GlueStick: Robust Image Matching by Sticking Points and Lines Together

Rémi Pautrat* [1] Iago Suárez* [2] Yifan Yu[1] Marc Pollefeys[1,3] Viktor Larsson[4]
[1] Department of Computer Science, ETH Zurich [2] Qualcomm XR Labs Europe
[3] Microsoft Mixed Reality and AI Zurich lab [4] Lund University

Abstract

Line segments are powerful features complementary to points. They offer structural cues, robust to drastic viewpoint and illumination changes, and can be present even in texture-less areas. However, describing and matching them is more challenging compared to points due to partial occlusions, lack of texture, or repetitiveness. This paper introduces a new matching paradigm, where points, lines, and their descriptors are unified into a single wireframe structure. We propose GlueStick, a deep matching Graph Neural Network (GNN) that takes two wireframes from different images and leverages the connectivity information between nodes to better glue them together. In addition to the increased efficiency brought by the joint matching, we also demonstrate a large boost of performance when leveraging the complementary nature of these two features in a single architecture. We show that our matching strategy outperforms the state-of-the-art approaches independently matching line segments and points for a wide variety of datasets and tasks. The code is available at https://github.com/cvg/GlueStick.

1. Introduction

Line segments are high-level geometric structures useful in a wide range of computer vision tasks such as SLAM [18, 81, 41], pose estimation [68], construction monitoring [25, 4], and 3D reconstruction [21, 74, 80]. Lines are ubiquitous in structured scenes and offer stronger constraints than feature points. In particular, lines shine in low-textured scenes where point-based approaches struggle.

However, compared to keypoints, line segments are often poorly localized in the image and suffer from lower repeatability. Line segments are also more challenging to describe since they can cover a large spatial extent in the image and suffer from occlusions and perspective effects due to viewpoint changes. Furthermore, lines often appear as part of repetitive structures in human-made environments, making classical descriptor-based matching fail. For this reason, typical matching heuristics such as mutual nearest neighbor and Lowe's ratio test [34] are often less effective for lines.

Recently, deep learning has ushered in a new paradigm for feature point matching using Graph Neural Networks (GNNs) [47, 57]. This new approach bypasses the need

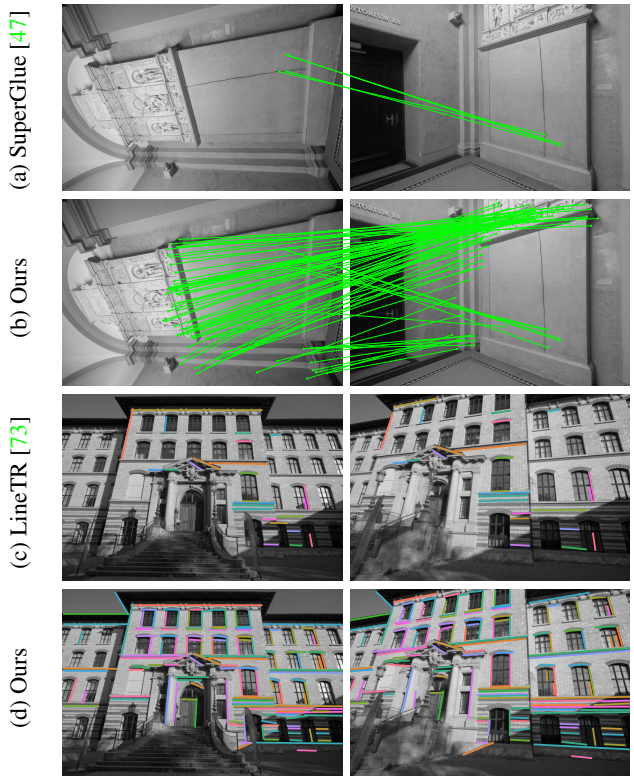


Figure 1: **Joint matching of points and lines.** Matching feature points often fails in textureless areas (a), while current line matching methods struggle with large viewpoint changes (c). We propose GlueStick, a network jointly matching points and lines. While none of the methods were trained on the rotations of (a)(b), line matches can guide GlueStick while SuperGlue [47] fails, and vice-versa in (d) where points can complement the line matching. For clarity reasons, we show here only the correct matches.

* Authors contributed equally.

for matching heuristics or even outlier removal techniques, thanks to the high precision of the predicted matches [47]. A key component to achieve this is to leverage the positional encoding of keypoints directly in the network and to let it combine visual features with geometric information [47, 57, 59, 24]. Letting the GNN reason with all features simultaneously brings in additional context and can disambiguate repetitive structures (Fig. 1).

Even though recent advances leveraged similar ideas to enrich line descriptors [73], directly transferring this GNN approach to line matching is not trivial. The large extent of lines and their lack of repeatability make it hard to find a good feature representation for them. In this paper, we take inspiration from SuperGlue [47] and introduce GlueStick, to jointly match keypoints and line segments. Our goal is to leverage their complementary nature in the matching process. By processing them together in a single GNN, the network can learn to resolve ambiguous line matches by considering nearby distinctive keypoints, and vice versa. We propose to leverage the connectivity between points and lines via a unified wireframe structure, effectively replacing previous handcrafted heuristics for line matching [75, 49, 32] by a data-driven approach.

Our network takes as input sparse keypoints, lines, and their descriptors extracted from an image pair, and outputs a set of locally discriminative descriptors enriched with the context from all features in both images, before establishing the final matches. Inside the network, keypoints and line endpoints are represented as nodes of a wireframe. The network is composed of self-attention layers between nodes, cross-attention layers exchanging information across the two images, and a new line message passing module enabling communication between neighboring nodes of the wireframe. After the GNN, points and lines are split into two separate matching matrices and a dual-softmax is used to find the final assignment of the features. Overall, our contributions are as follows:

1. We replace heuristic geometric strategies for line matching with a data-driven approach, by jointly matching points and lines within a single network.
2. We offer a novel architecture exploiting the local connectivity of the features within an image.
3. We experimentally show large improvements of our method over previous state-of-the-art point and line matchers on a wide range of datasets and tasks.

2. Related Work

Line segment detection is a classical problem in computer vision that can be traced back to the Hough Transform [22] and its improvements [37, 17, 77]. Local line segment detectors [19, 2, 56, 55] are efficient alternatives that fit segments to local regions with a prominent gradient. With more computational cost, deep line segment detectors offer better detection results, in particular for specialized tasks such as wireframe parsing [23, 79, 70, 76, 71, 69]. In this work, we train our network with the LSD detector [19], due to its high accuracy and versatility.

Line segment description is classically performed by using the image gradients to describe the texture around each segment locally [63, 65, 75, 62, 32]. More recently, deep learning models have emerged. Early works mimic keypoint patch descriptors by extracting a patch around each line and describing it via a neural network [29, 28, 1]. An alternative approach is to sample points along the line and to describe them separately [60, 40]. SOLD^2 [40] introduces a joint detection and description of line segments, as well as a mechanism to handle the partial occlusion of lines during matching. In this paper, we use the (point-based) SuperPoint [14] dense descriptors, interpolated at the two line endpoints. While these might not capture the full visual context of the line, having comparable descriptors for both points and lines is crucial in our network.

Since descriptor-based matching for line segments is generally more difficult than for points, several methods in the literature complemented the descriptor matching with geometric scene information [31]: global rotation between images [75]; properties of pairs of matched lines like the angle between segments, intersection ratios or projection ratios [75, 63]; line-point invariants [16]; cross-ratio [43] or consistency with a fundamental matrix estimated from points [49, 32]. However, estimating the fundamental matrix to perform matching generates a chicken-and-egg problem, and these heuristics often fail in realistic scenarios. For this reason, recent point matchers are learning the geometric relationships between the points of two images, thus implicitly learning the underlying epipolar geometry [47].

Matching with transformers. SuperGlue [47] uses a GNN to process keypoints and their descriptors from two input images, adding a positional encoding to better disambiguate repetitive patterns. Several variations of this method have been proposed later, with higher efficiency [11, 51] and with dense predictions [57, 59, 24, 64, 12, 15].

WGLSM [35] combines a CNN and a GNN to match line segments, but without feature points. In the GNN, each line is represented with a single node, and the assignation is solved using a single Sinkhorn matrix. LineTR [73] proposes to use attention inside points sampled for each line to deal with the line scale changes and occlusions. HDPL [20] mixes points and lines in the same GNN, each line being represented with a single node in the GNN. They only use a single Sinkhorn matrix, allowing point-line assignments.

In contrast to these methods, we model each line segment endpoint as a separate node in the GNN. The endpoints are, in most cases, consistent with the underlying epipolar geometry, allowing the network to leverage both points and

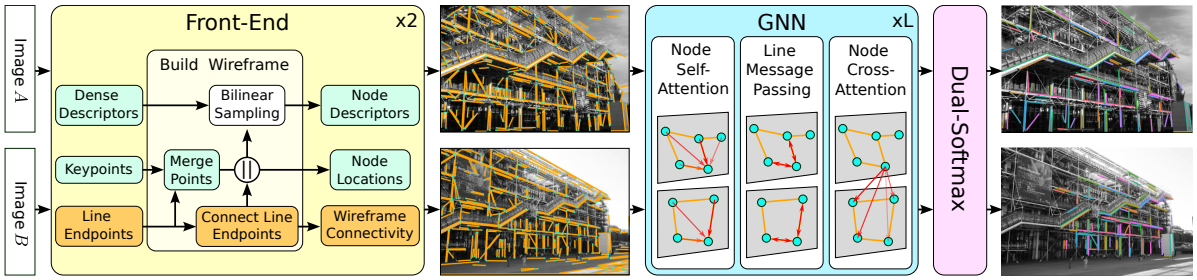


Figure 2: **Overview of GlueStick.** Keypoints, dense descriptors, and lines are extracted from two images, and unified into two wireframes (front-end). We then enrich the features of the nodes of both wireframes via self, line, and cross-attention inside a Graph Neural Network (GNN). Finally, points and lines are matched separately via two dual-softmax modules.

lines to disambiguate the matching. In our ablation study, we show that matching points and line endpoints together already greatly improves the matching performance.

3. GlueStick

In this section, we show how to combine points and lines within the same network. The motivation for this is that each feature can leverage cues from the neighbouring features to improve the matching performance. For example, a line using the surrounding points or vice-versa. Furthermore, the network can automatically discover combinations of points and lines that are useful for matching, instead of heuristically mining them as in previous works [32]. Our architecture, displayed in Fig. 2, consists in three blocks:

1. **Front-End**: We extract points, lines, and their descriptors with common feature detectors, then combine them into a single wireframe (Sec. 3.1).

2. **GNN**: The goal of this block, described in Sec. 3.2, is to combine the visual and spatial information of each feature, and to allow interaction between all features, regardless of their original receptive field. The output is a set of updated descriptors, enriched by the knowledge of relevant features within and across images, as well as within nodes connected by a line segment.

3. **Dual-Softmax**: The final assignation is solved separately for points and lines, using two independent dual-softmax modules [44, 57], as detailed in Sec. 3.3.

3.1. From Points and Lines to Wireframes

The input to our GNN is a set of points, their associated local descriptors, and a connectivity matrix indicating which points are connected by a line. The first step is to establish this connectivity and build the wireframe graph.

We use SuperPoint (SP) [14] to predict keypoints and a dense descriptor map, and we detect segments with the general-purpose LSD [19] detector. Keypoints located close to line endpoints are redundant, so we remove SP keypoints that are within a small distance d to existing line endpoints.

Furthermore, line segments generated by generic detectors such as LSD are usually disconnected. To give more structure to the input and to explicitly encourage the network to reason in terms of line connectivity, we merge close-by endpoints, again with a distance threshold d. This process lifts the unstructured line cloud into an interconnected wireframe. After this step, each keypoint and line endpoint is represented as a node in the wireframe, with different connectivities for each node: 0 for an isolated keypoint, 2 for a corner, etc. We then interpolate the dense SP feature map at the node locations to equip them with a visual descriptor. Note that this endpoint merging is modifying the position of the endpoints but not the number of lines. For downstream tasks requiring high precision, we use the original position of the endpoints, to keep the subpixel accuracy of the original detector.

3.2. Attention-based Graph Neural Network (GNN)

A key part of our method is the GNN, which aggregates visual and spatial information to predict a set of *enriched* feature descriptors, that are used to establish the final matches via descriptor similarity. Within the network, each node (either a keypoint, or a line endpoint) is associated with an initial descriptor that is based on the visual appearance as well as the position in the image.

Let A and B be a pair of images. For each image, the inputs of the network are: a set of nodes $\mathbf{p}$, with coordinates (x_p, y_p), confidence score s_p and visual descriptors $\mathbf{d}^{vis} \in \mathbb{R}^D$; and a set of line segments $\mathbf{l}$ defined as a pair of nodes (x_p, y_p) and (x'_p, y'_p), and with a line score s_l. This line score can be any value returned by the line detector indicating the quality of the line, or simply the length of the line to put more emphasis on longer lines. The node score s_p is either coming from the keypoint detector, or is equal to s_l when it is a line endpoint.

Positional and Directional Encoding. The first step is to encode the spatial information of each feature. To this end,

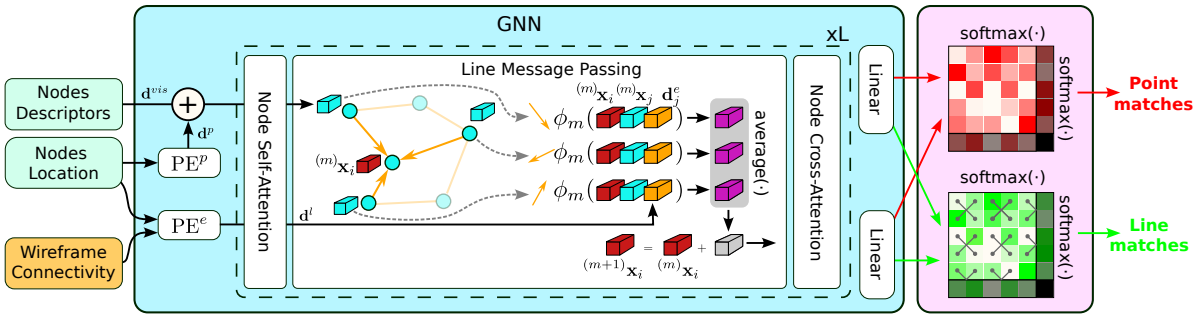


Figure 3: **Graph Neural Network (GNN) architecture.** Node features of the wireframe are enriched via several communication layers. Our Line Message Passing exchanges information between neighboring nodes that are connected together.

we learn two positional encoders (PE^p and PE^e) with Multi-layer Perceptron (MLP) that generate a *spatial* descriptor $\mathbf{d}^p$ for each node and an *edge*-descriptor $\mathbf{d}^e$ for each line segment originating from this node. A node with connectivity 3 will for instance get assigned one $\mathbf{d}^p$ and 3 $\mathbf{d}^e$ (one for each outgoing line segment). The edge-positional encoding takes as additional information the offset to the other endpoint of its line segment, allowing it to have access to the angle and length of the line segment:

$$\begin{aligned}\mathbf{d}^p &= \text{PE}^p([x_p, y_p, s_p]^\top) \\ \mathbf{d}^e &= \text{PE}^e([x_p, y_p, x'_p - x_p, y'_p - y_p, s_l]^\top).\end{aligned} \tag{1}$$

The spatial-descriptor $\mathbf{d}^p$ is used to initialize the node features, while the edge-descriptors $\mathbf{d}^e$ are used in the line message passing (see below).

Network Architecture. Our GNN is a complete graph with three types of undirected edges (See Fig. 2). Self-attention edges $\mathcal{E}_{\text{self}}$, connect nodes of one image with all the nodes of the same image. Line edges $\mathcal{E}_{\text{line}}$, connect nodes that are endpoints of the same line. Cross attention edges $\mathcal{E}_{\text{cross}}$, connect nodes of one image to the other image nodes.

A node i is initially assigned a feature descriptor fusing its spatial and visual information: ${}^{(0)}\mathbf{x}_i = \mathbf{d}_i^p + \mathbf{d}_i^{vis}$. This node descriptor is then iteratively enriched and refined with the context of all the other descriptors in L iterations of Self, Line, and Cross layers. Finally, the features of each node are linearly projected to obtain the output features. The next paragraphs detail each type of layer.

Self and Cross Layers. $\mathcal{E}_{\text{self}}$ and $\mathcal{E}_{\text{cross}}$ edges are similarly defined as in [47]. The m-th feature update is defined by a residual message passing:

$${}^{(m+1)}\mathbf{x}_i = {}^{(m)}\mathbf{x}_i + \psi_m\left(\left[{}^{(m)}\mathbf{x}_i || a_m({}^{(m)}\mathbf{x}_i; \mathcal{E})\right]\right), \tag{2}$$

where $||$ denotes concatenation, the function ψ_m is modeled with an MLP, and $a_m({}^{(m)}\mathbf{x}_i; \mathcal{E})$ is the Multi-Head Attention mechanism from [61] applied to the set of edges $\mathcal{E}$:

$$a_m(\mathbf{x}_i; \mathcal{E}) = \sum_{j:(i,j)\in\mathcal{E}} \text{softmax}_j\left(\frac{\mathbf{q}_i^\top \mathbf{k}_j}{\sqrt{D}}\right)\mathbf{v}_j, \tag{3}$$

where the keys $\mathbf{k}_j$, queries $\mathbf{q}_i$, and values $\mathbf{v}_j$ are computed as linear projections of the node features $\mathbf{x}_i$ and $\mathbf{x}_j$. In self-attention layers, $\mathbf{k}_j$ and $\mathbf{v}_j$ will come from the same image, whereas in cross-attention they will come from the other image. Self-attention allows the network to leverage the context of the full image, and to resolve repetitive structures. Cross-attention moves corresponding features closer in descriptor space and can search for similar node structures in the other image to fully leverage spatial information.

Line Message Passing. We describe here our novel Line Message Passing (LMP) transferring information across the line edges $\mathcal{E}_{\text{line}}$. By connecting line segments in a wireframe structure, we allow the i-th node to leverage the local edge connectivity to the set $\mathcal{N}_i$ of neighboring nodes, and to look for the same type of connectivities in the other image. This mechanism is enabled by the m-th LMP update which aggregates the information contained in the two endpoint features ${}^{(m)}\mathbf{x}_i$ and ${}^{(m)}\mathbf{x}_j$ and the corresponding endpoint positional encoding $\mathbf{d}_j^e$:

$${}^{(m+1)}\mathbf{x}_i = {}^{(m)}\mathbf{x}_i + \sum_{j\in\mathcal{N}_i} \frac{\phi_m([{}^{(m)}\mathbf{x}_i || {}^{(m)}\mathbf{x}_j || \mathbf{d}_j^e])}{|\mathcal{N}_i|}, \tag{4}$$

where ϕ_m denotes again an MLP and $|\mathcal{N}_i|$ is the number of neighbors of node i. We use here a simple average across all neighbors. An attention mechanism could also have been applied, but we empirically found that it only increased the complexity of the model, for no gain in performance.

3.3. Dual-Softmax for Points and Lines

Recent works [44, 57] show that dual-softmax approach obtains similar or better results than the usual Sinkhorn algorithm [53, 47], being also more efficient. We observed

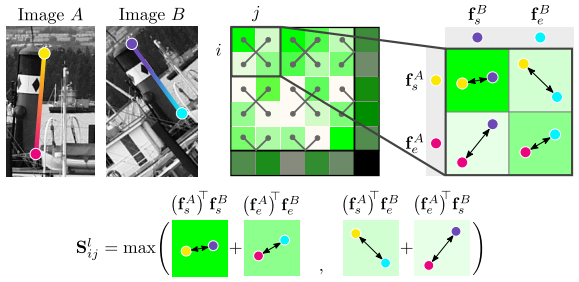


Figure 4: **Line matching with order-agnostic endpoints.** We consider the maximum score assignment between the two possible configurations of endpoint matching.

similar behaviour in our experiments and opted for the dual-softmax assignment. GlueStick provides both point and line matches in a single forward pass. We match nodes and lines separately through two independent dual-softmax assignments. On the one hand, all nodes (keypoints and line endpoints) are matched against each other using the final features output by the GNN: $\mathbf{f}_i^A \in \mathbb{R}^D$ for node i in image A and $\mathbf{f}_j^B \in \mathbb{R}^D$ for node j in image B. Each element of the assignment matrix $\mathbf{S}^p$ is formed by:

$$\mathbf{S}^p_{ij} = (\mathbf{f}_i^A)^\top \mathbf{f}_j^B. \tag{5}$$

We add a dustbin row and column at the end of $\mathbf{S}^p$, filled with a learnable parameter representing the threshold below which a node is considered unmatched, as [47] also does. We then apply softmax on all rows and all columns, and compute their geometric mean:

$$\mathbf{S}^p_{\text{final}} = \sqrt{\text{softmax}_{\text{row}}(\mathbf{S}^p) \odot \text{softmax}_{\text{col}}(\mathbf{S}^p)}. \tag{6}$$

Where $\odot$ means the element-wise product. Given this final assignation matrix, we keep the mutual nearest neighbors that have a matching score above a given threshold η.

On the other hand, lines are matched in a similar way, except that each line is represented by its two endpoints features $\mathbf{f}_s \in \mathbb{R}^D$ and $\mathbf{f}_e \in \mathbb{R}^D$. To make the matching agnostic of the endpoint ordering, we take the maximum of the two configurations in the line assignation matrix (see Fig. 4):

$$\begin{aligned}\mathbf{S}^l_{ij} = \max\big(&\left(\mathbf{f}_s^A\right)^\top \mathbf{f}_s^B + \left(\mathbf{f}_e^A\right)^\top \mathbf{f}_e^B, \\ &\left(\mathbf{f}_s^A\right)^\top \mathbf{f}_e^B + \left(\mathbf{f}_e^A\right)^\top \mathbf{f}_s^B\big).\end{aligned} \tag{7}$$

Finally, we get $\mathbf{S}^l_{\text{final}}$ by applying the dual-softmax of Eq. 6 and match lines with mutual nearest neighbors.

3.4. Ground Truth Generation

A challenging task in line matching is to generate high-quality labels handling line fragmentation, assignation, and partial visibility. To obtain the Ground Truth (GT) point matches $\mathcal{M}^p$, we use the same methodology as in [47]. In a nutshell, we leverage camera poses and depth to re-project keypoints from one image to another, and we add a new match whenever a re-projection falls within a small neighborhood of an existing keypoint.

For lines, we also leverage depth, but with a more complex setup. Let images A and B contain M and N line segments indexed by $\mathcal{A} := \{1, \dots, M\}$ and $\mathcal{B} := \{1, \dots, N\}$. We will denote the generated GT line matches $\mathcal{M}^l = \{(i, j)\} \subset \mathcal{A} \times \mathcal{B}$. For each segment $\mathbf{l}_i^A$ detected on image A, we sample K points $\left[\mathbf{x}_{i,1}^A, \dots, \mathbf{x}_{i,K}^A\right]$ along it. A point is considered invalid if it has either no depth or its projection $\mathbf{x}_i^B$ in the other image has no depth. A point is also considered non-valid if it is occluded. We detect these cases by comparing the depth $d(\mathbf{X}_i)$ of the unprojection $\mathbf{X}_i$ in 3D of point $\mathbf{x}_i^A$ with its expected depth d^B in image B:

$$\text{Occluded} = \frac{|d(\mathbf{X}_i) - d^B|}{d^B} > T_{\text{occlusion}}, \tag{8}$$

where $T_{\text{occlusion}}$ defines the tolerance threshold of depth variations. Segments with more than 50% of invalid points are labeled as `IGNORE` and will not affect the loss function.

Next, we generate a closeness matrix $\mathbf{C}^B \in \mathbb{N}^{M \times N}$ keeping track of how many sampled points of line i in A are reprojected close to a line j in B:

$$\mathbf{C}^B_{i,j} = \sum_{k=1}^{K} \mathbb{1}\left(\text{valid}(\mathbf{x}_{i,k}^B) \wedge d_\perp\left(\mathbf{x}_{i,k}^B, \mathbf{l}_j^B\right) < T_{\text{dist}}\right), \tag{9}$$

where $\mathbb{1}(\cdot)$ is the indicator function and $d_\perp(\cdot, \cdot)$ the perpendicular point-line distance. T_{dist} is a distance threshold in pixels that controls how demanding the GT is. $\mathbf{C}^A$ is defined analogously, and thus, we can define a cost matrix $\mathbf{C}$ with a minimum overlap threshold T_{overl}:

$$\mathbf{C}_{i,j} = \begin{cases} \infty, & \text{if } \mathbf{C}^A_{i,j} < T_{\text{overl}} \vee \mathbf{C}^B_{j,i} < T_{\text{overl}} \\ -\mathbf{C}^A_{i,j}\mathbf{C}^B_{j,i}, & \text{otherwise.} \end{cases} \tag{10}$$

Last, we solve the assignation problem defined by $\mathbf{C}$ with the Hungarian algorithm [26]. The resulting assignations $(i, j) \in \mathcal{M}^l$ are the `MATCHED` features, whereas all the valid entries $\mathcal{I} \subseteq \mathcal{A}$ and $\mathcal{J} \subseteq \mathcal{B}$ that were not assigned are labeled as `UNMATCHED`.

3.5. Loss Function

A classical approach for descriptor learning is to apply the triplet-ranking-loss [6, 54] with hard negative mining [38]. However, repetitive structures are often present along lines, which may produce detrimental hard negatives. We resort instead to minimizing the negative log-likelihood of point and line assignments $\mathbf{S}^p_{\text{final}}$ and $\mathbf{S}^l_{\text{final}}$:

$$\mathcal{L} = \frac{\text{NLL}(\mathbf{S}^p_{\text{final}}, \mathcal{M}^p) + \text{NLL}(\mathbf{S}^l_{\text{final}}, \mathcal{M}^l)}{2}, \tag{11}$$

where for an assignment matrix $\mathbf{A}$ and GT matches $\mathcal{M}$:

$$\text{NLL}(\mathbf{A}, \mathcal{M}) = -\sum_{(i,j)\in\mathcal{M}} \log \mathbf{A}_{i,j} - \sum_{i\in\mathcal{I}} \log \mathbf{A}_{i,N+1} - \sum_{j\in\mathcal{J}} \log \mathbf{A}_{M+1,j}. \quad (12)$$

4. Experiments

We pre-train our model on pairs of images synthetically warped by a homography, using the one million distractor images of [42], increasing the difficulty of the homographies gradually and speeding up convergence. We then fine-tune the model on MegaDepth [33] that contains 195 scenes of outdoor landmarks. We select image pairs with a minimum overlap of 10% of 3D points and resize each to 640×640 px. The wireframe threshold d to merge nodes is set to 3 pixels, and to generate the GT: $T_{\text{occlusion}} = 0.1$, $T_{\text{dist}} = 5$, and $T_{\text{overl}} = 0.2$. Our GNN contains 9 blocks of [self-attention, line message passing, cross-attention], and the matching threshold is set to $\eta = 0.2$. Features inside the network have size $D = 256$. We optimize our network using Adam with learning rate 10^{-4} for the homography pre-training and 10^{-5} for MegaDepth. To limit computational cost during training, we set a maximum number of 1000 keypoints and 250 line segments per image. Training takes 10 days on 2 NVIDIA RTX2080 GPUs.

4.1. Baselines

In the following, we compare GlueStick with several state-of-the-art line matchers: the handcrafted Line Band Descriptor (LBD)* [75], the self-supervised SOLD^2 [40], the transformer-based LineTR [73], and the learned L2D2 [1] descriptors. SOLD^2 uses its own detector since it is integrated with the descriptor. For all the other methods we use LSD [19]. We also compare to PL-Loc [73], the point-line matcher combining SuperPoint [14] and LineTR [73]. Whenever possible, we also compare to two additional point-based matchers: ClusterGNN [51]† and LoFTR [57].

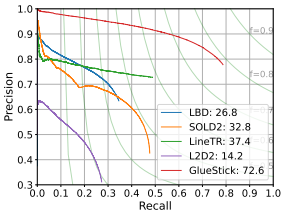
4.2. Ablation Study

Line segments are especially challenging to match in 3D due to occlusions, background changes, or partial visibility. We advocate for a proper evaluation of line matching covering these scenarios. Our ablation study is thus led on the ETH3D [50] dataset, an indoor-outdoor dataset of multiple scenes with GT LiDAR depth, and poses. We use the 13 scenes of the training set of the high-resolution multi-view images (downsampled by a factor of 8) , and sample all pairs of images with at least 500 GT keypoints in common, similarly as in [40]. We apply the same methodology as in Sec. 3.4 to compute the GT line matches. Given this GT, we can compute the precision-recall curve of the line matching by ordering lines by decreasing matching score.

*We use the authors' code instead of the binary version from OpenCV.
†We reuse the numbers of the paper as the code is not publicly available.

	P	L	W	LMP	AP (↑)
SG + Endpts	✓	✓			54.5
SG + W	✓	✓	✓		67.6
SG + LMP	✓	✓		✓	69.9
GlueStick-L		✓	✓	✓	64.0
GlueStick	✓	✓	✓	✓	**72.6**

(a) Ablation study



(b) Comparison to SOTA

Figure 5: **Ablation study and comparison to the state of the art (SOTA) on the ETH3D dataset [50].** We compute the line matching precision-recall curves and average precision (AP), displayed in the legend. (a) We compare several variations of our method using points (P), lines (L), wireframe connectivity (W), and Line Message Passing (LMP). (b) GlueStick surpasses all SOTA line matchers.

We compare several variations of our method in Fig. 5a. *SG + Endpts* refers to the pre-trained outdoor model of SuperGlue [47] to match the line endpoints, and use our proposed line association of Sec. 3.3 agnostic to the ordering of endpoints. *SG + W* is similar, but with our proposed wireframe preprocessing connecting line segments together. *SG + LMP* represents a SuperGlue backbone with the addition of our Line Message Passing (LMP), but no wireframe preprocessing. Finally, *GlueStick-L* is our proposed model without keypoints and matching lines only. The average precision (AP) shows that both the wireframe preprocessing and LMP bring a large boost of performance on the SuperGlue baseline. Their combination - our proposed model GlueStick - obtains the highest performance. *GlueStick-L* loses performance, but remains competitive, showing that the line matching is not relying only on points.

4.3. Line Matching Evaluation on ETH3D

We compare our method with previous state-of-the-art line matchers on the ETH3D dataset [50], and show the blatant superiority of GlueStick in Fig. 5b. It recovers almost 80% of the GT line matches, whereas the best previous methods do not manage to reach 50% of recall. At equivalent recalls, it outperforms the previous best method, LineTR, by more than 15% in precision, and is almost doubling the AP. This major improvement is due to the possibility of leveraging points in the matching, and to the rich signal provided by the wireframe structure. Note that GlueStick without points (in Fig. 5a), is still significantly better than other pure line matchers. It obtains good results thanks to the inclusion of a graph matching strategy combining appearance similarities and geometric consistencies. Despite having powerful descriptors, L2D2 and SOLD^2 ob-

tain worse results, because they neither use scene points nor geometric consistency between matches.

In terms of run time, GlueStick is also competitive. It runs in 258 ms on average on the images of ETH3D (around 775×515 pixels), which is similar to the execution time of SuperGlue of 235 ms. Other line matchers are even slower, with 419 ms for SOLD^2 and 304 ms for LineTR.

4.4. Homography Estimation

We evaluate our method on the task of homography estimation. While HPatches [5] is the most popular dataset, it is now very saturated [47, 57], and contains few structural lines that would be necessary to properly estimate a homography. Thus, lines do not help much to improve the current performance obtained by point methods. Nevertheless, GlueStick ranks first among all considered methods on HPatches. We show these results in the supplementary material. To circumvent this, we implement two meaningful experiments evaluating the homography estimation task in real-world scenarios: relative pose from planar surfaces (Sec. 4.4.1), and relative pose with pure rotations (Sec. 4.4.2).

4.4.1 Dominant Plane on ScanNet

ScanNet [13] is a large-scale RGB-D indoor dataset with GT camera poses, which pictures some hard cases for feature points with low texture, and where lines are expected to provide better constraints. We use the same test set of 1500 images as in [47], where the overlap between image pairs is computed from GT poses and depth. For each image pair, we match them with different state-of-the-art point, line, and point-line matchers. We then use a hybrid RANSAC [48, 10] to estimate a homography from these feature correspondences. This is a common way to initialize SLAM systems [39]. Since the GT homography is not known, we rely on the GT relative pose to evaluate the quality of the retrieved homography, as was done in previous works [7]. The relative pose corresponding to the predicted homography can be extracted using [36]. We report the pose error, computed as the maximum of the angular error in translation and rotation [72, 8, 47], as well as the corresponding pose AUC at error thresholds 10 / 20 / 30 degrees error. Note that this evaluation is valid regardless of the plane selected by each method to estimate the homography: all planes lead to the same relative pose.

The results are shown in Tab. 1. It can be seen first that GlueStick matching points only obtains better results than SuperGlue. This shows that our re-trained network is able to match and even outperform SuperGlue network for keypoint matching. Secondly, when matching lines only, GlueStick significantly exceeds the previous state of the art for line matching. This demonstrates that leveraging context from neighboring lines and being aware of their interconnection is highly beneficial. Finally, we obtain the best results overall when combining points and lines. The network can leverage both kinds of features and may rely more on the accurate points on well-textured images, while using lines in scenarios with scarce points.

		Pose error (↓)	Pose AUC (↑)
Points	SuperGlue (SG) [47]	18.1	15.6 / 29.8 / 39.4
	LoFTR [57]	16.8	15.8 / 30.9 / 41.4
	GlueStick	**15.7**	**17.4 / 32.8 / 42.9**
Lines	LBD [75]	49.2	3.7 / 8.2 / 13.4
	SOLD^2 [40]	55.6	4.9 / 10.8 / 16.1
	LineTR [73]	51.6	4.5 / 11.0 / 16.8
	L2D2 [1]	60.0	2.8 / 6.5 / 10.5
	SG + Endpts (no KP)	36.0	7.1 / 15.0 / 22.2
	GlueStick	**27.6**	**9.4 / 20.0 / 28.6**
Points + Lines	PL-Loc [73]	26.2	12.2 / 24.1 / 32.2
	SG + Endpts	17.1	17.5 / 31.8 / 41.2
	GlueStick	**14.1**	**19.3 / 35.4 / 46.0**

Table 1: **Homography estimation on ScanNet [13].** We first estimate a homography based on point-only, line-only or points+lines matches, then decompose it into the corresponding relative pose. We report the median pose error in degrees, as well as the AUC at 10° / 20° / 30° error.

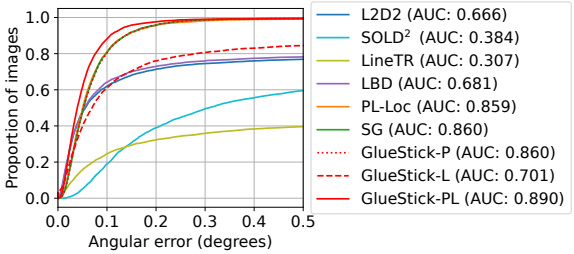


Figure 6: **Camera rotation estimation in SUN360 [67].** We show the cumulative angular error for all pairs of images. We report the AUC up to an error threshold of 0.5°.

4.4.2 Pure Rotations on SUN360

We also evaluate our method in estimating pure camera rotations, which are the cornerstone of some applications such as image stitching, or visual-guided sensor fusion. We use the SUN360 [67] dataset containing 360º images. From each original 360º image, we crop 10 pairs of perspective images (640×480 pixels) with a field-of-view of 80º. Pairs are randomly sampled with an angular difference in range $\pm\,[50°, 70°]$ for yaw and $\pm\,[0°, 30°]$ for pitch. We first extract the local feature matches, then we estimate a rotation using Hybrid RANSAC [48, 10]. We evaluate the angular error between the predicted and GT relative rotations.

In Fig. 6, GlueStick-PL (with points and lines) obtains the best results because lines help to match pairs where there is not enough texture. Specifically, long lines can be

detected very precisely, thus contributing to an accurate estimation. Point-based methods (SG and GlueStick-P) obtain already 3 points less of AUC. PL-Loc [73] is ranked fourth because it effectively uses points and lines, though independently and without spatial reasoning for point matches.

4.5. Visual Localization

We introduce here the downstream task of localizing a query image, given the known poses of database images. We follow the *hloc* pipeline [46, 45], and integrate line features and our own matcher in the existing code. We use NetVLAD [3] for image retrieval, detect SuperPoint [14] feature points and LSD [19] lines, and match these features with either SuperGlue [47] + a line matcher, or with GlueStick. We use the GT depth to back-project lines in 3D: points are sampled along each line, un-projected to 3D, and a 3D line is fit to these un-projected points. We use the solvers of [27, 78, 30] to generate poses from a minimal set of 3 features (3 points, 2 points and 1 line, 1 point and 2 lines, or 3 lines), then combine them in a hybrid RANSAC [48, 10] to recover the query camera poses.

Datasets. We compare our method to other baselines on two datasets. The 7Scenes dataset [52] is a famous RGB-D dataset for visual localization, displaying 7 indoor scenes with GT poses and depth. It is however limited in scale, and most scenes are already saturated for point-based localization. One scene remains extremely challenging for feature points: the Stairs scene, as illustrated in Fig. 7. Due to the lack of texture and repeated steps of the stairs, current point-based methods are still struggling on this scene [9]. We report median translation and rotation error, as well as the percentage of successfully recovered poses under a 5 cm / 5° threshold. InLoc [66, 58] is a large-scale indoor dataset with GT poses and depth, with two test scenes: DUC1 and DUC2. It is challenging for point-based methods due to images with low texture and large viewpoint changes. We report the pose AUC at 0.25m / 0.5m / 1m and 10°.

Results. The results can be found in Tab. 2. GlueStick with points only is able to surpass SuperGlue, confirming the strong matching performance of isolated keypoints already. In particular, GlueStick obtains an improvement of 44% in pose accuracy over SuperGlue on Stairs. Adding line features significantly improves the performance for Stairs and brings a small improvement on InLoc as well. This demonstrates the importance of lines in texture-less areas and with repeated structures. Combining points and lines in a single network allows GlueStick to reason about neighboring features and can thus beat the other methods that are independently matching points and lines.

5. Conclusion

Matching points across two views and matching line segments are traditionally treated as two separate independent problems. In this work, we challenge this paradigm and present GlueStick, a learned matcher that jointly establishes correspondences between points and lines. By processing both types of features together, the network is able to propagate strong matches of either type to neighboring features that might have less discriminative appearance.

In our experiments, we show an improved matching performance across the board, for both points and lines. In particular for line matching, GlueStick provides a significant leap forward compared to the current descriptor-based state-of-the-art. The key insight in our work is that line segments do not appear randomly scattered in the image, but rather form connected structures. This connectivity is explicitly encoded and exploited in our network architecture. Finally, we show that the improved matches we obtain directly translate to better results in downstream tasks such as homography and camera pose estimation.

		7Scenes [52]		InLoc [58]	
		T / R err.	Acc.	DUC 1	DUC 2
P	SuperGlue [47]	4.7 / 1.25	53.4	48.5 / 68.2 / 80.3	53.4 / 75.6 / 82.4
	ClusterGNN [51]	-	-	47.5 / 69.7 / 79.8	53.4 / 77.1 / 84.7
	LoFTR [57]	**4.4 / 0.95**	53.9	47.5 / **72.2 / 84.8**	54.2 / 74.8 / 85.5
	GlueStick	**4.4** / 1.21	**55.4**	**49.0** / 70.2 / 84.3	**55.0 / 83.2 / 87.0**
P+L	SOLD^2 [40]	3.2 / 0.83	75.8	44.9 / 69.7 / 79.8	54.2 / 75.6 / 80.2
	LineTR [73]	3.7 / 1.02	66.6	46.0 / 67.2 / 76.3	53.4 / 77.1 / 80.9
	L2D2 [1]	4.1 / 1.15	55.8	46.5 / 68.7 / 80.3	51.9 / 75.6 / 79.4
	SG + Endpts	3.1 / 0.81	75.6	45.5 / 71.2 / 81.8	45.5 / 71.2 / 81.8
	GlueStick	**2.9 / 0.79**	**79.7**	**47.5 / 73.7 / 85.9**	**57.3 / 83.2 / 87.0**

Table 2: **Visual localization on 7Scenes [52] and InLoc [58].** We report the median translation (cm), rotation error (deg), and pose accuracy at a 5 cm / 5° threshold for the scene Stairs of 7Scenes, and the pose AUC at 0.25m / 0.5m / 1m and 10° error for InLoc. GlueStick ranks first both for points-only (P) and point-line (P+L) results.

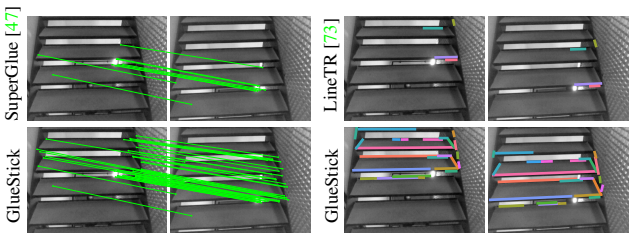


Figure 7: **Correct matches on 7Scenes Stairs [52].** Lines can guide the point matching in very challenging scenarios with low texture and repeated patterns.

Acknowledgement

We would like to thank P.E. Sarlin and P. Lindenberger for their great support, as well as L. Cavalli, J.M. Buenaposada, and L. Baumela for helping to review this paper. V. Larsson was supported by the strategic research project ELLIIT.

References

[1] Hichem Abdellali, Robert Frohlich, Viktor Vilagos, and Zoltan Kato. L2D2: learnable line detector and descriptor. In *IEEE International Conference on 3D Vision (3DV)*, 2021. 2, 6, 7, 8

[2] Cuneyt Akinlar and Cihan Topal. Edlines: A real-time line segment detector with a false detection control. *Pattern Recognition Letters*, 32(13):1633–1642, 2011. 2

[3] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2016. 8

[4] Khashayar Asadi, Hariharan Ramshankar, Mojtaba Noghabaei, and Kevin Han. Real-time image localization and registration with bim using perspective alignment for indoor monitoring of construction. *Journal of Computing in civil Engineering*, 33(5):04019031, 2019. 1

[5] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. HPatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2017. 7

[6] Vassileios Balntas, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *Brit. Mach. Vis. Conf. (BMVC)*, 2016. 5

[7] Dániel Baráth, Dmytro Mishkin, Michal Polic, Wolfgang Förstner, and Jiri Matas. A large scale homography benchmark. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2023. 7

[8] Eric Brachmann and Carsten Rother. Neural-guided RANSAC: Learning where to sample model hypotheses. In *Int. Conf. Comput. Vis. (ICCV)*, 2019. 7

[9] Eric Brachmann and Carsten Rother. Visual camera relocalization from RGB and RGB-D images using DSAC. *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)*, 44(9):5847–5865, 2021. 8

[10] Federico Camposeco, Andrea Cohen, Marc Pollefeys, and Torsten Sattler. Hybrid camera pose estimation. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018. 7, 8

[11] Hongkai Chen, Zixin Luo, Jiahui Zhang, Lei Zhou, Xuyang Bai, Zeyu Hu, Chiew-Lan Tai, and Long Quan. Learning to match features with seeded graph matching network. In *Int. Conf. Comput. Vis. (ICCV)*, 2021. 2

[12] Hongkai Chen, Zixin Luo, Lei Zhou, Yurun Tian, Mingmin Zhen, Tian Fang, David N. R. McKinnon, Yanghai Tsin, and Long Quan. Aspanformer: Detector-free image matching with adaptive span transformer. In *Eur. Conf. Comput. Vis. (ECCV)*, 2022. 2

[13] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2017. 7

[14] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018. 2, 3, 6, 8

[15] Johan Edstedt, Ioannis Athanasiadis, Mårten Wadenbäck, and Michael Felsberg. DKM: Dense kernelized feature matching for geometry estimation. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2023. 2

[16] Bin Fan, Fuchao Wu, and Zhanyi Hu. Robust line matching through line–point invariants. *Pattern Recognition*, 45(2):794–805, 2012. 2

[17] Leandro AF Fernandes and Manuel M Oliveira. Real-time line detection through an improved hough transform voting scheme. *Pattern recognition*, 41(1):299–314, 2008. 2

[18] Ruben Gomez-Ojeda, Francisco-Angel Moreno, David Zuñiga-Noël, Davide Scaramuzza, and Javier Gonzalez-Jimenez. PL-SLAM: A stereo SLAM system through the combination of points and line segments. *IEEE Transactions on Robotics*, 35(3):734–746, 2019. 1

[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 Trans. Pattern Anal. Mach. Intell. (PAMI)*, 32(4):722–732, 2010. 2, 3, 6, 8

[20] Zirui Guo, Huimin Lu, Qinghua Yu, Ruibin Guo, Junhao Xiao, and Hongshan Yu. HDPL: a hybrid descriptor for points and lines based on graph neural networks. *Industrial Robot: the international journal of robotics research and application*, 2021. 2

[21] Manuel Hofer, Michael Maurer, and Horst Bischof. Efficient 3D scene abstraction using line segments. *Computer Vision and Image Understanding (CVIU)*, 157:167–178, 2017. 1

[22] Paul VC Hough. Method and means for recognizing complex patterns, Dec. 18 1962. US Patent 3,069,654. 2

[23] Kun Huang, Yifan Wang, Zihan Zhou, Tianjiao Ding, Shenghua Gao, and Yi Ma. Learning to parse wireframes in images of man-made environments. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018. 2

[24] Wei Jiang, Eduard Trulls, Jan Hosang, Andrea Tagliasacchi, and Kwang Moo Yi. COTR: Correspondence Transformer for Matching Across Images. In *Int. Conf. Comput. Vis. (ICCV)*, 2021. 2

[25] Christopher Kropp, Christian Koch, and Markus König. Interior construction state recognition with 4D BIM registered image sequences. *Automation in Construction*, 86, 2018. 1

[26] Harold W Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 5

[27] Zuzana Kukelova, Jan Heller, and Andrew Fitzgibbon. Efficient intersection of three quadrics and applications in computer vision. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2016. 8

[28] Manuel Lange, Claudio Raisch, and Andreas Schilling. WLD: A Wavelet and Learning based Line Descriptor for Line Feature Matching. In Jens Krüger, Matthias Niessner, and Jörg Stückler, editors, *Vision, Modeling, and Visualization*. The Eurographics Association, 2020. 2

[29] Manuel Lange, Fabian Schweinfurth, and Andreas Schilling. DLD: A deep learning based line descriptor for line feature matching. In *International Conference on Intelligent Robots and Systems (IROS)*, 2019. 2

[30] Viktor Larsson. PoseLib - Minimal Solvers for Camera Pose Estimation. https://github.com/vlarsson/PoseLib, 2020. 8

[31] Kai Li, Jian Yao, Mengsheng Lu, Yuan Heng, Teng Wu, and Yinxuan Li. Line segment matching: a benchmark. In *Winter Conference on Applications of Computer Vision (WACV)*, 2016. 2

[32] Kai Li, Jian Yao, Xiaohu Lu, Li Li, and Zhichao Zhang. Hierarchical line matching based on line–junction–line structure descriptor and local homography estimation. *Neurocomputing*, 184:207–220, 2016. 2, 3

[33] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018. 6

[34] David G Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis. (IJCV)*, 60(2):91–110, 2004. 1

[35] Quanmeng Ma, Guang Jiang, Jiajie Wu, Changshuai Cai, Dianzhi Lai, Zixuan Bai, and Hao Chen. WGLSM: An end-to-end line matching network based on graph convolution. *Neurocomputing*, 453:195–208, 2021. 2

[36] Ezio Malis and Manuel Vargas. Deeper understanding of the homography decomposition for vision-based control. Research Report RR-6303, INRIA, 2007. 7

[37] Jiri Matas, Charles Galambos, and Josef Kittler. Robust detection of lines using the progressive probabilistic hough transform. *Computer Vision and Image Understanding (CVIU)*, 78(1):119–137, 2000. 2

[38] Anastasiia Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. In *Adv. Neural Inform. Process. Syst. (NeurIPS)*, volume 30, 2017. 5

[39] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. 7

[40] Rémi Pautrat, Juan-Ting Lin, Viktor Larsson, Martin R. Oswald, and Marc Pollefeys. SOLD^2: Self-supervised occlusion-aware line description and detection. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2021. 2, 6, 7, 8

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

[42] F. Radenović, A. Iscen, G. Tolias, Y. Avrithis, and O. Chum. Revisiting oxford and paris: Large-scale image retrieval benchmarking. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018. 6

[43] Srikumar Ramalingam, Michel Antunes, Dan Snow, Gim Hee Lee, and Sudeep Pillai. Line-sweep: Cross-ratio for wide-baseline matching and 3D reconstruction. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2015. 2

[44] I. Rocco, M. Cimpoi, R. Arandjelović, A. Torii, T. Pajdla, and J. Sivic. Neighbourhood consensus networks. In *Adv. Neural Inform. Process. Syst. (NeurIPS)*, 2018. 3, 4

[45] Paul-Edouard Sarlin. Visual localization made easy with hloc. https://github.com/cvg/Hierarchical-Localization. 8

[46] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2019. 8

[47] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2020. 1, 2, 4, 5, 6, 7, 8

[48] Torsten Sattler et al. RansacLib - A Template-based *SAC Implementation. https://github.com/tsattler/RansacLib, 2019. 7, 8

[49] Cordelia Schmid and Andrew Zisserman. Automatic line matching across views. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 666–671, 1997. 2

[50] Thomas Schops, Johannes L. Schonberger, 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 *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2017. 6

[51] Yan Shi, Jun-Xiong Cai, Yoli Shavit, Tai-Jiang Mu, Wensen Feng, and Kai Zhang. ClusterGNN: Cluster-based coarse-to-fine graph neural network for efficient feature matching. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2022. 2, 6, 8

[52] 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 *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2013. 8

[53] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967. 4

[54] Iago Suárez, José M Buenaposada, and Luis Baumela. Revisiting binary local image description for resource limited devices. *IEEE Robotics and Automation Letters (RAL)*, 6(4):8317–8324, 2021. 5

[55] Iago Suárez, José M Buenaposada, and Luis Baumela. ELSED: Enhanced line segment drawing. *Pattern Recognition*, 127:108619, 2022. 2

[56] Iago Suárez, Enrique Muñoz, José M Buenaposada, and Luis Baumela. FSG: A statistical approach to line detection via fast segments grouping. In *International Conference on Intelligent Robots and Systems (IROS)*, 2018. 2

[57] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2021. 1, 2, 3, 4, 6, 7, 8

[58] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. Inloc: Indoor visual localization with dense matching and view synthesis. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018. 8

[59] Prune Truong, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning accurate dense correspondences and when

to trust them. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2021. 2

[60] Alexander Vakhitov and Victor Lempitsky. Learnable line segment descriptor for visual slam. *IEEE Access*, 7:39923–39934, 2019. 2

[61] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Adv. Neural Inform. Process. Syst. (NeurIPS)*, 2017. 4

[62] Bart Verhagen, Radu Timofte, and Luc Van Gool. Scale-invariant line descriptors for wide baseline matching. In *Winter Conference on Applications of Computer Vision (WACV)*, 2014. 2

[63] Lu Wang, Ulrich Neumann, and Suya You. Wide-baseline image matching using line signatures. In *Int. Conf. Comput. Vis. (ICCV)*. IEEE, 2009. 2

[64] Qing Wang, Jiaming Zhang, Kailun Yang, Kunyu Peng, and Rainer Stiefelhagen. Matchformer: Interleaving attention in transformers for feature matching. In *Asian Conf. on Comput. Vision (ACCV)*, 2022. 2

[65] Zhiheng Wang, Fuchao Wu, and Zhanyi Hu. MSLD: A robust descriptor for line matching. *Pattern Recognition*, 42(5):941–953, 2009. 2

[66] Erik Wijmans and Yasutaka Furukawa. Exploiting 2D floorplan for building-scale panorama rgbd alignment. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2017. 8

[67] Jianxiong Xiao, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Recognizing scene viewpoint using panoramic place representation. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2012. 7

[68] Chi Xu, Lilian Zhang, Li Cheng, and Reinhard Koch. Pose estimation from line correspondences: A complete analysis and a series of solutions. *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)*, 39(6), 2017. 1

[69] Yifan Xu, Weijian Xu, David Cheung, and Zhuowen Tu. Line segment detection using transformers without edges. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 4257–4266, 2021. 2

[70] Nan Xue, Song Bai, Fudong Wang, Gui-Song Xia, Tianfu Wu, and Liangpei Zhang. Learning attraction field representation for robust line segment detection. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2019. 2

[71] Nan Xue, Tianfu Wu, Song Bai, Fudong Wang, Gui-Song Xia, Liangpei Zhang, and Philip H.S. Torr. Holistically-attracted wireframe parsing. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2020. 2

[72] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018. 7

[73] Sungho Yoon and Ayoung Kim. Line as a visual sentence: Context-aware line descriptor for visual localization. *IEEE Robotics and Automation Letters (RAL)*, 6(4):8726–8733, 2021. 1, 2, 6, 7, 8

[74] Huayi Zeng, Kevin Joseph, Adam Vest, and Yasutaka Furukawa. Bundle pooling for polygonal architecture segmentation problem. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2020. 1

[75] 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):794–805, 2013. 2, 6, 7

[76] Ziheng Zhang, Zhengxin Li, Ning Bi, Jia Zheng, Jinlei Wang, Kun Huang, Weixin Luo, Yanyu Xu, and Shenghua Gao. PPGNet: Learning point-pair graph for line segment detection. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2019. 2

[77] Kai Zhao, Qi Han, Chang-Bin Zhang, Jun Xu, and Ming-Ming Cheng. Deep hough transform for semantic line detection. *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)*, 2021. 2

[78] Lipu Zhou, Jiamin Ye, and Michael Kaess. A stable algebraic camera pose estimation for minimal configurations of 2D/3D point and line correspondences. In *Asian Conf. on Comput. Vision (ACCV)*, 2018. 8

[79] Yichao Zhou, Haozhi Qi, and Yi Ma. End-to-end wireframe parsing. In *Int. Conf. Comput. Vis. (ICCV)*, 2019. 2

[80] Yichao Zhou, Haozhi Qi, Yuexiang Zhai, Qi Sun, Zhili Chen, Li-Yi Wei, and Yi Ma. Learning to reconstruct 3D manhattan wireframes from a single image. In *Int. Conf. Comput. Vis. (ICCV)*, 2019. 1

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