LightGlueStick: a Fast and Robust Glue for Joint Point-Line Matching

Aidyn Ubingazhibov[1]   Rémi Pautrat[2]   Iago Suárez[3]   Shaohui Liu[1]
Marc Pollefeys[1,2]   Viktor Larsson[4]
[1]ETH Zurich   [2]Microsoft Spatial AI Lab   [3]Qualcomm XR Labs Europe   [4]Lund University

Abstract

Lines and points are complementary local features, whose combination has proven effective for applications such as SLAM and Structure-from-Motion. The backbone of these pipelines are the local feature matchers, establishing correspondences across images. Traditionally, point and line matching have been treated as independent tasks. Recently, GlueStick proposed a GNN-based network that simultaneously operates on points and lines to establish matches. While running a single joint matching reduced the overall computational complexity, the heavy architecture prevented real-time applications or deployment to edge devices.

Inspired by recent progress in point matching, we propose LightGlueStick, a lightweight matcher for points and line segments. The key novel component in our architecture is the Attentional Line Message Passing (ALMP), which explicitly exposes the connectivity of the lines to the network, allowing for efficient communication between nodes. In thorough experiments we show that LightGlueStick establishes a new state-of-the-art across different benchmarks. The code is available at https://github.com/aubingazhib/LightGlueStick.

1. Introduction

Visual feature matching is a fundamental component in numerous computer vision tasks, including visual localization, pattern detection, Structure-from-Motion (SfM), and Simultaneous Localization and Mapping (SLAM). These tasks are essential for navigation and positioning in fields such as robotics, augmented reality, architecture, and manufacturing. Classical handcrafted keypoint detectors and descriptors powered some of the first advances in these areas. However, these methods typically require well-textured regions and struggle with viewpoint and illumination changes. Conversely, line segments are abundant in texture-less areas, provide robust structural cues, and are resilient against occlusion and illumination changes. Their complementary nature makes the joint matching of points and lines an interesting paradigm for robust sparse image
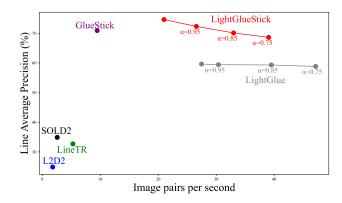

Figure 1. **LightGlueStick outperforms other state-of-the-art line matchers with a speed comparable to LightGlue [39]**. The depth adaptivity mechanism allows to predict line matches earlier. The parameter α controls the tradeoff between speed and accuracy. Here, LightGlue [39] matches lines using endpoints.

matching.

However, line matching presents unique challenges. Line segments are frequently partially occluded or fragmented, and in general less repeatable compared to points, at least in terms of endpoint consistency. As each line can span a large region of the image, their local descriptors are also more affected by foreshortening and other perspective effects. This yields a higher sensitivity to viewpoint changes, making them harder to both detect and describe. This is further complicated by lines frequently appearing in texture-poor areas, e.g. delineating walls or other planar surfaces.

Moreover, many image-matching applications require on-device execution with strict latency and power constraints. These conditions limit the practicality of recent dense [16, 18, 69] and semi-dense [28] matching methods, which typically demand significant computational resources. Consequently, sparse matching techniques become preferable due to their fast execution with minimal hardware requirements. A lightweight scene representation is also a requirement for real-time mapping purposes when the map needs to be compact or efficiently transmitted over

limited-bandwidth networks.

Recently, deep learning methods have significantly outperformed classical matching and filtering techniques, such as mutual nearest neighbor and ratio test [42]. Their key to success is to relate not only to the visual descriptors representing the image appearance but also to consider the local feature geometry, i.e. how the points and lines are spatially distributed in the images. These learned matchers have since replaced classical hand-crafted heuristics based on local planarity, coherence, or filtering [19, 44, 56, 78]. HDPL [24] and GlueStick [50] introduced the first learning approaches for joint matching of points and lines, yet the computational cost limits their applicability in real-time, on-device scenarios. Recently, LightGlue [39] has gained attention for significantly accelerating point-only matching through efficient attention mechanisms and reduced redundant computations.

In this paper, we introduce the next generation of point-line matching with LightGlueStick, a novel matcher that marries the matching capabilities of GlueStick with the computational efficiency of LightGlue. This facilitates effective joint matching of points and lines, making it suitable for real-time and on-device applications. Considering the importance of speed for downstream tasks, our approach ensures that joint matching remains practical by significantly reducing inference latency. As we show in our experiments, LightGlueStick surpasses current state-of-the-art sparse methods in both accuracy and runtime performance.

We re-design the joint point-line matching strategy with optimized attention mechanisms [66] into a unified architecture. Our method represents keypoints and line endpoints as nodes within a graph-based structure, where edges dynamically adapt across layers. We leverage self-attention and cross-attention techniques from transformers [61, 76], complemented by our novel Attentional Line Message Passing (ALMP) layer. This layer explicitly encodes line connectivity, enhancing communication between endpoints while allowing the network to ignore non-repeatable line endpoints. Extensive evaluation across multiple benchmarks demonstrates that LightGlueStick achieves state-of-the-art performance in multiple tasks. The code will be released upon publication. Our key contributions are:

- We achieve significant speedups while maintaining superior performance compared to state-of-the-art point and line matchers.
- We introduce a new Attentional Line Message Passing layer to improve the communication between line endpoints.
- Our approach is flexible, with a tunable threshold to trade off matching accuracy and efficiency.
- Extensive experiments demonstrate that our approach sets new state-of-the-art results across multiple benchmarks.

2. Related Work

Point matching. Feature point matching has a long history in computer vision, and was originally performed by extracting handcrafted descriptors from image patches [6, 42, 59]. Point matches were then obtained with either mutual nearest neighbor or ratio test [42]. The resurgence of deep learning introduced a series of more robust descriptors by extracting learned embeddings of image patches [43, 47, 48, 70, 71, 82]. Later, feature descriptors started to be extracted densely [13, 15, 17, 21, 51, 58, 68, 73, 85, 86], so that point features can be retrieved with bi-linear interpolation from the dense map. Learned matchers were originally introduced with the pioneering work SuperGlue [61], which proposed to combine Graph Neural Networks (GNNs) and transformers to enrich the point descriptors with the context of the other points. Subsequent works improved the original architecture to make it less costly to use [9, 39, 64]. LightGlue [39] proposed in particular an adaptivity mechanism allowing to finish the inference earlier for easy image pairs, thus significantly enhancing efficiency. OmniGlue [27] improved the robustness of the matching even further, by incorporating features from a vision foundation model. Instead of matching sparse keypoints, another line of research aims at matching image pairs densely. Initiated by LoFTR [69], this field has been very active in recent years [10, 16, 18, 25, 28, 33, 69, 72, 79]. These matches obtain good results at a higher computational cost. In this work, we take inspiration from the breakthroughs in efficiency brought by LightGlue [39], and incorporate them into a sparse joint point-line matcher.

Line matching. Similarly as for points, line matching has originally been performed by extracting image patches around lines and by computing handcrafted heuristics to describe the image gradient [34, 77, 78, 80, 84]. In the same spirit, early line matchers followed the same trend by running neural networks on the image patches around lines [1, 30, 31], bringing additional robustness to the line descriptors. Further improvements of the learned line descriptors involved extracting dense line descriptors [49, 74], sampling and describing points along the lines to increase the robustness to occlusions [1, 49], and jointly extracting lines and their descriptors with a single network [1, 49]. With the advent of transformers and GNNs, recent methods used attention within a line to enrich its line descriptor [83], while others connected the line segments into a graph and added self- and cross-attention between line nodes [24, 45, 50]. While WGLSM [45] only used line features, HDPL [24] and GlueStick [50] combined points and lines within the same graph. The former represents lines as a single node in the graph, while the latter leverages the line endpoint descriptors to ease communication between point and line features. Although GlueStick [50] is the current state-of-the-art point and line matcher, its execution time
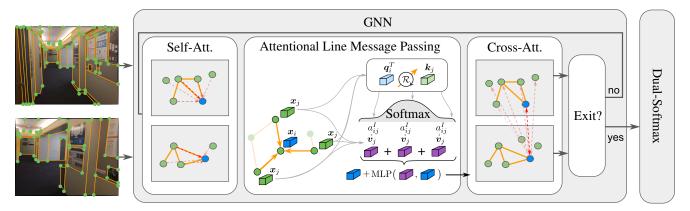
Figure 2. **Overview of the proposed pipeline**: LightGlueStick takes as input two images where points and lines have been detected and described. Our Graph Neural Network (GNN) is composed of three stages: 1) Self-Attention, 2) our novel Attentional Line Message Passing, and 3) Cross-Attention. Depending on the achieved matching result at each layer, we can stop the execution to save computation. The last step of Dual-Softmax generates two assignation matrices for points and lines, respectively.

remains slow, making it unpracticable for real-time applications. Our work improves the architecture of GlueStick, by making it more efficient, while maintaining or improving its performance.

Joint use of points and lines. Jointly matching point and line features unleashes multiple applications fusing the two kinds of features. SLAM has a long history of leveraging both points and line segments [20, 22, 35, 38, 52, 57, 81, 90]. The long range of lines is indeed a strong asset to reliably track features in a sequence and to reduce the drift over time. Line by-products can also be used in SLAM to add constraints, such as their junctions [57] and the vanishing points [38, 67]. Liu et al. [40] proposed to reconstruct lines in 3D, based on an existing point reconstruction, and further extended it into a full SfM pipeline integrating points, lines, and vanishing points [41]. Jointly using point and line features can also benefit visual localization and the related solvers by providing additional constraints, as demonstrated in [2, 40, 55, 75, 88, 89]. Finally, while lines do not bring constraints for relative pose estimation, [26] showed that one could combine points, line endpoints, and vanishing points to better constrain the relative pose between two images.

3. LightGlueStick

Our goal in this work is to match a given set of point and line features between two images A and B. We use I to refer generically to either image. We adopt the same wireframe formulation as in [50], where we represent lines with their two endpoints and initialize a graph where nodes consist of keypoints and line endpoints, and edges connect endpoints that are part of the same line. We first extract a dense descriptor map of dimension D for the entire image using an off-the-shelf network, from which we can interpolate the initial embedding $x_i^I \in \mathbb{R}^D$ of each node i in image I. We describe in Sec. 3.1 how we enrich these initial node embeddings with the visual and spatial context of the other nodes, through a Graph Neural Network (GNN) [7] and successive layers of transformer operations [76]. After the GNN, we leverage the enriched node embeddings to associate the point and line features of both images (Sec. 3.2). The overview of our method is available in Fig. 2.

3.1. Transformer Backbone

We describe in the following how node features are enriched in our proposed GNN, which consists of a series of 3 layers: self-attention, cross-attention, and a novel Attention Line Message Passing (ALMP). Each of these three layers is repeated up to L times to successively enrich the representation of each node before establishing matches.

Residual feature update. In each layer, the feature vector x_i^I of a node i is updated by applying a Multi-Layer Perceptron (MLP) to the concatenation of x_i^I and $m_i^{I \leftarrow S}$, where $m_i^{I \leftarrow S}$ represents a message received from a source image $S \in \{A, B\}$:

$$x_i^I \leftarrow x_i^I + \text{MLP}\left(\left[x_i^I | m_i^{I \leftarrow S}\right]\right), \quad (1)$$

where $[\cdot | \cdot]$ represents concatenation. In self-attention layers, messages are aggregated only from nodes within the same image, implying $S = I$. In cross-attention layers, node states are updated using nodes from the other image, meaning $S = \{A, B\} \setminus I$.

Self- and cross-attention layers. We use a lightweight attention mechanism [39, 66] for the implementation of the self- and cross-attention. The message is computed by an attention mechanism:

$$m_i^{I \leftarrow S} = \sum_{j \in S} \text{Softmax}_{k \in S}(a_{ik}^{IS})_j v_j^S, \quad (2)$$

where a_{ij}^{IS} is an attention weight between point $i \in I$ and point $j \in S$, while $v_j^S \in \mathbb{R}^D$ is the value vector obtained by applying a linear transformation to x_j^S. We define next how this attention is computed.

Consider nodes i and j at positions $p_i, p_j \in \mathbb{R}^2$. For self layers, we define the attention score as

$$a_{ij}^I = q_i^T \, \mathcal{R}(p_j - p_i) \, k_j, \tag{3}$$

where $q_i, k_j \in \mathbb{R}^K$ are obtained from x_i^I and x_j^I via separated linear transforms. Here, $\mathcal{R}(d)$ is a rotary positional encoding [66] that increases attention between nearby elements in a learned distance space. To ensure translation equivariance, we use the relative position $d = p_j - p_i$ and project it onto learned 2D vectors b_k to obtain $K/2$ angles θ_k. This defines

$$\mathcal{R}(d) = \begin{pmatrix} \hat{\mathcal{R}}(b_1^\top d) & & 0 \\ & \ddots & \\ 0 & & \hat{\mathcal{R}}(b_{K/2}^\top d) \end{pmatrix}, \hat{\mathcal{R}}(\theta_k) = \begin{pmatrix} \cos\theta_k & -\sin\theta_k \\ \sin\theta_k & \cos\theta_k \end{pmatrix}. \tag{4}$$

Thus, $\mathcal{R}(d)$ splits the K-dimensional space into $K/2$ 2D subspaces and rotates each by θ_k (akin to Fourier Features [36, 54]), combining spatial relative location with the query-key alignment in a_{ij}^I. In our implementation, we use $K = 64$ with four parallel attention heads.

For cross layers, the attention score a_{ij}^{IS} is computed as

$$a_{ij}^{IS} = (k_i^I)^\top \, k_j^S, \tag{5}$$

where the cross layers are bidirectional to save computation.

Attentional Line Message Passing. GlueStick introduced the Line Message Passing (LMP) layer, which averages messages from neighboring nodes based on line connectivity. However, line detectors can produce non-repeatable lines across views, causing inconsistencies in line connectivity for some endpoints that are otherwise matchable. To account for this, we propose Attentional Line Message Passing (ALMP), which aggregates messages with an attention mechanism.

Line endpoint feature vectors are updated as in Eq. (1), where the messages are passed between nodes of the same image. The message is computed as:

$$m_i^I = \sum_{j \in \mathcal{N}(i) \cup \{i\}} \text{Softmax}_{k \in \mathcal{N}(i) \cup \{i\}}(a_{ik})_j \, v_j^I, \tag{6}$$

where $\mathcal{N}(i)$ denotes the set of neighboring endpoint nodes of i based on its line connectivity, that is, endpoints directly connected to i by wireframe construction. The term a_{ik} represents the attention score between endpoints i and k, calculated as in Eq. (3), and $v_j^I \in \mathbb{R}^D$ is a value vector obtained by applying a linear transformation to the endpoint feature vector x_j^I.

By rotating feature vectors based on the relative positions of line endpoints, the network effectively captures the angular relationships between lines meeting at a junction, thereby facilitating the matching of junctions with similar angular connectivity across views. The learned attention further enables the network, after the cross-layer, to identify endpoints that are non-repeatable but connected to repeatable ones, and to ignore them for improved matching.

3.2. Correspondence Prediction

Points. To predict point correspondences, we compute an assignment matrix based on point-wise similarity scores. The similarity score s_{ij}^p for points i in A and j in B is:

$$s_{ij}^p = \text{Linear}(x_i^A)^T \text{Linear}(x_j^B), \tag{7}$$

where $\text{Linear}(\cdot)$ is a learned linear transformation. We compute a matchability score for each point $i \in I$ as

$$\sigma_i^{p,I} = \text{Sigmoid}(\text{Linear}(x_i^I)). \tag{8}$$

Finally, the assignment matrix S_{ij}^p combines the point similarity and matchability:

$$S_{ij}^p = \sigma_i^{p,A}\sigma_j^{p,B}\text{Softmax}_{j \in B}(s_{ij}^p)\text{Softmax}_{i \in A}(s_{ij}^p). \tag{9}$$

Lines. Lines are matched similarly, but by matching endpoints in an order-agnostic manner. Suppose x_s^I, x_e^I are feature vectors of two endpoints of the same line in image I. We first project the features with a linear transform:

$$y_s^I = \text{Linear}(x_s^I), \quad y_e^I = \text{Linear}(x_e^I) \tag{10}$$

The similarity score s_{ij}^l for lines i in A and j in B is then:

$$s_{ij}^l = \max \left((y_s^A)^T y_s^B + (y_e^A)^T y_e^B, \\ (y_s^A)^T y_e^B + (y_e^A)^T y_s^B \right). \tag{11}$$

We define the matchability score of an endpoint as in Equation (8), but with a distinct linear layer. This ensures that the noise from matching less repeatable endpoints does not affect the point matching. The matchability score σ_i^l for line i is then computed as the average of its endpoint matchabilities:

$$\sigma_i^l = \frac{\text{Sigmoid}(\text{Linear}(x_s)) + \text{Sigmoid}(\text{Linear}(x_e))}{2}, \tag{12}$$

where x_s and x_e are the endpoint descriptors. Similar to keypoints, the assignment matrix for lines S_{ij}^l combines line similarity and matchability:

$$S_{ij}^l = \sigma_i^{l,A}\sigma_j^{l,B}\text{Softmax}_{j \in B}(s_{ij}^l)\text{Softmax}_{i \in A}(s_{ij}^l) \tag{13}$$

Match filtering. A pair of points or lines (i, j) is considered a valid correspondence if both points or lines are predicted to be matchable and if their similarity is higher than that of any other point or line in either image. We select pairs where S_{ij} exceeds a threshold τ and is greater than all other elements in its respective row and column.

3.3. Supervision

We supervise our model using ground truth point and line matches derived from homographies or computed based on pixel-wise depth and relative pose, following the ground truth generation method of GlueStick [50]. This results in the following ground truth matches: a set of positive matches, represented by index pairs $\mathcal{M}^p$ for points and $\mathcal{M}^l$ for lines, as well as sets of unmatchable point and line indices for both views, denoted as $\tilde{A}^p$, $\tilde{B}^p$ for points and $\tilde{A}^l$, $\tilde{B}^l$ for lines.

To encourage early prediction of correspondences, the model is supervised after each layer. Dropping the index for points or lines, we define the negative log-likelihood of the predicted assignments for a given feature (points or lines) at layer ℓ:

$$
\begin{aligned}
\mathcal{L}^\ell(\boldsymbol{S}_{ij}, \mathcal{M}, \tilde{A}, \tilde{B}) = & -\frac{1}{|\mathcal{M}|} \sum_{(i,j)\in\mathcal{M}} \log{}^\ell \boldsymbol{S}_{ij} \\
& -\frac{1}{2|\tilde{A}|} \sum_{i\in\tilde{A}} \log\left(1 - {}^\ell \sigma_i^A\right) \\
& -\frac{1}{2|\tilde{B}|} \sum_{j\in\tilde{B}} \log\left(1 - {}^\ell \sigma_j^B\right).
\end{aligned}
\tag{14}
$$

The final loss is an average of the point and line losses at all layers, where L is the total number of layers:

$$
\mathcal{L} = \frac{\sum_\ell \mathcal{L}^\ell(\boldsymbol{S}_{ij}^p, \mathcal{M}^p, \tilde{A}^p, \tilde{B}^p) + \mathcal{L}^\ell(\boldsymbol{S}_{ij}^l, \mathcal{M}^l, \tilde{A}^l, \tilde{B}^l)}{2L}.
\tag{15}
$$

3.4. Adaptive Depth and Width

We adopt adaptivity mechanisms from LightGlue, specifically reducing the number of layers based on the difficulty of image pairs and pruning keypoints that are confidently predicted as unmatchable.

Confidence Classifier: If an image pair is easy (i.e., has large visual overlap and minimal appearance shift), early-layer predictions will closely match those of later layers, so that we can stop the prediction of the network early. After each block ℓ, we estimate the confidence that each node's assignment remains unchanged in the final output:

$$
c_i^\ell = \text{Sigmoid}(\text{MLP}(\boldsymbol{x}_i)).
\tag{16}
$$

Exit criterion: Following LightGlue, we terminate inference early when a sufficient fraction of nodes are confident. A node is considered confident if its predicted confidence exceeds a threshold λ_ℓ at layer ℓ. Assuming N nodes in image A and M nodes in image B, inference is halted if the fraction of confident nodes surpasses a predefined threshold α:

$$
\text{exit} = \left(\frac{1}{N+M} \sum_{I\in\{A,B\}} \sum_{i\in I} \mathbb{I}[c_i^\ell > \lambda_\ell]\right) > \alpha.
\tag{17}
$$

Note that we do not compute confidence scores for lines directly; instead, we rely on keypoint and endpoint confidences to obtain reliable line matches from earlier layers.

Pruning. When testing the point and line pruning proposed in LightGlue [39], we did not observe a significant speedup brought by pruning strategies. In particular for lines, given that images often contain much fewer lines than points, the overhead brought by the line pruning is often surpassing the gains in speedup. Therefore, we did not further explore the pruning of nodes during inference.

4. Experiments

We pre-train LightGlueStick on image pairs obtained by synthetic homography warps, sampled from the 1M distractor images of [53]. Following this, we fine-tune the model on the Megadepth dataset [37], which contains 196 scenes of tourist landmarks. We use gradient checkpointing [11] to fit a batch of 32 image pairs on a single 40Gb Nvidia A100 GPU. The images are resized to 1024x1024 px. We used Adam optimizer with a learning rate of 10^{-4}. Training is conducted on homographies for 40 epochs, followed by fine-tuning on MegaDepth for an additional 50 epochs. The learning rate is exponentially decayed after the 20th and 30th epochs, respectively. The entire training process took approximately 9 days.

Implementation details. LightGlueStick has $L = 9$ blocks of self-attention, Attentional Line Message Passing (ALMP), and cross-attention layers. Each attention unit has 4 heads. The network is trained with 1500 keypoints and 250 lines, with nodes represented by $D = 256$ dimensional features. We train LightGlueStick with Superpoint [13] local features and use LSD [23] to detect lines.

Baselines. We compare LightGlueStick with the SOTA sparse point matcher LightGlue [39]; the dense matchers LoFTR [69] and RoMa [18]; the line matchers: SOLD^2 [49], LineTR [83], and L2D2 [1]; and two methods combining points and lines: PL-Loc [83] and GlueStick [50]. SOLD^2 [49] employs its own line detector, while all other methods including LightGlueStick use LSD [23] for line detection, if not otherwise specified. "LG+Endpoints" refers to matching lines by first matching their endpoints with LightGlue [39], then finding the best association of pairs of endpoints as in [50].

Architecture	Line AP	Time (ms)
GlueStick	70.9	105.6
LightGlueStick (ALMP)	**74.6**	47
↪ a) no LMP	68.4	**39**
↪ b) mean LMP	73.3	54

Table 1. **Ablation study on the ETH3D dataset [63].** We compare our ALMP against no LMP and GlueStick's mean LMP.
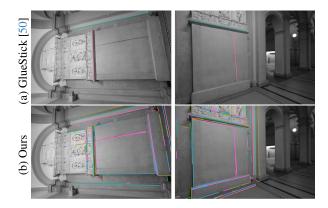
Figure 3. LightGlueStick successfully predicts matches on rotated images, despite not being trained on them, whereas GlueStick [50] fails to predict correct line matches.

4.1. Ablation Study

We evaluate the improvements brought by our contributions on the ETH3D dataset [63]. Since our improvements are not touching point matching, we show only the performance of line matching here. We use the images of the 13 training scenes of the high-resolution multi-view dataset, downsample them by a factor of 8, and sample pairs of images with at least 500 keypoints in common in the official reconstruction. The ground truth line matches are obtained by leveraging the dataset LiDAR depth and poses, using the same methodology as was used to supervise our network. Given predicted line matches, we rank them by matching score, and compute a precision-recall curve. We report the Average Precision (AP) in Tab. 1, computed as the Area Under the Curve (AUC) of the precision-recall curve.

We compare the original GlueStick architecture [50] with our final model (LightGlueStick (ALMP)), and also ablate the latter with either no LMP, or the same LMP as was used in GlueStick (referred to "mean LMP"). Starting from the original GlueStick, we can observe that incorporating the learnings from LightGlue [39] for line matching (rotary encoding, bi-directional attention, flash attention, matchability prediction, shown in line "b) mean LMP") already brings a significant boost of performance and efficiency. Using no LMP incurs however a severe drop of performance. On the contrary, our final model equipped with our proposed ALMP gives the best balance between performance and execution time.

4.2. Line Matching Evaluation on ETH3D

We evaluate the quality of our point and line matches on the ETH3D dataset [63] by computing the Average Precision (AP) of either point or line matches. We also report the average running time per image pair for each method. The results can be found in Tab. 2. LightGlueStick obtains the best performance across the board. One can note in par-

ticular that it surpasses the previous best point-line matcher GlueStick [50] on both points and lines, while being more than twice as fast. Note, that flash attention works better with larger number of keypoints, hence there might not be significant differences when using only points.

We also highlight in Fig. 3 an example where GlueStick fails on a rotated image pair, while LightGlueStick manages to correctly match lines, even though it was not trained on such rotations. We hypothesize that the rotary encoding is a strong asset in this situation because of its translation equivariance that lets the network focus on modeling hard transformations like the rotation in Fig. 3.

	Method	Point AP (↑)	Line AP (↑)	Time (ms) (↓)
Points	LightGlue (LG) [39]	76.4	–	**39**
	GlueStick [50]	**77.0**	–	92
	LightGlueStick	**77.0**	–	44
Points + Lines	L2D2 [1]	–	24.9	546
	LineTR [83]	–	32.7	189
	SOLD2 [49]	–	34.9	388
	LG+Endpoints	76.7	58.6	**38**
	GlueStick [50]	76.5	70.9	106
	LightGlueStick	**78.1**	**74.6**	47

Table 2. **Point and line matching evaluation on the ETH3D dataset [63].** We report the point and line Average Precision (AP) in percentage, as well as the average execution time.

4.3. Homography Estimation on HPatches

HPatches [4] is a standard dataset used to benchmark homography estimation. It consists in 108 sequences displaying either homography viewpoint or illumination changes between image pairs. We match points and/or lines with the matchers to evaluate, use the 4-point homography solver implemented in PoseLib [32] when solving for the homography from points only, and use the same three minimal solvers combining 4 points, 2 points and 2 lines, or 4 lines in a hybrid RANSAC implementation [8, 62] as in [50] for the joint estimation. Using the ground truth homography, one can compute the precision and recall of the matching for both points and lines, as well as evaluate the quality of the retrieved homographies. For this, we compute the AUC of the reprojection error of the four image corners at three error thresholds: 1 / 3 / 5 pixels.

The results can be found in Tab. 3. We compare separately point-only, line-only, and joint point-line methods. On points, the dense matchers [18, 69] obtain the best results, as could be expected from such powerful methods. However, these methods remain costly to use and are significantly slower than lightweight sparse matchers. For line matchers, LightGlueStick obtains the best precision and recall and is on par with GlueStick [50] for homography estimation. The best performance is obtained after combining points and lines in the estimation, as the two features

	Method	AUC (↑)			Points (↑)		Lines (↑)	
		1px	3px	5px	P	R	P	R
P	RoMa	**44.0**	**72.3**	**81.3**	95.8	97.0	–	–
	LoFTR	40.6	69.1	78.5	**98.2**	**99.6**	–	–
	LightGlue	39.0	69.1	78.9	95.7	88.7	–	–
	GlueStick	38.6	69.1	79.0	94.8	89.3	–	–
	LightGlueStick	38.9	69.1	79.0	96.3	87.5	–	–
L	L2D2	22.5	50.8	62.3	–	–	59.4	42.3
	LineTR	15.6	42.2	54.2	–	–	81.1	53.8
	SOLD2	14.3	32.2	41.5	–	–	82.3	77.6
	GlueStick	**22.8**	**52.8**	64.9	–	–	91.0	84.2
	LightGlueStick	22.5	52.8	**65.5**	–	–	**92.2**	**86.8**
P+L	PL-Loc	35.7	63.6	74.1	86.3	68.3	80.9	53.5
	GlueStick	38.5	**69.9**	**79.9**	95.3	**89.4**	90.4	86.7
	LightGlueStick	**39.2**	69.8	79.4	**97.3**	87.2	**92.7**	**87.2**

Table 3. **Homography estimation on HPatches [4].** We report the success rate AUC at 1 / 3 / 5 px thresholds, along with precision (P) and recall (R) for point and line matching.

	Method	Pose Error (↓)	Pose AUC (↑)	Time (ms)
P	RoMa [18]	**6.77**	**34.8 / 54.7 / 65.6**	593
	LoFTR [69]	15.5	18.6 / 33.7 / 43.8	80
	LightGlue [39]	12.3	21.8 / 38.4 / 48.9	**28**
	GlueStick [50]	13.9	19.6 / 35.7 / 45.8	47
	LightGlueStick	**12.0**	**22.5 / 39.1 / 49.3**	**28**
L	L2D2 [1]	63.3	3.8 / 8.8 / 13.7	496
	LineTR [83]	52.6	3.9 / 9.8 / 15.4	84
	SOLD2 [49]	52.6	5.1 / 11.4 / 17.2	454
	GlueStick [50]	**27.0**	9.5 / 19.9 / 29.0	60
	LightGlueStick	27.2	**10.5 / 21.0 / 29.7**	**46**
P + L	PL-Loc [83]	24.2	12.1 / 24.6 / 33.3	169
	GlueStick [50]	12.2	21.7 / 38.8 / 49.3	72
	LightGlueStick	**10.8**	**24.3 / 42.0 / 52.5**	**49**

Table 4. **Relative pose estimation by dominant plane on ScanNet [12].** 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.

can complement each other depending on the scene. There, LightGlueStick scores again similarly or better compared to GlueStick [50], while being faster.

4.4. Dominant Plane Estimation on ScanNet

One of the standard applications combining points and lines is the estimation of a homography between two images. However, as noticed in previous works [50], the existing datasets used to benchmark homography estimation such as HPatches [4] are already saturated, and a comparison on such datasets is moderately meaningful. Thus, we follow the example of these previous works [5, 50], in which an evaluation of homography estimation is proposed, by considering a pair of images, estimating the homography between the two dominant planes from feature matches, and then converting the retrieved homography into a relative pose [46] and comparing it to the ground truth pose. We use the same solvers as in the previous experiment on HPatches to obtain the homography.

We reuse the 1500 image pairs selected in [61], originally taken from the ScanNet dataset [12]. These images come with the ground truth relative pose between the two images, making it possible to compare the homography-based relative pose with the ground truth one. Similarly to previous works [50, 61], we report the pose error consisting of the maximum angular error in translation and rotation, as well as the pose AUC at 10° / 20° / 30° error thresholds. For enhanced performance, the image resolution was reduced by half for LoFTR [69].

The results are displayed in Tab. 4. The dense matcher RoMa [18] is the best, but at a prohibitive running time: it is more than 10 times slower than our method. For all three categories of sparse matched features, points-only, line-

only, and points+lines, LightGlueStick performs the best. One can note that even though the point matching architecture of LightGlueStick remains similar to the one from LightGlue [39], it can slightly improve upon the original LightGlue. The best results overall are obtained when combining points and lines, due to the complementary nature of the features: points excel in textured areas, while lines help on the frequent texture-less areas present in indoor data such as in ScanNet. Notably, LightGlueStick is able to improve the scores of GlueStick [50] by 3 points in AUC.

4.5. Visual Localization

We also evaluate our joint point-line matcher on a visual localization task, where the goal is to find the camera pose of a query image with respect to a pre-built map coming from a set of database images. The 7Scenes dataset [65] is a well-known dataset for this task, consisting of seven indoor scenes with RGB-D images and ground truth poses. We follow the hloc [60, 61] and LIMAP [40] frameworks by detecting SuperPoint feature points [13]; extracting LSD lines [23]; for every query image, retrieve the top 10 closest images in the database with NetVlad [3]; and match them with the different matchers we are evaluating. When evaluating a pure line matcher, we use LightGlue [39] to match the points. Since ground truth depth is available, we use it to back-project lines to 3D on the database images, so that we get 2D-3D matches and can use the P3P solver of PoseLib [14, 32] for points, and the joint point-line solvers of PoseLib [29, 32, 87] to estimate the absolute pose from points and lines. The 7Scenes is however quite saturated already, but the scene stairs remains the most challenging one, especially for point features due to the lack of texture
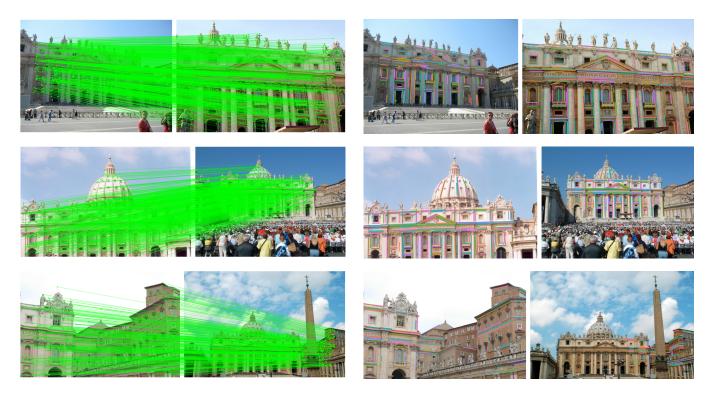
Figure 4. **Depth Adaptivity of LightGlueStick.** LightGlueStick adaptively adjusts its depth based on image difficulty, exiting after the 4th layer for the top pair, the 5th layer for the middle pair, and the 7th layer for the bottom pair. The bottom pair requires more layers due to its smaller visual overlap, making it more challenging to match. Processing times for the top, middle, and bottom pairs are 27ms, 34ms, and 42ms, respectively.

and the repeated pattern of steps. Thus, we only show the results on this scene in Tab. 5.

As shown in the table, point-only methods perform similarly, but adding line features yields a notable boost due to their long range across the image and their presence even in texture-less areas. Among point+line methods, Light-GlueStick matches the previous best, GlueStick [50], but is nearly four times faster: GlueStick processes 7.0 pairs per second, while LightGlueStick manages to process 26.9 pairs per second.

	Method	T / R err.	Acc.
Points	LightGlue [39]	4.3 / 1.14	57.9
	GlueStick [50]	**4.1 / 1.08**	**60.3**
	LightGlueStick	**4.1** / 1.13	59.8
Points + Lines	SOLD2 [49]	3.1 / 0.82	78.1
	LineTR [83]	3.1 / 0.81	76.8
	L2D2 [1]	2.9 / 0.77	78.4
	GlueStick [50]	**2.7 / 0.73**	78.8
	LightGlueStick	**2.7** / 0.74	**79.1**

Table 5. **Visual localization on scene stairs of the 7Scenes dataset [65].** We report the median translation / rotation errors (cm / deg) and pose accuracy at 5cm / 5° threshold.

5. Conclusion

In this work, we introduced a deep network for joint point and line matching that operates at near real-time speeds (20 FPS). Introducing novel architecture changes, we in particular explicitly encoded the line connectivity into the network, and let the network reason about non-repeatable features through a novel Attention Line Message Passing. We demonstrate that our method maintains or even improves upon the performance of state-of-the-art sparse matchers, while also being among the fastest point matchers available. Furthermore, our approach surpasses the fastest line matchers by a significant margin, setting a new benchmark for speed and accuracy. Additionally, by incorporating the adaptivity mechanism, we enable the network to predict line matches earlier, further reducing the runtime of our matcher, with only a marginal trade-off in performance. By enabling fast and accurate feature matching, our method unlocks the potential for real-time point-line feature fusion in embedded systems, such as robotics and mobile devices.

References

[1] Hichem Abdellali, Robert Frohlich, Viktor Vilagos, and Zoltan Kato. L2D2: learnable line detector and descriptor. In *3DV*, 2021. 2, 5, 6, 7, 8

[2] Sérgio Agostinho, João Gomes, and Alessio Del Bue. Cvx-pnpl: A unified convex solution to the absolute pose estimation problem from point and line correspondences, 2019. 3

[3] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *CVPR*, 2016. 7

[4] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. HPatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *CVPR*, 2017. 6, 7

[5] Dániel Baráth, Dmytro Mishkin, Michal Polic, Wolfgang Förstner, and Jiri Matas. A large scale homography benchmark. In *CVPR*, 2023. 7

[6] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *CVIU*, 2008. 2

[7] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Process. Mag.*, 34:18–42, 2016. 3

[8] Federico Camposeco, Andrea Cohen, Marc Pollefeys, and Torsten Sattler. Hybrid camera pose estimation. In *CVPR*, 2018. 6

[9] 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 *ICCV*, 2021. 2

[10] 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 *ECCV*, 2022. 2

[11] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost, 2016. 5

[12] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *CVPR*, 2017. 7

[13] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPRW*, 2018. 2, 5, 7

[14] Yaqing Ding, Jian Yang, Viktor Larsson, Carl Olsson, and Kalle Åström. Revisiting the p3p problem. In *CVPR*, 2023. 7

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

[16] Johan Edstedt, Ioannis Athanasiadis, Mårten Wadenbäck, and Michael Felsberg. DKM: Dense kernelized feature matching for geometry estimation. In *CVPR*, 2023. 1, 2

[17] Johan Edstedt, Georg Bökman, Mårten Wadenbäck, and Michael Felsberg. DeDoDe: Detect, Don't Describe — Describe, Don't Detect for Local Feature Matching. In *3DV*, 2024. 2

[18] Johan Edstedt, Qiyu Sun, Georg Bökman, Mårten Wadenbäck, and Michael Felsberg. RoMa: Robust Dense Feature Matching. In *CVPR*, 2024. 1, 2, 5, 6, 7

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

[20] Qiang Fu, Jialong Wang, Hongshan Yu, Islam Ali, Feng Guo, Yijia He, and Hong Zhang. Pl-vins: Real-time monocular visual-inertial slam with point and line features, 2020. 3

[21] Pierre Gleize, Weiyao Wang, and Matt Feiszli. Silk – simple learned keypoints. In *ICCV*, 2023. 2

[22] Ruben Gomez-Ojeda, Francisco-Angel Moreno, David Zuniga-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. 3

[23] Rafael Grompone von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. LSD: A fast line segment detector with a false detection control. *IEEE TPAMI*, 32(4):722–732, 2010. 5, 7

[24] 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

[25] Xingyi He, Hao Yu, Sida Peng, Dongli Tan, Zehong Shen, Hujun Bao, and Xiaowei Zhou. Matchanything: Universal cross-modality image matching with large-scale pre-training, 2025. 2

[26] Petr Hrubý, Shaohui Liu, Rémi Pautrat, Marc Pollefeys, and Dániel Baráth. Handbook on leveraging lines for two-view relative pose estimation. In *3DV*, 2023. 3

[27] Hanwen Jiang, Arjun Karpur, Bingyi Cao, Qixing Huang, and Andre Araujo. Omniglue: Generalizable feature matching with foundation model guidance. In *CVPR*, 2024. 2

[28] Wei Jiang, Eduard Trulls, Jan Hosang, Andrea Tagliasacchi, and Kwang Moo Yi. COTR: Correspondence Transformer for Matching Across Images. In *ICCV*, 2021. 1, 2

[29] Zuzana Kukelova, Jan Heller, and Andrew Fitzgibbon. Efficient intersection of three quadrics and applications in computer vision. In *CVPR*, 2016. 7

[30] Manuel Lange, Fabian Schweinfurth, and Andreas Schilling. DLD: A deep learning based line descriptor for line feature matching. In *IROS*, 2019. 2

[31] Manuel Lange, Claudio Raisch, and Andreas Schilling. WLD: A Wavelet and Learning based Line Descriptor for Line Feature Matching. In *Vision, Modeling, and Visualization*. The Eurographics Association, 2020. 2

[32] Viktor Larsson. PoseLib - Minimal Solvers for Camera Pose Estimation. https://github.com/PoseLib/PoseLib. 6, 7

[33] Vincent Leroy, Yohann Cabon, and Jerome Revaud. Grounding image matching in 3d with mast3r, 2024. 2

[34] 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

[35] Wanting Li, Yuanbin Shao, Yongcai Wang, Shuo Wang, Xuewei Bai, and Deying Li. Idls: Inverse depth line based visual-inertial slam, 2023. 3

[36] Yang Li, Si Si, Gang Li, Cho-Jui Hsieh, and Samy Bengio. Learnable Fourier Features for Multi-dimensional Spatial Positional Encoding. In *NeurIPS*, 2021. 4

[37] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *CVPR*, 2018. 5

[38] Hyunjun Lim, Jinwoo Jeon, and Hyun Myung. Uv-slam: Unconstrained line-based slam using vanishing points for structural mapping. *IEEE Robotics and Automation Letters*, 7(2):1518–1525, 2022. 3

[39] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. LightGlue: Local Feature Matching at Light Speed. In *ICCV*, 2023. 1, 2, 3, 5, 6, 7, 8

[40] Shaohui Liu, Yifan Yu, Rémi Pautrat, Marc Pollefeys, and Viktor Larsson. 3d line mapping revisited. In *CVPR*, 2023. 3, 7

[41] Shaohui Liu, Yidan Gao, Tianyi Zhang, Rémi Pautrat, Johannes L. Schönberger, Viktor Larsson, and Marc Pollefeys. Robust incremental structure-from-motion with hybrid features. In *ECCV*, 2024. 3

[42] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 2

[43] Zixin Luo, Tianwei Shen, Lei Zhou, Jiahui Zhang, Yao Yao, Shiwei Li, Tian Fang, and Long Quan. Contextdesc: Local descriptor augmentation with cross-modality context. In *CVPR*, 2019. 2

[44] Jiayi Ma, Ji Zhao, Junjun Jiang, Huabing Zhou, and Xiaojie Guo. Locality preserving matching. *IJCV*, 127:512–531, 2019. 2

[45] 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

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

[47] Anastasiya Mishchuk, Dmytro Mishkin, Filip Radenović, and Jiri Matas. Working hard to know your neighbor's margins:local descriptor learning loss. In *NeurIPS*, 2017. 2

[48] Yuki Ono, Eduard Trulls, Pascal Fua, and Kwang Moo Yi. Lf-net: Learning local features from images. In *NeurIPS*, 2018. 2

[49] Rémi Pautrat, Juan-Ting Lin, Viktor Larsson, Martin R. Oswald, and Marc Pollefeys. SOLD^2: Self-supervised occlusion-aware line description and detection. In *CVPR*, 2021. 2, 5, 6, 7, 8

[50] Rémi* Pautrat, Iago* Suárez, Yifan Yu, Marc Pollefeys, and Viktor Larsson. GlueStick: Robust image matching by sticking points and lines together. In *ICCV*, 2023. 2, 3, 5, 6, 7, 8

[51] Guilherme Potje, Felipe Cadar, André Araujo, Renato Martins, and Erickson R. Nascimento. Xfeat: Accelerated features for lightweight image matching. In *CVPR*, 2024. 2

[52] Albert Pumarola, Alexander Vakhitov, Antonio Agudo, Alberto Sanfeliu, and Francese Moreno-Noguer. Pl-slam: Real-time monocular visual slam with points and lines. In *ICRA*, 2017. 3

[53] F. Radenović, A. Iscen, G. Tolias, Y. Avrithis, and O. Chum. Revisiting oxford and paris: Large-scale image retrieval benchmarking. In *CVPR*, 2018. 5

[54] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *NeurIPS*, 20, 2007. 4

[55] Srikumar Ramalingam, Sofien Bouaziz, and Peter Sturm. Pose estimation using both points and lines for geo-localization. In *ICRA*, 2011. 3

[56] Srikumar Ramalingam, Michel Antunes, Dan Snow, Gim Hee Lee, and Sudeep Pillai. Line-sweep: Cross-ratio for wide-baseline matching and 3D reconstruction. In *CVPR*, 2015. 2

[57] Guangli Ren, Zhiqiang Cao, Xilong Liu, Min Tan, and Junzhi Yu. Plj-slam: Monocular visual slam with points, lines, and junctions of coplanar lines. *IEEE Sensors Journal*, 22 (15), 2022. 3

[58] Jerome Revaud, Philippe Weinzaepfel, César Roberto de Souza, and Martin Humenberger. R2D2: repeatable and reliable detector and descriptor. In *NeurIPS*, 2019. 2

[59] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *ICCV*, 2011. 2

[60] Paul-Edouard Sarlin. Visual localization made easy with hloc. https://github.com/cvg/Hierarchical-Localization. 7

[61] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 2, 7

[62] Torsten Sattler et al. RansacLib - A Template-based *SAC Implementation. https://github.com/tsattler/RansacLib. 6

[63] 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 *CVPR*, 2017. 5, 6

[64] Yanxing Shi, Junxiong 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 *CVPR*, 2022. 2

[65] 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 *CVPR*, 2013. 7, 8

[66] Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021. Sections 3, 4, 13. 2, 3, 4

[67] Iago Suárez, Enrique Muñoz, José M Buenaposada, and Luis Baumela. Fsg: A statistical approach to line detection via fast segments grouping. In *IROS*, pages 97–102, 2018. 3

[68] Iago Suárez, José M Buenaposada, and Luis Baumela. Revisiting binary local image description for resource limited devices. *IEEE Robotics and Automation Letters*, 6(4):8317–8324, 2021. 2

[69] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. In *CVPR*, 2021. 1, 2, 5, 6, 7

[70] Yurun Tian, Bin Fan, and Fuchao Wu. L2-net: Deep learning of discriminative patch descriptor in euclidean space. In *CVPR*, 2017. 2

[71] Yurun Tian, Xin Yu, Bin Fan, Fuchao Wu, Huub Heijnen, and Vassileios Balntas. Sosnet: Second order similarity regularization for local descriptor learning. In *CVPR*, 2019. 2

[72] Prune Truong, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning accurate dense correspondences and when to trust them. In *CVPR*, 2021. 2

[73] Michał Tyszkiewicz, Pascal Fua, and Eduard Trulls. Disk: Learning local features with policy gradient. In *NeurIPS*, 2020. 2

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

[75] Alexander Vakhitov, Jan Funke, and Francesc Moreno-Noguer. Accurate and linear time pose estimation from points and lines. In *ECCV*, 2016. 3

[76] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 2, 3

[77] Bart Verhagen, Radu Timofte, and Luc Van Gool. Scale-invariant line descriptors for wide baseline matching. In *WACV*, 2014. 2

[78] Lu Wang, Ulrich Neumann, and Suya You. Wide-baseline image matching using line signatures. In *ICCV*. IEEE, 2009. 2

[79] Qing Wang, Jiaming Zhang, Kailun Yang, Kunyu Peng, and Rainer Stiefelhagen. Matchformer: Interleaving attention in transformers for feature matching. In *ACCV*, 2022. 2

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

[81] Lei Xu, Hesheng Yin, Tong Shi, Di Jiang, and Bo Huang. Eplf-vins: Real-time monocular visual-inertial slam with efficient point-line flow features. *IEEE Robotics and Automation Letters*, 8(2), 2023. 3

[82] Kwang Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In *ECCV*, 2016. 2

[83] Sungho Yoon and Ayoung Kim. Line as a visual sentence: Context-aware line descriptor for visual localization. *IEEE Robotics and Automation Letters*, 6(4):8726–8733, 2021. 2, 5, 6, 7, 8

[84] 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

[85] Xiaoming Zhao, Xingming Wu, Jinyu Miao, Weihai Chen, Peter C. Y. Chen, and Zhengguo Li. Alike: Accurate and lightweight keypoint detection and descriptor extraction. In *IEEE TMM*, 2022. 2

[86] Xiaoming Zhao, Xingming Wu, Weihai Chen, Peter C. Y. Chen, Qingsong Xu, and Zhengguo Li. Aliked: A lighter keypoint and descriptor extraction network via deformable transformation. In *TIM*, 2023. 2

[87] Lipu Zhou, Jiamin Ye, and Michael Kaess. A stable algebraic camera pose estimation for minimal configurations of 2D/3D point and line correspondences. In *ACCV*, 2018. 7

[88] Lipu Zhou, Jiamin Ye, and Michael Kaess. A stable algebraic camera pose estimation for minimal configurations of 2d/3d point and line correspondences. In *ACCV*, 2018. 3

[89] Lipu Zhou, Shengze Wang, and Michael Kaess. A fast and accurate solution for pose estimation from 3d correspondences. In *ICRA*, 2020. 3

[90] Xingxing Zuo, Xiaojia Xie, Yong Liu, and Guoquan Huang. Robust visual slam with point and line features. In *IROS*, 2017. 3