

# GlueStick: Robust Image Matching by Sticking Points and Lines Together

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

## Abstract

Line segments are powerful features complementary to points. They offer structural cues, robust to drastic view-point 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 [20, 88, 48], pose estimation [75], construction monitoring [28, 4], and 3D reconstruction [23, 81, 87]. 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 view-point changes. Furthermore, lines often appear as part of repetitive structures in human-made environments, making

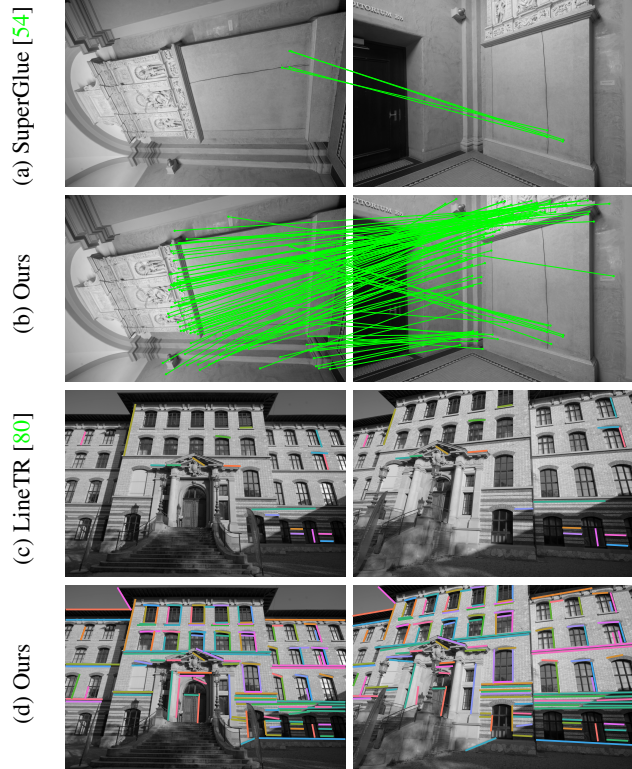


Figure 1: **Joint matching of points and lines.** Matching feature points often fails in textureless areas (a), while current line matching methods struggle with large view-point changes (c). We propose *GlueStick*, a network jointly matching points and lines. While none of the methods were trained on the rotations of (a)(b), line matches can guide *GlueStick* while SuperGlue [54] fails, and vice-versa in (d) where points can complement the line matching. For clarity reasons, we show here only the correct matches.

classical descriptor-based matching fail. For this reason, typical matching heuristics such as mutual nearest neighbor and Lowe’s ratio test [38] are often less effective for lines.

Recently, deep learning has ushered in a new paradigm for feature point matching using Graph Neural Networks (GNNs) [54, 64]. This new approach bypasses the need

\* Authors contributed equally.

for matching heuristics or even outlier removal techniques, thanks to the high precision of the predicted matches [54]. 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 [54, 64, 66, 26]. 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 [80], 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 [54] 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 [82, 56, 35] 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 [24] and its improvements [41, 19, 84]. Local line segment detectors [21, 2, 63, 62, 46] 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 [25, 86, 77, 83, 78, 76]. In this work, we train our network with the LSD detector [21], 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 [70, 72, 82, 69, 35]. 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 [32, 31, 1]. An alternative approach is to sample points along the line and to describe them separately [67, 47]. SOLD<sup>2</sup> [47] 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 [16] 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 [34]: global rotation between images [82]; properties of pairs of matched lines like the angle between segments, intersection ratios or projection ratios [82, 70]; line-point invariants [18]; cross-ratio [50] or consistency with a fundamental matrix estimated from points [56, 35]. 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 [54].

**Matching with transformers.** SuperGlue [54] 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 [12, 58] and with dense predictions [64, 66, 26, 71, 13, 17].

WGLSM [39] 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 assignment is solved using a single Sinkhorn matrix. LineTR [80] proposes to use attention inside points sampled for each line to deal with the line scale changes and occlusions. HDPL [22] 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

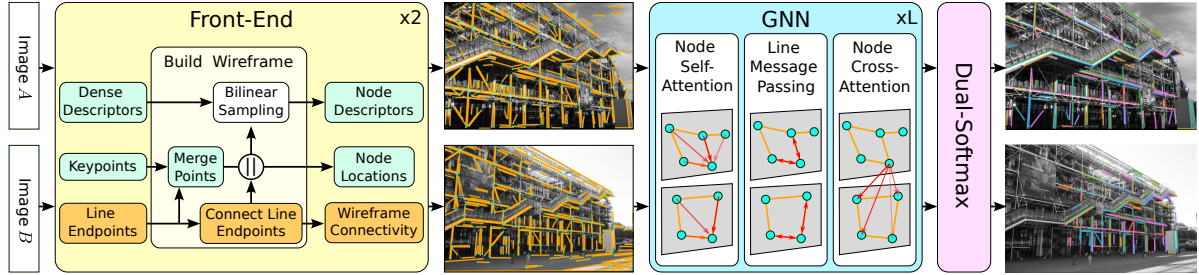


Figure 2: **Overview of GlueStick.** Keypoints, dense descriptors, and lines are extracted from two images, and unified into two wireframes (front-end). We then enrich the features of the nodes of both wireframes via self, line, and cross-attention inside a Graph Neural Network (GNN). Finally, points 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 [35]. 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 assignment is solved separately for points and lines, using two independent dual-softmax modules [51, 64], 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) [16] to predict keypoints and a dense descriptor map, and we detect segments with the general-purpose LSD [21] 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 sub-pixel 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,

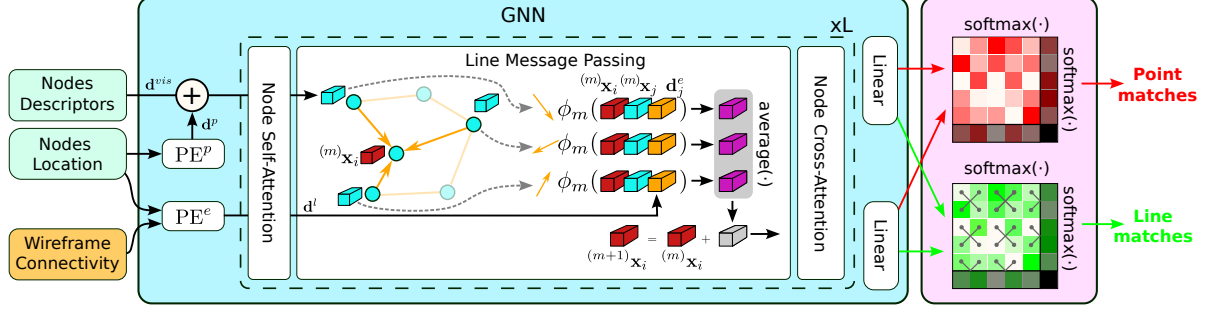


Figure 3: **Graph Neural Network (GNN) architecture.** Node features of the wireframe are enriched via several communication layers. Our Line Message Passing exchanges information between neighboring nodes that are connected together.

we learn two positional encoders ( $PE^p$  and  $PE^e$ ) with Multi-layer Perceptron (MLP) that generate a *spatial* descriptor  $d^p$  for each node and an *edge*-descriptor  $d^e$  for each line segment originating from this node. A node with connectivity 3 will for instance get assigned one  $d^p$  and 3  $d^e$  (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:

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

The spatial-descriptor  $d^p$  is used to initialize the node features, while the edge-descriptors  $d^e$  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  $\mathcal{E}_{self}$ , connect nodes of one image with all the nodes of the same image. Line edges  $\mathcal{E}_{line}$ , connect nodes that are endpoints of the same line. Cross attention edges  $\mathcal{E}_{cross}$ , 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.**  $\mathcal{E}_{self}$  and  $\mathcal{E}_{cross}$  edges are similarly defined as in [54]. The  $m$ -th feature update is defined by a residual message passing:

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

where  $\parallel$  denotes concatenation, the function  $\psi_m$  is modeled with an MLP, and  $a_m((m)\mathbf{x}_i; \mathcal{E})$  is the Multi-Head Attention

mechanism from [68] applied to the set of edges  $\mathcal{E}$ :

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

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  $\mathcal{E}_{line}$ . 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  $\mathbf{d}_j^e$ :

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

where  $\phi_m$  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 [51, 64] show that dual-softmax approach obtains similar or better results than the usual Sinkhorn algorithm [60, 54], being also more efficient. We observed

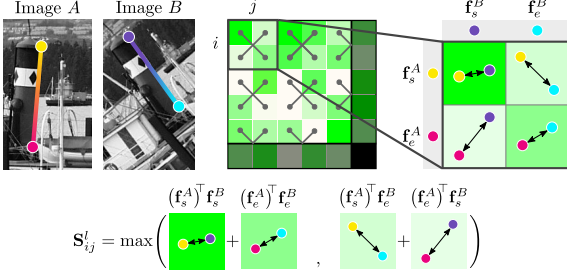


Figure 4: **Line matching with order-agnostic endpoints.** We consider the maximum score assignment between the two possible configurations of endpoint matching.

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:

$$\mathbf{S}_{ij}^p = (\mathbf{f}_i^A)^\top \mathbf{f}_j^B. \quad (5)$$

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 [54] also does. We then apply softmax on all rows and all columns, and compute their geometric mean:

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

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

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 assignment matrix (see Fig. 4):

$$\mathbf{S}_{ij}^l = \max\left( (\mathbf{f}_s^A)^\top \mathbf{f}_e^B + (\mathbf{f}_e^A)^\top \mathbf{f}_s^B, \right. \\ \left. (\mathbf{f}_s^A)^\top \mathbf{f}_s^B + (\mathbf{f}_e^A)^\top \mathbf{f}_e^B \right). \quad (7)$$

Finally, we get  $\mathbf{S}_{\text{final}}^l$  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, assignment, and partial visibility. To obtain the Ground Truth (GT) point matches  $\mathcal{M}^p$ , we use the same methodology as in [54]. 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  $[\mathbf{x}_{i,1}^A, \dots, \mathbf{x}_{i,K}^A]$  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  $B$ :

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

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  $B$ :

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

where  $\mathbb{1}(\cdot)$  is the indicator function and  $d_{\perp}(\cdot, \cdot)$  the perpendicular point-line distance.  $T_{\text{dist}}$  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  $T_{\text{overl}}$ :

$$\mathbf{C}_{i,j} = \begin{cases} \infty, & \text{if } \mathbf{C}_{i,j}^A < T_{\text{overl}} \vee \mathbf{C}_{j,i}^B < T_{\text{overl}} \\ -\mathbf{C}_{i,j}^A \mathbf{C}_{j,i}^B, & \text{otherwise.} \end{cases} \quad (10)$$

Last, we solve the assignment problem defined by  $\mathbf{C}$  with the Hungarian algorithm [29]. The resulting assignments  $(i, j) \in \mathcal{M}^l$  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, 61] with hard negative mining [43]. 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  $\mathbf{S}_{\text{final}}^p$  and  $\mathbf{S}_{\text{final}}^l$ :

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

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

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

## 4. Experiments

We pre-train our model on pairs of images synthetically warped by a homography, using the one million distractor images of [49], increasing the difficulty of the homographies gradually and speeding up convergence. We then fine-tune the model on MegaDepth [36] that contains 195 scenes of outdoor landmarks. We select image pairs with a minimum overlap of 10% of 3D points and resize each to  $640 \times 640$  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

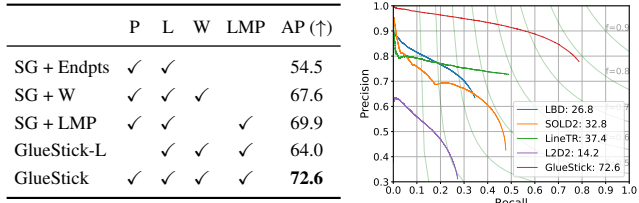
In the following, we compare GlueStick with several state-of-the-art line matchers: the handcrafted Line Band Descriptor (LBD)\* [82], the self-supervised SOLD<sup>2</sup> [47], the transformer-based LineTR [80], and the learned L2D2 [1] descriptors. SOLD<sup>2</sup> uses its own detector since it is integrated with the descriptor. For all the other methods we use LSD [21]. We also compare to PL-Loc [80], the point-line matcher combining SuperPoint [16] and LineTR [80]. Whenever possible, we also compare to two additional point-based matchers: ClusterGNN [58]<sup>†</sup> and LoFTR [64].

### 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 [57] 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 [47]. We apply the same methodology

\*We use the authors' code instead of the binary version from OpenCV.

<sup>†</sup>We reuse the numbers of the paper as the code is not publicly available.



(a) Ablation study

(b) Comparison to SOTA

Figure 5: **Ablation study and comparison to the state of the art (SOTA) on the ETH3D dataset [57].** 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.

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 compare several variations of our method in Fig. 5a. *SG + Endpts* refers to the pre-trained outdoor model of SuperGlue [54] 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 [57], 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<sup>2</sup> 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  $775 \times 515$  pixels), which is similar to the execution time of SuperGlue of 235 ms. Other line matchers are even slower, with 419 ms for SOLD<sup>2</sup> 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 [54, 64], 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 [14] 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 [54], 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 [55, 10] to estimate a homography from these feature correspondences. This is a common way to initialize SLAM systems [44]. 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 [40]. We report the pose error, computed as the maximum of the angular error in translation and rotation [79, 8, 54], 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 key-point matching. Secondly, when matching lines only, GlueStick significantly exceeds the previous state of the art for line matching. This demonstrates that leveraging context

		Pose error ( $\downarrow$ )	Pose AUC ( $\uparrow$ )
Points	SuperGlue (SG) [54]	18.1	15.6 / 29.8 / 39.4
	LoFTR [64]	16.8	15.8 / 30.9 / 41.4
	GlueStick	<b>15.7</b>	<b>17.4 / 32.8 / 42.9</b>
Lines	LBD [82]	49.2	3.7 / 8.2 / 13.4
	SOLD <sup>2</sup> [47]	55.6	4.9 / 10.8 / 16.1
	LineTR [80]	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	<b>27.6</b>	<b>9.4 / 20.0 / 28.6</b>
Points + Lines	PL-Loc [80]	26.2	12.2 / 24.1 / 32.2
	SG + Endpts	17.1	17.5 / 31.8 / 41.2
	GlueStick	<b>14.1</b>	<b>19.3 / 35.4 / 46.0</b>

Table 1: **Homography estimation on ScanNet [14].** 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.

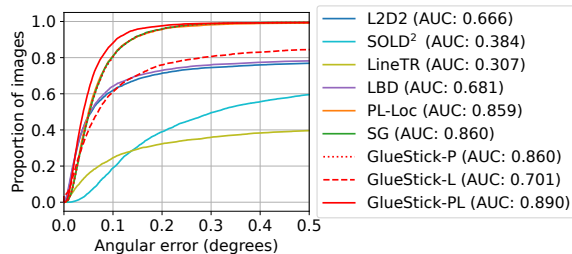


Figure 6: **Camera rotation estimation in SUN360 [74].** We show the cumulative angular error for all pairs of images. We report the AUC up to an error threshold of 0.5°.

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.

##### 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 [74] dataset containing 360° images. From each original 360° image, we crop 10 pairs of perspective images ( $640 \times 480$  pixels) with a field-of-view of 80°. Pairs are randomly sampled with an angular difference in range  $\pm [50^\circ, 70^\circ]$  for yaw and  $\pm [0^\circ, 30^\circ]$  for pitch. We first extract the local feature matches, then we estimate a rotation using Hybrid RANSAC [55, 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 [80] 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 pipeline of LIMAP [37], which integrates line features into *hloc* [53, 52]. We use NetVLAD [3] for image retrieval, detect SuperPoint [16] feature points and LSD [21] lines, and match these features with either SuperGlue [54] + 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 [30, 85, 33] 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 [55, 10] to recover the query camera poses.

**Datasets.** We compare our method to other baselines on two datasets. The 7Scenes dataset [59] 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 [73, 65] 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

		7Scenes [59]		InLoc [65]	
		T / R err.	Acc.	DUC 1	DUC 2
P	SuperGlue [54]	4.7 / 1.25	53.4	48.5 / 68.2 / 80.3	53.4 / 75.6 / 82.4
	ClusterGNN [58]	-	-	47.5 / 69.7 / 79.8	53.4 / 77.1 / 84.7
	LoFTR [64]	<b>4.4 / 0.95</b>	53.9	47.5 / <b>72.2 / 84.8</b>	54.2 / 74.8 / 85.5
	GlueStick	<b>4.4 / 1.21</b>	<b>55.4</b>	<b>49.0 / 70.2 / 84.3</b>	<b>55.0 / 83.2 / 87.0</b>
P+L	SOLD <sup>2</sup> [47]	3.2 / 0.83	75.8	44.9 / 69.7 / 79.8	54.2 / 75.6 / 80.2
	LineTR [80]	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	<b>2.9 / 0.79</b>	<b>79.7</b>	<b>47.5 / 73.7 / 85.9</b>	<b>57.3 / 83.2 / 87.0</b>

Table 2: **Visual localization on 7Scenes [59] and InLoc [65].** 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.

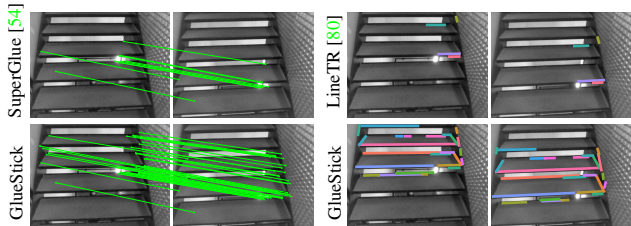


Figure 7: **Correct matches on 7Scenes Stairs [59].** Lines can guide the point matching in very challenging scenarios with low texture and repeated patterns.

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.

### 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.



## Supplementary Material

In the following, we provide additional details regarding GlueStick, our point-line matcher. Appendix A offers a visualization of the generation of our line ground truth. Appendix B gives additional insights and ablation studies motivating our choices. Appendix C specifies some experimental details to reproduce our experiments and brings additional results. Appendix D shows matching results, as well as failure cases of our method. Finally, Appendix E provides visualizations of the attention for various kinds of nodes.

### A. Ground Truth Generation

Designing a line matching ground truth (GT) is challenging, due to partial occlusions and lack of repeatability of line detectors. We provide here some visualizations of the GT generation process.

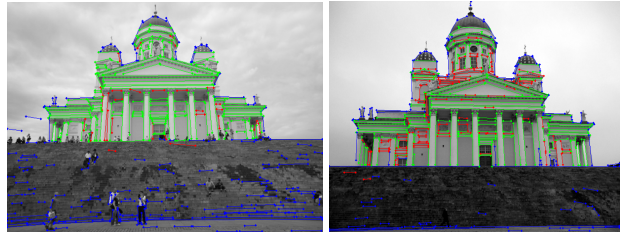
Fig. 8 shows an example of the ground truth between two images. Line segments can be either **MATCHED** (green), **UNMATCHED** (red), or **IGNORED** (blue). The latter case happens when the depth along the line is too uncertain or when its reprojection in the other image is occluded. The generation process is also illustrated in Fig. 9 for one pair of line segments. The advantage of the proposed method is that it recovers a large number of matches for each pair of images, providing a strong matching signal. In comparison to the method proposed in [1] which does robust 3D reconstruction of line segments, our method is faster and simpler. By avoiding the 3D line reconstruction step, we can train in larger scenes with potentially noisy depth, like the MegaDepth dataset [36].

### B. Additional Insights on GlueStick

In this section, we give extra insights and motivations for our design choices.

**Choice of the Line Segment Detector.** In all our training and experiments, we used the Line Segment Detector (LSD) [21] to extract line segments. For a certain application, such as indoor wireframe parsing [25], learned methods largely overtake classic ones [15, 62, 76, 78]. However, learned methods struggle to generalize this power to other contexts, tasks, or types of images. For this reason, we have chosen LSD as the generic method to train GlueStick. Furthermore, we believe that our line pre-processing, turning an unordered set of lines into a connected graph, is beneficial to make the endpoints more repeatable across views, thus potentially making LSD more repeatable.

We ran a small experiment to compute the line repeatability of different line detectors on the HPatches [5] and ETH3D [57] datasets. We define line repeatability for a



(a) GT line assignments, shown as **MATCHED**, **UNMATCHED**, and **IGNORED**.



(b) Each color identifies a match  $(i, j) \in \mathcal{M}^l$  of the GT.

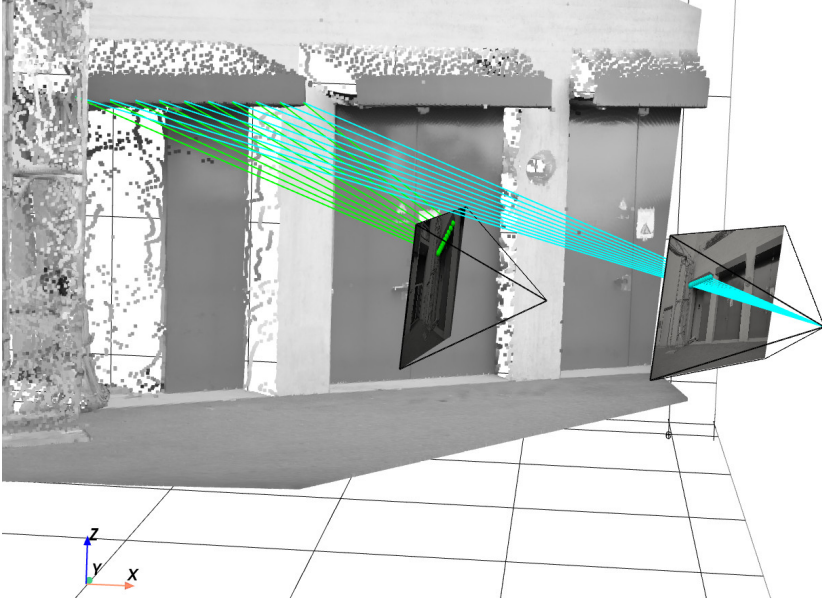
Figure 8: **Ground truth (GT) line assignments.** Examples of GT line matches. Note that blue lines are located in uncertain regions and depth discontinuities, and they are ignored during training.

	HPatches		ETH3D	
	#Lines	Rep. (%)	#Lines	Rep. (%)
LSD [21]	307	68.72	578	47.49
ELSED [62]	243	64.66	464	48.03
HAWP [78]	366	60.86	420	38.69
SOLD <sup>2</sup> [47]	167	65.49	332	39.86
F-Clip [15]	139	72.91	444	50.90
LETR [76]	95	70.65	311	47.70

Table 3: **Line segment detection comparison.** We compare different line segment detectors in terms of their repeatability in the HPatches [5] and ETH3D [57] datasets.

pair of images as the percentage between the number of line correspondences and the number of lines in the pair of images [42]. We establish line correspondences between images with the protocol defined in Sec. 3.4 of the main paper and Appendix A. Tab. 3 shows that the learned baseline F-Clip [15] obtains the highest repeatability, but detects few lines, due to the fact that it was trained on the ground truth lines of the Wireframe dataset [25]. On the contrary, LSD provides the best trade-off in terms of repeatability and number of lines. Thus, this good trade-off, as well as its low localization error and versatility, make LSD a very suitable choice for our approach.

We provide in Fig. 11 additional visualizations of line segments for two traditional methods: LSD [21] and



(a) 3D Visualization of a scene in ETH3D [57], with the depth associated with each point and the camera poses.



(b) GT Matching result

Figure 9: **Visualization of the ground truth (GT) generation.** To check if a pair of line segments  $I_i^A$  and  $I_j^B$  correspond to each other, we sample points along the segment: cyan points in the right image of (a). Points are lifted using depth and re-projected in the second image: green points in the left image of (a). We use the number of points that lie close to the segment in the second image to build each entry  $C_{i,j}^B$  of the cost matrix. Together with the reciprocal matrix  $C^A$  we define an assignment problem whose solution is our GT shown in (b).

ELSEED [62], and two learned methods: SOLD<sup>2</sup> [47] and HAWP [78]. While traditional ones sometimes detect noisy lines (for example in the sky), learned ones are often biased towards their training set and do not generalize very well to different settings, such as outdoor images.

However, it is important for GlueStick to generalize and perform well with other line segment detectors. In Fig. 10 we run GlueStick using either LSD or SOLD<sup>2</sup> [47] lines, and we evaluate the precision-recall of both methods on the ETH3D dataset [57]. The latter are generic lines extracted by a deep network, with strong repeatability and low localization error [47]. It can be seen from the precision-recall curves that 1) our GlueStick model trained on LSD lines is able to generalize to other lines such as SOLD<sup>2</sup> [47], and 2) the performance is slightly better with LSD lines. This is reasonable, since GlueStick was already trained on these lines. In summary, we chose LSD as base detector for downstream tasks since it remains one of the most accurate detector currently available, by directly relying on the image gradient at a sub-pixel level.

**Effect of the Fine-tuning.** Again in Fig. 10, we compare our final GlueStick model with its pre-trained version on homographies, *GlueStick - H*. The plot shows that pre-training on homographies is already sufficient to get very high per-

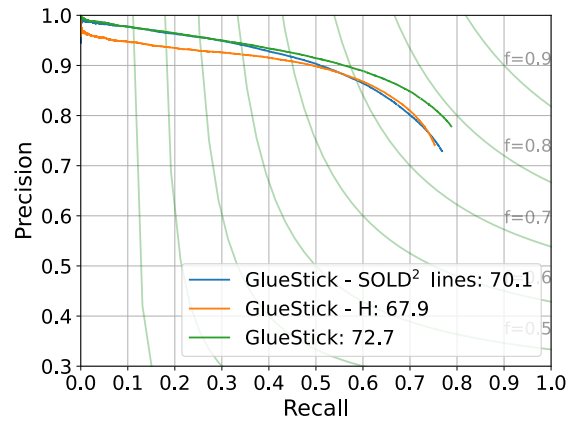


Figure 10: **Additional ablation study on the ETH3D dataset [57].** We report the precision-recall curve of the line matching, as well as Average Precision (AP) in the legend. Our final *GlueStick* model running with LSD [21] lines is compared to its pre-trained version on homographies (*GlueStick - H*), and the final model using SOLD<sup>2</sup> lines [47] (*GlueStick - SOLD<sup>2</sup> lines*).

formance on ETH3D - better than the previous state-of-the-art line matchers. Fine-tuning on MegaDepth [36] with real



(a) LSD [21]

(b) ELSED [62]

(c) SOLD<sup>2</sup>[47]

(d) HAWP [78]

Figure 11: **Comparison of line segment detectors.** Learned methods such as SOLD<sup>2</sup> [47] and HAWP [78] may not generalize well in all situations, such as outdoors. Traditional ones such as LSD [21] and ELSED [62] produce a lot of overlapping segments and sometimes noisy ones. We decided to use LSD for its high versatility and accuracy.

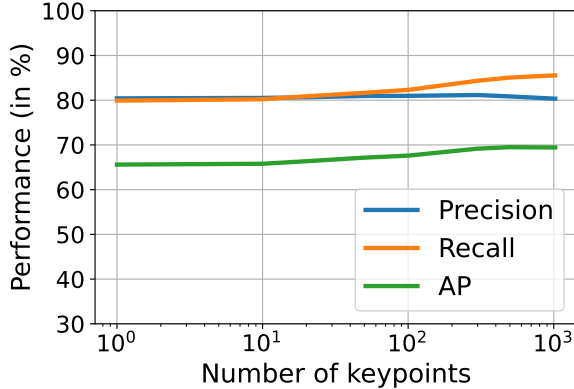


Figure 12: **Analysis of the dependence on keypoints.** We run GlueStick on 1000 image pairs warped by a homography (taken from the 1M distractor images of [49]), with varying numbers of keypoints, and report the precision, recall and Average Precision (AP) of the line matching. GlueStick robustly matches lines even when few or no keypoints are present.

viewpoint changes can however further improve the robustness of our matcher, as demonstrated by the stronger performance of the final model.

**Dependence on Point Matches.** When jointly matching two kinds of features, one caveat is often that one type of feature takes the lead and the other relies mainly on the first one. While we know from SuperGlue [54] that point-only matching is already very strong on its own, we show here that our architecture is very robust to the absence of keypoints and that line-only matching is still possible. We ran an evaluation of the precision, recall, and average precision (AP) of the line matching on 1000 validation images of our homography dataset (images taken from the 1M distractor images of [49]), and tested different maximum numbers of keypoints per image. The results showed in Fig. 12, highlight that our line matching is extremely robust to the lack of keypoints. The precision remains indeed constant, and the recall and AP are decreasing by at most 5% when switching from 1000 keypoints to no keypoints. Thus, this study confirms that our matcher can be used in texture-less areas where no keypoints are present, and is still able to match lines with high accuracy.

**Impact of the Line Length.** While we adopted a line representation based on the endpoints, one may wonder whether GlueStick can handle very long lines, and how it performs with respect to the line length. We studied this on the ETH3D [57] by categorising lines into three categories of length (in pixels): *Short* ( $[0, 50)$ ), *Medium* ( $[50, 150)$ ), and *Long* ( $[150, +\infty)$ ). The results are shown in Fig. 13. It can be seen that the best performance is obtained for long lines, showing that GlueStick is still able to match lines even without context in the middle of the line. This result is due to

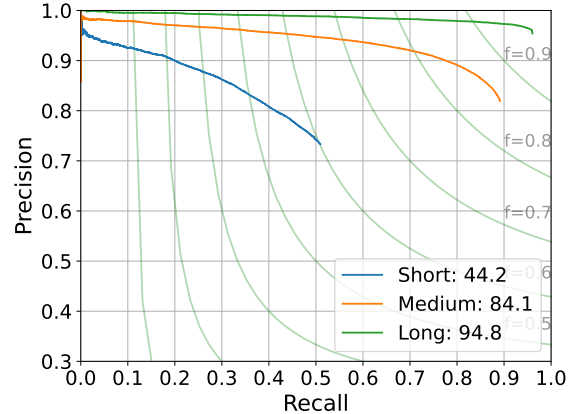


Figure 13: **Analysis of the impact of line length.** We run GlueStick on the ETH3D dataset [57] and evaluate separately the matching of Short, Medium, and Long lines. The best performance is obtained for long lines, as they are more stable than short ones. GlueStick can thus match long lines even with an endpoint representation.

the fact that long lines are more stable across views, while short ones are often noisy and not very repeatable.

**Robustness to Small Image Overlaps.** The image overlap and scale changes between images can play a large role in matching. To study the effect of image overlap on GlueStick, we revisited our line matching experiment on the ETH3D dataset [57] and separated the pairs of images into three categories of image overlap: *Small* ( $[0, 0.33)$ ), *Medium* ( $[0.33, 0.66)$ ), and *Large* ( $[0.66, 1]$ ). Overlap is defined as the proportion of pixels falling into the other image after reprojection. It is computed symmetrically between the two images, and the minimum of the two values is kept. Results are available in Fig. 14. While the performance naturally decreases with smaller overlaps, GlueStick maintains a strong performance on such hard cases.

## C. Experimental Details

In this section, we first provide additional baselines to the experiment on ScanNet for homography and relative pose estimation. Secondly, we give details and visualizations of the pure rotation estimation between two image pairs, and its application to image stitching. Thirdly, we provide a comparison of methods on homography estimation on the HPatches dataset [5]. Finally, we give the full results of visual localization on the 7Scenes dataset [59], though the performance on most scenes is already saturated.

### C.1. Relative Pose through Homographies

The experiment in Tab. 1 of the main paper was meant to evaluate the quality of homographies retrieved from points, lines or points+lines features. Homographies were evalu-

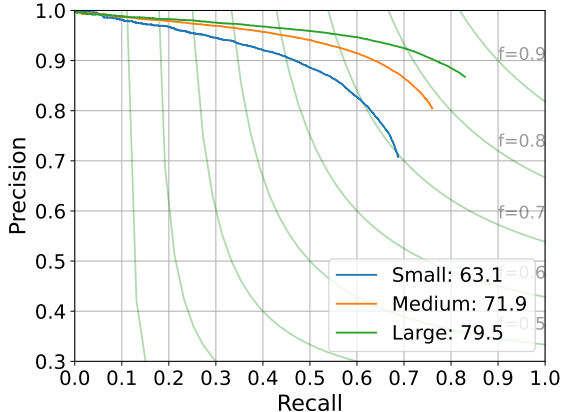


Figure 14: **Analysis of the impact of the image overlap.** We run GlueStick on the ETH3D dataset [57] and classify image pairs into three categories: Small, Medium, and Large overlap. While the performance of GlueStick decreases with smaller overlaps, it is able to maintain a high performance for all kinds of overlaps.

		Pose error ( $\downarrow$ )	Pose AUC ( $\uparrow$ )
Pose from H	SuperGlue (SG) [54]	18.1	15.6 / 29.8 / 39.4
	LoFTR [64]	16.8	15.8 / 30.9 / 41.4
	GlueStick	<b>14.1</b>	<b>19.3 / 35.4 / 46.0</b>
Pose from E	SuperGlue (SG) [54]	8.6	30.5 / 46.0 / 54.1
	LoFTR [64]	11.7	23.6 / 39.6 / 48.4
	GlueStick	<b>8.4</b>	<b>30.9 / 46.8 / 55.1</b>

Table 4: **Using essential matrices instead of homographies on ScanNet [14].** While our experiment on ScanNet is meant to evaluate homographies, we display here the results that would be obtained when using essential matrices to get a relative pose, instead of homographies. We report the median pose error in degrees, as well as the AUC at  $10^\circ / 20^\circ / 30^\circ$  error. Essential matrices are naturally more robust and obtain better results when evaluated on relative pose estimation.

ated by decomposing into the corresponding relative pose and evaluating the latter on the ScanNet dataset [14]. For the sake of completeness, we also report here the results that would be obtained with a more efficient method to obtain the relative pose: the 5-point algorithm to obtain the essential matrix [45], later decomposed as a relative pose. Tab. 4 demonstrates that the quality of relative poses retrieved through essential matrices is much higher than with homographies - as could be expected. GlueStick remains nevertheless the top performing method among all baselines. Note that we use here the outdoor models for all methods, for fairness reasons as GlueStick was only trained on outdoor data.

## C.2. Pure Rotation Estimation

In this section, we describe the details of the pure rotation algorithm used to perform the experiment of Sec. 4.4.2 of the main paper. Inside a Hybrid RANSAC [11], we design minimal and least square solvers to estimate the rotation based on points, lines, or a combination of both.

Images are cropped from the panorama images of SUN360 [74] and projected with a calibration matrix  $\mathbf{K} \in \mathbb{R}^{3 \times 3}$ . Fig. 15 shows some examples. The relation between both images is defined by:

$$\mathbf{x}^B = \mathbf{K}\mathbf{R}\mathbf{K}^{-1}\mathbf{x}^A, \quad (13)$$

where  $\mathbf{R} \in SO(3)$  is the rotation matrix between the cameras. Thus, we can calibrate the features detected on each image, multiplying them by  $\mathbf{K}^{-1}$ . This way, we only have to robustly estimate the 3 Degrees-of-Freedom (DoF) of the rotation.

Point features are sampled uniformly, and lines are sampled proportionally to the square root of their length to give priority to larger lines. The probability of choosing one type of feature is proportional to its number of matches. For example, if a pair has 60 point matches and 40 line matches, the probability of choosing a point is 60% and 40% for lines.

The minimal solver randomly chooses 2 feature matches (point-point, point-line, or line-line) and estimates the rotation based on them. This can be seen as aligning 2 sets of 3D vectors. Homogeneous points are already 3D vectors going from the camera center to the plane  $Z = 1$ . To get a vector from a line segment with homogeneous endpoints  $\mathbf{x}_s$  and  $\mathbf{x}_e$  we use its line-plane normal  $\mathbf{n} = \mathbf{x}_s \times \mathbf{x}_e$ . To make the method invariant to the order of the endpoints, we force the normals of the segments to have a positive dot product. This simple heuristic works as long as the sought rotation is less than  $180^\circ$ . Therefore, we can obtain a 3D vector from any of the types of features. For two (or more) correspondences, the optimal rotation can then be found with SVD using the Kabsch algorithm [27].

We provide a more extensive table of results than in the main paper in Tab. 5, as well as visual examples of the point and line matches obtained by GlueStick, and the resulting image stitching output in Fig. 15.

## C.3. Homography Estimation in HPatches

HPatches [5] is one of the most frequently used datasets to evaluate image matching. It contains 108 sequences where the scene contains only one dominant plane, with 6 images per sequence. Each sequence has either illumination or viewpoint changes. Similarly as in [54], we compute a homography from point and/or line correspondences and RANSAC, and compute the Area Under the Curve (AUC) of the reprojection error of the four image corners. We report the results for thresholds 3 / 5 / 10 pixels. We also

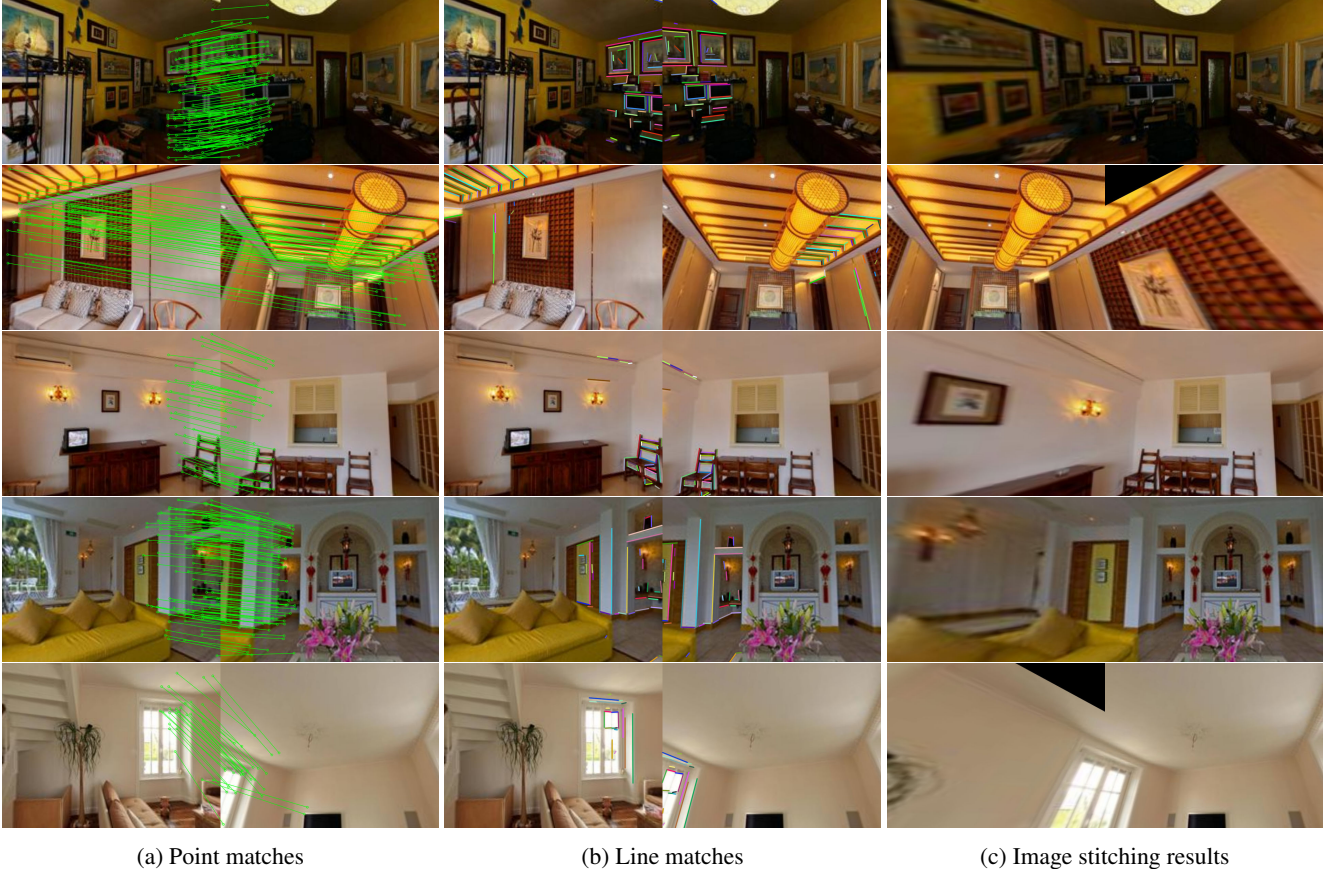


Figure 15: **Examples of GlueStick matches on image pairs of SUN360 [74].** We provide the point and line matches, as well as the stitching of the two images using the resulting matches.

	Rotation error (↓)		AUC (↑)					
	Avg	Med	0.25°	0.5°	1°	2°	5°	10°
LineTR [80]	60.37	10.80	0.239	0.307	0.366	0.414	0.455	0.474
LBD [82]	19.57	0.054	0.593	0.681	0.736	0.768	0.790	0.799
SOLD <sup>2</sup> [47]	23.74	0.308	0.232	0.384	0.515	0.609	0.682	0.713
L2D2 [1]	18.31	0.056	0.578	0.667	0.725	0.765	0.795	0.808
GlueStick-L	8.79	0.070	0.579	0.701	0.780	0.830	0.869	0.885
SG [54]	<b>0.135</b>	0.052	0.730	0.860	0.929	0.964	0.985	<b>0.992</b>
GlueStick-P	0.230	0.052	0.729	0.860	0.928	0.963	0.984	0.991
PL-Loc [80]	0.338	0.050	0.733	0.859	0.927	0.961	0.982	0.989
GlueStick-PL	0.327	<b>0.039</b>	<b>0.789</b>	<b>0.890</b>	<b>0.943</b>	<b>0.970</b>	<b>0.986</b>	0.991

Table 5: **Pure rotation estimation on SUN360 [74].** We estimate a rotation based on point-only, line-only, or points+lines matches. We report the average and median rotation error in degrees, as well as the Area Under the Curve (AUC) at 0.25 / 0.5 / 1 / 2 / 5 / 10 degrees error.

compute the precision and recall of the ground truth (GT) matches obtained by the GT homography.

We report the results in Tab. 6 and Fig. 16. HPatches is clearly saturated and the precision/recall metrics are already

very high for point-based methods. Note the strong performance of GlueStick on line matching, with an increase by nearly 10% in precision compared to the previous state of the art. Regarding point-based methods and homography scores, GlueStick obtains a very similar performance as the previous point matchers SuperGlue [54] and LoFTR [64], and ranks first (with a very small margin) in terms of homography estimation. In addition to the fact that HPatches is saturated, it contains also very few structural lines that could have been useful to refine the homography fitting. Thus, the improvement brought by line segments is not significant here.

#### C.4. Visual Localization on the Full 7Scenes Dataset

As stated in the main paper, 7Scenes [59] is a rather small-scale dataset for visual localization, which is already largely saturated for point-based methods. Adding line segments into the pipeline can improve the results only in a few scenes such as Fire, Office and mostly on Stairs. We demonstrate this in Tab. 7, where we used the same setup as described in the main paper. While all methods obtain close

	AUC ( $\uparrow$ )			Points ( $\uparrow$ )		Lines ( $\uparrow$ )		
	3px	5px	10px	P	R	P	R	
L	L2D2 [1]	43.73	55.98	69.39	-	-	55.55	38.76
	SOLD <sup>2</sup> [47]	26.60	36.41	48.82	-	-	80.57	77.43
	LineTR [80]	42.90	55.74	69.26	-	-	78.78	58.74
	LBD [82]	46.82	59.13	71.82	-	-	82.73	56.38
	GlueStick-L	46.61	61.45	76.32	-	-	<b>90.27</b>	76.69
P	SuperGlue [54]	66.21	77.77	88.05	<b>98.85</b>	97.44	-	-
	LoFTR [64]	66.15	75.28	84.54	97.60	<b>99.38</b>	-	-
	GlueStick-P	65.88	77.41	87.72	<b>98.85</b>	97.08	-	-
P+L	PL-Loc [80]	60.03	71.44	83.08	90.80	77.60	80.33	50.35
	GlueStick-PL	<b>66.88</b>	<b>78.14</b>	<b>88.12</b>	98.00	94.86	89.54	<b>80.44</b>

Table 6: **Homography estimation in HPatches [5].** We report the Area Under the Curve (AUC) of the cumulative error curve generated by the re-projection error of the four image corners at different thresholds (3px, 5px, 10px), as well as the precision (P) and recall (R) of the matches.

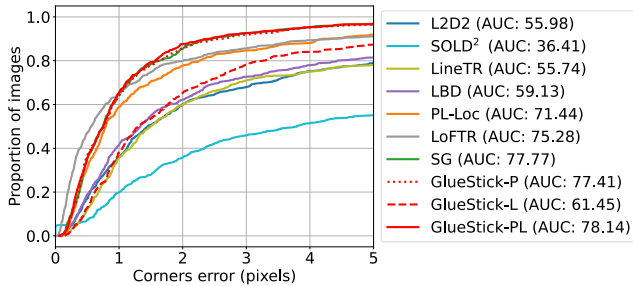


Figure 16: **HPatches [5] cumulative error curve.** We report the percentage of images where the homography is correctly predicted for various pixel error thresholds.

results on such a saturated dataset, GlueStick is slightly ahead of the baselines, and largely outperforms them on the most challenging scene, Stairs.

## D. Qualitative Examples

### D.1. Feature Matches

Fig. 17 displays some examples of line matching on the ETH3D dataset [57]. We plot in green the correct matches and in red the incorrect ones. Thanks to its spatial reasoning in the GNN and context-awareness, GlueStick is consistently matching more lines and with a higher precision than previous works. This is in particular true for scenes with repeated structures, such as the one in the right column, where the descriptors of SOLD<sup>2</sup>[47] and L2D2[1] do not have context from neighboring lines, and can only match a few lines.

### D.2. Visualization of the Camera Pose Estimation

We visualize the re-projection of points and lines on the scene Stairs of the 7Scenes dataset [59] in Fig. 18. We plot

in green the points and lines that were originally detected in 2D, and re-project in red the corresponding 3D features using the estimated camera pose. The reprojections of GlueStick are almost perfectly aligned compared to the ones of hloc [53, 52], highlighting the quality of the poses retrieved by our method.

### D.3. Failure Cases and Limitations

While jointly matching keypoints and lines in the same matching network helps disambiguating many challenging scenarios, GlueStick may still underperform in some scenarios. We list in the following some limitations of our method, and report some failure cases.

**Limitations.** Currently, the main performance bottleneck of GlueStick lies in the line segment detection. While this field has seen great advances in recent years, existing line detectors are still not as repeatable and accurate as point features, making the line matching more challenging. Partially occluded lines are also a potential issue for GlueStick, as it represents lines with their two endpoints. However, we observed a surprisingly good robustness of GlueStick to partially occluded lines, probably thanks to the neighboring points and lines that are not occluded. Note that it is also possible to equip GlueStick with a similar mechanism as in SOLD<sup>2</sup> [47], by sampling several points along the line segments, and matching them with the Needleman-Wunsch algorithm. We tried this option and observed a small increase in performance (notably in areas with occluded lines), but at the cost of higher running time. Therefore, we did not incorporate this feature in our final method.

Another issue is that points and lines are still detected with different methods for now. Thus, three networks / algorithms need to be run to detect and describe keypoints, detect lines, and finally match them. Jointly detecting and describing points and lines would be an interesting future direction of research. Furthermore, the extraction of discrete features such as points and lines is usually non-differentiable, such that one cannot get a fully differentiable pipeline going from the feature extraction to their matching. Enabling such end-to-end training could potentially make features better specialized for matching.

Finally, our current supervision requires ground truth correspondences of points and lines across images (usually obtained through reprojection with depth and camera poses). Other supervision signals such as epipolar constraints and using two-view geometry would be an interesting direction of improvement in the future.

**Failure cases.** We display in Fig. 19 a few examples of scenarios where GlueStick may still fail or underperform. First, in scenes with repeated patterns, GlueStick is able to find a consistent matching, but can be displaced by one pattern if there is no additional hint to disambiguate the transform between the two images. This is for example the case

	Points			Points + Lines				
	SuperGlue [54]	LoFTR [64]	GlueStick - P	SOLD <sup>2</sup> [47]	LineTR [80]	L2D2 [1]	SG + Endpts	GlueStick - PL
Chess	<b>2.4</b> / 0.81 / 94.5	2.5 / 0.86 / 93.8	<b>2.4</b> / <b>0.80</b> / 94.3	<b>2.4</b> / 0.82 / 94.4	<b>2.4</b> / 0.81 / 94.5	<b>2.4</b> / 0.83 / 94.5	<b>2.4</b> / 0.82 / <b>94.6</b>	<b>2.4</b> / 0.82 / 94.5
Fire	1.9 / 0.76 / 96.4	1.7 / <b>0.66</b> / 96.8	2.0 / 0.78 / 96.6	<b>1.6</b> / 0.69 / 96.8	<b>1.6</b> / 0.69 / 97.0	<b>1.6</b> / 0.69 / 96.4	1.7 / 0.69 / 97.2	1.7 / 0.69 / <b>97.4</b>
Heads	1.1 / 0.74 / 99.0	1.1 / 0.78 / 98.2	1.1 / 0.74 / 99.2	<b>1.0</b> / <b>0.72</b> / <b>99.4</b>	1.1 / 0.75 / 99.1	<b>1.0</b> / 0.73 / 99.3	1.1 / 0.74 / 99.2	<b>1.0</b> / 0.74 / <b>99.4</b>
Office	2.7 / 0.83 / 83.9	2.7 / 0.83 / 82.0	2.7 / 0.83 / 83.6	<b>2.6</b> / 0.80 / <b>84.7</b>	<b>2.6</b> / <b>0.79</b> / 84.4	<b>2.6</b> / 0.81 / 83.9	<b>2.6</b> / <b>0.79</b> / 84.4	<b>2.6</b> / <b>0.79</b> / 84.6
Pumpkin	4.0 / 1.05 / 62.0	<b>3.9</b> / 1.12 / <b>62.4</b>	<b>3.9</b> / <b>1.04</b> / 62.2	4.0 / 1.07 / 60.2	4.0 / 1.08 / 61.5	4.0 / 1.05 / 61.3	4.0 / 1.06 / 61.2	4.0 / 1.06 / 61.5
Red kitchen	3.3 / <b>1.12</b> / 72.5	3.3 / 1.14 / <b>73.8</b>	3.3 / <b>1.12</b> / 72.8	<b>3.2</b> / 1.15 / 72.6	3.3 / 1.15 / 72.6	<b>3.2</b> / 1.14 / 72.9	<b>3.2</b> / 1.14 / 72.8	<b>3.2</b> / 1.13 / 73.0
Stairs	4.7 / 1.25 / 53.4	4.4 / 0.95 / 53.9	4.4 / 1.21 / 55.4	3.2 / 0.83 / 75.8	3.7 / 1.02 / 66.6	4.1 / 1.15 / 55.8	3.1 / 0.81 / 75.6	<b>2.9</b> / <b>0.79</b> / <b>79.7</b>
Total	2.9 / 0.94 / 80.2	2.8 / 0.91 / 80.1	2.8 / 0.93 / 80.6	2.6 / 0.87 / 83.4	2.7 / 0.90 / 82.2	2.7 / 0.91 / 80.6	2.6 / <b>0.86</b> / 83.6	<b>2.5</b> / <b>0.86</b> / <b>84.3</b>

Table 7: **Visual localization on the full 7Scenes dataset [59].** We report the median translation error (cm) / median rotation error (deg) / pose AUC at a 5 cm / 5 deg threshold. Most scenes are already saturated for point methods, and lines can hardly make a difference.

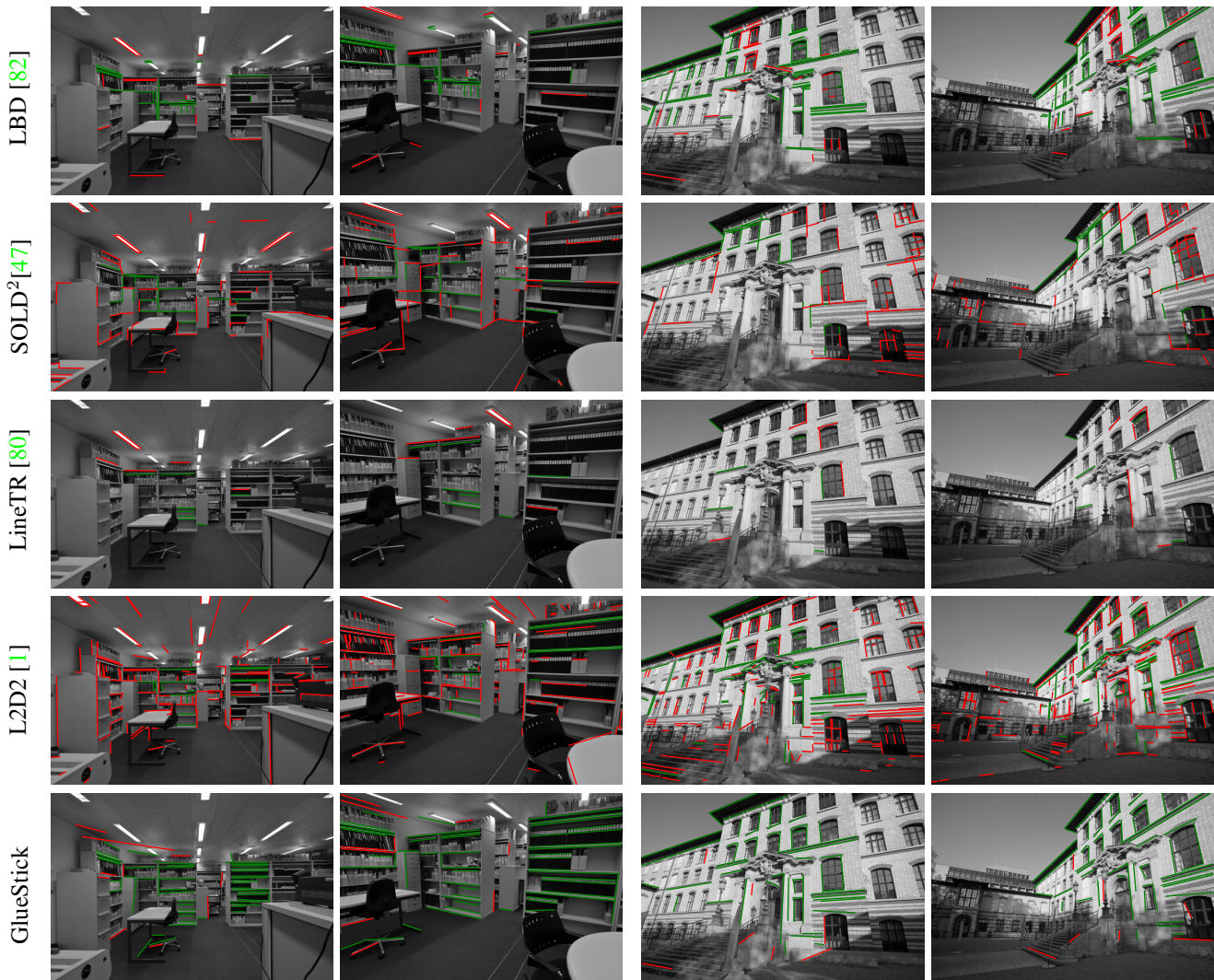


Figure 17: **Line matches examples on ETH3D [57].** We display correct line matches in green and incorrect ones in red for several state-of-the-art line matchers.

on 7Scenes Stairs [59], when the camera is only seeing several steps, and it is unclear which step should be matching

with which one in the other image.

Secondly, GlueStick has not been trained for large rota-



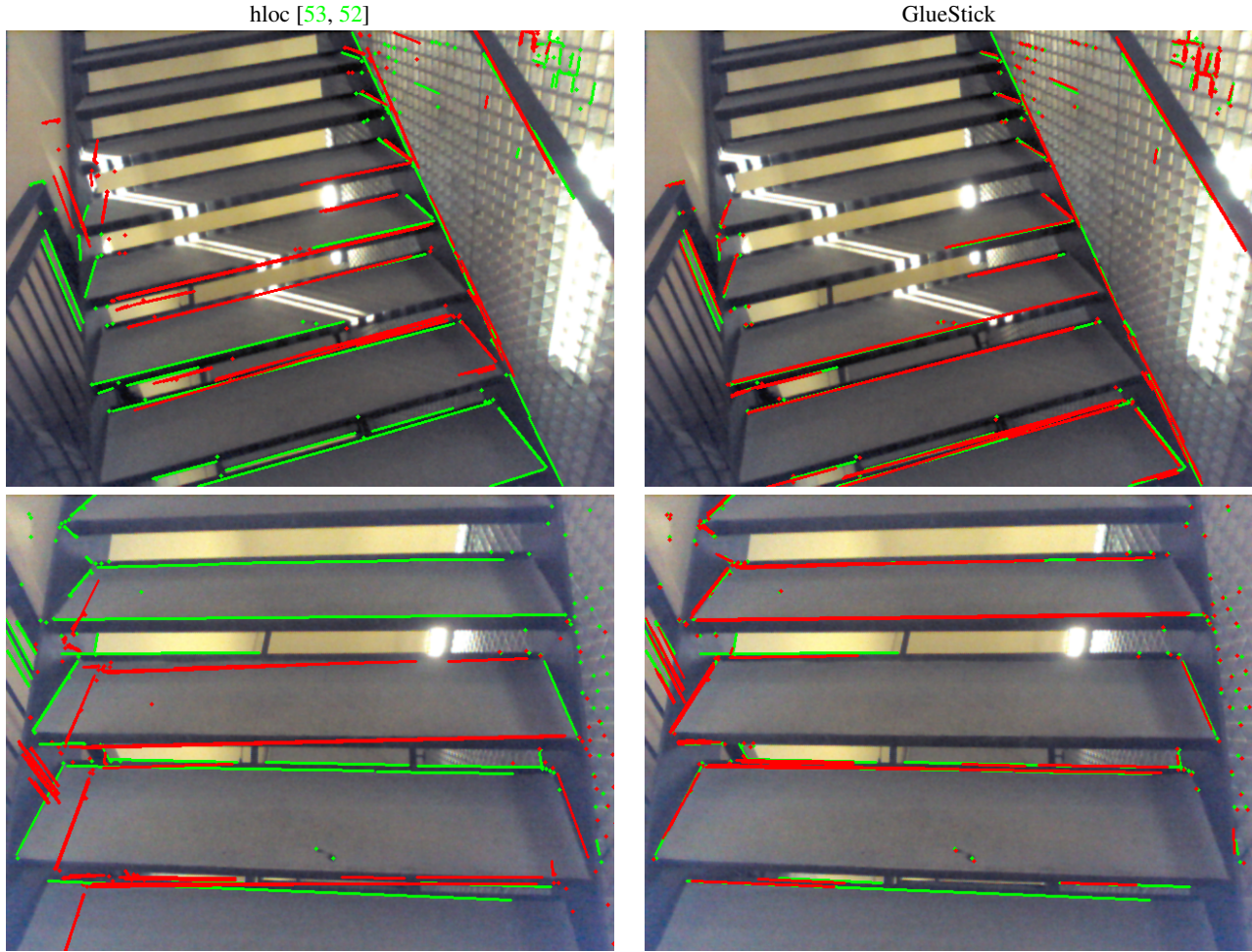


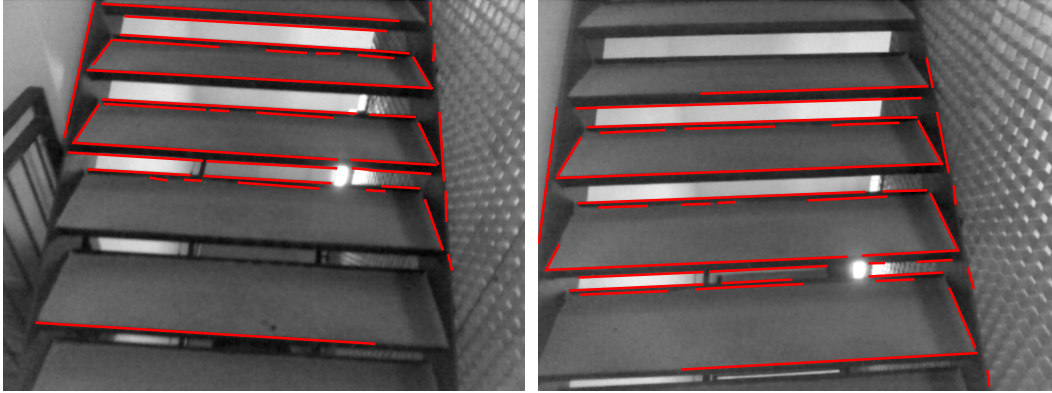
Figure 18: **Camera pose estimation visualizations.** We compare the originally detected keypoints and lines (in green) with the re-projected points and lines using the estimated camera pose (in red), for hloc [53, 52] with SuperPoint [16] + SuperGlue [54] and our method. The reprojections of GlueStick align almost perfectly with the 2D detections, showing a high quality estimated pose.

tions beyond  $45^\circ$  and often fails in these scenarios. The pre-training with homographies was done with rotations lower than  $45^\circ$ , and real viewpoint changes are rarely with such rotations. Nonetheless, a simple fix is to rotate one of the two images by  $0^\circ, 90^\circ, 180^\circ$  and  $270^\circ$ , match it with the other image, and keep the best matching among the four.

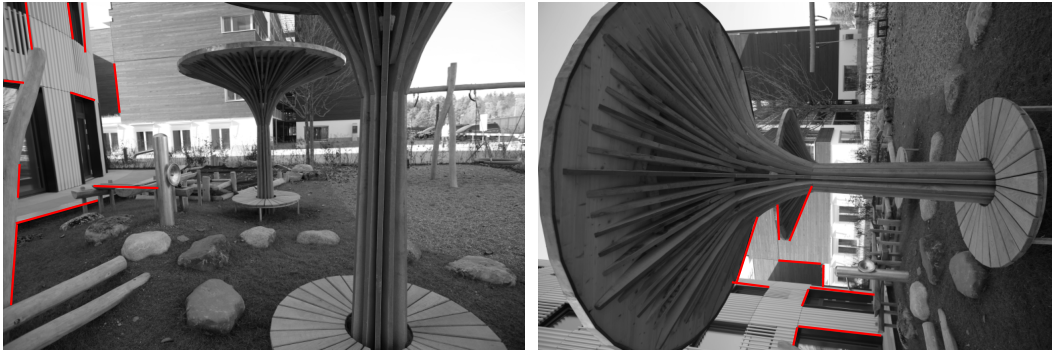
Finally, a challenging scenario happens when the images have low texture in combination with symmetric structures. The former makes visual descriptors less reliable, while the latter makes it harder to disambiguate matches from the spatial context. The performance is then degraded in such situations. Having access to sequential data and feature tracking may help solving such cases.

## E. Attention visualization

We display the attention for some nodes in Fig. 20. This visualization is obtained by taking the attention matrix at various cross layers, averaging it across all heads, and taking the top 20 activated nodes. Green lines are used for nodes with connectivity greater than 0 (i.e. line endpoints), and cyan for nodes that are isolated keypoints. It can be seen from the left column that keypoint attention is leveraging the line structure to look for the right points along the line. In the right column, we can see that line endpoints can benefit from both keypoint and line endpoint attention. The attention is initially looking broadly at the image, before gradually focusing on the corresponding node in the other image. Thus, both points and line endpoints can complement each other to disambiguate the matching process.



(a) Repeated patterns: GlueStick finds consistent line matches, but displaced by one step.



(b) GlueStick is not trained on large rotations.



(c) The lack of texture / symmetric structures make visual descriptors / spatial descriptors less reliable.

Figure 19: **Failure cases.** We display correct line matches in green and wrong ones in red. GlueStick may still fail or underperform in some situations, such as (a) perfectly repeated patterns that are hard to disambiguate, (b) large rotation (e.g.  $> 45^\circ$ ), and (c) lack of texture and symmetric structures.

## 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, 9, 14, 15, 16

[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

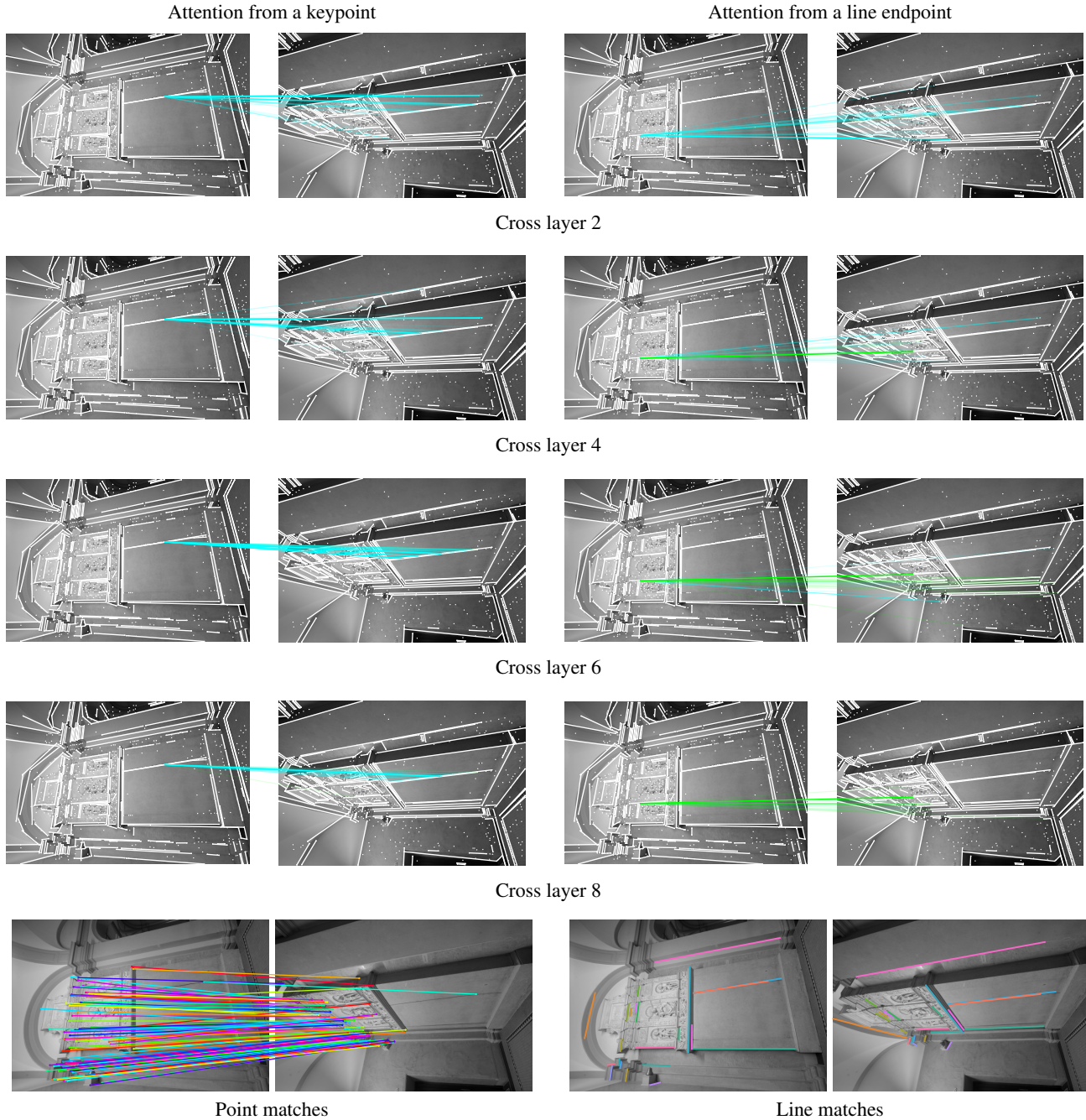


Figure 20: **Cross attention visualization.** We plot in the first column the attention from a keypoint and in the right column the attention of a line endpoint, for various layers in the Graph Neural Network. We compute here the average attention across all heads and keep the top 20 activated nodes. More opaque lines means higher attention, green matches are connected to a line endpoint in the second image, and cyan matches are connected to an isolated keypoint. The last row pictures the final matches.

[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 Mikołajczyk. HPatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *IEEE Conf.*

- Comput. Vis. Pattern Recog. (CVPR)*, 2017. 7, 9, 12, 13, 15
- [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 re-localization 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] Federico Camposeco, Andrea Cohen, Marc Pollefeys, and Torsten Sattler. Hybrid camera pose estimation. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018. 13
- [12] 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
- [13] 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
- [14] 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, 13
- [15] Xili Dai, Haigang Gong, Shuai Wu, Xiaojun Yuan, and Ma Yi. Fully convolutional line parsing. *Neurocomputing*, 506:1–11, 2022. 9
- [16] 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, 17
- [17] 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
- [18] Bin Fan, Fuchao Wu, and Zhanyi Hu. Robust line matching through line–point invariants. *Pattern Recognition*, 45(2):794–805, 2012. 2
- [19] 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
- [20] 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
- [21] 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, 9, 10, 11
- [22] 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
- [23] 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
- [24] Paul VC Hough. Method and means for recognizing complex patterns, Dec. 18 1962. US Patent 3,069,654. 2
- [25] 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, 9
- [26] 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
- [27] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976. 13
- [28] Christopher Kropp, Christian Koch, and Markus König. Interior construction state recognition with 4D BIM registered image sequences. *Automation in Construction*, 86, 2018. 1
- [29] Harold W Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 5
- [30] 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
- [31] 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
- [32] 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
- [33] Viktor Larsson. PoseLib - Minimal Solvers for Camera Pose Estimation. <https://github.com/vlarsson/PoseLib>, 2020. 8
- [34] 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
- [35] 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

- [36] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018. 6, 9, 10
- [37] Shaohui Liu, Yifan Yu, Rémi Pautrat, Marc Pollefeys, and Viktor Larsson. 3d line mapping revisited. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2023. 8
- [38] David G Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis. (IJCV)*, 60(2):91–110, 2004. 1
- [39] 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
- [40] Ezio Malis and Manuel Vargas. Deeper understanding of the homography decomposition for vision-based control. Research Report RR-6303, INRIA, 2007. 7
- [41] 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
- [42] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kadir, and L Van Gool. A comparison of affine region detectors. *Int. J. Comput. Vis. (IJCV)*, 65:43–72, 2005. 9
- [43] 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
- [44] 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
- [45] David Nistér. An efficient solution to the five-point relative pose problem. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2003. 13
- [46] Rémi Pautrat, Daniel Barath, Viktor Larsson, Martin R. Oswald, and Marc Pollefeys. Deeplsd: Line segment detection and refinement with deep image gradients. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2023. 2
- [47] Rémi Pautrat, Juan-Ting Lin, Viktor Larsson, Martin R. Oswald, and Marc Pollefeys. SOLD<sup>2</sup>: Self-supervised occlusion-aware line description and detection. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2021. 2, 6, 7, 8, 9, 10, 11, 14, 15, 16
- [48] 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
- [49] F. Radenović, A. Iscen, G. Toliás, Y. Avrithis, and O. Chum. Revisiting oxford and paris: Large-scale image retrieval benchmarking. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018. 6, 12
- [50] 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
- [51] 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
- [52] Paul-Edouard Sarlin. Visual localization made easy with hloc. <https://github.com/cvg/Hierarchical-Localization>. 8, 15, 17
- [53] 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, 15, 17
- [54] 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, 12, 13, 14, 15, 16, 17
- [55] Torsten Sattler et al. RansacLib - A Template-based \*SAC Implementation. <https://github.com/tsattler/RansacLib>, 2019. 7, 8
- [56] Cordelia Schmid and Andrew Zisserman. Automatic line matching across views. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 666–671, 1997. 2
- [57] 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, 9, 10, 12, 13, 15, 16
- [58] 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
- [59] 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, 12, 14, 15, 16
- [60] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967. 4
- [61] 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
- [62] Iago Suárez, José M Buenaposada, and Luis Baumela. ELSEd: Enhanced line segment drawing. *Pattern Recognition*, 127:108619, 2022. 2, 9, 10, 11
- [63] 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
- [64] 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, 13, 14, 15, 16
- [65] 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
- [66] 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
- [67] Alexander Vakhitov and Victor Lempitsky. Learnable line segment descriptor for visual slam. *IEEE Access*, 7:39923–39934, 2019. 2
- [68] 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
- [69] 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
- [70] Lu Wang, Ulrich Neumann, and Suya You. Wide-baseline image matching using line signatures. In *Int. Conf. Comput. Vis. (ICCV)*. IEEE, 2009. 2
- [71] Qing Wang, Jiaming Zhang, Kailun Yang, Kunyu Peng, and Rainer Stiefelwagen. Matchformer: Interleaving attention in transformers for feature matching. In *Asian Conf. on Comput. Vision (ACCV)*, 2022. 2
- [72] Zhiheng Wang, Fuchao Wu, and Zhanyi Hu. MSLD: A robust descriptor for line matching. *Pattern Recognition*, 42(5):941–953, 2009. 2
- [73] Erik Wijmans and Yasutaka Furukawa. Exploiting 2D floorplan for building-scale panorama rgbd alignment. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2017. 8
- [74] 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, 13, 14
- [75] 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
- [76] 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, 9
- [77] 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
- [78] 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, 9, 10, 11
- [79] 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
- [80] 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, 14, 15, 16
- [81] 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
- [82] 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, 14, 15, 16
- [83] 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
- [84] 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
- [85] 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
- [86] Yichao Zhou, Haozhi Qi, and Yi Ma. End-to-end wireframe parsing. In *Int. Conf. Comput. Vis. (ICCV)*, 2019. 2
- [87] 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
- [88] 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