DeepLSD: Line Segment Detection and Refinement with Deep Image Gradients 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 [15] 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 1 for the detailed architecture. Each convolution layer is followed by ReLU activation [1] and Batch Normalization [9], 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 [22] 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 [20] as base detector (coined DeepELSED) and compare it to the original ELSED in Table 1. We give the results for the raw lines without any refinement on low-level line detection metrics on the HPatches [2] and RDNIM [12] 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.

			LSD [22]		ELSED [20]	
			Traditional	DeepLSD	Traditional	DeepELSED
HPatches [2]	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 [12]	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 1. **Generalization to other traditional detectors.** Our method is not limited to LSD [22], but can also be applied to the ELSED [20] line detector for example. We show the comparison between our approach and the original detectors on the HPatches [2] and RDNIM [12] datasets. The first three columns are identical to Table 1 in the main paper and our results are given without the final line refinement.

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 [22] and ELSED [20] 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 2). 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

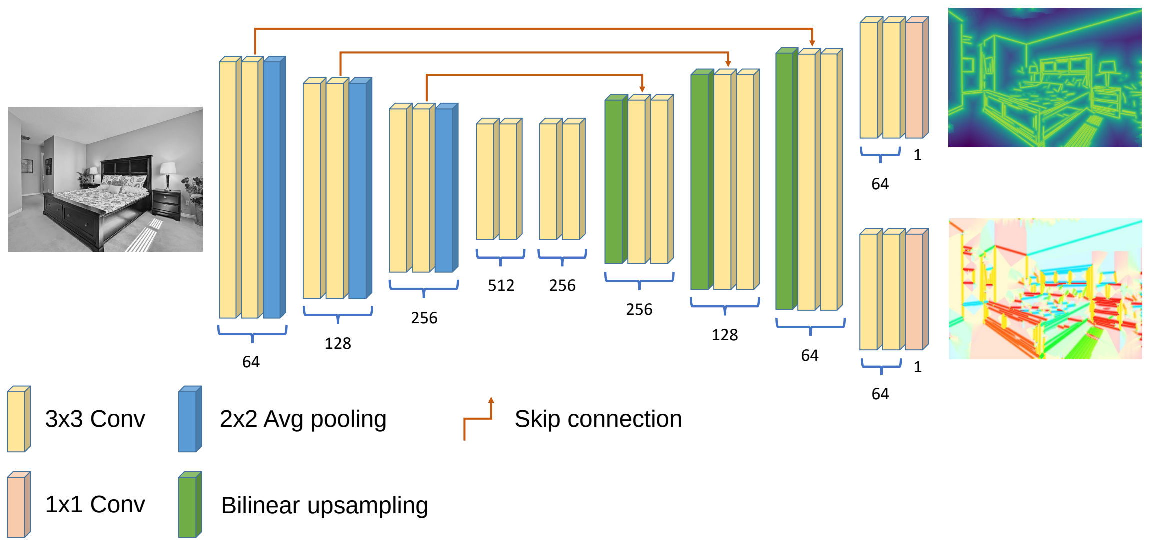


Figure 1. **Network architecture.** We use a standard UNet [15] architecture to predict the distance and angle fields.

		Struct		Orth		H estimation	# lines / img	Time [ms] ↓
		Rep ↑	LE ↓	Rep ↑	LE ↓			
LSD [22]	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 [20]	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 2. **Line refinement of traditional methods on the Wireframe dataset [7].** 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.

as shown by the large boost in homography estimation for ELSED.

B.3. Training Learned Baselines with our Supervision Strategy

In the main paper, we proposed an ablation study by re-training the HAWP [23] 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 2 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 [19], Stairs, we provide here the results of visual localization on the full dataset. As described in the main paper, we detect keypoints with SuperPoint [5], match them with SuperGlue [18], and build on top of hloc [16, 17] by adding line features and using them in the pose estimation. The lines are again matched with the SOLD2 [13] line detector. Table 3 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 [3] 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}

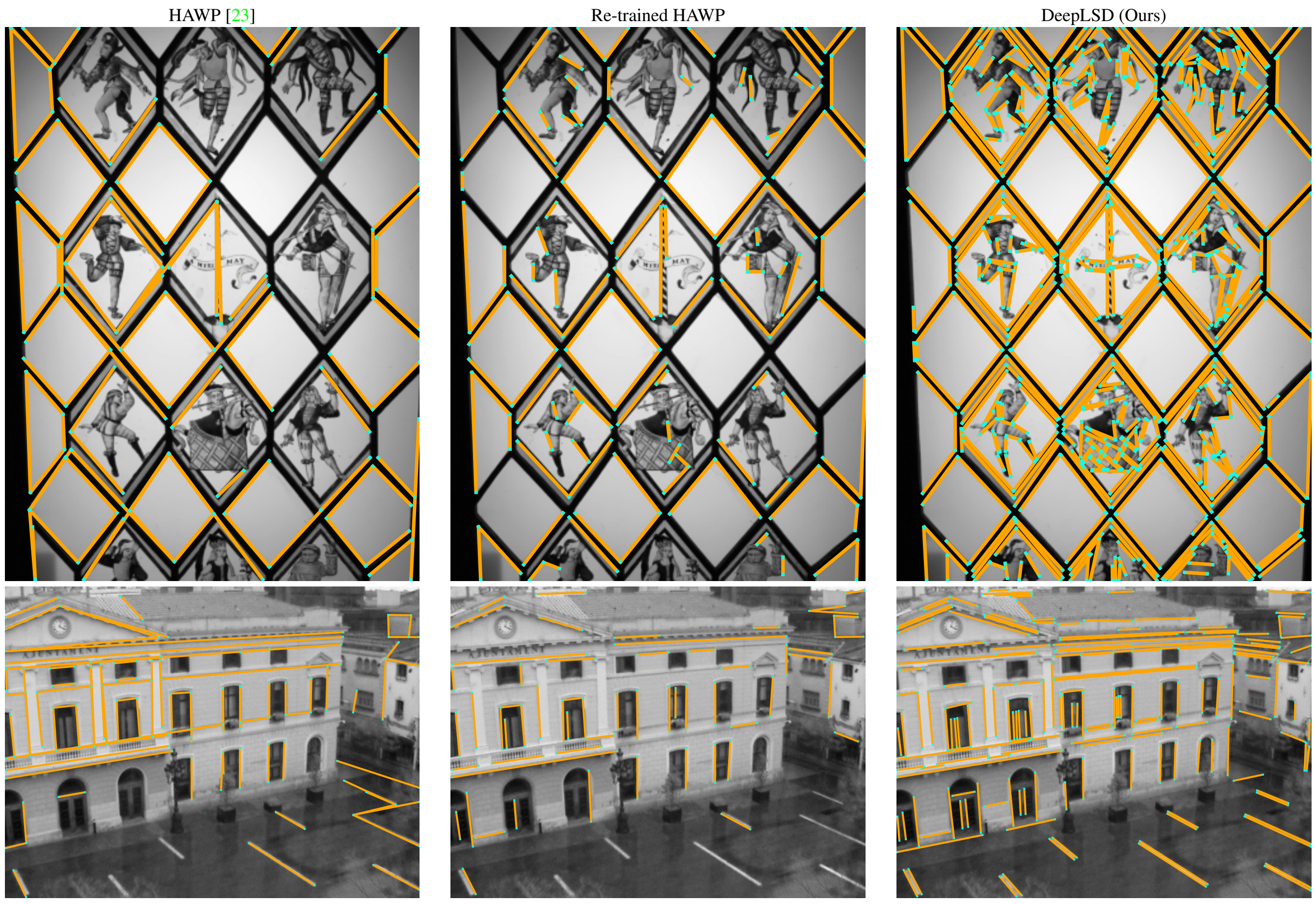


Figure 2. **Re-training the HAWP detector [23] with the proposed pseudo ground truth lines.** It yields unsatisfactory lines compared to the DeepLSD approach, mainly because detecting line endpoints with a network prediction is challenging for high densities of line segments.

	Point-only SP [5] + SG [18]	LSD [22]	SOLD2 [13]	TP-LSD [8]	HAWPv3 [24]	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 3. **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.

metric [21]. 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 [4] 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+ [10], which labels up to 8 VPs per image. The second dataset is adapted from the NYU Depth dataset V2 [11] by [10], consisting of 1449 images (we keep the last 49 for parameter tuning), each

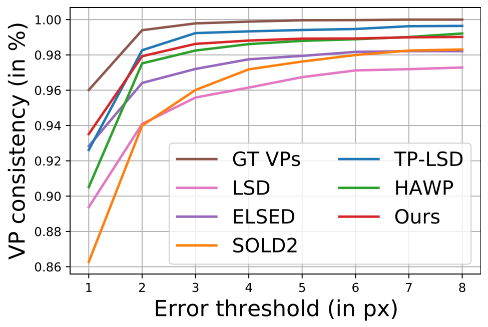


Figure 3. **VP consistency on the York Urban dataset [4].** DeepLSD ranks first on the 1 pixel threshold of VP consistency, meaning that it leads to the largest number of highly accurate VPs.

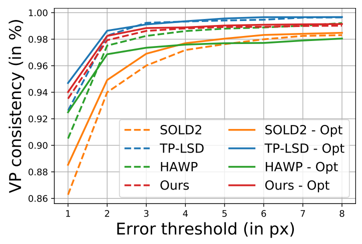
	YUD+ [4]		NYU-VP [10, 11]	
	VP error ↓	AUC ↑	VP error ↓	AUC ↑
LSD [22]	2.05	82.9 (5.3)	3.29	68.6 (6.3)
ELSED [20]	1.88	81.9 (6.0)	**3.24**	68.3 (6.6)
HAWP [23]	1.76	84.2 (4.2)	3.35	68.0 (5.7)
TP-LSD [8]	1.73	85.1 (5.0)	3.35	68.0 (4.5)
SOLD2 [13]	2.59	75.4 (6.4)	4.46	56.9 (7.6)
DeepLSD (Ours)	**1.63**	**85.6** (3.6)	**3.24**	**69.1** (6.2)

Table 4. **VP estimation on York Urban [4] and NYU-VP [10, 11].** 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.

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 of the predicted VPs [21]. 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 [10]. We show the average AUC and its standard deviation over 5 runs.

Results are shown in Figure 3 and Table 4. The wireframe methods TP-LSD [8] and HAWP [23] 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 line detectors provide the best accuracy since they can detect all types of lines. Our proposed DeepLSD outperforms all



	VP error ↓
HAWP [23]	**1.76**
HAWP [23] - Opt	1.78
TP-LSD [8]	1.73
TP-LSD [8] - Opt	**1.59**
SOLD2 [13]	2.59
SOLD2 [13] - Opt	**2.28**
DeepLSD (Ours)	1.63
DeepLSD (Ours) - Opt	**1.59**

Figure 4. **Effect of the line refinement on VP estimation on YorkUrbanDB [4, 10].** The line optimization improves the VP consistency and error of most deep methods.

baselines in terms of VP error and AUC, and obtains the most consistent lines with the GT VPs at small thresholds in Figure 3.

We additionally study the effect of refinement on the VP estimation task in Figure 4. We show again the difference in VP consistency on the YorkUrbanDB dataset [4] and VP error on YUD+ [10], 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 5 a qualitative comparison of the 3D line reconstructions of our lines and some baselines for the first 4 scenes of the Hypersim dataset [14]. TP-LSD [8] can reconstruct fewer lines as it is trained on wireframe lines only and cannot recover subtle details of the scene. While LSD [22] is usually the traditional detector being used for 3D reconstruction [6], 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

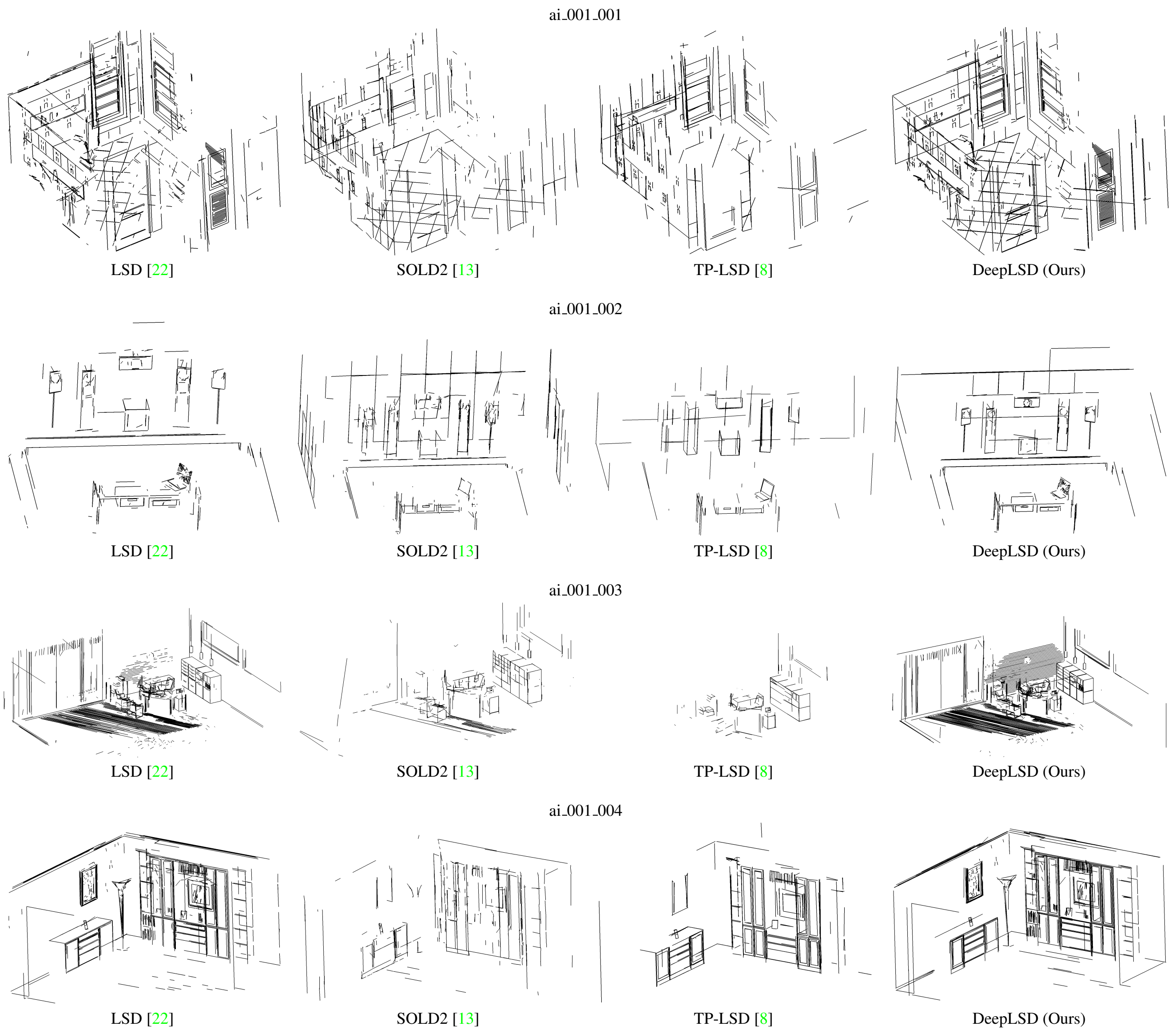


Figure 5. **Line 3D reconstruction on Hypersim [14].** We leverage the line 3D mapping software Line3D++ [6] on the first 4 scenes of Hypersim [14]. DeepLSD produces more complete and accurate reconstructions than all baselines.

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 [22]. 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 [5].

- 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 6 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. 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 6. We first show line detection examples from the YorkUrbanDB dataset [4], 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 [25], where DeepLSD provides more lines than the other baselines in challenging scenarios such as night time, over-exposition and low image quality.

References

[1] Abien Fred Agarap. Deep learning using rectified linear units (ReLU). In *arXiv*, 2018. 1

[2] 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. 1

[3] Daniel Barath and Jiri Matas. Progressive-X: Efficient, anytime, multi-model fitting algorithm. In *International Conference on Computer Vision (ICCV)*, 2019. 2

[4] 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. 3, 4, 6, 8

[5] 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, 3, 5

[6] Manuel Hofer, Michael Maurer, and Horst Bischof. Efficient 3d scene abstraction using line segments. *Computer Vision and Image Understanding (CVIU)*, 157, 2017. 4, 5

[7] 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. 2

[8] 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. 3, 4, 5, 8

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015. 1

[10] 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. 3, 4

[11] 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. 3, 4

[12] 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. 1

[13] 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. 2, 3, 4, 5, 8

[14] 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. 4, 5

[15] 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. 1, 2

[16] Paul-Edouard Sarlin. Visual localization made easy with hloc. `https://github.com/cvg/Hierarchical-Localization/`. 2

[17] 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. 2

[18] 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. 2, 3

[19] 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. 2

[20] Iago Suárez, José M. Buenaposada, and Luis Baumela. ELSED: Enhanced line segment drawing. *Pattern Recognition*, 2022. 1, 2, 4

[21] Jean-Philippe Tardif. Non-iterative approach for fast and accurate vanishing point detection. In *International Conference on Computer Vision (ICCV)*, 2009. 3, 4

[22] 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, 8

[23] 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. 2, 3, 4, 8

[24] 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. 3

[25] Hao Zhou, Torsten Sattler, and David W. Jacobs. Evaluating local features for day-night matching. In *European Conference on Computer Vision Workshops (ECCVW)*, 2016. 6, 8

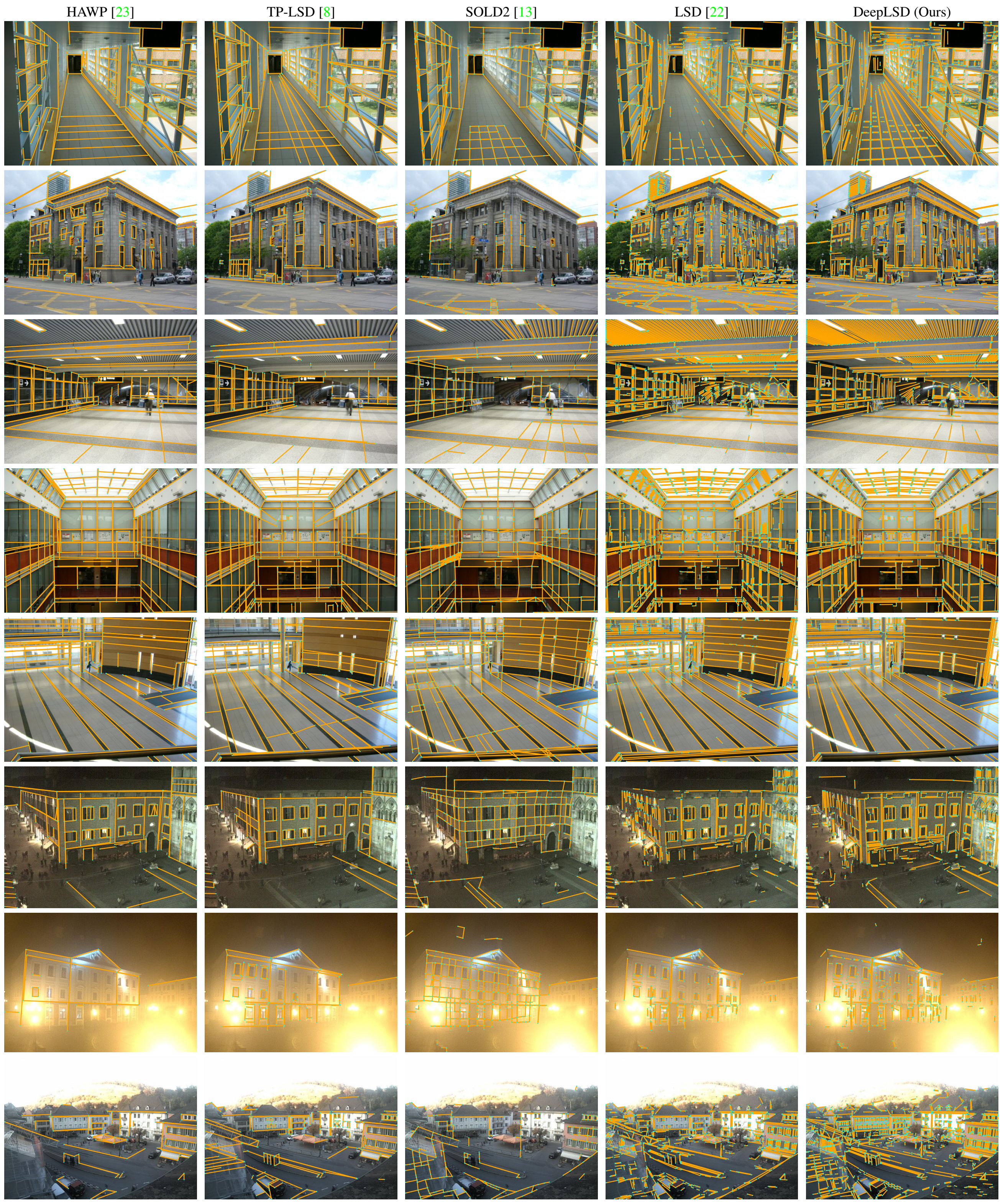


Figure 6. **Visual comparison of line detectors. First five rows:** the lines of DeepLSD (here, without line refinement) are more complete and accurate in urban scenarios (images from the YorkUrbanDB dataset [4]). **Last three rows:** when employed in challenging scenarios such as by night, over-exposition and low image quality, DeepLSD can detect more relevant lines than the other baselines (images from the Day-Night Image Matching (DNIM) dataset [25]).