaai.v36i1.19953.

[11] K. He, X. Zhang, S. Ren, and J. Sun, *Deep Residual Learning for Image Recognition*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778. https://doi.org/10.1109/CVPR.2016.90.

[12] K. Huang, Y. Wang, Z. Zhou, T. Ding, S. Gao, and Y. Ma, *Learning to Parse Wireframes in Images of Man-Made Environments*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 626–635. https://doi.org/10.1109/CVPR.2018.00072.

[13] S. Huang, F. Qin, P. Xiong, N. Ding, Y. He, and X. Liu, *Tp-Lsd: Tri-Points Based Line Segment Detector*, in European Conference on Computer Vision (ECCV), Springer, 2020, pp. 770–785. https://doi.org/10.1007/978-3-030-58583-9_46.

[14] Z. Kou, Z. Shi, and L. Liu, *Airport Detection Based on Line Segment Detector*, in International Conference on Computer Vision in Remote Sensing, IEEE, 2012, pp. 72–77. https://doi.org/10.1109/CVRS.2012.6421236.

[15] H. Li, H. Yu, J. Wang, W. Yang, L. Yu, and S. Scherer, *ULSD: Unified Line Segment Detection Across Pinhole, Fisheye, and Spherical Cameras*, ISPRS Journal of Photogrammetry and Remote Sensing, 178 (2021), pp. 187–202. https://doi.org/10.1016/j.isprsjprs.2021.06.004.

[16] X. Lu, J. Yao, K. Li, and L. Li, *CannyLines: A Parameter-Free Line Segment Detector*, in IEEE International Conference on Image Processing (ICIP), IEEE, 2015, pp. 507–511. https://doi.org/10.1109/ICIP.2015.7350850.

[17] R. Pautrat, D. Barath, V. Larsson, M. R. Oswald, and M. Pollefeys, *DeepLSD: Line Segment Detection and Refinement with Deep Image Gradients*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023, https://doi.org/https://doi.org/10.1109/CVPR52729.2023.01662.

[18] R. Pautrat, J.-T. Lin, V. Larsson, M. R. Oswald, and M. Pollefeys, *SOLD2: Self-Supervised Occlusion-Aware Line Description and Detection*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 11368–11378. https://doi.org/10.1109/CVPR46437.2021.01121.

[19] O. Ronneberger, P. Fischer, and T. Brox, *U-Net: Convolutional Networks for Biomedical Image Segmentation*, in International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, 2015, pp. 234–241. https://doi.org/10.1007/978-3-319-24574-4_28.

[20] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, *MobileNetv2: Inverted Residuals and Linear Bottlenecks*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4510–4520. https://doi.org/10.1109/CVPR.2018.00474.

[21] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, *LSD: A Fast Line Segment Detector with a False Detection Control*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 32 (2008), pp. 722–732. https://doi.org/10.1109/TPAMI.2008.300.

[22] J. Xiao, K. A. Ehinger, A. Oliva, and A. Torralba, *Recognizing Scene Viewpoint Using Panoramic Place Representation*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2012, pp. 2695–2702. https://doi.org/10.1109/CVPR.2012.6247991.

[23] Y. Xu, W. Xu, D. Cheung, and Z. Tu, *Line Segment Detection Using Transformers Without Edges*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 4257–4266. https://doi.org/10.1109/CVPR46437.2021.00424.

[24] N. Xue, S. Bai, F. Wang, G.-S. Xia, T. Wu, and L. Zhang, *Learning Attraction Field Representation for Robust Line Segment Detection*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 1595–1603. https://doi.org/10.1109/CVPR.2019.00169.