# Unified and Efficient Point-Line Local Features

François Costa[1,⋆], Raphael Kreft[1,*], Eckhard Goedeke[1], Felix Möller[1], Hardik Shah[1], Ramanathan Rajaraman[1], Shaohui Liu[1], Rémi Pautrat[2], and Marc Pollefeys[1,2]

[1] Department of Computer Science, ETH Zurich, Switzerland
[2] Microsoft Spatial AI Lab

**Abstract.** Multi-view computer vision pipelines typically rely on accurate sparse keypoints and robust descriptors. While incorporating line features has shown clear benefits for matching and pose estimation, existing point-line approaches remain inefficient: they detect points and lines separately, use increasingly heavy networks, and depend on CPU-bound heuristics that hinder real-time performance. We introduce a Unified Efficient Points and Lines (UPAL) feature extractor that jointly extracts keypoints, line segments, and feature descriptors within a single lightweight architecture. A shared backbone provides common representations that feed different branches for point and line features. Line segments are recovered through an accelerated post-processing stage, an enhanced and highly efficient variant of the LSD algorithm. UPAL matches or exceeds state-of-the-art performance in both point and line applications while significantly reducing computational cost, achieving, for instance, a 4× speedup and 10× smaller memory footprint over the ALIKED + DeepLSD pipeline. Code is publicly available at `https://github.com/francois141/upal`.

**Keywords:** Fast local features · Point and line detectors and descriptors

## 1 Introduction

Extracting salient points and their corresponding descriptors is a fundamental step in many vision tasks involving multi-view geometry. Beyond 2D points, line segments are another type of feature that can complement points due to their prevalence in human-made structures, resulting in more informative image representations for geometric tasks. Accurately localizing line segments in images remains an active field of research due to several challenges, such as their extended size across images, their susceptibility to occlusion, and their sensitivity to perspective changes. In spite of these challenges, recent methods combining points and lines have demonstrated state-of-the-art results in multiple fields, such as feature matching [24, 41, 45, 67], 3D reconstruction [38, 39], visual localization [2, 19, 39, 51, 68, 81, 82], relative pose estimation [26], and Simultaneous Localization and Mapping (SLAM) [18, 22, 34, 36, 49, 52, 69, 70, 85].

⋆ François Costa and Raphael Kreft contributed equally to this work.

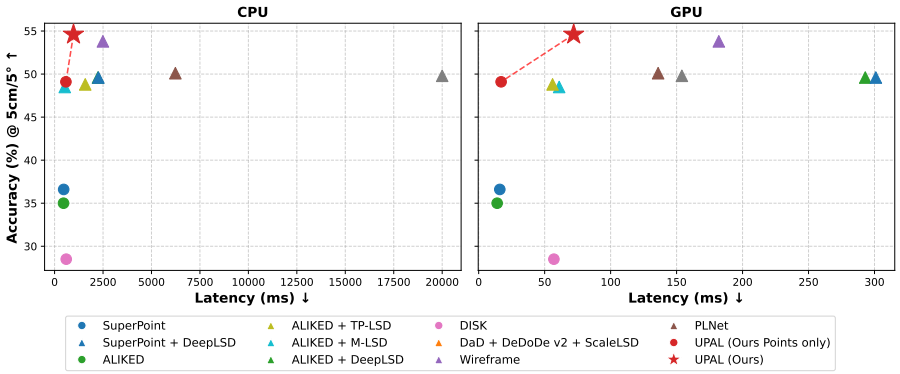



**Fig. 1: In this paper, we propose UPAL, a joint point-line feature extractor** that outperforms most existing methods (here, on hybrid point-line visual localization with pose accuracy under 5cm / 5°) **while being substantially more lightweight**. This is enabled by our compact architecture, joint point-line training with distilled supervision, and an efficient variant of the LSD algorithm. Results are shown on the *Stairs* scene of the 7Scenes dataset [62] with the setup described in Sec. 4.4.

Despite their effectiveness, such vision pipelines using both points and lines suffer from notable computational drawbacks. They often require two separate extractors: one for keypoint detection and description, and another for line localization. This approach is inefficient, as the two tasks are strongly related and could share computations, since points and lines are both localized in regions with strong image gradients. Moreover, recent network architectures tend to grow in size, making both point and line extraction computationally expensive, requiring high-end GPUs to achieve satisfactory throughput. While recent line detectors have pushed performance for feature extraction and matching, little effort has been invested in reducing the size of the networks and making them efficient. Furthermore, some state-of-the-art deep line detectors reuse classical algorithms such as LSD [20] to extract their line segments [44, 61, 65]. However, LSD is a CPU-only algorithm that contains operations that could be implemented more efficiently on GPU. In addition, it suffers from quadratic complexity with respect to the size of the image.

Motivated by these observations, we introduce *Unified Efficient Points and Lines (UPAL)* local features, a novel lightweight feature extractor that extracts keypoints, line segments, and feature descriptors jointly in a single forward pass. UPAL leverages the architecture of a point feature extractor and adds a lightweight line detection branch on top of it. Without resorting to costly learned line descriptors or matchers [45, 67, 75], we argue that lines can be efficiently matched by first matching their endpoints and then finding the optimal assignment, thus requiring no extra line descriptors. While we are not the first to propose joint point-line feature extraction [17, 69], previous attempts failed to reach state-of-the-art results for both features, despite maintaining high throughput. With UPAL, we show how to reach such goals with a streamlined design. Through distillation, UPAL combines the best of both worlds: it leverages

an efficient and lightweight neural network while being supervised by powerful state-of-the-art models. We carefully reviewed the best point and line extractors to date, and selected the best-performing combination to distill their knowledge into our architecture. By eliminating redundant computations in the extraction of the two kinds of features (points and lines), UPAL preserves state-of-the-art performance while improving efficiency and memory footprint. In particular, we demonstrate that **state-of-the-art results in line detection can be achieved by adding only three convolutional layers to an existing point extractor.**

Inspired by DeepLSD [44], we leverage a post-processing step based on the LSD [20] algorithm to extract the final line segments. We introduce three major improvements over the original LSD and DeepLSD implementations. First, we show that discarding the branch predicting an angle field can simplify the architecture and at the same time improve the performance. Second, we move most of LSD's image processing to the GPU and leave only the line-growing stage on CPU. Lastly, we propose a simple but effective heuristic to reduce the number of seed points in LSD, increasing the efficiency of the detector without any loss in performance.

We evaluate UPAL on both low-level metrics and geometric downstream tasks, demonstrating its effectiveness across multiple challenging datasets. Combined with our optimized line extraction module, UPAL achieves a $4\times$ speedup over the current state of the art, with similar or better performance. Moreover, approaches with comparable performance have a memory footprint that is at least one order of magnitude higher.

In summary, our contributions are as follows:

1. We demonstrate how to distill state-of-the-art point and line detectors and descriptors into a single lightweight network, preserving, or even improving, their original accuracy. This design substantially simplifies existing pipelines, requiring only a few additional convolutions for line extraction.
2. We introduce an enhanced variant of LSD that provides significantly higher efficiency through GPU-accelerated processing.
3. UPAL achieves state-of-the-art performance for both point and line features, across low-level metrics and downstream tasks, while running much faster than most prior approaches and being significantly lighter, making it more suitable for embedded applications.

## 2 Related work

**Point Feature Extractors.** Feature points are traditionally detected and described using patterns in the image gradient [5, 25, 40, 54]. The deep learning era has subsequently introduced novel methods to jointly detect and describe feature points within a single network. The pioneering work SuperPoint [10] introduced the process of homography adaptation to create a pseudo ground truth of corner-like keypoints. D2-Net [11] showed how to describe first, then extract keypoints from the features. R2D2 [53] introduced the concept of reliability

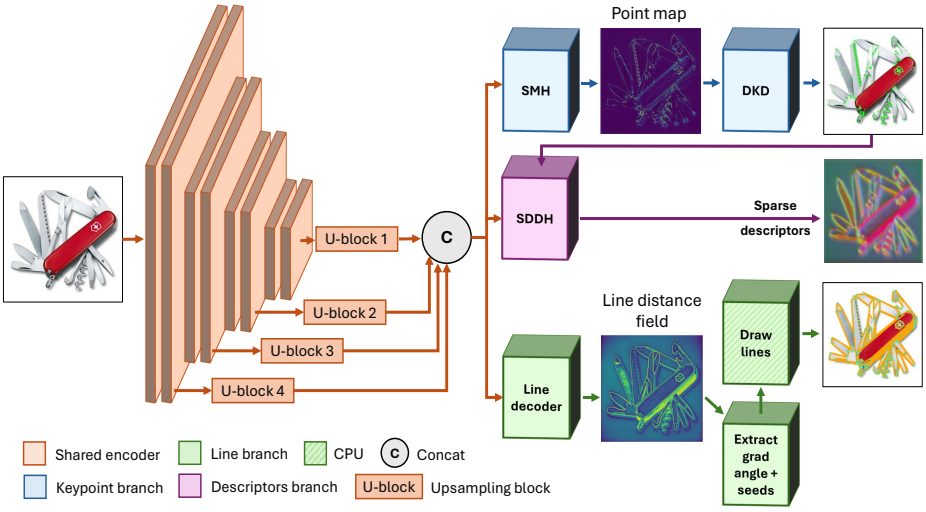



**Fig. 2: Overview of our network architecture.** We use a shared convolutional encoder to generate a multi-scale feature map (orange). We then use three decoder branches: a keypoint detection branch (blue) with a Score Map Head (SMH) followed by Differentiable Keypoint Detection (DKD), a Sparse Deformable Descriptor Head (SDDH) to obtain descriptors (purple), and a branch predicting a distance field (green). The latter is then passed to our fast LSD to extract the line segments.

to only keep the most relevant keypoints. DISK [66] revisited how to learn reliable keypoints through reinforcement learning. ALIKED [79] was proposed as a lightweight network and remains among the top performers, which motivated our decision to follow the same architecture. SiLK [21] demonstrated that a simple setup can already lead to a strong performance. Progress has also been made in terms of efficiency with XFeat [48], which made it possible to run feature extraction even on embedded devices. The DeDoDe work [14] advocates for using independent networks for detection and description to maximize performance. Finally, a recent keypoint detector, DaD [13], revisited the reinforcement learning objective to learn more robust points. In this work, we reuse the architecture of ALIKED including its sparse descriptors branch, and leverage both the SuperPoint and DaD keypoints to supervise our keypoint detector.

**Line Feature Extractors.** Line features have recently emerged as an alternative to point features. As with points, traditional line detectors rely on image gradients to group pixels with strong, coherent gradient magnitude and orientation [3, 16, 20, 55, 64, 77]. With the rise of deep learning, wireframe extraction, detecting interconnected lines that capture the main structure of indoor scenes, has become a popular task [27], leading to numerous wireframe-detection networks [27, 42, 72–74, 78, 83]. In parallel, more general deep line detectors have been proposed to detect all kinds of lines, not only structural ones [9, 23, 28, 30, 44, 61, 65, 71]. Following a trend similar to point features, joint line detection and description

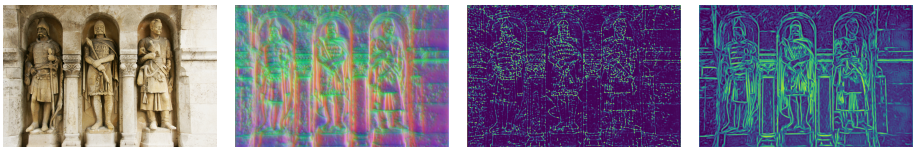
has also been explored [1, 47, 76]. Among these methods, DeepLSD [44] and ScaleLSD [30] stand out for geometric tasks: the former combines a learned image gradient with LSD [20], while the latter scales network capacity and training data. In this work, we follow the DeepLSD design by predicting a deep image gradient and using LSD to extract segments, but introduce multiple performance and efficiency improvements to both the learned and handcrafted components.

**Joint Point-Line Applications.** Points and lines have long been shown to be complementary across a wide range of applications, consistently outperforming approaches relying on points or lines alone. PLNet [69] and Wireframe [17] pioneered the joint detection and description of points and lines within a single network. However, this joint prediction came at the cost of lower performance compared to individual best-in-class baselines. Joint point-line matching is a key component in many multi-view pipelines and has recently been addressed using graph neural networks [24, 41, 45, 67]. This complementarity is particularly valuable in SLAM, where the longer spatial extent of lines improves feature tracking and reduces drift [18, 22, 34, 36, 49, 52, 69, 70, 85]. Lines also provide additional geometric structure beneficial for 3D reconstruction, and recent work has shown that combining points and lines improves both accuracy and completeness [38,39]. Although lines alone do not fully constrain relative pose between two views, they offer useful cues through endpoints and vanishing points, effectively complementing points [26]. Finally, point-line combinations have been widely explored for visual localization [2, 19, 51, 68, 81, 82], where lines are advantageous in low-texture scenes, an established failure mode for point-based methods.

## 3 Method

We present a unified pipeline that jointly predicts points, lines, and feature descriptors using a single neural network, reducing inference time and memory footprint while preserving or exceeding the performance of separate detectors. The model is trained in a student-teacher framework that distills multiple state-of-the-art keypoint and line detectors. An overview of the architecture is provided in Fig. 2.

**Design Choices.** Our objective is to match the performance of independent feature extractors while distilling their capabilities into a single lightweight network for faster inference and a lower memory footprint. As dense matching remains computationally costly, we focus on the classical paradigm of detecting and describing features in a single image. We extensively evaluated point detectors, line detectors, and feature descriptors in isolation and identified the following strong performers among existing methods: SuperPoint [10] and DaD [13] for keypoints, ALIKED [79] for descriptors, and DeepLSD [44] for line detection. To unify all methods, we adopt the architecture of ALIKED [79] for its low latency and memory footprint and extend it with additional heads for the new feature types.



(a) Input image (b) Bottleneck features (c) Keypoint score map (d) Line distance field

**Fig. 3: Visualization of the outputs of the network.** The shared backbone receives an input image (a) and encodes it into bottleneck features (b). We observe that the predicted keypoint score map (c) and line distance field (d) show some similarity (e.g. keypoints are often line endpoints or intersections), justifying their joint prediction.

### 3.1 Encoder

The encoder follows ALIKED [79] and consists of four convolutional blocks with progressively reduced resolution; the last two employ Deformable Convolutions (DCN) [84] to improve geometric invariance. Each block's output is passed through a $1 \times 1$ convolution, then upsampled to the original resolution and finally concatenated with the other blocks' outputs to form the bottleneck $\mathbf{F}$. This representation captures both low- and high-level cues and supports accurate point detection, descriptor computation, and distance-field prediction. Figure 3 illustrates a keypoint score map, a line distance field, and a PCA visualization of $\mathbf{F}$, showing that the bottleneck can encode information for both points and lines, as keypoints frequently lie on line segments.

### 3.2 Point Extraction

Starting from the bottleneck features $\mathbf{F}$, we use the Score Map Head (SMH) from ALIKED [79], a lightweight convolutional head, to extract a keypoint probability heatmap $\mathbf{S} \in [0,1]^{H \times W}$. This map is then fed into a Differentiable Keypoint Detection (DKD) module [80], which applies Non-Maximum Suppression (NMS) thresholding to obtain a set of N keypoints $\{\mathbf{p_i}\}, i \in \{1, \ldots, N\}$, and softargmax within a $3 \times 3$ window to obtain subpixel-accurate coordinates.

As in ALIKED [79], we use the Sparse Deformable Descriptor Head (SDDH). For each predicted keypoint $\mathbf{p_i}$, a $K \times K$ window is extracted around $p_i$ from the bottleneck $\mathbf{F}$, and a small module predicts M sampling positions. These sampling positions are then used to sample the bottleneck features with bilinear interpolation, followed by a $1 \times 1$ convolution and SELU activation. The sampled features are then aggregated into a single descriptor $\mathbf{d_i} \in \mathcal{R}^D$ through a weighted sum with learned weights. This produces a deformable descriptor that can incorporate information from various parts of the image and thus easily adapt to geometric deformations.

### 3.3 Distance Field Prediction

State-of-the-art line detectors such as DeepLSD [44] and ScaleLSD [30] predict an attraction field before extracting line segments from it. DeepLSD regresses in

particular both a distance and an angle field, corresponding to dense maps where each pixel value encodes the distance to the nearest line and the orientation of that line. While experimenting with DeepLSD, we discovered that the angle field predicted by DeepLSD is detrimental, as will be shown in Table 8. Instead of predicting an angle field, we observed that directly using image-gradient orientation yielded better performance, while reducing the complexity of the network. Thus, the line decoder branch of our architecture only predicts a line distance field and we compute the angle field using a GPU-based image gradient angle computation.

Furthermore, DeepLSD relies on a deep and computationally expensive U-Net backbone, which contains roughly 10× more learnable parameters than ALIKED. Since predicting a distance field is a relatively low-level task, we argue and demonstrate that a smaller-scale backbone is sufficient to produce equivalent results. In Sec. 4 we show that our lighter backbone network, followed by the same line decoder branch as in DeepLSD, can obtain results similar to or better than those of DeepLSD, while reducing the number of learnable parameters by a factor of 10. This line decoder consists only of 3 convolutions with batch normalization and ReLU activation, making it extremely lightweight. Overall, this makes the network architecture very lightweight and efficient.

### 3.4 Line Extraction

Similar to DeepLSD [44], our network predicts the distance field and delegates line extraction to the handcrafted LSD detector [20]. LSD is an expensive, two-stage, CPU-only algorithm. It first computes the image gradient (or receives it from a learned detector), and performs an approximate sorting of the gradient magnitude of the pixels. Then, it iterates over the nearly sorted points to start drawing lines, starting from the most promising ones with the highest gradient magnitudes. This sorting-and-seeding step is expensive on CPU and the current LSD implementation has a quadratic cost with respect to the image side lengths, as it is proportional to the number of pixels in the image.

In this work, we propose an accelerated variant of LSD, which removes the expensive approximate sorting by moving the extraction of useful seed points directly to the GPU. Following the observation that most points with low gradient values never contribute to lines and can therefore be discarded, we reduce the number of seed points. We empirically determined that downsampling the grid of seed points with a stride of 2 and keeping the top 20% of pixels with the lowest distance-field values has no impact on the performance of the detector, while improving the efficiency by a factor of 3.

### 3.5 Losses

The keypoint heatmap $\mathbf{S}$ is supervised using the teacher prediction $\hat{\mathbf{S}}$ with a weighted binary cross-entropy loss:

$$\mathcal{L}_{\text{wbce}}(\mathbf{S}, \hat{\mathbf{S}}) = -\lambda\, \hat{\mathbf{S}}\, \log(\mathbf{S}) \;-\; (1 - \hat{\mathbf{S}}) \log(1 - \mathbf{S}). \tag{1}$$


**Fig. 4: Line detection examples.** In spite of its efficiency, UPAL outputs generic lines with quality similar to more expensive methods.

The weighting factor $\lambda$ compensates for the imbalance between positive keypoint pixels and the predominantly near-zero background values in the pseudo-ground truth. Motivated by our empirical analysis of keypoint detectors, we construct the teacher heatmap by taking the element-wise maximum of the predictions from SuperPoint [10] and DaD [13].

The line distance field $\mathbf{D}$ is supervised with an L1 loss against the ground-truth map $\hat{\mathbf{D}}$ generated by DeepLSD [44]. Following DeepLSD, we restrict this loss to pixels within a radius $r = 5$ of each ground-truth line, which improves convergence and accuracy by focusing the supervision on the most informative regions. We also normalize both the prediction and the ground truth to yield more meaningful gradients in the vicinity of lines:

$$\mathcal{L}_D = \left\| \hat{\mathbf{D}}_\mathbf{n} - \mathbf{D}_\mathbf{n} \right\|, \qquad \mathbf{D}_\mathbf{n} = -\log\left(\frac{\mathbf{D}}{r}\right). \tag{2}$$

Local descriptors $\mathbf{d_i}$ are supervised using ground-truth descriptors $\hat{\mathbf{d}}_\mathbf{i}$ with an L1 loss averaged over the top $n = 1000$ detected keypoints:

$$\mathcal{L}_{\text{desc}} = \frac{1}{n} \sum_{i=1}^{n} \left\| \hat{\mathbf{d}}_\mathbf{i} - \mathbf{d_i} \right\|. \tag{3}$$

The ground-truth descriptors are obtained by querying ALIKED-n32 [79] at the predicted keypoint locations.

The final training objective is the sum of all components:

$$\mathcal{L} = \mathcal{L}_{\text{wbce}} + \mathcal{L}_D + \mathcal{L}_{\text{desc}}. \tag{4}$$

### 3.6 Implementation Details

Our encoder and keypoint branch follow the ALIKED-n16 architecture [79], while the distance-field branch is adapted from DeepLSD [44]. We train on 10k distractor images from the Oxford-Paris dataset [50] at a resolution of $800 \times 800$. This dataset contains a mix of indoor and outdoor scenes, making our model

**Table 1: Homography estimation on HPatches [4].** We report the latency in ms and the Area Under the Curve (AUC) of correctly predicted homographies for various pixel error thresholds. Methods are grouped into lightweight and heavyweight categories. Within each category, the best result is shown in **bold**, and the second best is underlined.

| Method | AUC @1px ↑ | AUC @3px ↑ | AUC @5px ↑ | Latency (ms) ↓ |
|---|---|---|---|---|
| **Lightweight Methods** | | | | |
| SIFT [40] | 35.1 | 65.0 | 75.3 | 237 |
| SuperPoint [10] | 35.6 | 65.2 | 75.7 | 50 |
| DISK [66] | 29.9 | 61.8 | 73.2 | 100 |
| XFeat [48] | 34.4 | 62.2 | 73.3 | **44** |
| Wireframe [17] | 38.3 | 66.4 | 76.1 | 308 |
| PLNet [69] | 35.5 | 67.0 | 77.4 | 158 |
| UPAL (Ours) | **39.2** | **67.9** | **77.7** | 62 |
| **Heavyweight Methods** | | | | |
| DeDoDe v2 [15] | **40.9** | **69.1** | 78.4 | 348 |
| DaD [13] + DeDoDe v2 [15] | 39.7 | 68.6 | **78.6** | **164** |
| Ripe [31] | 37.5 | 58.9 | 69.3 | 193 |
| RDD [6] | 34.6 | 60.0 | 68.4 | 522 |

suitable for both conditions. We set $K = 3$ in the SDDH, use 128-dimensional descriptors, apply a keypoint loss weight of $\lambda = 200$, and optimize with Adam at a learning rate of 1e−4.

Training proceeds in two stages: we first learn the distance-field branch for about one hour, using early stopping after a single epoch, then freeze the encoder and distance-field module and train the remaining components for 60 epochs (about 12 hours). ALIKED-derived components are initialized with their pretrained weights. Training is performed on four NVIDIA TITAN GPUs (24 GB each) with a batch size of 4 per GPU.

## 4 Experiments

We evaluate UPAL, first showing that it matches or exceeds the state of the art in point features, line segment detection, and point-line applications. We then highlight its efficiency benefits in terms of inference speed and memory footprint and provide an ablation study of our design choices. A visual comparison of predicted lines is shown in Figure 4.

### 4.1 Point Evaluation

We begin by assessing UPAL against state-of-the-art point extractors, verifying that adding the line-detection branch does not degrade point performance. We use three benchmarks that evaluate point matching, jointly measuring keypoint and descriptor quality. For each benchmark, point features are extracted from image pairs and matched via mutual nearest neighbors. We compare UPAL to the following lightweight methods: SIFT [40], SuperPoint [10], DISK [66], ALIKED-n16 [79] (sharing the same architecture as ours), XFeat [48], the Wireframe point

**Table 2: Relative pose estimation on MegaDepth [35].** The latency and AUC of correctly retrieved poses under various angular error thresholds are reported. Methods are grouped into lightweight and heavyweight categories. Within each category, the best result is highlighted in **bold** and the second best is underlined. Results reported in parentheses are not considered for ranking due to overlap between the training data and the MegaDepth benchmark test set.

| Method | AUC @5° ↑ | AUC @10° ↑ | AUC @20° ↑ | Latency (ms) ↓ |
|---|---|---|---|---|
| **Lightweight Methods** | | | | |
| SIFT [40] | 40.1 | 53.6 | 64.4 | 729 |
| SuperPoint [10] | 51.3 | 64.4 | 73.7 | 78 |
| DISK [66] | 53.9 | 66.1 | 75.3 | 335 |
| ALIKED [79] | **59.8** | **72.0** | **81.1** | 79 |
| XFeat [48] | 44.6 | 59.0 | 70.2 | **28** |
| Wireframe [17] | 47.7 | 60.7 | 70.8 | 307 |
| PLNet [69] | 37.1 | 51.6 | 63.9 | 288 |
| UPAL (Ours) | 58.2 | 70.8 | 79.4 | 80 |
| **Heavyweight Methods** | | | | |
| Ripe [31] | 58.2 | 70.7 | 79.8 | 751 |
| DeDoDe v2 [15] | 57.2 | 69.6 | 78.8 | 1142 |
| DaD [13]+DeDoDe v2 [15] | **60.3** | **72.8** | **81.6** | **485** |
| RDD [6][3] | *(64.8)* | *(77.2)* | *(85.8)* | $\infty$[4] |

detector and descriptor [17], and PLNet [69]. We also add more heavyweight baselines (more than 10M params) as a comparison: DeDoDe v2 [15], DaD [13] keypoints paired with DeDoDe v2 descriptors, Ripe [31], and RDD [6].

**Homography Estimation.** The HPatches [4] dataset contains 580 image pairs, with either illumination or viewpoint variations. We resize the longest side of the images to 800 pixels while keeping the aspect ratio, and extract the top 1024 keypoints per image. The benchmark evaluates the estimation of the dominant planar homography from keypoint correspondences. After matching, we use PoseLib [32] to estimate the homography, and evaluate the Area Under the Curve (AUC) of the reprojection error of the four image corners at thresholds of 1, 3, and 5 pixels, as in [58]. Results can be found in Tab. 1.

**Outdoor Relative Pose Estimation.** The MegaDepth [35] benchmark for feature matching contains 1,500 image pairs sampled with various levels of overlap, taken from two outdoor landmark scenes of the original MegaDepth dataset. For this benchmark, we resize images to 1600 pixels on the long side and use the top 2048 keypoints for each extractor. After matching, we leverage PoseLib [32] to estimate the relative translation and rotation from the correspondences, using the 5-point algorithm [43] to obtain a robust estimate. The results are returned as the AUC of the pose error (maximum of the angular error in both translation and rotation) at three error thresholds. Results are shown in Tab. 2.

**Table 3: Relative pose estimation on ScanNet [8].** The latency and AUC of correctly estimated poses under various angular error thresholds are reported. Methods are grouped into lightweight and heavyweight categories. Within each category, the best result is highlighted in **bold** and the second best is underlined.

| Method | AUC @5° ↑ | AUC @10° ↑ | AUC @20° ↑ | Latency (ms) |
|---|---|---|---|---|
| **Lightweight Methods** | | | | |
| SIFT [40] | 6.7 | 14.9 | 24.2 | 742 |
| SuperPoint [10] | 15.5 | 29.8 | 43.4 | 83 |
| DISK [66] | 10.1 | 19.2 | 29.5 | 348 |
| ALIKED [79] | 6.9 | 13.3 | 20.1 | 82 |
| XFeat [48] | **16.4** | **31.2** | **45.7** | **27** |
| Wireframe [17] | 11.5 | 22.3 | 34.5 | 293 |
| PLNet [69] | 9.5 | 20.7 | 34.8 | 427 |
| UPAL (Ours) | 15.6 | 28.9 | 41.4 | 83 |
| **Heavyweight Methods** | | | | |
| Ripe [31] | 5.5 | 11.6 | 18.9 | 782 |
| DeDoDe v2 [15] | 9.6 | 18.5 | 28.4 | 1135 |
| DaD [13]+DeDoDe v2 [15] | **14.6** | 27.9 | 41.6 | **462** |
| RDD [6] | 14.2 | **28.2** | **42.7** | $\infty$[5] |

**Indoor Relative Pose Estimation.** ScanNet [8] is a large-scale RGB-D indoor dataset, from which [58] sampled 1,500 overlapping image pairs with ground-truth poses. Indoor scenes are particularly challenging due to repetitive structures and texture-less regions, although line features are abundant and can effectively complement point features. The evaluation protocol is identical to that of the previous section, and Tab. 3 shows the results.

**Discussion of the Point Evaluation.** While UPAL is mainly distilled from existing methods for the point branch, it consistently ranks in the top three among all these benchmarks. The first reason is that we explicitly selected the best teacher for each component. For example, while ALIKED [79] has generally the best descriptors overall, we noticed that its points underperform in indoor scenarios, as can be seen in Tab. 3, probably because of its unsupervised training and lack of indoor training data. In contrast, SuperPoint [10] keypoints are supervised as corners and naturally work better in indoor scenarios where corners are abundant. We also empirically found that mixing SuperPoint with DaD [13] keypoints could further boost the overall performance. A second reason for the increased performance of our approach is that jointly detecting points and lines benefits the network by helping it identify more stable keypoints along line structures, which are naturally abundant in indoor environments.

[3] RDD is reported separately since its training data overlaps with the MegaDepth benchmark test set, making these results not directly comparable to the other methods.
[4] RTX 2080 Ti ran out of memory
[5] RTX 2080 Ti ran out of memory

**Table 4: Line detection evaluation on HPatches [4] and RDNIM [46].** We compare the line localization error (Loc Err) on a fixed set of l lines, homography estimation (H Estim), and line repeatability (Rep) at various pixel thresholds. Horizontal lines separate Traditional / Learned / Joint methods.

| | HPatches | | | | | | | | | RDNIM | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Loc Err | | H Estim | | | Rep | | | Time | Loc Err | | H Estim | | | Rep | | | Time |
| | 50l | 300l | 1px | 3px | 5px | 1px | 3px | 5px | (ms) | 50l | 300l | 1px | 3px | 5px | 1px | 3px | 5px | (ms) |
| LSD [20] | 0.50 | 1.42 | **58.3** | **88.3** | 90.9 | 26.4 | 53.6 | 61.6 | 150 | 1.78 | 1.85 | 7.4 | 44.8 | 57.6 | 9.2 | 30.0 | 37.7 | 85 |
| ELSED [64] | 0.71 | 1.63 | 40.7 | 65.8 | 67.9 | 18.8 | 41.8 | 50.8 | 67 | 1.92 | 1.96 | 5.9 | 30.8 | 40.9 | 8.9 | 25.9 | 30.9 | **40** |
| TP-LSD [28] | 1.49 | 1.61 | 10.9 | 34.6 | 42.6 | 11.6 | 42.9 | 57.2 | 65 | 1.93 | 1.93 | 0.2 | 6.3 | 14.2 | 4.9 | 33.3 | **47.5** | 52 |
| SOLD2 [47] | 1.09 | **1.29** | 23.7 | 48.5 | 55.4 | 19.8 | 44.9 | 51.5 | 745 | 1.39 | **1.41** | 0.6 | 9.4 | 16.0 | 8.8 | 20.0 | 24.0 | 546 |
| M-LSD [23] | 2.10 | 2.21 | 15.0 | 54.1 | 63 | 8.9 | 35.6 | 51.2 | 91 | 2.41 | 2.41 | 0.5 | 13.4 | 26.0 | 4.3 | 22.6 | 34.9 | 70 |
| DeepLSD [44] | 0.55 | 1.43 | 58.1 | 87.4 | 90.6 | **27.2** | **54.1** | 61.8 | 473 | 1.73 | 1.81 | 9.4 | 45.3 | 60.5 | 10.1 | 31.8 | 39.5 | 294 |
| Linea [29] | 1.81 | 2.03 | 15.9 | 46.5 | 53.0 | 8.3 | 35.3 | 50.0 | **63** | 2.25 | 2.26 | 0.3 | 9.0 | 17.1 | 2.6 | 25.0 | 40.0 | 44 |
| ScaleLSD [30] | 0.57 | 1.56 | 53.3 | 87.6 | **91.1** | 20.2 | 46.5 | 55.2 | 177 | 1.71 | 1.83 | **9.8** | 47.4 | 63.4 | 9.1 | 25.0 | 31.4 | 166 |
| Wireframe [17] | 1.05 | 1.67 | 33.7 | 72.8 | 79.4 | 16.7 | 43.7 | 52.3 | 294 | 1.88 | 1.90 | 2.0 | 20.4 | 31.4 | 7.3 | 25.0 | 31.9 | 291 |
| PLNet [69] | **0.39** | 1.57 | 41.9 | 67.0 | 71.7 | 11.5 | 30.4 | 40.0 | 240 | **0.99** | 2.35 | 8.3 | **49.4** | **66.8** | 4.0 | 16.1 | 24.4 | 199 |
| UPAL (Ours) | 0.51 | 1.42 | 57.0 | 88.1 | 90.9 | **27.2** | **54.1** | **61.9** | 141 | 1.62 | 1.69 | 9.6 | 45.3 | 58.0 | **13.3** | **35.9** | 43.6 | 83 |

### 4.2 Line Evaluation

We evaluate our line detection on two standard datasets to test the robustness of the methods: HPatches [4] and RDNIM [46]. The latter consists of 1,722 image pairs related by a homography and with challenging day-night variations.

Following [44, 47], we evaluate the repeatability and localization error metrics. Since our teacher's lines are semi-supervised and generic [44], evaluating on a biased set of lines such as in the Wireframe dataset [27] would not provide a fair comparison for generic line detection, and we instead report quality metrics of our lines for downstream tasks. Given a pair of images with ground-truth homography, we detect line segments in both images and establish one-to-one correspondences by warping lines between frames. To decide which pairs should be considered as a match, we compute the symmetric orthogonal distance: for each pair of lines, we calculate the average distance from both endpoints of a segment to the other line, compute this bidirectionally, and take the mean. Repeatability measures the ratio of lines whose match has an error below 1, 3, and 5 pixels, and the localization error returns the average distance of the 10, 50, and 300 most accurate matches. We also compute a homography estimation score, as in [10, 44]. Reusing the previous line matches, we sample minimal sets of 4 line matches and run LO-RANSAC [33] with the orthogonal line distance as reprojection error to estimate a homography.

Tab. 4 compares UPAL to two classical detectors: LSD [20] and ELSED [64]; the recent learned baselines TP-LSD [28], SOLD2 [47], M-LSD tiny [23], DeepLSD [44], Linea [29], and ScaleLSD [30]; and the joint point-line extractors Wireframe [17] and PLNet [69]. For LSD, we use the implementation available at `https://github.com/iago-suarez/pytlsd` and otherwise use the authors' implementations with default parameters for all the other models. Despite its light backbone, UPAL has the highest repeatability across the board and often ranks second in terms of localization error and homography estimation. This shows that line detection is a low-level task that can be solved with as few as 780k

**Table 5: Visual localization on Stairs scene of the 7Scenes dataset [62].** We report the median translation / rotation errors (cm / deg) and pose accuracy at 5 cm / 5° threshold. Please refer to our supplementary material for results on all scenes.

| | Method | T / R err. ↓ | Acc. ↑ |
|---|---|---|---|
| **Points** | SuperPoint [10] | 6.2 / 1.65 | 36.6 |
| | DISK [66] | 7.2 / 2.16 | 28.5 |
| | ALIKED [79] | 7.8 / 1.98 | 35.0 |
| | PLNet [69] | 6.2 / 1.68 | 38.2 |
| | UPAL (Ours) | **5.1 / 1.39** | **49.1** |
| **Points + Lines** | SuperPoint [10]+DeepLSD [44] | 5.0 / 1.33 | 49.6 |
| | ALIKED [79]+TP-LSD [28] | 5.2 / 1.48 | 48.8 |
| | ALIKED [79]+M-LSD [23] | 5.2 / 1.52 | 48.5 |
| | ALIKED [79]+DeepLSD [44] | 5.1 / 1.48 | 49.6 |
| | Wireframe [17] | 4.6 / 1.30 | 53.8 |
| | PLNet [69] | 5.0 / 1.36 | 50.1 |
| | UPAL (Ours) | **4.4 / 1.23** | **54.6** |

parameters. Notably, UPAL improves over its teacher, DeepLSD [44], on the RDNIM dataset. This is due to the removal of the angle field prediction, which yields inaccurate predictions on such challenging images.

### 4.3 Point-Line Applications

**Visual Localization.** We evaluate the combination of points and lines for the task of visual localization using lightweight detectors. We use the 7Scenes dataset [62], consisting of 7 indoor scenes with RGB images and ground-truth poses. We focus here on the scene with the most lines, Stairs, and report the other scenes in the supplementary. We do not use depth information in this experiment. We run the point SfM on database images and find the camera poses of query images using the hloc [56, 57] framework. We then add line segments to the 3D model and perform pose estimation of the query images with a combination of points and lines using the LIMAP framework [39]. Line matching is performed as originally proposed in [45]: we extract descriptors at line endpoints, create an assignment matrix between the lines of both images by summing the matching scores of both endpoints, and finally solve for the best assignment with the Sinkhorn algorithm [7, 63]. For each image, we extract a maximum of 2048 keypoints and 3000 lines, and resize the maximum side of the image to 1024 pixels. Camera poses are estimated using the solvers in PoseLib [32]. We report the median translation and rotation error on the retrieved camera pose, as well as the pose estimation accuracy at a 5cm / 5° threshold.

In Tab. 5 we compare the results of point-only pipelines with point-line approaches and first observe that UPAL achieves the best results among all point baselines. This aligns with Sec. 4.1, where UPAL excels in indoor scenarios. We attribute this to joint training with lines, encouraging keypoints to be better localized on lines, which are abundant in indoor scenarios. Second, the addition of lines improves results over points in isolation, especially in scenes that are challenging for keypoints but rich in lines, like the Stairs scene. This aligns with existing works [39, 45] which demonstrated the complementarity of lines with

**Table 6: Evaluation on multi-view point triangulation on ETH3D [60].**

| Method | Completeness (%) | | | Accuracy (%) | | |
|---|---|---|---|---|---|---|
| | 1cm | 2cm | 5cm | 1cm | 2cm | 5cm |
| SuperPoint [10] | 0.17 | 0.81 | 4.34 | 49.52 | 64.73 | 80.01 |
| DISK [66] | **0.33** | **1.18** | 4.41 | **69.22** | **81.10** | **91.29** |
| ALIKED [79] | 0.13 | 0.56 | 2.76 | 56.61 | 70.78 | 85.18 |
| Wireframe [17] | 0.19 | 0.98 | **5.41** | 50.98 | 66.44 | 82.50 |
| UPAL (Ours) | <u>0.30</u> | <u>1.09</u> | <u>4.45</u> | <u>64.80</u> | <u>77.57</u> | <u>88.87</u> |

respect to points. Finally, UPAL with points and lines has the best overall results, showcasing the synergy of jointly learned features.

**3D Reconstruction.** We further evaluate 3D point-cloud reconstruction from multi-view posed images. We evaluate the accuracy and completeness of the point triangulation [59] on ETH3D [60] following existing practices [12,37]. All detectors are tested with mutual nearest neighbor matching with a maximum number of keypoints of 2048. Tab. 6 shows that we achieve strong performance both in terms of accuracy and completeness, consistently exceeding SuperPoint [10] and ALIKED [79], while falling behind DISK [66], which performs particularly well under this setup. This further proves the applicability of UPAL for accurate 3D reconstruction with a compact, unified architecture.

### 4.4 Efficiency

We benchmark efficiency against combinations of state-of-the-art models on an NVIDIA RTX 2080 Ti GPU (11 GB) as well as on an Intel Core i9-13900K CPU, on 500 images from the Oxford-Paris dataset [50] with a batch size of one. Independent point and line extractors are run sequentially. Images are resized to $800 \times 800$. As shown in Tab. 7, UPAL achieves a mean GPU inference time of 70 ms, providing roughly a $4\times$ speedup over the SOTA ALIKED [79] + DeepLSD [44]. This efficiency stems from joint inference, a smaller network architecture, and an optimized line post-processing pipeline. UPAL is also competitive with lightweight approaches such as ALIKED [79] + M-LSD [23], while obtaining better performance (e.g. in Tab. 4). Furthermore, our number of parameters is lower by an order of magnitude compared to previous joint extractors (Wireframe [17] and PLNet [69]), and GPU latency is 37% lower.

### 4.5 Ablation study

We validate our design choices for the line detection implementation on the HPatches dataset [4]. In Tab. 8 we compare our final model to various versions of our line post-processing. First, we show that DeepLSD without an angle field prediction (2) obtains a higher repeatability and lower localization error than the vanilla DeepLSD (1). Next, we show the benefits of our enhancements over the

**Table 7: Runtime evaluation.** Parameter counts are reported, and latencies are averaged over 500 sequential detections on an NVIDIA RTX 2080 Ti GPU and Intel Core i9-13900K CPU.

| | Nb params ↓ | Latency (ms) ↓ | |
|---|---|---|---|
| | | GPU | CPU |
| SuperPoint [10] + DeepLSD [44] | 9.8M | 293 | 2259 |
| ALIKED [79] + DeepLSD [44] | 9.2M | 286 | 2239 |
| ALIKED [79] + M-LSD [23] | 1.3M | **54** | **525** |
| ALIKED [79] + TP-LSD [28] | 25M | 56 | 1582 |
| DaD [13] + DeDoDe v2 [15] + ScaleLSD [30] | 120M | 147 | >200K |
| Wireframe [17] | 5.8M | 266 | 2493 |
| PLNet [69] | 7.5M | 112 | 6233 |
| UPAL (Ours) | **0.78M** | 70 | 976 |

**Table 8: Ablation study on the HPatches dataset [4].** We compare the localization error, homography estimation, repeatability, and latency for variants of our approach.

| | Loc Error ↓ | H estimation ↑ | Repeatability ↑ | Latency (ms) ↓ |
|---|---|---|---|---|
| *(1)* DeepLSD [44] | 1.432 | 90.6 | 27.2 | 233 |
| *(2)* DeepLSD no AF | **1.341** | 90.6 | 28.2 | 218 |
| *(3)* LSD [20] | 1.414 | **91.1** | 26.3 | 36 |
| *(4)* (3) + parall. ang. comp. | 1.414 | **91.1** | 26.3 | 27 |
| *(5)* (4) + stride 2 | 1.455 | 90.6 | 26.3 | 19 |
| *(6)* (5) + subset | 1.476 | 90.6 | 26.3 | 17 |
| *(7)* (6) + GPU | 1.343 | 90.7 | **28.9** | **11** |

vanilla LSD (3). For each modification, computing the gradient angle in parallel (4), sampling seed points with a stride of 2 (5), selecting only a subset of the seed points (6), and offloading the image preprocessing to the GPU (7), we observe consistent improvements in latency, with no degradation in performance.

## 5 Conclusion

In this work, we introduced UPAL, a unified architecture capable of jointly extracting keypoints, line segments, and feature descriptors, while maintaining high efficiency and preserving the performance of each individual component. Our method achieves a 4× speedup compared to combining existing standalone components, while reducing the memory footprint by at least one order of magnitude. Looking ahead, we plan to integrate UPAL with a lightweight, jointly learned matcher to enable end-to-end point-line image matching. We believe that our efficient, joint feature extractor marks an important step toward broadening access to advanced geometric perception, ultimately facilitating the deployment of point-line applications on embedded platforms and in low-compute environments.

**Acknowledgments:** We would like to thank Viktor Larsson for valuable discussions regarding the LSD line detector. During our efforts to develop a faster variant of LSD, he suggested precomputing the distance field on the GPU. This insight ultimately helped shape the accelerated version of LSD presented in this work. We also thank Olivia Buset for her help during this project.

## References


1. Abdelbaki, H., Frohlich, R., Vilajosana, V., Katona, Z.: L2d2: Learnable line detector and descriptor. In: 3DV (2021)
2. Agostinho, S., Gomes, J., Del Bue, A.: Cvxpnpl: A unified convex solution to the absolute pose estimation problem from point and line correspondences (2019)
3. Akinlar, C., Topal, C.: Edlines: A real-time line segment detector with a false detection control. In: ICIP (2011)
4. Balntas, V., Lenc, K., Vedaldi, A., Mikolajczyk, K.: Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In: CVPR (2017)
5. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (surf). CVIU (2008)
6. Chen, G., Fu, T., Chen, H., Teng, W., Xiao, H., Zhao, Y.: Rdd: Robust feature detector and descriptor using deformable transformer. In: Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR). pp. 6394–6403 (June 2025)
7. Cuturi, M.: Sinkhorn distances: Lightspeed computation of optimal transport. NeurIPS (2013)
8. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In: CVPR (2017)
9. Dai, X., Yuan, X., Gong, H., Ma, Y.: Fully convolutional line parsing. In: IPIN (2019)
10. DeTone, D., Malisiewicz, T., Rabinovich, A.: Superpoint: Self-supervised interest point detection and description. CVPRW (2018)
11. Dusmanu, M., Rocco, I., Pajdla, T., Pollefeys, M., Sivic, J., Torii, A., Sattler, T.: D2-Net: A Trainable CNN for Joint Detection and Description of Local Features. In: CVPR (2019)
12. Dusmanu, M., Schönberger, J.L., Pollefeys, M.: Multi-view optimization of local feature geometry. In: ECCV (2020)
13. Edstedt, J., Bökman, G., Wadenbäck, M., Felsberg, M.: DaD: Distilled Reinforcement Learning for Diverse Keypoint Detection. arXiv (2025)
14. Edstedt, J., Bökman, G., Wadenbäck, M., Felsberg, M.: DeDoDe: Detect, Don't Describe - Describe, Don't Detect for Local Feature Matching. In: 3DV (2024)
15. Edstedt, J., Bökman, G., Zhao, Z.: DeDoDe v2: Analyzing and Improving the DeDoDe Keypoint Detector . In: CVPRW (2024)
16. Elder, J.H., Almazan, E.J., Qian, Y., Tal, R.: Mcmlsd: A probabilistic algorithm and evaluation framework for line segment detection. arXiv (2020)
17. Ferre, I., Baumela, L., Suárez, I.: Learning to detect and describe a wireframe. In: Proceedings of the 12th Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA) (2025)
18. Fu, Q., Wang, J., Yu, H., Ali, I., Guo, F., He, Y., Zhang, H.: Pl-vins: Real-time monocular visual-inertial slam with point and line features (2020)
19. Gao, S., Wan, J., Ping, Y., Zhang, X., Dong, S., Yang, Y., Ning, H., Li, J., Guo, Y.: Pose refinement with joint optimization of visual points and lines. In: IROS (2022)
20. von Gioi, R.G., Jakubowicz, J., Morel, J.M., Randall, G.: Lsd: A fast line segment detector with a false detection control. IEEE TPAMI (2010)
21. Gleize, P., Wang, W., Feiszli, M.: Silk – simple learned keypoints. In: ICCV (2023)
22. Gomez-Ojeda, R., Moreno, F.A., Zuniga-Noël, D., Scaramuzza, D., Gonzalez-Jimenez, J.: Pl-slam: A stereo slam system through the combination of points and line segments. IEEE Transactions on Robotics **35**(3), 734–746 (2019)

23. Gu, G., Ko, B., Go, S., Lee, S.H., Lee, J., Shin, M.: Towards real-time and lightweight line segment detection. In: AAAI (2022)
24. Guo, Z., Lu, H., Yu, Q., Guo, R., Xiao, J., Yu, H.: HDPL: a hybrid descriptor for points and lines based on graph neural networks. Industrial Robot: the International Journal of Robotics Research and Application (2021)
25. Harris, C., Stephens, M.: A combined corner and edge detector. In: Alvey Vision Conference (1988)
26. Hrubý, P., Liu, S., Pautrat, R., Pollefeys, M., Baráth, D.: Handbook on leveraging lines for two-view relative pose estimation. In: 3DV (2023)
27. Huang, K., Wang, Y., Zhou, Z., Ding, T., Gao, S., Ma, Y.: Learning to parse wireframes in images of man-made environments. In: CVPR (2018)
28. Huang, S., Qin, F., Xiong, P., Ding, N., He, Y., Liu, X.: Tp-lsd: Tri-points based line segment detector. In: Computer Vision – ECCV 2020 (2020)
29. Janampa, S., Pattichis, M.: Linea: Fast and accurate line detection using scalable transformers (2025), `https://arxiv.org/abs/2505.16264`
30. Ke, Z., Tan, B., Zheng, X., Shen, Y., Wu, T., Xue, N.: Scalelsd: Scalable deep line segment detection streamlined. In: CVPR (2025)
31. Künzel, J., Hilsmann, A., Eisert, P.: Ripe: Reinforcement learning on unlabeled image pairs for robust keypoint extraction (2025), `https://arxiv.org/abs/2507.04839`
32. Larsson, V.: Poselib-minimal solvers for camera pose estimation (2020), available at https://github.com/vlarsson/poselib
33. Lebeda, K., Matas, J., Chum, O.: Fixing the Locally Optimized RANSAC. In: BMVC (2012)
34. Li, W., Shao, Y., Wang, Y., Wang, S., Bai, X., Li, D.: Idls: Inverse depth line based visual-inertial slam (2023)
35. Li, Z., Snavely, N.: Megadepth: Learning single-view depth prediction from internet photos. CVPR (2018)
36. Lim, H., Jeon, J., Myung, H.: Uv-slam: Unconstrained line-based slam using vanishing points for structural mapping. IEEE Robotics and Automation Letters **7**(2), 1518–1525 (2022)
37. Lindenberger, P., Sarlin, P.E., Larsson, V., Pollefeys, M.: Pixel-perfect structure-from-motion with featuremetric refinement. In: ICCV (2021)
38. Liu, S., Gao, Y., Zhang, T., Pautrat, R., Schönberger, J.L., Larsson, V., Pollefeys, M.: Robust incremental structure-from-motion with hybrid features. In: ECCV (2024)
39. Liu, S., Yu, Y., Pautrat, R., Pollefeys, M., Larsson, V.: 3d line mapping revisited. In: CVPR (2023)
40. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV **60**(2), 91–110 (2004)
41. Ma, Q., Jiang, G., Wu, J., Cai, C., Lai, D., Bai, Z., Chen, H.: WGLSM: An end-to-end line matching network based on graph convolution. Neurocomputing **453**, 195–208 (2021)
42. Meng, Q., Zhang, J., Hu, Q., He, X., Yu, J.: Lgnn: A context-aware line segment detector. arXiv **abs/2008.05892** (2020)
43. Nistér, D.: An efficient solution to the five-point relative pose problem. In: CVPR (2003)
44. Pautrat, R., Larsson, V., Yu, Y., Pollefeys, M.: Deeplsd: Line segment detection and refinement with deep image gradients. In: CVPR (2023)
45. Pautrat*, R., Suárez*, I., Yu, Y., Pollefeys, M., Larsson, V.: GlueStick: Robust image matching by sticking points and lines together. In: ICCV (2023)

46. Pautrat, R., Larsson, V., Oswald, M.R., Pollefeys, M.: Online invariance selection for local feature descriptors. In: ECCV (2020)
47. Pautrat, R., Lin, J.T., Larsson, V., Oswald, M.R., Pollefeys, M.: SOLD2: Self-supervised occlusion-aware line description and detection. In: CVPR (2021)
48. Potje, G., Cadar, F., Araujo, A., Martins, R., Nascimento, E.R.: Xfeat: Accelerated features for lightweight image matching. In: CVPR (2024)
49. Pumarola, A., Vakhitov, A., Agudo, A., Sanfeliu, A., Moreno-Noguer, F.: Pl-slam: Real-time monocular visual slam with points and lines. In: ICRA (2017)
50. Radenović, F., Iscen, A., Tolias, G., Avrithis, Y., Chum, O.: Revisiting oxford and paris: Large-scale image retrieval benchmarking. In: CVPR (2018)
51. Ramalingam, S., Bouaziz, S., Sturm, P.: Pose estimation using both points and lines for geo-localization. In: ICRA (2011)
52. Ren, G., Cao, Z., Liu, X., Tan, M., Yu, J.: Plj-slam: Monocular visual slam with points, lines, and junctions of coplanar lines. IEEE Sensors Journal **22**(15) (2022)
53. Revaud, J., Weinzaepfel, P., de Souza, C.R., Humenberger, M.: R2D2: repeatable and reliable detector and descriptor. In: NeurIPS (2019)
54. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: An efficient alternative to sift or surf. In: ICCV (2011)
55. Salaün, Y., Marlet, R., Monasse, P.: Multiscale line segment detector for robust and accurate SfM. In: ICPR (2016)
56. Sarlin, P.E.: Visual localization made easy with hloc. `https://github.com/cvg/Hierarchical-Localization`
57. Sarlin, P.E., Cadena, C., Siegwart, R., Dymczyk, M.: From coarse to fine: Robust hierarchical localization at large scale. In: CVPR (2019)
58. Sarlin, P.E., DeTone, D., Malisiewicz, T., Rabinovich, A.: SuperGlue: Learning feature matching with graph neural networks. In: CVPR (2020)
59. Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: CVPR (2016)
60. Schops, T., Schönberger, J.L., Galliani, S., Sattler, T., Schindler, K., Pollefeys, M., Geiger, A.: A multi-view stereo benchmark with high-resolution images and multi-camera videos. In: CVPR (2017)
61. Sha, Z., Zhong, B., Chen, X., Wang, Z.: Dealsd: A deep edge assisted line segment detector. Expert Systems with Applications **297** (2025)
62. Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., Fitzgibbon, A.: Scene coordinate regression forests for camera relocalization in RGB-D images. In: CVPR (2013)
63. Sinkhorn, R., Knopp, P.: Concerning nonnegative matrices and doubly stochastic matrices. Pacific Journal of Mathematics **21**(2), 343–348 (1967)
64. Suárez, I., Buenaposada, J.M., Baumela, L.: Elsed: Enhanced line segment drawing. PR **127** (2022)
65. Teplyakov, L., Erlygin, L., Shvets, E.: Lsdnet: Trainable modification of lsd algorithm for real-time line segment detection. IEEE Access **10** (2022)
66. Tyszkiewicz, M., Fua, P., Trulls, E.: Disk: Learning local features with policy gradient. In: NeurIPS (2020)
67. Ubingazhibov, A., Pautrat, R., Suárez, I., Liu, S., Pollefeys, M., Larsson, V.: Lightgluestick: a fast and robust glue for joint point-line matching. In: CVPRW (2025)
68. Vakhitov, A., Funke, J., Moreno-Noguer, F.: Accurate and linear time pose estimation from points and lines. In: ECCV (2016)
69. Xu, K., Hao, Y., Yuan, S., Wang, C., Xie, L.: AirSLAM: An efficient and illumination-robust point-line visual slam system. arXiv preprint arXiv:2408.03520 (2024), `https://arxiv.org/abs/2408.03520`

70. Xu, L., Yin, H., Shi, T., Jiang, D., Huang, B.: Eplf-vins: Real-time monocular visual-inertial slam with efficient point-line flow features. IEEE Robotics and Automation Letters **8**(2) (2023)
71. Xu, Y., Xu, W., Cheung, D., Tu, Z.: Line segment detection using transformers without edges. arXiv **abs/2101.01909** (2021)
72. Xue, N., Bai, S., Wang, F., Xia, G.S., Wu, T., Zhang, L.: Learning attraction field representation for robust line segment detection. In: CVPR (2019)
73. Xue, N., Wu, T., Bai, S., Wang, F.D., Xia, G.S., Zhang, L., Torr, P.H.: Holistically-attracted wireframe parsing: From supervised to self-supervised learning. arXiv (2022)
74. Xue, N., Wu, T., Bai, S., Wang, F., Xia, G.S., Zhang, L., Torr, P.H.: Holistically-attracted wireframe parsing. In: CVPR (2020)
75. Yoon, S., Kim, A.: Line as a visual sentence: Context-aware line descriptor for visual localization. IEEE Robotics and Automation Letters (RAL) **6**(4), 8726–8733 (2021)
76. Zhang, H., Luo, Y., Qin, F., He, Y., Liu, X.: Elsd: Efficient line segment detector and descriptor. In: ICCV (2021)
77. Zhang, Y., Wei, D., Li, Y.: AG3line: Active grouping and geometry-gradient combined validation for fast line segment extraction. PR **113** (2021)
78. Zhang, Z., Li, Z., Bi, N., Zheng, J., Wang, J., Huang, K., Luo, W., Xu, Y., Gao, S.: Ppgnet: Learning point-pair graph for line segment detection. In: CVPR (2019)
79. Zhao, X., Wu, X., Chen, W., Chen, P.C., Xu, Q., Li, Z.: Aliked: A lighter keypoint and descriptor extraction network via deformable transformation. TIM (2023)
80. Zhao, X., Wu, X., Miao, J., Chen, W., Chen, P.C.Y., Li, Z.: Alike: Accurate and lightweight keypoint detection and descriptor extraction. IEEE Transactions on Multimedia (2022)
81. Zhou, L., Wang, S., Kaess, M.: A fast and accurate solution for pose estimation from 3d correspondences. In: ICRA (2020)
82. Zhou, L., Ye, J., Kaess, M.: A stable algebraic camera pose estimation for minimal configurations of 2d/3d point and line correspondences. In: ACCV (2018)
83. Zhou, Y., Qi, H., Ma, Y.: End-to-end wireframe parsing. ECCV (2019)
84. Zhu, X., Hu, H., Lin, S., Dai, J.: Deformable convnets v2: More deformable, better results. In: CVPR (2019)
85. Zuo, X., Xie, X., Liu, Y., Huang, G.: Robust visual slam with point and line features. In: IROS (2017)