arXiv:2212.07766v1 [cs.CV] 15 Dec 2022

DeepLSD: Line Segment Detection and Refinement with Deep Image Gradients

Rémi Pautrat[1] Daniel Barath[1] Viktor Larsson[2] Martin R. Oswald[1,3] Marc Pollefeys[1,4]
[1] Department of Computer Science, ETH Zurich [2] Lund University [3] University of Amsterdam
[4] Microsoft Mixed Reality and AI Zurich Lab

Abstract

Line segments are ubiquitous in our human-made world and are increasingly used in vision tasks. They are complementary to feature points thanks to their spatial extent and the structural information they provide. Traditional line detectors based on the image gradient are extremely fast and accurate, but lack robustness in noisy images and challenging conditions. Their learned counterparts are more repeatable and can handle challenging images, but at the cost of a lower accuracy and a bias towards wireframe lines. We propose to combine traditional and learned approaches to get the best of both worlds: an accurate and robust line detector that can be trained in the wild without ground truth lines. Our new line segment detector, DeepLSD, processes images with a deep network to generate a line attraction field, before converting it to a surrogate image gradient magnitude and angle, which is then fed to any existing handcrafted line detector. Additionally, we propose a new optimization tool to refine line segments based on the attraction field and vanishing points. This refinement improves the accuracy of current deep detectors by a large margin. We demonstrate the performance of our method on low-level line detection metrics, as well as on several downstream tasks using multiple challenging datasets. The source code and models are available at https://github.com/cvg/DeepLSD.

1. Introduction

Line segments are ubiquitous in human-made environments and encode the underlying scene structure in a compact way. As such, line features have been used in multiple vision tasks: 3D reconstruction and Structure-from-Motion (SfM) [18, 34, 36], Simultaneous Localization and Mapping [14, 16, 27, 40, 66], visual localization [15], tracking [41], vanishing point estimation [52], etc. Thanks to their spatial extent and presence even in textureless areas, they offer a good complement to feature points [15, 16, 40].

All these applications require a robust and accurate detector to extract line features from images. Traditionally, line segments are extracted from the image gradient using handcrafted heuristics, such as in the Line Segment Detector (LSD) [54]. These methods are fast and very accurate since they rely on low-level details of the image. However, they can suffer from a lack of robustness in challenging conditions such as in low illumination, where the image gradient is noisy. They also miss global knowledge from the scene and will detect any set of pixels with the same gradient orientation, including uninteresting and noisy lines.

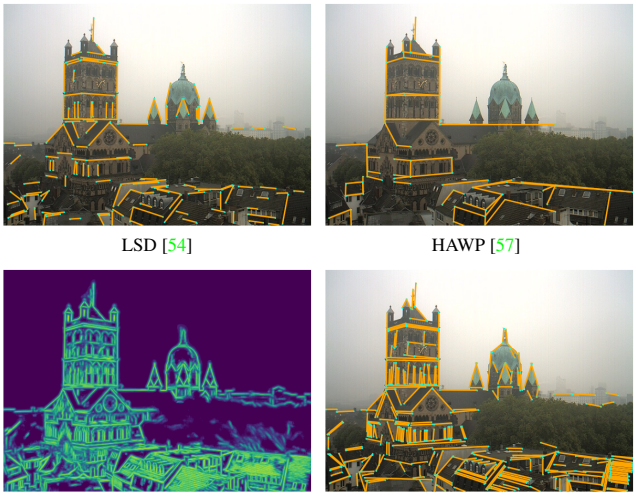


Figure 1. **Line detection in the wild. Top row:** on challenging images, handcrafted methods such as LSD [54] suffer from noisy image gradients, while current learned methods like HAWP [57] were trained on wireframe images and generalize poorly. **Bottom row:** we combine deep learning to regress a line attraction field and a handcrafted detector to get both accurate and robust lines.

Recently, deep networks offer new possibilities to tackle these drawbacks. This resurgence of line detection methods was initiated by the deep wireframe methods aiming at inferring the line structure of indoor scenes [20, 32, 56, 57, 65]. Since then, more generic deep line segment detectors have been proposed [10, 17, 21, 30, 53], including joint line detectors and descriptors [1, 39, 59]. These methods can, in theory, be trained on challenging images and, thus, gain robustness where classical methods fail. As they require a large receptive field to be able to handle the extent of line segments in

an image, they can also encode some image context and can distinguish between noisy and relevant lines. On the other hand, most of these methods are *fully* supervised and there exists currently only a single dataset with ground truth lines, the Wireframe dataset [20]. Initially designed for wireframe parsing, this dataset is biased towards structural lines and is limited to indoor scenes. Therefore, it is not a suitable training set for generic line detectors, as illustrated in Figure 1. Additionally, similarly as with feature points [33, 48], current deep detectors are lacking accuracy and are still outperformed by handcrafted methods on easy images. The exact localization of line endpoints is often hard to obtain, as lines can be fragmented and suffer from partial occlusion. Many applications using lines consequently consider infinite lines and ignore the endpoints [36].

Based on this assessment, we propose in this work to keep the best of both worlds: use deep learning to process the image and discard unnecessary details, then use handcrafted methods to detect the line segments. We thus retain the benefits of deep learning, namely, to abstract the image and gain more robustness to illumination and noise, while at the same time retaining the accuracy of classical methods. We achieve this goal by following the tracks of two previous methods that used a dual representation of line segments with attraction fields [56, 57]. The latter are continuous representations that are well-suited for deep learning, and we show how to leverage them as input to the traditional line detectors. Contrary to these two previous methods, we do not rely on ground truth lines to train our line attraction field, but propose instead to bootstrap existing methods to create a high-quality pseudo ground truth. Thus, our network can be trained on any dataset and be specialized towards specific applications, which we show in our experiments.

We additionally propose a novel optimization procedure to refine the detected line segments. This refinement is based on the attraction field output by the proposed network, as well as on vanishing points, optimized together with the segments. Not only can this optimization be used to effectively improve the accuracy of our prediction, but it can also be applied to other deep line detectors.

In summary, we propose the following contributions:

- We propose a method **bootstrapping current detectors to create ground truth line attraction fields** on any image.
- We introduce an optimization procedure that can simultaneously **refine line segments and vanishing points**. This optimization can be used as a stand-alone refinement to improve the accuracy of any existing deep line detector.
- We set a new record in several downstream tasks requiring line segments by combining the **robustness of deep learning approaches** with the **precision of handcrafted methods** in a single pipeline.

2. Related Work

Handcrafted Line Detectors. Detecting line segments in images is traditionally performed based on the image gradient. Early methods threshold the gradient magnitude to keep only strong edges and search for aligned sets of pixels sharing the same gradient angle. LSD [54] grows line regions, fits a rectangle to the resulting set of pixels, and finally extract a line segment. EDLines [4] grows the line regions in one direction only, orthogonal to the image gradient. Several extensions of these methods have been proposed, such as the multi-scale version of LSD, MLSD [44], and ELSED [51], a faster version of EDLines which avoids breaking lines in case of small discontinuities. AG3Line [61] proposes to actively group the seed points and adds line geometry constraints. Another approach consists in detecting full lines with the Hough transform [19] in a first step, then finding segments within these lines [13]. Since all these methods rely on low-level details of the image, they are highly accurate and fast, but lack robustness to noise and low illumination.

Learned Line Detectors. Deep line detection was first introduced through the task of wireframe parsing, i.e. estimating the structural lines of a scene [20]. Several approaches have been proposed to parameterize and represent the line segments, e.g., with two endpoints [65], attraction fields [56, 57], center and offset to the endpoints [21], graphs [35, 62], and transformers [55]. Wireframes can be further improved through a Deep Hough transform [32]. All these methods are trained on a single dataset, the Wireframe dataset [20], and they are not necessarily suitable for other tasks such as visual localization and SfM. Generic deep line segment detectors have also been proposed, with a focus on efficiency [10, 17], and can improve visual localization with points and lines [15]. However, these methods are again trained solely on the Wireframe dataset and their predicted lines are biased towards structural lines and indoor scenes.

Some works also perform a joint detection and description of line segments. SOLD2 [39] introduced a self-supervised training, using the homography adaptation technique initially described in SuperPoint [12]. ELSD [59] and L2D2 [1] both propose similar networks, but ELSD is again trained on the Wireframe dataset, while L2D2 uses a novel process to extract a line ground truth from LiDAR scans. Though these approaches are a first step towards unsupervised line detection, they still lack accuracy.

Attraction Fields. This work proposes to combine deep learning methods with classical line extractors. The key component for this is to use a dual representation of lines through an attraction field. This representation was first introduced by Xue et al. [56] for the wireframe task, and later improved with HAWP [57, 58]. They represent the set of discrete lines of an image with a continuous 2D vector field, suitable for deep networks. We adopt a similar approach, with small

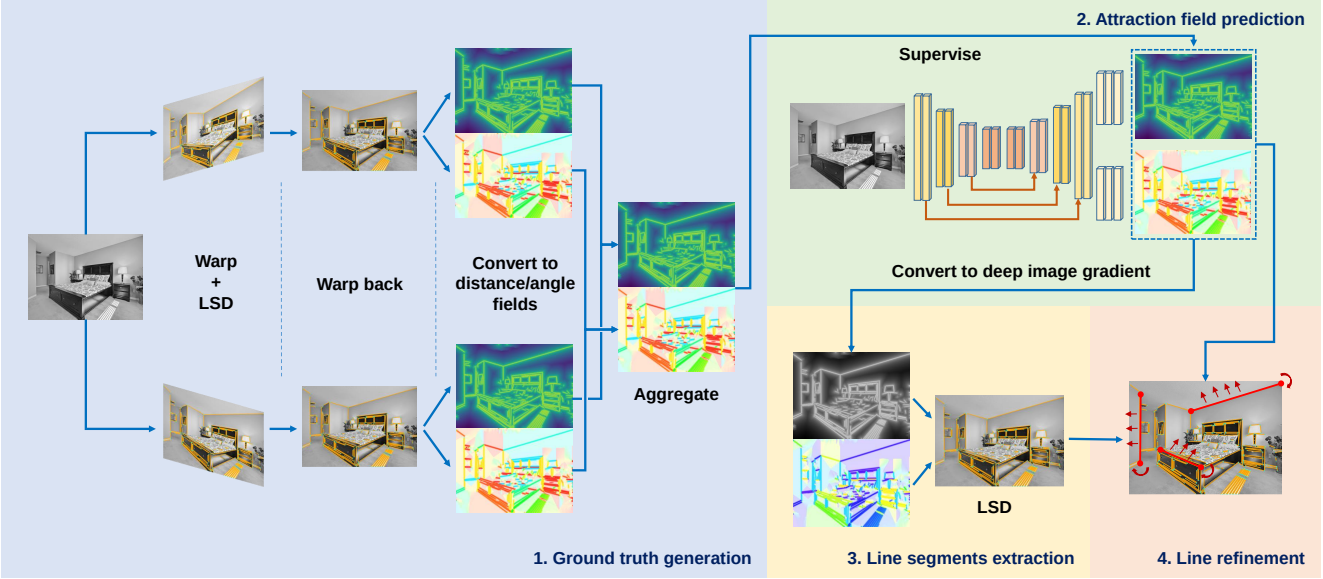


Figure 2. **Overview of the method.** (1) We generate ground truth line distance and angle fields (DF/AF) by bootstrapping LSD [54]. (2) A deep network is trained to predict the DF/AF, which is then converted to a surrogate image gradient. (3) Line segments are extracted with LSD and (4) refined based on the DF/AF.

modifications to make the prediction more accurate. While not exactly an attraction field, Teplyakov et al. [53] also proposed to predict a line mask and line angle field with a network, then used LSD [54] to get line segments. Our method obtains better accuracy by predicting a distance field instead of a simple binary mask. Attraction fields have also been leveraged for keypoint detection [22], where 2D vectors are voting for the closest keypoint in the image. These detections-by-voting offer a convenient way to represent discrete quantities through continuous ones, and are also a key aspect of our approach when it comes to generating a reliable ground truth for line detection.

3. Hybrid Line Detector

We demonstrate how to combine the robustness of deep networks together with the accuracy of handcrafted line detectors. We train a deep network to predict a line attraction field, convert it to a surrogate image gradient, and feed it to a handcrafted line detector to obtain the segments. Finally, an optimization based on the attraction field is used to refine the lines, as depicted in Figure 2.

3.1. Line Attraction Field

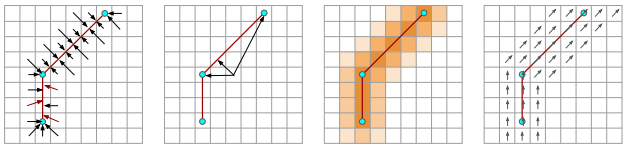
Representing line segments through an attraction field was first proposed by Xue et al. [56]. They initially proposed to regress a 2D vector field for each pixel of an image, indicating the relative position of the closest point on a line. This approach allows to represent discrete quantities (the line segments) as a smooth 2-channel image well suited for deep learning. In [57], the authors enriched the attraction field by adding two angles pointing at the endpoints of the closest line. Recovering the original segments from the attraction field is then straightforward.

However, this representation is not optimal to obtain accurate line segments, as illustrated in Figure 3. Directly predicting the position of the endpoints as done in HAWP [57] requires a larger receptive field to be able to get information from far-away endpoints, so that the network will focus on higher-level details instead of low-level ones. Additionally, deep networks are still struggling to yield accurate keypoint detections [33,48], which holds even more for line endpoints, which are notoriously noisy and unstable. On the contrary, handcrafted methods such as LSD [54] are very low-level and gradually grow a line, so that endpoints are recovered only at the end of the region growing process. In this work, we propose to restrict our network to a smaller receptive field and to let the traditional heuristics determine the endpoints.

We adopt a similar attraction field representation as HAWP [57] but without the additional two angles pointing at endpoints, yielding only a *line distance field* (DF) and a *line angle field* (AF). For every pixel in these two images, the line distance field $\mathcal{D}$ gives the distance from the current pixel to the closest point on a line, and the line angle field $\mathcal{A}$ returns the orientation of the closest line. These two quantities can be easily obtained from the 2D offset field $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{H \times W} \times \mathbb{R}^{H \times W}$ pointing at the closest point on a line, where (H, W) are the dimension of the image:

$$\mathcal{D} = \sqrt{\mathbf{x}^2 + \mathbf{y}^2}, \qquad \mathcal{A} = \arctan \frac{\mathbf{y}}{\mathbf{x}} + \pi/2 \mod \pi \,. \tag{1}$$

We define here the line angle modulo π so that a pixel above



(a) AFM [56] (b) HAWP [57] (c) Distance field (d) Angle field

Figure 3. **Attraction field parametrizations.** (a) Parametrizing with 2D vectors may produce noisy angles for small vector norms. (b) Adding offsets to the endpoints requires long-range information and is not robust to noisy endpoints. We propose to decouple the distance field (c) and line orientation field (d).

or below a line would have the same angle. Adopting this parametrization has the advantage of separating the norm from the angle of the 2D offset. Traditional detectors are leveraging the image gradient magnitude and angle, so we adopt a similar representation. Furthermore, both quantities are continuous close to line segments, and the line angle is even constant close enough to a line.

3.2. Ground Truth Generation

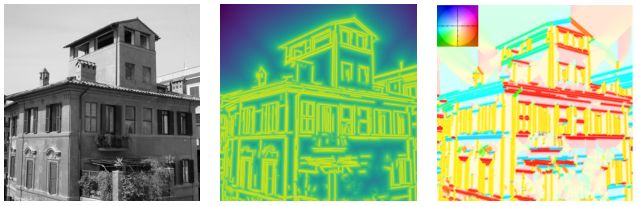
To learn the attraction field, a ground truth is needed. Both AFM [56] and HAWP [57] are supervised with the ground truth lines of the Wireframe dataset [20]. We explore a novel method to acquire our ground truth, by bootstrapping previous line detectors. Inspired by SuperPoint [12] and SOLD2 [39], we propose to generate the ground truth attraction field through *homography adaptation*. Given a single input image I, we warp it with N random homographies H_i, detect line segments in all the warped images I_i using any existing line detector, and then warp back the segments into I to get a set L_i of lines. We use LSD [54] to extract lines as it is currently among the most accurate existing line detector. The next step is to aggregate all the detections together, however, aggregating discrete quantities such as lines is non trivial. SOLD2 [39] proposed to aggregate the endpoints and line heatmaps, and recover the segments afterwards. Instead, we propose to convert the sets of lines L_i into a distance field $\mathcal{D}_i$ and angle field $\mathcal{A}_i$, and to aggregate them by taking the median value of each pixel (u, v) across all images:

$$\begin{cases} \mathcal{D}(u,v) = \text{median}_{i\in[1,N]}\mathcal{D}_i(u,v) \\ \mathcal{A}(u,v) = \text{median}_{i\in[1,N]}\mathcal{A}_i(u,v) \end{cases} . \tag{2}$$

By taking the median, we remove the noisy lines that were detected in only a few images, as shown in Figure 4.

3.3. Learning the Line Attraction Field

To regress our line distance and angle fields, we leverage a UNet-like neural network architecture [43]. The input image of size (H, W) is processed by several convolutional layers and gradually downsampled up to a factor of 8 through 3 successive average pooling operations. The features are then upscaled back to the original resolution through another series of convolutional layers and bilinear interpolation. The



Input image Distance field Angle field

Figure 4. **Pseudo GT visualization.** Given an input image, we generate a line distance and angle fields (color coded [5]) and use them to supervise a deep network. Noisy lines, such as the ones in the bush at the bottom, are averaged out and ignored.

resulting deep features are then split into two branches, one outputting the distance field $\hat{\mathcal{D}} \in \mathbb{R}^{H\times W}$ and the other one the angle field $\hat{\mathcal{A}} \in \mathbb{R}^{H\times W}$. Refer to Figure 2 and supp. material for the detailed architecture.

While all convolutions are followed by ReLU [2] and Batch Normalization [23], the last two outputs have different activations. The angle field is obtained through a sigmoid activation and is multiplied by π to get an angle within $]0, \pi[$. Since the distance field can get very small values close to lines, where we also want the highest accuracy, we adopt a special normalization. The distance field branch ends with a ReLU activation and outputs a normalized distance field $\hat{\mathcal{D}}_n \in (\mathbb{R}^+)^{H\times W}$. The final distance field is obtained through the following denormalization:

$$\hat{\mathcal{D}} = r \cdot e^{-\hat{\mathcal{D}}_n} , \tag{3}$$

where r is a parameter in pixels that defines a region around each line. Since handcrafted methods mainly need gradient information close to line segments, we supervise our network only on pixels at a distance of less than r pixels from a line. By selecting a small value for r, large portions of the image may not have any supervision, including areas where the pseudo ground truth was not able to detect real lines, e.g. lines with small contrast. Enforcing these lines to be in the background during training, i.e. with high distance field, provides a detrimental training signal and decreases the recall of the prediction. On the contrary, with our loose supervision, these low contrast lines are not penalized during training and our trained model can detect them, thus yielding a more complete prediction than the ground truth.

We compute the training loss by comparing with a normalized version of the ground truth: $\mathcal{D}_n = -\log\left(\frac{\mathcal{D}}{r}\right)$. Note that since we only supervise pixels with a distance field below r, $\frac{\mathcal{D}}{r} \in [0, 1]$ and so $\mathcal{D}_n \in \mathbb{R}^+$. We compute the total loss as the sum of the losses for the distance field and the angular field:

$$\mathcal{L} = \mathcal{L}_D + \mathcal{L}_A , \tag{4}$$

where $\mathcal{L}_D$ is an L1 loss between the normalized distance fields and $\mathcal{L}_A$ is an L2 angular loss that takes the circularity of the angles into account for the given predicted and ground

(a) A double edge

(b) HAWP [57]

(c) Ours

Figure 5. **Distinguishing double edges.** (a) An example of a bright-dark-bright edge and the oriented angle field. (b) Wireframe methods treat it as a single line. (c) We detect it as two lines for better accuracy.

truth angle fields $\hat{\mathcal{A}}, \mathcal{A} \in [0, \pi]^{H \times W}$:

$$\begin{aligned} \mathcal{L}_D &= ||\hat{\mathcal{D}}_n - \mathcal{D}_n||_1 \ , \\ \mathcal{L}_A &= \min(||\hat{\mathcal{A}} - \mathcal{A}||_2, ||\pi - |\hat{\mathcal{A}} - \mathcal{A}|||_2) \ . \end{aligned} \tag{5}$$

3.4. Extracting Line Segments

Since handcrafted detectors are based on the image gradient, we propose to convert our distance and angle fields into a surrogate image gradient magnitude $\mathbf{M}$ and angle θ:

$$\begin{cases} \mathbf{M} & = r - \hat{\mathcal{D}} \\ \theta & = \hat{\mathcal{A}} - \frac{\pi}{2} \end{cases} \ . \tag{6}$$

Our predicted angle follows the directions of the lines and is perpendicular to the image gradient, so we rotate it by $\frac{\pi}{2}$. The maximal magnitude of a pixel on a line is r.

An important difference between the approaches of AFM and LSD is the gradient orientation. For an edge separating a dark from a bright area, LSD keeps track of the dark-to-bright gradient direction, while AFM does not. This becomes important when several parallel lines occur next to each other in a dark-bright-dark or bright-dark-bright pattern, as illustrated in Figure 5. For better accuracy and scale-invariance, we advocate to detect these double edges and make our predicted angle *oriented*, based on the sign of the image gradient angle θ_I:

$$\theta_o = \begin{cases} \theta & \text{if } d(\theta, \theta_I) < d(\theta - \pi, \theta_I) \\ \theta - \pi & \text{otherwise} \end{cases} \ , \tag{7}$$

where $d(\cdot, \cdot)$ is a circular distance between two angles. Now equipped with an oriented angle θ_o and magnitude $\mathbf{M}$, we can directly apply any existing classical line segment detector. Unless stated otherwise, we always use the LSD [54] approach in the following, due to its high accuracy. In summary, the purpose of the deep net is to suppress image noise and detect low-contrast lines, while the line segments are accurately extracted by LSD afterwards.

We also add a filtering step, leveraging the DF and AF. We sample n_f points along each line, and compute the fraction p of samples whose distance function is below η_{DF} and angle is close enough to the line orientation with tolerance η_θ. Only the segments with enough inliers are kept.

3.5. Line Segment Refinement with Optimization

To make lines even more accurate, we propose an optimization step to refine them by leveraging the predicted DF and AF. This refinement can also be used to enhance the lines of any other detector, and we show in Section 4.4 how it can make current deep detectors much more precise.

While lines are detected independently, they usually appear in highly structured configurations in the image. In particular, lines that are parallel in 3D will share vanishing points. We propose to integrate this as soft constraints into our refinement, effectively reducing the degrees of freedom.

We first compute a set of vanishing points (VPs) associated with the predicted line segments, using the multi-model fitting algorithm Progressive-X [7]. We use a strict inlier threshold to be sure to associate only relevant lines to a VP. The optimization is then performed independently for each line and is a weighted unconstrained least square minimization of three different costs:

$$\mathcal{C} = \lambda_A \mathcal{C}_A + \lambda_D \mathcal{C}_D + \lambda_V \mathcal{C}_V \ . \tag{8}$$

Given a set P of n_{opt} points uniformly sampled along a line segment l, we denote each point by p_i, the orientation angle of the line as θ_l, and the VP associated with the line as $\mathbf{v}_l$. We use the following three costs:

$$\begin{aligned} \mathcal{C}_A &= \frac{1}{n_{opt}} \sum_{p_i \in P} \Big(1 - \big(\cos(\hat{\mathcal{A}}(p_i) - \theta_l)\big)\Big) \ , \\ \mathcal{C}_D &= \frac{1}{n_{opt}} \sum_{p_i \in P} \hat{\mathcal{D}}(p_i) \ , \qquad \mathcal{C}_V = d_{\text{VP}}(l, \mathbf{v}_l) \ , \end{aligned} \tag{9}$$

where d_{VP} is a distance measure between a line and a VP. We adopt the perpendicular distance of the line endpoints, projected onto the infinite line passing through the center of the line and the VP, as in [52]. These objectives are thus minimizing the difference between the sampled angle $\hat{\mathcal{A}}(p_i)$ and the line orientation angle, minimizing the average distance field value over the line, and minimizing the distance between the line and its VP. In case the closest VP is farther away from the line than a threshold t_{VP}, we drop the VP constraint as it would push the line towards a wrong VP. To avoid lines drifting or collapsing to a single point, we keep the length of the line fixed, and we only optimize the lines

Figure 6. **Line detection examples.** Wireframe methods [21, 57] only detect structural lines, while DeepLSD offers more generic detections.

over two degrees of freedom: the orientation angle of the line θ_l, and a translation of the middle point in the perpendicular direction of the line.

Since the VPs are already computed, we can even optimize the VPs as well, as a by-product of our approach. Jointly optimizing lines and VPs empirically led to inferior results, mainly because some lines require more refinement than others, so that a global refinement performs worse than independently optimizing the lines. We alternate, instead, between refining the lines and refining the VPs, for a fixed number of iterations k. The VP refinement is performed through a least square minimization of the distance d_{VP} between the VP and all associated lines, and the line-VP association is recomputed after each iteration.

3.6. Implementation Details

We train two versions of our network, one indoors on the Wireframe dataset [20], but without using the ground truth lines, and one outdoors on MegaDepth [31]. Given the large size of MegaDepth, we keep 150 scenes for training and 17 for validation, and only sample 50 images from each scene. We use the Adam optimizer [24] and an initial learning rate of $1e^{-3}$, which is divided by 10 each time the validation loss reaches a plateau. The training takes roughly 12 hours on a single NVIDIA RTX 2080 GPU. For the line detection, we set the line region r to 5 pixels and ignore magnitudes in $\mathbf{M}$ below 3 when applying LSD. We use $n_f = 50$ samples in the filtering step, $\eta_{\text{DF}} = 1.5$, $\eta_\theta = \frac{\pi}{9}$ and accept lines with more than 50% inliers. The parameters for VP estimation are tuned for each method on a validation set, but the usual threshold t_{VP} ranges from 1 to 2 pixels. The optimization weights are empirically chosen as $\lambda_D = 1$, $\lambda_A = 1$, and $\lambda_V = 0.2$. We adopt $n_{opt} = 10$ samples, perform a fixed set of $k = 5$ alternating iterations, and optimize with Ceres [3].

4. Experiments

To evaluate the performance of our method, we cannot use labeled lines as the existing ones are usually biased towards wireframes. We are more interested in evaluating the potential to use these lines for downstream applications, such as homography estimation, 3D line reconstruction, and visual localization. We also provide a visual comparison of various line detectors in Figure 6.

4.1. Evaluation on Low-Level Metrics

We first evaluate our line detection on two challenging datasets to test the robustness of the methods. First, the HPatches dataset [6], consisting of 580 pairs of images with ground truth homographies relating them and varying illumination and viewpoint changes. Second, the RDNIM dataset [38], also with image pairs related by a homography and with challenging day-night variations. We use the night reference in our experiments to get more challenging pairs.

Similarly as in [39], we assess the *repeatability* and *localization error* metrics. For both metrics, we compute a one-to-one matching of the detected line segments between the two images of a pair using the ground truth homography. For each match, one can then compute the distance between the line in the reference image and the line of the warped image reprojected into the reference frame. We consider two line distance measures: the structural distance evaluating the average distance between the endpoints, and the orthogonal distance measuring the average distance of each endpoint of one line to their orthogonal projection to the other line. Repeatability (Rep) measures the ratio of lines whose match has an error below 3 pixels, and the localization error (LE) returns the average distance of the 50 most accurate matches.

We also compute a *homography estimation score*, similarly as in [12]. We first match line segments between the two images, using the Line Band Descriptor (LBD) [60]. To estimate the homography, we sample minimal sets of 4 line matches and run LO-RANSAC [29] for up to 1M iterations, using the orthogonal line distance as reprojection error.

We compare in Table 1 our method to two classical detectors: LSD [54] and ELSED [51]; the best methods using attraction fields: HAWP [57], its recent update HAWPv3 trained in a self-supervised way [58], and LSDNet [53]: a similar approach as ours combining LSD and a deep network; and two generic deep line detectors: TP-LSD [21] and SOLD2 [39]. We use the implementation of the authors with the biggest model available and default parameters, except for HAWP where we use a threshold of 0.9, as it was not detecting enough lines otherwise. HAWPv3 was trained on ImageNet. For LSD, we use the implementation of Rafael Grompone[1] instead of the OpenCV one as it gets much better results. Our method is given without the final optimization in the following, unless otherwise specified.

[1] http://www.ipol.im/pub/art/2012/gjmr-lsd/

			Traditional		Learned				Hybrid		Traditional		Learned				Hybrid	
			LSD [54]	ELSED [51]	HAWP [57]	HAWPv3 [58]	TP-LSD [21]	SOLD2 [39]	LSDNet [53]	DeepLSD (Ours)	LSD [54]	ELSED [51]	HAWP [57]	HAWPv3 [58]	TP-LSD [21]	SOLD2 [39]	LSDNet [53]	DeepLSD (Ours)
HPatches [6]	Struct	Rep ↑	0.314	0.240	0.330	0.272	**0.413**	0.308	0.108	0.367	0.283	0.209	0.284	0.320	**0.344**	0.307	0.047	0.285
		LE ↓	1.309	1.551	2.019	2.132	1.500	1.741	2.860	**1.235**	2.039	2.303	2.206	1.939	1.779	1.879	3.331	**1.733**
	Orth	Rep ↑	0.468	0.465	0.337	0.309	0.444	0.395	0.200	**0.485**	**0.403**	0.392	0.284	0.354	0.377	0.386	0.130	0.394
		LE ↓	**0.793**	0.845	1.905	1.937	1.305	1.362	2.285	0.818	1.369	1.248	2.215	1.704	1.625	1.449	2.752	**1.098**
	H estimation ↑		0.697	0.617	0.260	0.231	0.388	0.421	0.316	**0.705**	0.468	0.200	0.006	0.026	0.030	0.182	0.027	**0.591**
	# lines / img		492.6	425.4	53.6	82.0	88.6	122.9	172.1	486.2	191.4	112.0	31.6	23.8	24.1	138.2	109.1	400.0
	Time [ms] ↓		104	**10**	61	51	179	334	48	271	34	**3**	42	47	75	199	44	96

RDNIM [38]

Table 1. **Line detection evaluation on the HPatches [6] and RDNIM [38] datasets.** We compare repeatability (Rep) and localization error (LE) in structural and orthogonal distances, together with homography estimation. We get the best score on homography estimation and a good trade-off between classical and learned methods for the all metrics. The best score is in bold and the second best is underlined.

From the results, the learned methods, led by TP-LSD [21], offer good repeatability, but suffer from a low localization error and inaccurate homography estimation. Handcrafted methods and our method are much more accurate, due to the fact that they do not directly regress the endpoints, but gradually grow the line segments using very low-level details. DeepLSD displays the best improvement over LSD when the changes become the most challenging, i.e. on RDNIM with strong day-night changes. It can significantly improve the localization error and homography estimation score. In spite of having a similar approach as ours, LSDNet [53] performs poorly for multiple reasons: they lose accuracy by rescaling images to a fixed low resolution, their line mask is less precise than our distance field, and their training is limited to the Wireframe dataset, while ours can be trained on more diverse images. Overall, our method offers the best trade-off between handcrafted and learned methods and consistently ranks first in the downstream task of homography estimation.

4.2. 3D Line Reconstruction

The aim of this work is to provide general-purpose lines and as such, the lines generated by DeepLSD should be suitable for 3D reconstruction. We leverage Line3D++ [18] that takes a collection of images with known poses and the associated 2D line segments, and outputs a 3D reconstruction of lines. We propose to compare our method with a few baselines on the first 4 scenes of the Hypersim dataset [42]. This synthetic - but highly realistic - dataset has the advantage of offering a ground truth mesh and 3D model, making it suitable for a quantitative evaluation. Given the ground truth mesh of the scene, we can compute the recall and precision of the 3D lines. Recall is the length in meters of all the portions of lines that are within 5 millimeters from the mesh. High values mean that many lines have been reconstructed. Precision is the percentage of predicted lines that are within 5 millimeters from the mesh. High values indicate that most of the predicted lines are on a real 3D surface.

The results can be seen in Table 2. DeepLSD obtains the best recall overall, and second best precision. While

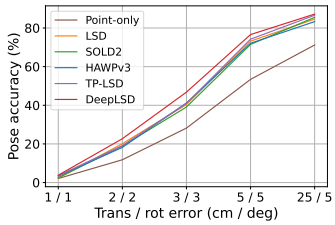
	ai_001_001		ai_001_002		ai_001_003		ai_001_004		Average	
	R	P	R	P	R	P	R	P	R	P
LSD [54]	183.6	95.8	61.8	95.3	**385.0**	88.9	225.3	91.5	213.9	92.9
SOLD2 [39]	109.9	94.7	89.3	92.8	62.0	89.0	58.6	89.1	80.0	91.4
HAWPv3 [58]	15.8	79.9	15.6	81.0	24.4	68.4	18.5	77.3	18.6	76.7
TP-LSD [21]	68.8	95.3	38.9	94.7	50.7	**98.2**	102.7	**94.3**	65.3	**95.6**
DeepLSD	**204.8**	**96.5**	**89.5**	**98.1**	378.8	88.0	**231.1**	91.9	**226.1**	93.6

Table 2. **Line 3D reconstruction evaluation.** We reconstruct lines in 3D with Line3D++ [18] and evaluate the line length recall in m (R ↑) and precision (P ↑) on the first 4 scenes of Hypersim [42].

TP-LSD [21] ranks first in precision, it is able to recover very few lines, as shows its average recall, which is 71% smaller than the one of DeepLSD. We provide qualitative examples of the reconstructions in the supp. material. Note that DeepLSD is able to reconstruct more lines and with a higher precision than LSD [54], the detector that is the most commonly used for line reconstruction [18].

4.3. Visual Localization

The 7Scenes dataset [50] is a well-known RGB-D dataset for visual localization, displaying 7 indoor scenes with GT poses and depth. While most scenes are already saturated for point-based localization, the Stairs scene remains very challenging for feature points. Due to the lack of texture and repeated patterns of the stairs, current point-based methods are still struggling on this scene [8]. We thus propose to evaluate our method and previous works on this particular scene, by following the pipeline of hloc [45, 46], enriched with line features. As points remain important features, we still detect SuperPoint features [12] and match them with SuperGlue [47]. We detect lines with different detectors, and match them between database and query images with the SOLD2 descriptor [39]. Since depth is available on 7Scenes, we can directly back-project lines in 3D and do not rely on line mapping. In practice, we sample points along each line, un-project them to 3D, and re-fit a line in 3D to these un-projected points. We use the solvers of [26,28,64] to generate poses from a minimal set of 3 features (3 points, 2 points and 1 line, 1 point and 2 lines, or 3 lines), then combine them in a hybrid RANSAC implementation [9, 49] to robustly recover the query camera poses. We report the median translation



	T / R err ↓	Acc ↑
Point-only	4.7 / 1.25	53.4
LSD [54]	3.4 / 0.94	73.2
SOLD2 [39]	3.5 / 0.96	71.5
HAWPv3 [58]	3.4 / 0.93	72.1
TP-LSD [21]	3.4 / 0.98	74.2
DeepLSD	**3.1 / 0.85**	**76.6**

Figure 7. **Visual localization on 7Scenes stairs [50].** We evaluate the median translation and rotation errors (cm / deg), the pose accuracy at a 5 cm / 5 deg threshold, and plot the pose accuracy curve for various thresholds.

		Struct		Orth		H estim	# lines / img	Time [ms] ↓
		Rep ↑	LE ↓	Rep ↑	LE ↓			
HAWP [57]	Baseline	0.253	1.34	0.253	1.43	0.701		**40**
	Opt w/o VP	0.300	1.293	0.399	1.067	0.864	95.2	142
	Opt w/ VP	**0.318**	**1.245**	**0.431**	**0.967**	**0.892**		300
TP-LSD [21]	Baseline	0.273	1.379	0.342	1.269	0.658		**46**
	Opt w/o VP	0.314	1.326	0.470	0.949	0.898	90.8	145
	Opt w/ VP	**0.331**	**1.277**	**0.512**	**0.861**	**0.913**		297
SOLD2 [39]	Baseline	**0.197**	**1.277**	0.333	0.894	0.848		**297**
	Opt w/o VP	0.172	1.388	0.339	0.814	**0.935**	166.7	426
	Opt w/ VP	0.185	1.330	**0.368**	**0.753**	0.920		697
DeepLSD (Ours)	Baseline	0.318	0.941	0.489	0.574	0.991		**68**
	Opt w/o VP	0.314	0.938	0.482	0.575	**0.994**	168.8	154
	Opt w/ VP	**0.319**	**0.927**	**0.501**	**0.544**	0.981		542

Table 3. **Line refinement on the Wireframe dataset [20].** We use an error threshold of 1 pixel for the repeatability metrics. The refinement can significantly improve the localization error and homography score of inaccurate methods.

and rotation error, as well as the percentage of successfully recovered poses under various thresholds.

Figure 7 shows that DeepLSD obtains the best performance on this challenging dataset. One can highlight the large boost of performance brought by line features compared to using points only. Lines are indeed still present and well localized in indoor environments such as in this scene, and can be matched even when in low-textured scenes.

4.4. Impact of the Line Refinement

We evaluate applying our proposed line refinement as a post-processing step for several learned detection methods. Classical detectors are usually already accurate enough, so that our refinement would not enhance them much. For each method, we compare the raw lines with the lines and VPs optimized by our line optimization. Table 3 shows results of line detectors on the 462 images of the test set of the Wireframe dataset [20]. The second image is obtained using a synthetic homographic warp of the first image. We use the Wireframe dataset as it has a lot of well-defined vanishing points, which can be leveraged during the optimization. We include results for our proposed optimization with and without the VP constraint to show the increased accuracy with VPs. As we want to highlight the gain in accuracy, we compute repeatability with an error threshold of only 1 pixel.

	Struct		Orth		H estim	# lines / img
	Rep ↑	LE ↓	Rep ↑	LE ↓		
Single edge	0.241	2.121	0.328	1.686	0.434	130.8
No DF normalization	0.344	1.343	0.475	0.879	0.674	439.6
HAWP with our lines	0.209	2.138	0.239	1.840	0.245	98.0
DeepLSD (Ours)	**0.367**	**1.235**	**0.485**	**0.818**	**0.705**	**486.2**

Table 4. **Ablation study on the HPatches dataset [6].** We compare DeepLSD to alternatives detecting single edges, without DF normalization and with HAWP re-trained on our line GT.

Results show that the refinement can significantly improve all metrics evaluating the accuracy of the lines, i.e. the localization error and homography estimation. This is particularly true for HAWP [57] and TP-LSD [21], with a decrease in localization error with orthogonal distance of up to 32% for both, and an improvement of homography score of 27% and 39%. The benefits brought by the refinement are lower for our method, as its raw predicted lines are already sub-pixel accurate and the optimization is limited by the resolution of the DF and AF. Nonetheless, it can slightly improve most metrics. A limitation of this refinement is the execution time, which grows linearly with the number of lines, and requires running two networks.

4.5. Ablation Study

We validate our design choices on the HPatches dataset [6] with low-level detector metrics. We compare our proposed approach with the same model detecting single edges instead of double ones, our network trained without the DF normalization, and a version of the HAWP [57] backbone re-trained on our line GT on the MegaDepth dataset [31]. The results of Table 4 emphasize the importance of each component. Note that re-training HAWP [57] on our lines yields poor results due to the higher number of lines compared to wireframe lines, and the fact that generic lines have often noisy endpoints, so that predicting an angle to the two endpoints is noisy as well.

5. Conclusion

We presented a hybrid line segment detector combining the robustness of deep learning and the accuracy of handcrafted detectors, using a learned surrogate image gradient as intermediate representation. Without the requirement of ground truth lines, our method can be trained on any dataset and is suitable for most tasks including line segments. Finally, we proposed a line refinement able to improve the accuracy of our method and to bridge the gap in line localization between deep line detectors and handcrafted ones. We believe that our general-purpose lines will open new possibilities to use line segments in the wild.

Acknowledgments. We would like to warmly thank Iago Suarez for reviewing this paper and for the insightful discussions, as well as Yifan Yu for sharing his code for visual localization.

Supplementary Material

In the following we provide additional results, insights and visualizations for DeepLSD. Section A describes in details our network architecture, Section B introduces additional ablation studies and insights about our approach, Section C provides an evaluation of visual localization with points and lines on the full 7Scenes dataset, Section D gives additional results about vanishing point estimation from the detected line segments, Section E displays visualizations of the 3D reconstruction, Section F highlights some limitations of our method, and finally Section G offers examples of the line detections.

A. Network Architecture

We provide more details about the network architecture that we used to predict attraction fields. We use a simple U-Net-like architecture [43] with several blocks of convolutions, downsampling the initial image by a factor of 8 and then upsampling it again to the initial resolution. Downsampling is performed through 3 successive 2×2 average poolings and upsampling is done with bilinear interpolation. A skip connection is added before each downsampling layer and is concatenated with the output of the corresponding upsampling layer. Please refer to Figure 8 for the detailed architecture. Each convolution layer is followed by ReLU activation [2] and Batch Normalization [23], except the final layer of each branch. The activations of the two output branches are ReLU for the distance field and Sigmoid for the angle field, without batch normalization.

B. Additional Ablation Studies

B.1. Generalization to Other Traditional Detectors

While DeepLSD is using LSD [54] as its base line detector, our approach can be applied to any other traditional detector leveraging the image gradient. We show here the results of our method using ELSED [51] as base detector (coined DeepELSED) and compare it to the original ELSED in Table 5. We give the results for the raw lines without any refinement on low-level line detection metrics on the HPatches [6] and RDNIM [38] datasets. For both traditional detectors LSD and ELSED, our deep version can improve most metrics, thanks to the additional robustness brought by the learned processing of the image.

B.2. Line Refinement on Traditional Methods

The proposed line refinement is mainly aiming at improving the accuracy of previous deep line detectors and DeepLSD, but one can wonder how it performs with traditional methods. When refining the lines output by LSD [54] and ELSED [51] on the Wireframe dataset, we did not observe any improvement in low-level metrics, except for a boost of performance in homography estimation for ELSED (see Table 6). Traditional detectors are indeed already subpixel accurate, so that the limited resolution of the distance field is not high enough to refine the lines further. The drop in performance in most metrics can be explained by the fact that some lines detected by these methods are in areas with high distance field values, so that these lines will rather drift than being optimized correctly. However, relevant lines for downstream tasks still seem to benefit from the refinement as shown by the large boost in homography estimation for ELSED.

			LSD [54]		ELSED [51]	
			Traditional	DeepLSD	Traditional	DeepELSED
HPatches [6]	Struct	Rep ↑	0.314	**0.367**	0.240	**0.263**
		LE ↓	1.309	**1.235**	**1.551**	1.585
	Orth	Rep ↑	0.468	**0.485**	0.465	**0.478**
		LE ↓	**0.793**	0.818	0.845	**0.839**
	H estimation ↑		0.697	**0.705**	0.617	**0.624**
	# lines / img		492.6	486.2	425.4	419.4
	Time [ms] ↓		**104**	271	**10**	144
RDNIM [38]	Struct	Rep ↑	0.283	**0.285**	0.209	**0.230**
		LE ↓	2.039	**1.733**	2.303	**2.258**
	Orth	Rep ↑	**0.403**	0.394	0.392	**0.407**
		LE ↓	1.369	**1.098**	**1.248**	1.361
	H estimation ↑		0.468	**0.591**	0.200	**0.221**
	# lines / img		191.4	400.0	112.0	162
	Time [ms] ↓		**34**	96	**3**	88

Table 5. **Generalization to other traditional detectors.** Our method is not limited to LSD [54], but can also be applied to the ELSED [51] line detector for example. We show the comparison between our approach and the original detectors on the HPatches [6] and RDNIM [38] datasets. The first three columns are identical to Table 1 in the main paper and our results are given without the final line refinement.

		Struct		Orth		H	# lines	Time
		Rep ↑	LE ↓	Rep ↑	LE ↓	estimation	/ img	[ms] ↓
LSD [54]	Baseline	**0.386**	**0.456**	**0.647**	**0.12**	**0.998**		**23**
	Opt w/o VP	0.332	0.593	0.485	0.35	0.994	352.1	217
	Opt w/ VP	0.332	0.589	0.494	0.325	0.994		545
ELSED [51]	Baseline	**0.185**	**1.238**	**0.564**	**0.36**	0.926		**3**
	Opt w/o VP	0.165	1.315	0.462	0.529	**0.989**	178.2	130
	Opt w/ VP	0.164	1.313	0.474	0.502	**0.989**		397

Table 6. **Line refinement of traditional methods on the Wireframe dataset [20].** The line refinement can be detrimental for some outlier lines outside of the distance field, but it is still able to improve the accuracy of most lines, as shown by the boost of performance of ELSED in homography estimation.

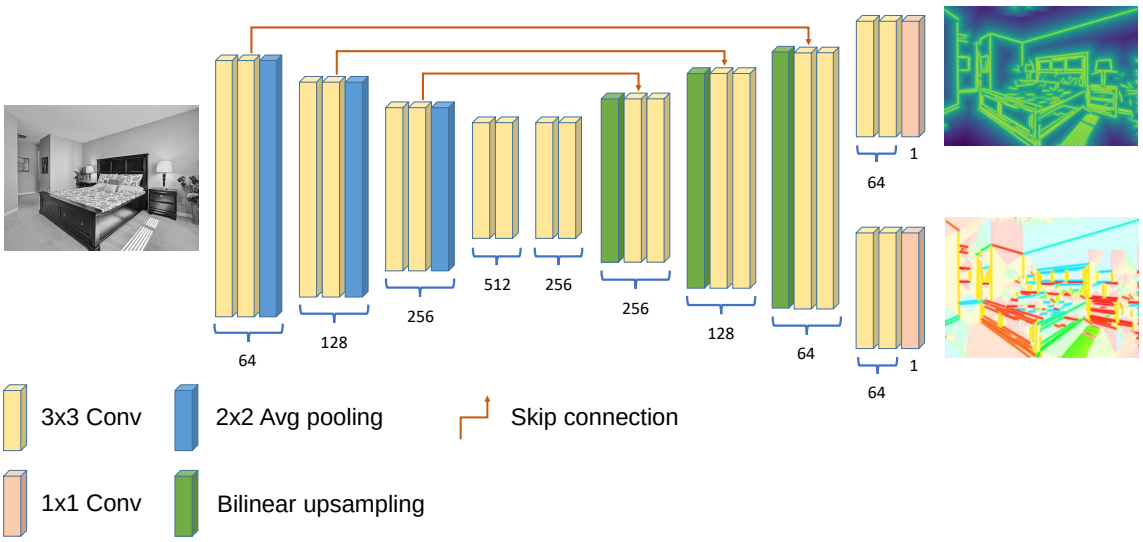


Figure 8. **Network architecture.** We use a standard UNet [43] architecture to predict the distance and angle fields.

B.3. Training Learned Baselines with our Supervision Strategy

In the main paper, we proposed an ablation study by re-training the HAWP [57] detector with our ground truth (GT) supervision. We provide here additional details and visualizations of this ablation. Instead of taking our DeepLSD approach of predicting the distance and angle fields and then applying LSD on top of it, one could also extract lines from the ground truth distance and angle fields, and then use these lines to supervise any existing deep line detector. Figure 9 shows two examples of lines detected by the original HAWP, the re-trained version using our GT lines, and DeepLSD. The latter remains the most satisfactory one, and thus justifies our approach of leveraging traditional line detectors instead of end-to-end line detection. One reason for the lower quality of the re-trained HAWP is that predicting the position of endpoints with an additional attraction field is not suitable for generic lines, as there are often too many of them in most images. This approach works better for wireframe lines, which are sparser and require less accuracy.

C. Additional Visual Localization Results

While the main paper focuses on the most challenging scene of the 7Scenes dataset [50], Stairs, we provide here the results of visual localization on the full dataset. As described in the main paper, we detect keypoints with SuperPoint [12], match them with SuperGlue [47], and build on top of hloc [45, 46] by adding line features and using them in the pose estimation. The lines are again matched with the SOLD2 [39] line detector. Table 7 displays the results of several state-of-the-art line detectors in terms of translation and rotation errors, as well as pose accuracy at a 5 cm / 5 degree threshold. DeepLSD obtains the best translation error on all scenes, as well as the best metrics on the full dataset. It can be noted that the improvement with respect to previous methods is rather small, due to the fact that 7Scenes is already very saturated for visual localization.

D. Vanishing Point Estimation

Another common application for line segments is the vanishing point (VP) estimation task. Given the line segments extracted by all the baselines and our method, we apply multi-model fitting with Progressive-X [7] to find an unconstrained number of (not necessarily orthogonal) VPs. A minimal set of 2 lines provides a VP candidate, and its consistency with the other lines is evaluated under the d_{VP} metric [52]. This distance is computed as the average orthogonal distance between the endpoints of a line segment and the infinite line going from the VP to the midpoint of the segment. Based on the inlier lines, we do a weighted least squares of the distance of all inliers to the VP, using the line length as weight. We tune the parameters of the model fitting algorithm for each method on a validation set.

We consider two benchmarks for vanishing point estimation. YorkUrbanDB [11] pictures 102 images (51 for validation and 51 for test) of urban scenes. It offers 2 or 3 ground truth VPs per image, ground truth lines, and the association between VPs and lines. Additionally, we consider the extended set of VPs proposed in YUD+ [25], which labels up to 8 VPs per image. The second dataset is adapted from the NYU Depth dataset V2 [37] by [25], consisting of 1449 images (we keep the last 49 for parameter tuning), each labelled with 1 to 8 VPs.

We consider three metrics. *VP consistency* counts the percentage of ground truth lines that are within a given threshold

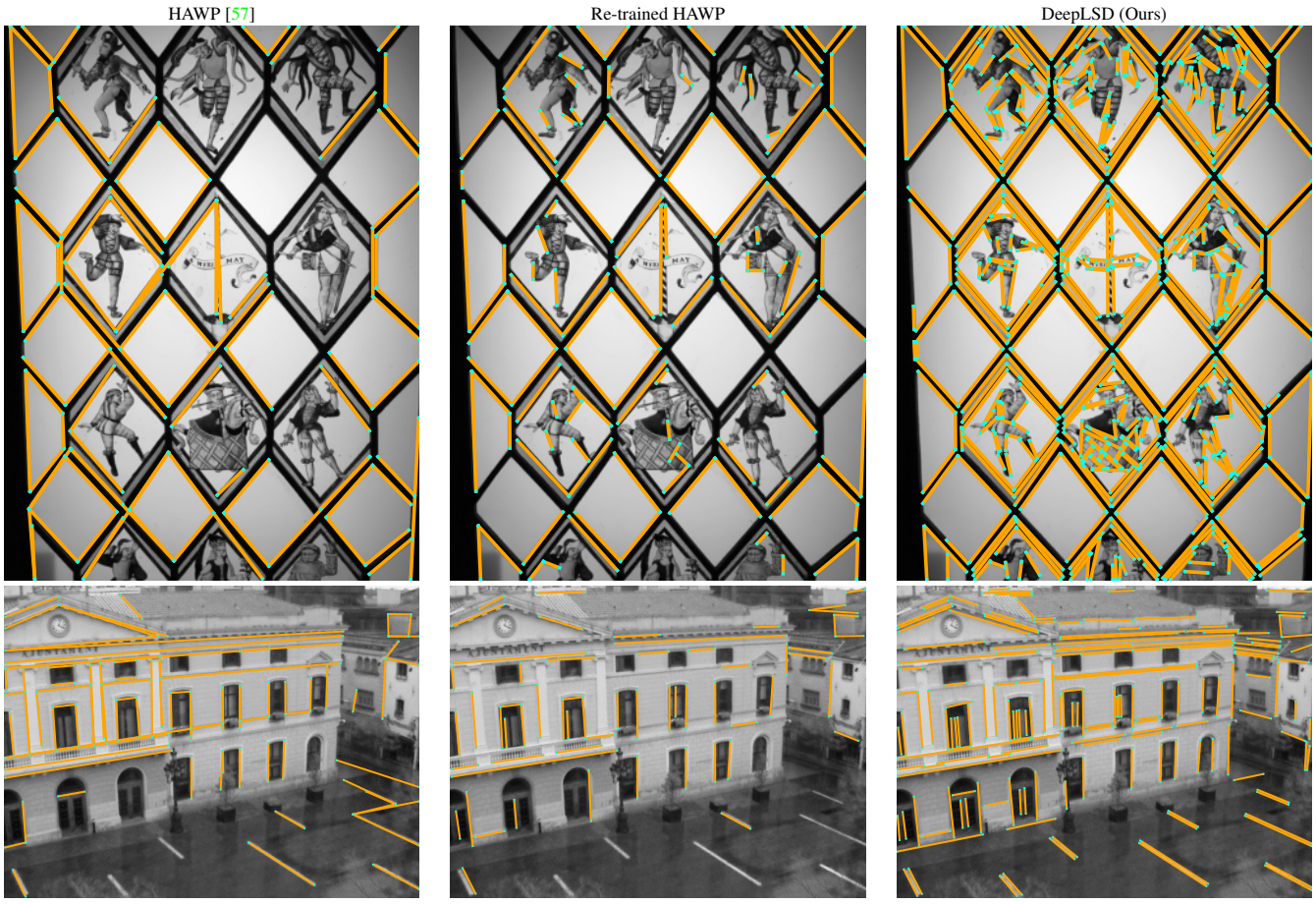


Figure 9. **Re-training the HAWP detector [57] with the proposed pseudo ground truth lines.** It yields unsatisfactory lines compared to the DeepLSD approach, mainly because detecting line endpoints with a network prediction is challenging for high densities of line segments.

	Point-only SP [12] + SG [47]	LSD [54]	SOLD2 [39]	TP-LSD [21]	HAWPv3 [58]	DeepLSD
Chess	**2.4** / 0.81 / **94.5**	**2.4** / 0.82 / 94.4	**2.4** / 0.81 / 94.0	**2.4** / **0.80** / 94.4	**2.4** / **0.80** / **94.5**	**2.4** / 0.82 / **94.5**
Fire	1.9 / 0.76 / 96.4	**1.7** / 0.73 / 96.5	1.8 / 0.76 / 95.9	1.8 / 0.76 / 95.8	1.9 / 0.77 / **97.1**	**1.7** / **0.70** / 96.7
Heads	1.1 / 0.74 / 99.0	1.1 / 0.74 / 99.4	1.1 / 0.76 / 99.3	1.1 / **0.73** / **99.5**	1.1 / 0.80 / 99.2	**1.0** / **0.73** / **99.5**
Office	2.7 / 0.83 / 83.9	**2.6** / **0.79** / 84.7	2.7 / 0.82 / 83.8	**2.6** / 0.81 / 84.1	2.7 / 0.82 / 83.8	**2.6** / 0.80 / **85.0**
Pumpkin	4.0 / 1.05 / 62.0	4.0 / 1.04 / 62.1	4.1 / 1.07 / 60.7	4.0 / 1.04 / **62.6**	4.0 / 1.04 / 62.3	**3.9** / **1.02** / 62.2
Redkitchen	3.3 / **1.12** / 72.5	**3.2** / 1.14 / 73.2	**3.2** / **1.12** / **73.5**	**3.2** / **1.12** / 73.2	3.3 / 1.13 / 73.0	**3.2** / 1.13 / 73.4
Stairs	4.7 / 1.25 / 53.4	3.4 / 0.94 / 73.2	3.5 / 0.96 / 71.5	3.4 / 0.93 / 72.1	3.4 / 0.98 / 74.2	**3.1** / **0.85** / **76.6**
Total	2.9 / 0.94 / 80.2	**2.6** / 0.89 / 83.4	2.7 / 0.90 / 82.7	**2.6** / 0.88 / 83.1	2.7 / 0.91 / 83.4	**2.6** / **0.86** / **84.0**

Table 7. **Visual localization on the 7Scenes dataset.** We report the translation error (in cm) / rotation error (in deg) / pose accuracy at a 5 cm / 5 deg threshold (in %) for the 7 scenes and the average score across all scenes.

of the predicted VPs [52]. Each set of ground truth lines is associated to a single predicted VP and each VP can be associated with at most one set of lines. We only show this metric for YorkUrbanDB as NYU does not have manually labelled lines. *VP error* measures how precise the estimated VPs are in 3D. It is the angular error between the directions in 3D of the ground truth VPs and the predicted ones. We perform again a 1:1 matching to optimally assign the predicted VPs to the ground truth ones. For each experiment, we run the VP detection algorithm 20 times and report the median results. *AUC* represents the Area Under the Curve (AUC) of the recall curve of the VPs, as described in [25]. We show the average AUC and its standard deviation over 5 runs.

Results are shown in Figure 10 and Table 8. The wireframe methods TP-LSD [21] and HAWP [57] are particularly good for vanishing point estimation, as they only detect structural lines, which are usually the only relevant ones for VP estimation. However, when evaluated on the more challenging and non-Manhattan scenes of NYU-VP, the handcrafted

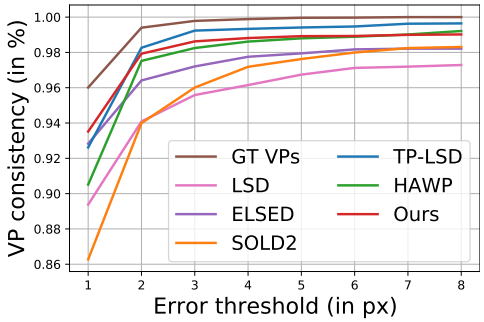
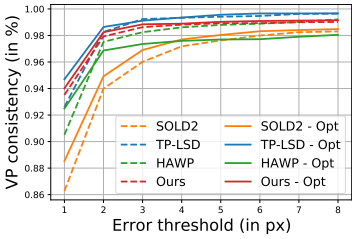


Figure 10. **VP consistency on the York Urban dataset [11].** DeepLSD ranks first on the 1 pixel threshold of VP consistency, meaning that it leads to the largest number of highly accurate VPs.

	YUD+ [11]		NYU-VP [25, 37]	
	VP error ↓	AUC ↑	VP error ↓	AUC ↑
LSD [54]	2.05	82.9 (5.3)	3.29	68.6 (6.3)
ELSED [51]	1.88	81.9 (6.0)	**3.24**	68.3 (6.6)
HAWP [57]	1.76	84.2 (4.2)	3.35	68.0 (5.7)
TP-LSD [21]	1.73	85.1 (5.0)	3.35	68.0 (4.5)
SOLD2 [39]	2.59	75.4 (6.4)	4.46	56.9 (7.6)
DeepLSD (Ours)	**1.63**	**85.6** (3.6)	**3.24**	**69.1** (6.2)

Table 8. **VP estimation on York Urban [11] and NYU-VP [25, 37].** We compare DeepLSD with other baselines in terms of median VP error and average recall AUC (and standard deviation). DeepLSD obtains the best performance overall.



	VP error ↓
HAWP [57]	**1.76**
HAWP [57] - Opt	1.78
TP-LSD [21]	1.73
TP-LSD [21] - Opt	**1.59**
SOLD2 [39]	2.59
SOLD2 [39] - Opt	**2.28**
DeepLSD (Ours)	1.63
DeepLSD (Ours) - Opt	**1.59**

Figure 11. **Effect of the line refinement on VP estimation on YorkUrbanDB [11, 25].** The line optimization improves the VP consistency and error of most deep methods.

line detectors provide the best accuracy since they can detect all types of lines. Our proposed DeepLSD outperforms all baselines in terms of VP error and AUC, and obtains the most consistent lines with the GT VPs at small thresholds in Figure 10.

We additionally study the effect of refinement on the VP estimation task in Figure 11. We show again the difference in VP consistency on the YorkUrbanDB dataset [11] and VP error on YUD+ [25], with the optimization objective including VPs. Except for HAWP, all methods benefit from the refinement, showing that our refinement can improve the lines as much as their associated VPs.

E. Line 3D Reconstruction

We show here in Figure 12 a qualitative comparison of the 3D line reconstructions of our lines and some baselines for the first 4 scenes of the Hypersim dataset [42]. TP-LSD [21] can reconstruct fewer lines as it is trained on wireframe lines only and cannot recover subtle details of the scene. While LSD [54] is usually the traditional detector being used for 3D reconstruction [18], the reconstructions produced by DeepLSD are overall more complete and the lines are cleaner compared to the LSD reconstruction. In addition, LSD has a tendency to break segments on higher resolution images, while DeepLSD will detect longer and cleaner lines. Thus, it is easy to merge all lines of a track into a nice long 3D line for DeepLSD, while LSD will generate a collection of dissociated small segments along the 3D line.

F. Limitations

Even though DeepLSD can produce repeatable and accurate lines by taking advantage of the benefits of both traditional and learned methods, it still suffers from a few limitations:

- The current approach of running a deep network, followed by handcrafted heuristics and line optimization is not fully differentiable. Making the full pipeline differentiable would mean making LSD differentiable, which is unclear how to do it. We plan to investigate this further in the future, as an end-to-end pipeline would certainly provide better training signals to the deep network processing the image.

- The generation of the pseudo ground truth lines is still limited by the performance of LSD [54]. If a line is almost never detected by LSD during homography adaptation, it will most likely not be detected in the ground truth attraction field. Similarly, a noisy but repeatable line will be kept in the pseudo ground truth. One way to overcome this issue could be to leverage the trained DeepLSD to re-generate a new pseudo ground truth with less noise, as was done in SuperPoint [12].

- In spite of our efforts to make the pseudo ground truth as clean as possible, there is always a trade-off between detecting all low-contrast lines and avoiding to detect noisy lines in the background. For example, DeepLSD misses some good lines at the bottom right of the image in the 5th row of Figure 13 and is also detecting some noisy lines in the sky of the image in the 7th row. We can influence this trade-off in two ways. First, by tuning the aggregation of the attraction field when generating the ground truth. We currently take the median value of the distance and angle fields, but one could also take a given percentile, to allow more or less outlier values.

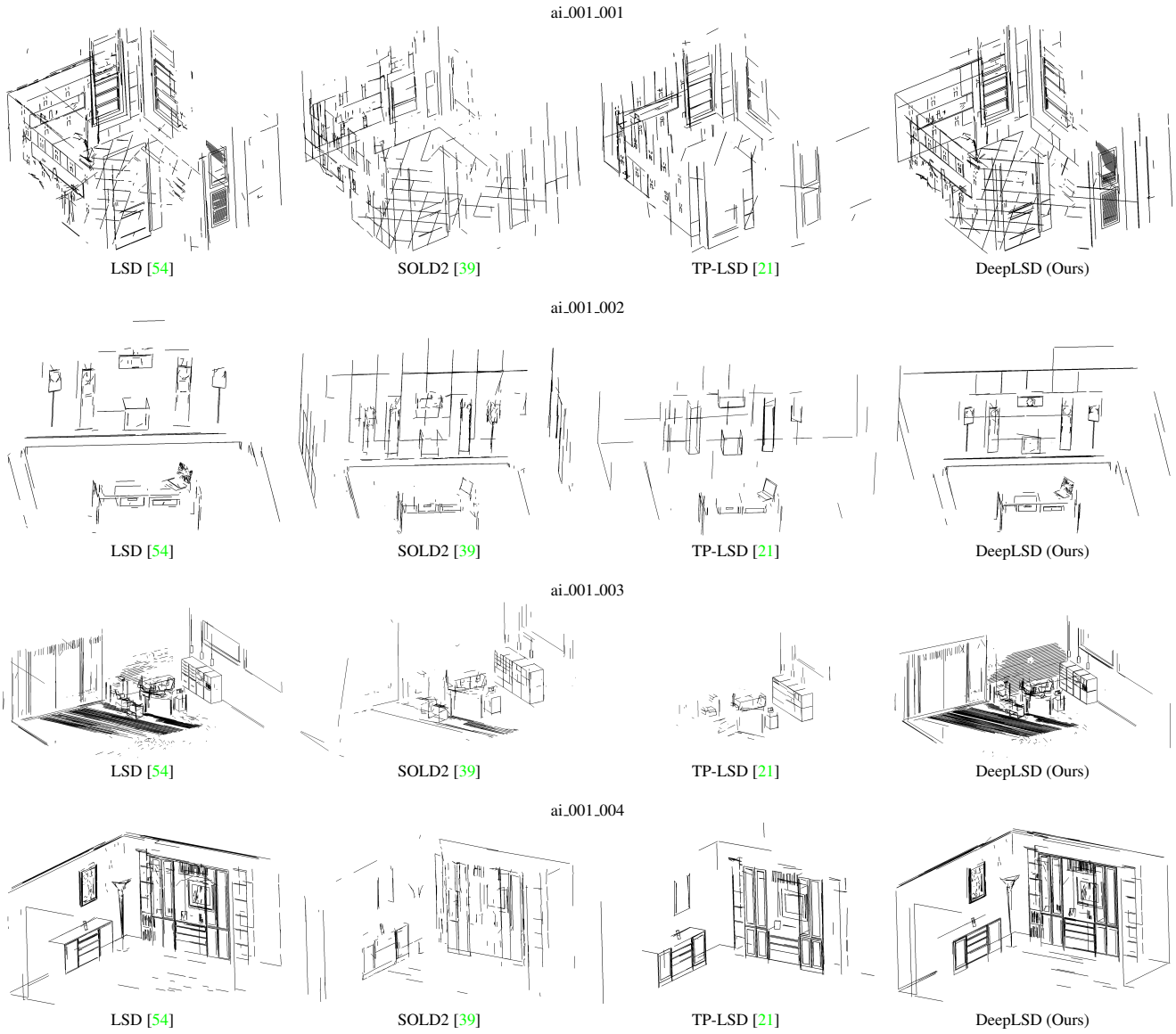


Figure 12. **Line 3D reconstruction on Hypersim [42].** We leverage the line 3D mapping software Line3D++ [18] on the first 4 scenes of Hypersim [42]. DeepLSD produces more complete and accurate reconstructions than all baselines.

Second, one can enforce more or less constraints to the distance field for background areas. Enforcing a high distance field for pixels far away from the ground truth lines will reduce the number of noisy lines in the background, but will also ignore the lines with low contrast. The parameters proposed in this paper are the ones visually yielding the best trade-off between the two.

- Though the input image is processed through a deep network, there is still no proper semantic understanding of the detected lines, so that DeepLSD will detect any kind of lines. Depending on the application, one could imagine adding some semantic filtering in the ground truth generation to keep only a specific kind of lines (e.g. avoiding lines in the sky or on dynamic objects such as humans).

- The proposed line refinement is for now rather slow, especially when it is applied to other deep line detectors, as it requires running two networks. However, we believe that it is still valuable for applications that can run offline and that require high precision, such as for 3D reconstruction. Our current implementation can also certainly be optimized, and our network compressed to run on embedded devices, without sacrificing too much performance.

G. Additional Visualizations

We provide a visual comparison of our method and the other baselines for line detection in Figure 13. We first show line detection examples from the YorkUrbanDB dataset [11], picturing indoor and outdoor urban scenes. DeepLSD offers more complete and accurate lines than its competitors. We also compare our method to the other line detectors on some images of the Day-Night Image Matching dataset [63], where DeepLSD provides more lines than the other baselines in challenging scenarios such as night time, over-exposition and low image quality.

References

[1] Hichem Abdellali, Robert Frohlich, Viktor Vilagos, and Zoltan Kato. L2D2: Learnable line detector and descriptor. In *International Conference on 3D Vision (3DV)*, 2021. 1, 2

[2] Abien Fred Agarap. Deep learning using rectified linear units (ReLU). In *arXiv*, 2018. 4, 9

[3] Sameer Agarwal and Keir Mierle. Ceres solver. http://ceres-solver.org. 6

[4] C. Akinlar and C. Topal. EDLines: Real-time line segment detection by edge drawing. In *International Conference on Image Processing (ICIP)*, 2011. 2

[5] Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. In *International Conference on Computer Vision (ICCV)*, 2007. 4

[6] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 6, 7, 8, 9

[7] Daniel Barath and Jiri Matas. Progressive-X: Efficient, anytime, multi-model fitting algorithm. In *International Conference on Computer Vision (ICCV)*, 2019. 5, 10

[8] Eric Brachmann and Carsten Rother. Visual camera relocalization from rgb and rgb-d images using dsac. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 44, 2022. 7

[9] Federico Camposeco, Andrea Cohen, Marc Pollefeys, and Torsten Sattler. Hybrid Camera Pose Estimation. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 7

[10] Xili Dai, Xiaojun Yuan, Haigang Gong, and Yi Ma. Fully convolutional line parsing. In *arXiv*, 2021. 1, 2

[11] Patrick Denis, James H Elder, and Francisco J Estrada. Efficient edge-based methods for estimating manhattan frames in urban imagery. In *European Conference on Computer Vision (ECCV)*, 2008. 10, 12, 14, 17

[12] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperPoint: Self-supervised interest point detection and description. In *Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018. 2, 4, 6, 7, 10, 11, 12

[13] James H. Elder, Emilio J. Almazán, Yiming Qian, and Ron Tal. MCMLSD: A probabilistic algorithm and evaluation framework for line segment detection. In *arXiv*, 2020. 2

[14] Qiang Fu, Jialong Wang, Hongshan Yu, Islam Ali, Feng Guo, Yijia He, and Hong Zhang. PL-VINS: Real-time monocular visual-inertial SLAM with point and line features. In *arXiv*, 2020. 1

[15] Shuang Gao, Jixiang Wan, Yishan Ping, Xudong Zhang, Shuzhou Dong, Jijunnan Li, and Yandong Guo. Pose refinement with joint optimization of visual points and lines. In *arXiv*, 2021. 1, 2

[16] Ruben Gomez-Ojeda, Francisco-Angel Moreno, David Zuñiga-Noël, Davide Scaramuzza, and Javier Gonzalez-Jimenez. PL-SLAM: A stereo SLAM system through the combination of points and line segments. *IEEE Transactions on Robotics*, 35, 2019. 1

[17] Geonmo Gu, Byungsoo Ko, SeoungHyun Go, Sung-Hyun Lee, Jingeun Lee, and Minchul Shin. Towards real-time and light-weight line segment detection. In *Conference on Artificial Intelligence (AAAI)*, 2022. 1, 2

[18] Manuel Hofer, Michael Maurer, and Horst Bischof. Efficient 3d scene abstraction using line segments. *Computer Vision and Image Understanding (CVIU)*, 157, 2017. 1, 7, 12, 13

[19] Paul VC Hough. Method and means for recognizing complex patterns, 1962. US Patent 3,069,654. 2

[20] Kun Huang, Yifan Wang, Zihan Zhou, Tianjiao Ding, Shenghua Gao, and Yi Ma. Learning to parse wireframes in images of man-made environments. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 4, 6, 8, 9

[21] Siyu Huang, Fangbo Qin, Pengfei Xiong, Ning Ding, Yijia He, and Xiao Liu. TP-LSD: Tri-points based line segment detector. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 2, 6, 7, 8, 11, 12, 13, 17

[22] Zhaoyang Huang, Han Zhou, Yijin Li, Bangbang Yang, Yan Xu, Xiaowei Zhou, Hujun Bao, Guofeng Zhang, and Hongsheng Li. VS-Net: Voting with segmentation for visual localization. In *Computer Vision and Pattern Recognition (CVPR)*, 2021. 3

[23] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015. 4, 9

[24] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2014. 6

[25] Florian Kluger, Eric Brachmann, Hanno Ackermann, Carsten Rother, Michael Ying Yang, and Bodo Rosenhahn. CONSAC: Robust multi-model fitting by conditional sample consensus. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. 10, 11, 12

[26] Zuzana Kukelova, Jan Heller, and Andrew Fitzgibbon. Efficient intersection of three quadrics and applications in computer vision. In *Computer Vision and Pattern Recognition (CVPR)*, 2016. 7

[27] Manuel Lange, Claudio Raisch, and Andreas Schilling. LVO: Line only stereo visual odometry. In *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2019. 1

[28] Viktor Larsson. PoseLib - Minimal Solvers for Camera Pose Estimation, 2020. 7

[29] Karel Lebeda, Jiri Matas, and Ondrej Chum. Fixing the Locally Optimized RANSAC. In *British Machine Vision Conference (BMVC)*, 2012. 6

[30] Hao Li, Huai Yu, Jinwang Wang, Wen Yang, Lei Yu, and Sebastian Scherer. ULSD: Unified line segment detection across pinhole, fisheye, and spherical cameras. *Journal of Photogrammetry and Remote Sensing (ISPRS)*, 178, 2021. 1

[31] Zhengqi Li and Noah Snavely. MegaDepth: Learning single-view depth prediction from internet photos. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 6, 8

[32] Yancong Lin, Silvia L Pintea, and Jan C van Gemert. Deep hough-transform line priors. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 2

[33] Philipp Lindenberger, Paul-Edouard Sarlin, Viktor Larsson, and Marc Pollefeys. Pixel-Perfect Structure-from-Motion with Featuremetric Refinement. In *International Conference on Computer Vision (ICCV)*, 2021. 2, 3

[34] André Mateus, Omar Tahri, A. Pedro Aguiar, Pedro U. Lima, and Pedro Miraldo. On incremental structure from motion using lines. *IEEE Transactions on Robotics*, 38, 2022. 1

[35] Quan Meng, Jiakai Zhang, Qiang Hu, Xuming He, and Jingyi Yu. LGNN: A context-aware line segment detector. In *ACM International Conference on Multimedia*, 2020. 2

[36] Branislav Micusik and Horst Wildenauer. Structure from motion with line segments under relaxed endpoint constraints. *International Journal of Computer Vision (IJCV)*, 124, 2017. 1, 2

[37] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision (ECCV)*, 2012. 10, 12

[38] Rémi Pautrat, Viktor Larsson, Martin R. Oswald, and Marc Pollefeys. Online invariance selection for local feature descriptors. In *European Conference on Computer Vision (ECCV)*, 2020. 6, 7, 9

[39] Rémi Pautrat, Juan-Ting Lin, Viktor Larsson, Martin R. Oswald, and Marc Pollefeys. SOLD2: Self-supervised occlusion-aware line description and detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2, 4, 6, 7, 8, 10, 11, 12, 13, 17

[40] Albert Pumarola, Alexander Vakhitov, Antonio Agudo, Alberto Sanfeliu, and Francese Moreno-Noguer. PL-SLAM: Real-time monocular visual SLAM with points and lines. In *International Conference on Robotics and Automation (ICRA)*, 2017. 1

[41] Meixiang Quan, Zheng Chai, and Xiao Liu. LOF: Structure-aware line tracking based on optical flow. In *arXiv*, 2021. 1

[42] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *International Conference on Computer Vision (ICCV)*, 2021. 7, 12, 13

[43] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015. 4, 9, 10

[44] Yohann Salaün, Renaud Marlet, and Pascal Monasse. Multiscale line segment detector for robust and accurate SfM. In *International Conference on Pattern Recognition (ICPR)*, 2016. 2

[45] Paul-Edouard Sarlin. Visual localization made easy with hloc. https://github.com/cvg/Hierarchical-Localization/. 7, 10

[46] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 7, 10

[47] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, June 2020. 7, 10, 11

[48] Paul-Edouard Sarlin, Ajaykumar Unagar, Måns Larsson, Hugo Germain, Carl Toft, Victor Larsson, Marc Pollefeys, Vincent Lepetit, Lars Hammarstrand, Fredrik Kahl, and Torsten Sattler. Back to the Feature: Learning Robust Camera Localization from Pixels to Pose. In *Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 3

[49] Torsten Sattler et al. RansacLib - A Template-based *SAC Implementation, 2019. 7

[50] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Computer Vision and Pattern Recognition (CVPR)*, 2013. 7, 8, 10

[51] Iago Suárez, José M. Buenaposada, and Luis Baumela. ELSED: Enhanced line segment drawing. *Pattern Recognition*, 2022. 2, 6, 7, 9, 12

[52] Jean-Philippe Tardif. Non-iterative approach for fast and accurate vanishing point detection. In *International Conference on Computer Vision (ICCV)*, 2009. 1, 5, 10, 11

[53] Lev Teplyakov, Leonid Erlygin, and Evgeny Shvets. Lsdnet: Trainable modification of lsd algorithm for real-time line segment detection. *IEEE Access*, 10, 2022. 1, 3, 6, 7

[54] Rafael Grompone Von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. LSD: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(4):722–732, 2008. 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 17

[55] Yifan Xu, Weijian Xu, David Cheung, and Zhuowen Tu. Line segment detection using transformers without edges. In *Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[56] Nan Xue, Song Bai, Fudong Wang, Gui-Song Xia, Tianfu Wu, and Liangpei Zhang. Learning attraction field representation for robust line segment detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 3, 4

[57] Nan Xue, Tianfu Wu, Song Bai, Fudong Wang, Gui-Song Xia, Liangpei Zhang, and Philip HS Torr. Holistically-attracted wireframe parsing. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 17

[58] Nan Xue, Tianfu Wu, Song Bai, Fu-Dong Wang, Gui-Song Xia, Liangpei Zhang, and Philip H.S. Torr. Holistically-attracted wireframe parsing: From supervised to self-supervised learning. *arXiv*, 2022. 2, 6, 7, 8, 11

[59] Haotian Zhang, Yicheng Luo, Fangbo Qin, Yijia He, and Xiao Liu. Elsd: Efficient line segment detector and descriptor. In *International Conference on Computer Vision (ICCV)*, 2021. 1, 2

[60] Lilian Zhang and Reinhard Koch. An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency. *Journal of Visual Communication and Image Representation*, 24, 2013. 6

[61] Yongjun Zhang, Dong Wei, and Yansheng Li. AG3line: Active grouping and geometry-gradient combined validation for fast line segment extraction. *Pattern Recognition*, 113, 2021. 2

[62] Ziheng Zhang, Zhengxin Li, Ning Bi, Jia Zheng, Jinlei Wang, Kun Huang, Weixin Luo, Yanyu Xu, and Shenghua Gao. Ppgnet: Learning point-pair graph for line segment detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[63] Hao Zhou, Torsten Sattler, and David W. Jacobs. Evaluating local features for day-night matching. In *European Conference on Computer Vision Workshops (ECCVW)*, 2016. 14, 17

[64] Lipu Zhou, Jiamin Ye, and Michael Kaess. A stable algebraic camera pose estimation for minimal configurations of 2d/3d point and line correspondences. In *Asian Conference on Computer Vision (ACCV)*, 2018. 7

[65] Yichao Zhou, Haozhi Qi, and Yi Ma. End-to-end wireframe parsing. In *International Conference on Computer Vision (ICCV)*, 2019. 1, 2

[66] Xingxing Zuo, Xiaojia Xie, Yong Liu, and Guoquan Huang. Robust visual SLAM with point and line features. In *International Conference on Intelligent Robots and Systems (IROS)*, 2017. 1

Figure 13. **Visual comparison of line detectors. First five rows:** the lines of DeepLSD (here, without line refinement) are more complete and accurate in urban scenarios (images from the YorkUrbanDB dataset [11]). **Last three rows:** when employed in challenging scenarios such as by night, over-exposition and low image quality, DeepLSD can detect more relevant lines than the other baselines (images from the Day-Night Image Matching (DNIM) dataset [63]).