Appendix

In the following, we provide additional details about our Self-supervised Occlusion-aware Line Descriptor and Detector (SOLD^2). Section A describes the generation of the synthetic dataset used to pretrain the network. Section B details our network architecture. Section C refers to the multi-task approach used to balance our different losses. Section D explains into details some parts of the line segment detection module. Section E gives the exact equations used to compute the evaluation metrics considered in this work. Section F provides proof that our results are statistically meaningful. Section G describes how we preprocessed the ETH3D dataset. Section H discusses the feasibility of applying our method to the homography estimation task. Finally section I displays qualitative examples of our line detections and matches compared to other baselines.

A. Synthetic dataset examples and homography generation

We provide here some examples of the images in our synthetic dataset and the homographies we used in both data augmentation and homography adaptation. These shapes include polygons, cubes, stars, lines, checkerboards, and stripes. Figure 1 shows some examples of these rendered shapes.

We follow the same process as in SuperPoint [1] to generate the random homographies. They are generated as a composition of simple transformations with pre-defined ranges: scaling (normal distribution $\mathcal{N}(1., 0.1)$), translation (uniform distribution within the image boundaries), rotation (uniformly in $[-90°, +90°]$), and perspective transform. Examples of the difficulty of the test set can be observed in Figures 4 and 5 of the supplementary material.

B. Network architecture

We provide here more details about our architecture and parameter choices. To have a fair comparison with most wireframe parsing methods [22, 20, 11], we use the same stacked hourglass network as in [12]. Given an image with resolution $h \times w$, the output of the backbone encoder is a $\frac{h}{4} \times \frac{w}{4} \times 256$ feature map. The three heads of the network are implemented as follows:

Junction branch: It is composed of a 3×3 convolution with stride 2 and 256 channels, followed by a 1×1 convolution with stride 1 and 65 channels, to finally get the $\frac{h}{8} \times \frac{w}{8} \times 65$ junction map.

Heatmap branch: To keep a light network and avoid artifacts from transposed convolutions, we perform two consecutive subpixel shuffles [17] blocks to perform a $\times 4$ upsampling. More precisely, we use two 3×3 conv layers of output channel sizes 256 and 64, each of them followed by batch normalization, ReLU activation and a $\times 2$ subpixel shuffle for upsampling. A final 1×1 convolution with output channel 1 and sigmoid activation is then used to get the final line heatmap of resolution $h \times w$.

Descriptor branch: The backbone encoding is processed by two consecutive convolutions of kernels 3×3 and 1×1, and output channels 256 and 128, to produce a $\frac{h}{4} \times \frac{w}{4} \times 128$ feature descriptor map. This semi-dense map can be later bilinearly interpolated at any point location. The triplet loss is optimized with a margin $M = 1$ and a minimal distance to the hardest negative of $T = 8$ pixels.

We use ReLU activations after each convolution and optimize the network with the Adam solver [8]. Images are resized to a 512×512 resolution and converted to grayscale during training.

C. Multi-task learning

The tasks of detecting lines, their junctions, and describing them are diverse, and we assume them to have a different homoscedastic aleatoric uncertainty. Additionally, they can have different orders of magnitude and their relative values are changing during training, in particular when the descriptor branch is added to the pre-trained detector network. Therefore, we chose to use the multi-task loss introduced by Kendall *et al*. [7] and successfully used in other geometrical tasks [6, 14], to automatically adjust the weights of the losses during training.

The final weights of Equation (8) gracefully converged towards the inverse of each loss, such that the value of each loss multiplied by its weight is around 1. The final weight values are the following: $e^{-w_{junc}} = 7.2$, $e^{-w_{line}} = 16.3$ and $e^{-w_{desc}} = 8.2$. To show the effectiveness of the dynamic weighting, we tried two variants: (1) all loss weights are 1, and (2) we used the final values from the dynamic weighting as static loss weights. In the first case, the detection and description results are worse by at least 10% and 5.5%, respectively. In the second case, the detection and description results are worse by at least 6.7% and 76.2%, respectively.

D. Line segment detection details

To convert the raw junctions and line heatmap predicted by our network into line segments, the following steps are performed sequentially: regular sampling of points along each line, adaptive local-maximum search, and accepting the lines verifying a minimum average score and inlier ratio. Additionally, an initial step called candidate selection can be used to pre-filter some of the line candidates $\hat{L}$. We describe here two of these steps into more details: the adaptive local-maximum search, and the candidate selection.

Adaptive local-maximum search. Given a set of points sampled along a candidate line segment, one wants to extract

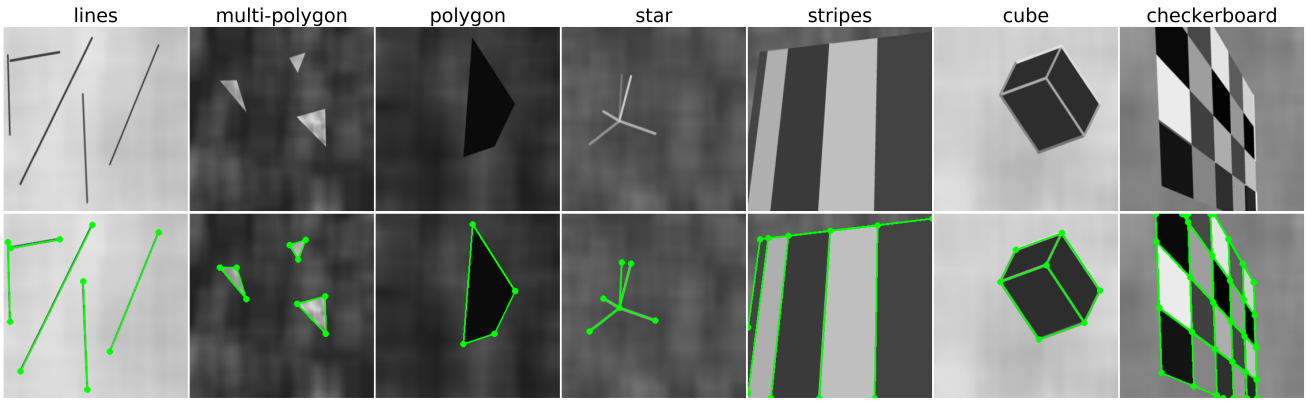

Figure 1: **Image examples from the synthetic dataset.** First row: rendered images. Second row: images with estimated junctions and line segment labels.

the line heatmap values at these sampling locations. However, since the heatmap is limited to a resolution of one pixel, some samples may get a lower heatmap value if they land next to the actual line. Thus, we instead use an adaptive local-maximum search to find the highest activation of the line heatmap around each sampling location. Given a line segment $\hat{l} = (\hat{e}^1, \hat{e}^2)$ from the candidate set $\hat{L}$ in an image of size $h \times w$, the search radius r is defined as:

$$r = r_{min} + \lambda \frac{\|\hat{e}^1 - \hat{e}^2\|}{\sqrt{h^2 + w^2}} \tag{1}$$

where $r_{min} = \frac{\sqrt{2}}{2}$ is the minimum search radius and λ is a hyper-parameter to adjust the linear dependency on the segment lengths. We used $\lambda = 3$ pixels in all experiments. The optimal line parameters were selected by a grid search on the validation set. The r_{min} parameter can in particular be kept constant across different image resolutions, without performance degradation.

Candidate selection (CS). In some application requiring line matching, having multiple overlapping segments may hinder the matching as the descriptor will have a harder job at discriminating close lines. Therefore, a non-maximum suppression (NMS) mechanism is necessary for lines. Unlike point or bounding box NMS, there is no well-established procedure for line NMS. Contrary to usual NMS methods which are used as postprocessing steps, we implement our line NMS as a preprocessing step, which actually speeds up the overall line segment detection as it removes some line candidates. Starting from the initial line candidates set $\hat{L}_{cand}$, we remove the line segments containing other junctions between their two endpoints. To identify whether a junction lies on a line segment, we first project the junction on the line and check if it falls within the segment boundaries. When it does, the junction is considered to be on the line segment if it is at a distance of less than ξ_{cs} pixels from the line. Through out our experiments, we adopted $\xi_{cs} = 3$ pixels.

E. Detector evaluation metrics

Similarly to the metrics introduced in SuperPoint [1], we propose line segments repeatability and localization error metrics. Both of these metrics are computed using pairs of images I_1 and I_2, where I_2 is a warped version of I_1 under a homography $\mathcal{H}$. Each image is associated with a set of line segments $L_1 = \{l_m^1\}_{m=1}^{M_1}$ and $L_2 = \{l_m^2\}_{m=1}^{M_2}$, and d refers to one of the two line distances defined in this work: structural distance $\mathrm{d_s}$ and orthogonal distance $\mathrm{d_{orth}}$.

Repeatability: The repeatability measures how often a line can be re-detected in different views. The repeatability with tolerance ϵ is defined as:

$$\forall l \in L_1, \mathrm{C}_{L_2}(l) = \begin{cases} 1 & \text{if}(\min_{l_j^2 \in L_2} \mathrm{d}(l, l_j^2)) \leq \epsilon, \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$\text{Rep-}\epsilon = \frac{\sum_{i=1}^{M_1} \mathrm{C}_{L_2}(l_i^1) + \sum_{j=1}^{M_2} \mathrm{C}_{L_1}(l_j^2)}{M_1 + M_2} \tag{3}$$

Localization error: The localization error with tolerance ϵ is the average line distance between a line and its re-detection in another view:

$$\text{LE-}\epsilon = \frac{\sum_{j \in Corr} \min_{l_i^1 \in L1} \mathrm{d}(l_i^1, l_j^2)}{|Corr|} \tag{4}$$

$$Corr = \{j \mid C_{L_1}(l_j^2) = 1,\ l_j^2 \in L_2\} \tag{5}$$

where $|\cdot|$ measures the cardinality of a set.

F. Statistical evaluation of our method

All the experiments displayed in the main paper are from a single training run. To justify our statistical improvement over the baselines, we re-trained the full detector and descriptor network 5 times with different random seeds

	d_s		d_{orth}	
	Rep-5 ↑	LE-5 ↓	Rep-5 ↑	LE-5 ↓
LCNN [22]	0.336	2.777	0.637	1.878
HAWP [20]	0.451	2.625	0.537	1.738
DeepHough [11]	0.370	2.676	0.652	1.777
TP-LSD [5]	0.563	2.467	0.746	1.450
LSD [19]	0.358	2.079	0.707	0.825
Ours (mean)	**0.577**	**1.955**	**0.891**	**0.804**
Std deviation	0.020	0.070	0.0055	0.023

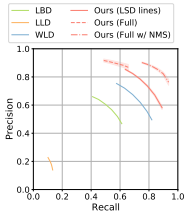

Figure 2: **Statistical significance of the evaluation.** We report the average value μ and standard deviation σ of 5 different training runs of our method on the Wireframe dataset [4]. Left: repeatability and localization error of the detector. Right: matching precision-recall curve with confidence interval $[\mu - \sigma, \mu + \sigma]$.

each time. Figure 2 displays the same evaluation on the Wireframe dataset [4] as in our paper with the mean and standard deviation of the performance of our method over these 5 runs, and thus shows statistically meaningful results.

G. ETH3D dataset preprocessing

The ETH3D dataset [16] is composed of 13 scenes taken in indoor as well as outdoor environments. Each image comes with the corresponding camera intrinsics and depth map, and a 3D model of each scene built with Colmap [15] is provided as well. We use the undistorted image downsampled by a factor of 8 to run the line detection and description. We then use the depth maps and camera intrinsics to reproject the lines in 3D and compute the descriptor metrics in 3D space. While the depth maps have been obtained from a high-precision laser scanner, they contain some holes, in particular close to depth discontinuities. Since these discontinuities are actually where lines are often located, we inpaint the depth in all of the invalid areas at up to 10 pixels from a valid depth pixel. We used NLSPN [13], the current state of the art in deep depth inpainting guided with RGB images.

H. Homography estimation experiment

To validate the real-world applications of our method, we used the line segment detections and descriptors to match segments across pairs of images of the Wireframe dataset related by a homography [4] and estimate the homography with RANSAC [3]. We sample minimal sets of 4 lines to fit a homography and run up to $1,000,000$ iterations with the LO-RANSAC [10] implementation of Sattler *et al.*[1]. The reprojection error is computed with the orthogonal line distance. We compute the accuracy of the homography estimation similarly as in SuperPoint [1] by warping the four corners of the image with the estimated homography, warping them back to the initial image with the ground truth homography and computing the reprojection error of the corners. We consider the estimated homography to be correct if the average reprojection error is less than 3 pixels. The results are listed in Table 1.

[1] https://github.com/tsattler/RansacLib

		Homography estimation		
Lines	Desc	Accuracy↑	# inliers	Reproj. error↓
LSD [19]	LBD [21]	0.781	80	0.791
	LLD [18]	0.201	21	0.927
	WLD [9]	0.920	116	0.868
	Ours	**0.948**	116	0.863
Ours	Ours	0.935	200	**0.780**
SuperPoint [1]		0.582	173	0.996

Table 1: **Evaluation results of homography estimation.** The homography between images of the Wireframe dataset [4] is estimated from line matches using RANSAC. We use a threshold of 5 pixels in orthogonal line distance to consider a match to be an inlier.

When compared on LSD line [19], our descriptors provide the highest accuracy among all baselines, and our full pipeline achieves a similar performance. When using our lines, we use a similar refinement of the junctions as in LSD [19]: we sample small perturbations of the endpoints by a quarter of a pixel and keep the perturbed endpoints maximizing the line average score. Similarly to feature point methods [1, 2], this experiment shows that learned features are still on par or slightly worse than handcrafted detections in terms of localization error.

We also add to the comparison the results of homography estimation for a learned feature point detector and descriptor, SuperPoint [1]. The point-based approach performs significantly worse than our method, due to the numerous textureless scenes and repeated structures present in the Wireframe dataset. We also found that SuperPoint is not robust to rotations above 45 degrees, while our line descriptor can leverage its ordered sequence of descriptors to achieve invariance with respect to any rotation.

I. Qualitative results of line segment detections and matches

We provide some visualizations of the line segment detection results in Figure 4 and of the line matching in Figure 3. Figure 5 also offers a comparison of line matches with point matches in challenging images with low texture, and repeated structures. Our method is able to match enough lines to obtain an accurate pose estimation, while point-based methods such as SuperPoint [1] fail in such scenarios.

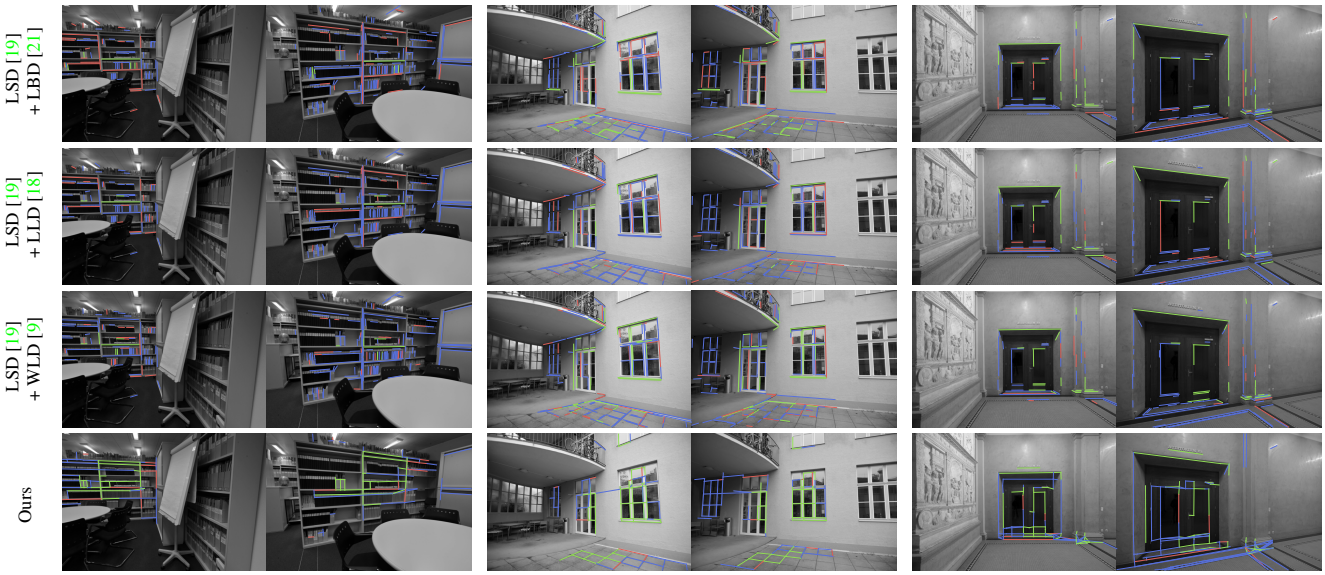

Figure 3: **Qualitative results of line segment matching.** We display line segment matches on the ETH3D dataset [16] with correct matches, wrong matches and unmatched lines. Only lines shared between the two views are shown. Our full pipeline is compared to three line descriptor baselines computed on LSD lines [19]: LBD [21], LLD [18] and WLD [9].

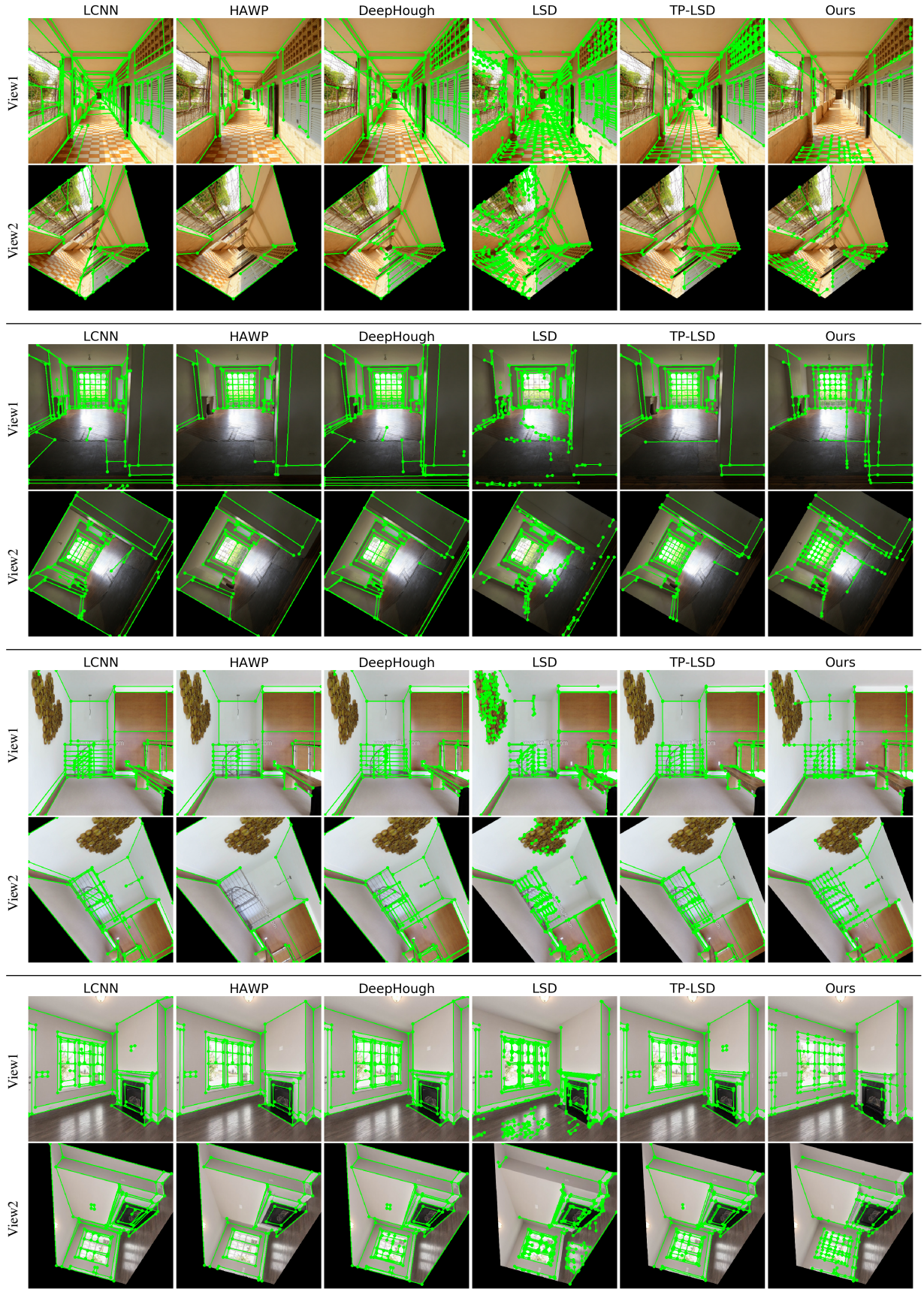

Figure 4: **Qualitative results of line segment detections.** We show examples of line detections on the Wireframe dataset [4] for the following methods: LCNN [22], HAWP [20], DeepHough [11], LSD [19], TP-LSD [5] and ours.

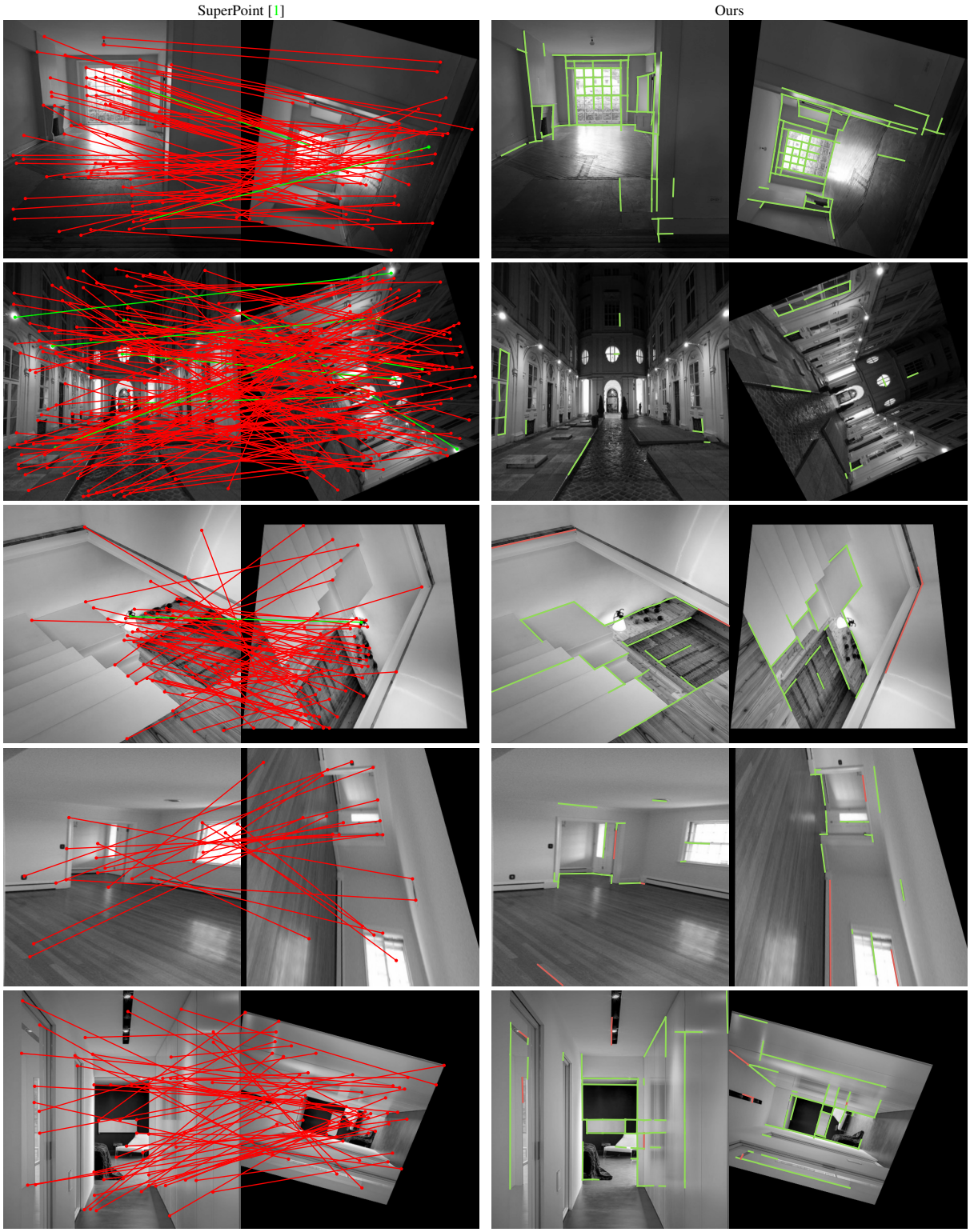

Figure 5: **Benefits of lines compared to feature points.** We compare our method with point matching from SuperPoint [1] on challenging images of the Wireframe dataset [4] with correct and wrong matches. We use a distance threshold of 5 pixels to determine if a match is correct, using the orthogonal line distance in the case of lines. Lines can be matched even in the presence of textureless areas, as well as repeated and symmetrical structures.

References

[1] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018. 1, 2, 3, 6

[2] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-Net: A Trainable CNN for Joint Detection and Description of Local Features. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[3] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communication of the ACM*, 24(6), 1981. 3

[4] 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. 3, 5, 6

[5] 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, 5

[6] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[7] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 1

[8] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2014. 1

[9] Manuel Lange, Claudio Raisch, and Andreas Schilling. Wld: A wavelet and learning based line descriptor for line feature matching. In *International Symposium on Vision, Modeling, and Visualization (VMV)*, 2020. 3, 4

[10] Karel Lebeda, Jiri Matas, and Ondrej Chum. Fixing the Locally Optimized RANSAC. In *British Machine Vision Conference (BMVC)*, 2012. 3

[11] Yancong Lin, Silvia L Pintea, and Jan C van Gemert. Deep hough-transform line priors. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 3, 5

[12] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision (ECCV)*, 2016. 1

[13] Jinsun Park, Kyungdon Joo, Zhe Hu, Chi-Kuei Liu, and In So Kweon. Non-local spatial propagation network for depth completion. In *Proc. of European Conference on Computer Vision (ECCV)*, 2020. 3

[14] 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. 1

[15] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[16] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 3, 4

[17] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[18] A. Vakhitov and V. Lempitsky. Learnable line segment descriptor for visual slam. *IEEE Access*, 7, 2019. 3, 4

[19] 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. 3, 4, 5

[20] 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, 3, 5

[21] 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(7), 2013. 3, 4

[22] Yichao Zhou, Haozhi Qi, and Yi Ma. End-to-end wireframe parsing. In *International Conference on Computer Vision (ICCV)*, 2019. 1, 3, 5