

# LightGlue: Local Feature Matching at Light Speed

Philipp Lindenberger<sup>1</sup> Paul-Edouard Sarlin<sup>1</sup> Marc Pollefeys<sup>1,2</sup>

<sup>1</sup> ETH Zurich <sup>2</sup> Microsoft Mixed Reality & AI Lab

## Abstract

We introduce LightGlue, a deep neural network that learns to match local features across images. We revisit multiple design decisions of SuperGlue, the state of the art in sparse matching, and derive simple but effective improvements. Cumulatively, they make LightGlue more efficient – in terms of both memory and computation, more accurate, and much easier to train. One key property is that LightGlue is adaptive to the difficulty of the problem: the inference is much faster on image pairs that are intuitively easy to match, for example because of a larger visual overlap or limited appearance change. This opens up exciting prospects for deploying deep matchers in latency-sensitive applications like 3D reconstruction. The code and trained models are publicly available at [github.com/cvg/LightGlue](https://github.com/cvg/LightGlue).

## 1. Introduction

Finding correspondences between two images is a fundamental building block of many computer vision applications like camera tracking and 3D mapping. The most common approach to image matching relies on sparse interest points that are matched using high-dimensional representations encoding their local visual appearance. Reliably describing each point is challenging in conditions that exhibit symmetries, weak texture, or appearance changes due to varying viewpoint and lighting. To reject outliers that arise from occlusion and missing points, such representations should also be discriminative. This yields two conflicting objectives, robustness and uniqueness, that are hard to satisfy.

To address these limitations, SuperGlue [56] introduced a new paradigm – a deep network that considers both images at the same time to jointly match sparse points and reject outliers. It leverages the powerful Transformer model [74] to learn to match challenging image pairs from large datasets. This yields robust image matching in both indoor and outdoor environments. SuperGlue is highly effective for visual localization in challenging conditions [59, 55, 58, 57] and generalizes well to other tasks like aerial matching [83], object pose estimation [69], and even fish re-identification [47].

These improvements are however computationally ex-

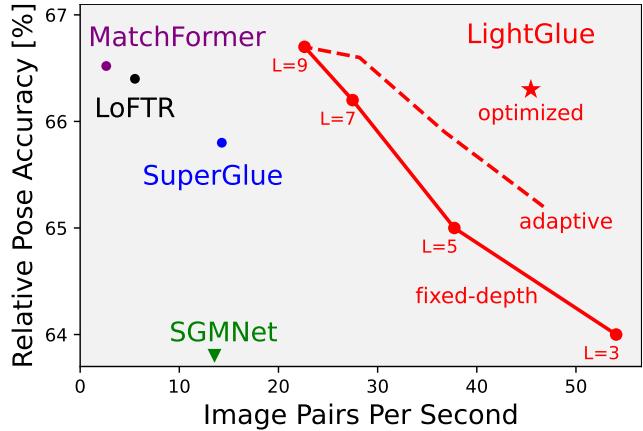


Figure 1. **LightGlue matches sparse features faster and better** than existing approaches like SuperGlue. Its adaptive stopping mechanism gives a fine-grained control over the speed vs. accuracy trade-off. Our final, optimized model  $\star$  delivers an accuracy closer to the dense matcher LoFTR at an  $8\times$  higher speed, here in typical outdoor conditions.

pensive, while the efficiency of image matching is critical for tasks that require a low latency, like tracking, or a high processing volume, like large-scale mapping. Additionally, SuperGlue, as with other Transformer-based models, is notoriously hard to train, requiring computing resources that are inaccessible to many practitioners. Follow-up works [8, 65] have thus failed to reach the performance of the original SuperGlue model. Yet, since its initial publication, Transformers have been extensively studied, improved, and applied to numerous language [17, 51, 13] and vision [18, 6, 29] tasks.

In this paper, we draw on these insights to design LightGlue, a deep network that is more accurate, more efficient, and easier to train than SuperGlue. We revisit its design decisions and combine numerous simple, yet effective, architecture modifications. We distill a recipe to train high-performance deep matchers with limited resources, reaching state-of-the-art accuracy within just a few GPU-days. As shown in Figure 1, LightGlue is Pareto-optimal on the efficiency-accuracy trade-off when compared to existing sparse and dense matchers.

Unlike previous approaches, LightGlue is adaptive to the difficulty of each image pair, which varies based on the

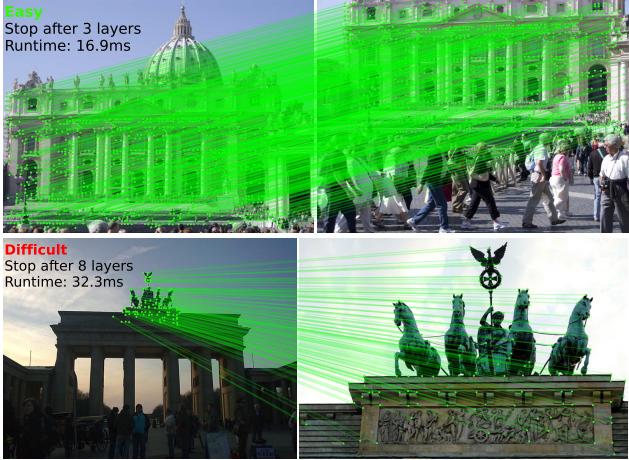


Figure 2. **Depth adaptivity.** LightGlue is faster at matching easy image pairs (top) than difficult ones (bottom) because it can stop at earlier layers when its predictions are confident.

amount of visual overlap, appearance changes, or discriminative information. Figure 2 shows that the inference is thus much faster on pairs that are intuitively easy to match than on challenging ones, a behavior that is reminiscent of how humans process visual information. This is achieved by 1) predicting a set of correspondences after each computational blocks, and 2) enabling the model to introspect them and predict whether further computation is required. LightGlue also discards at an early stage points that are not matchable, thus focusing its attention on the covisible area.

Our experiments show that LightGlue is a plug-and-play replacement to SuperGlue: it predicts strong matches from two sets of local features, at a fraction of the run time. This opens up exciting prospects for deploying deep matchers in latency-sensitive applications like SLAM [45, 5] or reconstructing larger scenes from crowd-sourced data [25, 60, 39, 57]. The LightGlue model and its training code will be released publicly with a permissive license.

## 2. Related work

**Matching images** that depict the same scene or object typically relies on local features, which are sparse keypoints each associated with a descriptor of its local appearance. While classical algorithms rely on hand-crafted criteria and gradient statistics [41, 23, 4, 53], much of the recent research has focused on designing Convolutional Neural Networks (CNNs) for both detection [81, 16, 19, 52, 73] and description [42, 72]. Trained with challenging data, CNNs largely improve the accuracy and robustness of matching. Local features now come in many flavors: some are better localized [41], highly repeatable [16], cheap to store and match [54], invariant to specific changes [46], or ignore unreliable objects [73].

Local features are then matched with a nearest neighbor search in descriptor space. Because of non-matchable key-

points and imperfect descriptors, some correspondences are incorrect. Those are filtered out by heuristics, like Lowe’s ratio test [41] or the mutual check, inlier classifiers [44, 82], and by robustly fitting geometric models [22, 7]. This process requires extensive domain expertise and tuning and is prone to failure when conditions are too challenging. These limitations are largely solved by deep matchers.

**Deep matchers** are deep networks trained to jointly match local features and reject outliers given an input image pair. The first of its kind, SuperGlue [56] combines the expressive representations of Transformers [74] with optimal transport [48] to solve a partial assignment problem. It learns powerful priors about scene geometry and camera motion and is thus robust to extreme changes and generalizes well across data domains. Inheriting the limitations of early Transformers, SuperGlue is hard to train and its complexity grows quadratically with the number of keypoints.

Subsequent works make it more efficient by reducing the size of the attention mechanism. They restrict it to a small set of seed matches [8] or within clusters of similar keypoints [65]. This largely reduces the run time for large numbers of keypoints but yields no gains for smaller, standard input sizes. This also impairs the robustness in the most challenging conditions, failing to reach the performance of the original SuperGlue model. LightGlue instead brings large improvements for typical operating conditions, like in SLAM, without compromising on performance for any level of difficulty. This is achieved by dynamically adapting the network size instead of reducing its overall capacity.

Conversely, dense matchers like LoFTR [68] and follow-ups [9, 78] match points distributed on dense grids rather than sparse locations. This boosts the robustness to impressive levels but is generally much slower because it processes many more elements. This limits the resolution of the input images and, in turn, the spatial accuracy of the correspondences. While LightGlue operates on sparse inputs, we show that fair tuning and evaluation makes it competitive with dense matchers, for a fraction of the run time.

**Making Transformers efficient** has received significant attention following their success in language processing. As the memory footprint of attention is a major limitation to handling long sequences, many works reduce it using linear formulations [79, 32, 33] or bottleneck latent tokens [35, 30]. This enables long-range context but can impair the performance for small input sizes. Selective checkpointing [49] reduces the memory footprint of attention and optimizing the memory access also drastically speeds it up [14].

Other, orthogonal works instead adaptively modulate the network depth by predicting whether the prediction of a token at a given layer is final or requires further computations [15, 20, 62]. This is mostly inspired by adaptive schemes developed for CNNs by the vision commu-

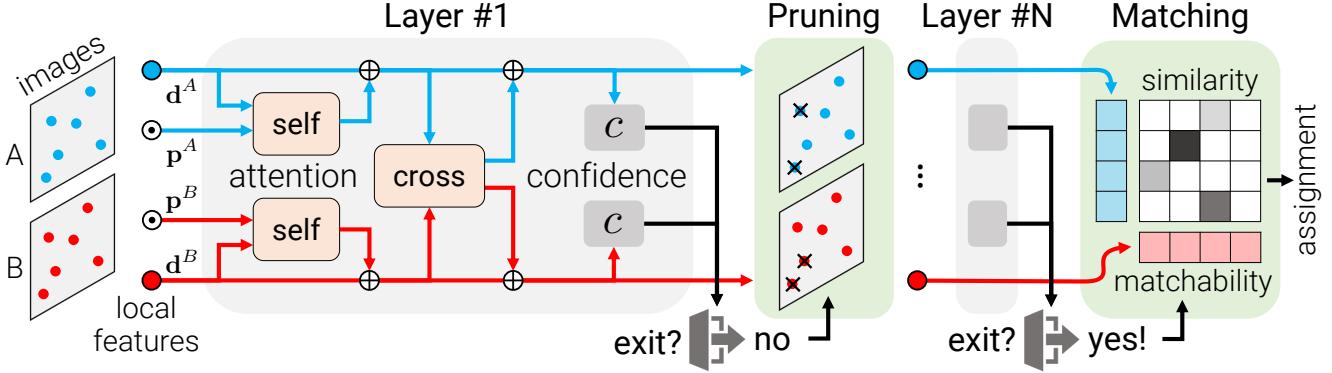


Figure 3. **The LightGlue architecture.** Given a pair of input local features  $(\mathbf{d}, \mathbf{p})$ , each layer augments the visual descriptors ( $\bullet, \circ$ ) with context based on self- and cross-attention units with positional encoding  $\odot$ . A confidence classifier  $c$  helps decide whether to stop the inference. If few points are confident, the inference proceeds to the next layer but we prune points that are confidently unmatched. Once a confident state is reached, LightGlue predicts an assignment between points based on their pairwise similarity and unary matchability.

nity [71, 80, 40, 21, 36, 76]. In Transformers, the type of positional encoding has a large impact on the accuracy. While absolute sinusoidal [74] or learned encodings [17, 51] were initially prevalent, recent works have studied relative encodings [63, 67] to stabilize the training and better capture long-range dependencies.

LightGlue adapts some of these innovations to 2D feature matching and shows gains in both efficiency and accuracy.

### 3. Fast feature matching

**Problem formulation:** LightGlue predicts a partial assignment between two sets of local features extracted from images  $A$  and  $B$ , following SuperGlue. Each local feature  $i$  is composed of a 2D point position  $\mathbf{p}_i := (x, y)_i \in [0, 1]^2$ , normalized by the image size, and a visual descriptor  $\mathbf{d}_i \in \mathbb{R}^d$ . Images  $A$  and  $B$  have  $M$  and  $N$  local features, indexed by  $\mathcal{A} := \{1, \dots, M\}$  and  $\mathcal{B} := \{1, \dots, N\}$ , respectively.

We design LightGlue to output a set of correspondences  $\mathcal{M} = \{(i, j)\} \subset \mathcal{A} \times \mathcal{B}$ . Each point is matchable at least once, as it stems from a unique 3D point, and some keypoints are unmatchable, due to occlusion or non-repeatability. As in previous works, we thus seek a soft partial assignment matrix  $\mathbf{P} \in [0, 1]^{M \times N}$  between local features in  $A$  and  $B$ , from which we can extract correspondences.

**Overview – Figure 3:** LightGlue is made of a stack of  $L$  identical layers that process the two sets jointly. Each layer is composed of self- and cross-attention units that update the representation of each point. A classifier then decides, at each layer, whether to halt the inference, thus avoiding unnecessary computations. A lightweight head finally computes a partial assignment from the set of representations.

#### 3.1. Transformer backbone

We associate each local feature  $i$  in image  $I \in \{A, B\}$  with a state  $\mathbf{x}_i^I \in \mathbb{R}^d$ . The state is initialized with the cor-

responding visual descriptor  $\mathbf{x}_i^I \leftarrow \mathbf{d}_i^I$  and subsequently updated by each layer. We define a layer as a succession of one self-attention and one cross-attention units.

**Attention unit:** In each unit, a Multi-Layer Perceptron (MLP) updates the state given a message  $\mathbf{m}_i^{I \leftarrow S}$  aggregated from a source image  $S \in \{A, B\}$ :

$$\mathbf{x}_i^I \leftarrow \mathbf{x}_i^I + \text{MLP}([\mathbf{x}_i^I | \mathbf{m}_i^{I \leftarrow S}]) , \quad (1)$$

where  $[\cdot | \cdot]$  stacks two vectors. This is computed for all points in both images in parallel. In a self-attention unit, each image  $I$  pulls information from points of the same image and thus  $S = I$ . In a cross-attention unit, each image pulls information from the other image and  $S = \{A, B\} \setminus I$ .

The message is computed by an attention mechanism as the weighted average of all states  $j$  of image  $S$ :

$$\mathbf{m}_i^{I \leftarrow S} = \sum_{k \in S} \text{Softmax}(a_{ik}^{IS})_j \mathbf{W} \mathbf{x}_j^S , \quad (2)$$

where  $\mathbf{W}$  is a projection matrix and  $a_{ij}^{IS}$  is an attention score between points  $i$  and  $j$  of images  $I$  and  $S$ . How this score is computed differs for self- and cross-attention units.

**Self-attention:** Each point attends to all points of the same image. We perform the same following steps for each image  $I$  and thus drop the superscript  $I$  for clarity. For each point  $i$ , the current state  $\mathbf{x}_i$  is first decomposed into key and query vectors  $\mathbf{k}_i$  and  $\mathbf{q}_i$  via *different* linear transformations. We then define the attention score between points  $i$  and  $j$  as

$$a_{ij} = \mathbf{q}_i^\top \mathbf{R}(\mathbf{p}_j - \mathbf{p}_i) \mathbf{k}_j , \quad (3)$$

where  $\mathbf{R}(\cdot) \in \mathbb{R}^{d \times d}$  is a rotary encoding [67] of the relative position between the points. We partition the space into  $d/2$  2D subspaces and rotate each of them by an angle corresponding, following Fourier Features [37], to the projection

onto a learned basis  $\mathbf{b}_k \in \mathbb{R}^2$ :

$$\mathbf{R}(\mathbf{p}) = \begin{pmatrix} \hat{\mathbf{R}}(\mathbf{b}_1^\top \mathbf{p}) & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \hat{\mathbf{R}}(\mathbf{b}_{d/2}^\top \mathbf{p}) \end{pmatrix}, \hat{\mathbf{R}}(\theta) = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}. \quad (4)$$

Positional encoding is a critical part of attention as it allows addressing different elements based on their position. We note that, in projective camera geometry, the position of visual observations is equivariant w.r.t. a translation of the camera within the image plane: 2D points that stem from 3D points on the same fronto-parallel plane are translated in an identical way and their relative distance remains constant. This calls for an encoding that only captures the relative but not the absolute position of points.

The rotary encoding [67] enables the model to retrieve points  $j$  that are located at a learned relative position from  $i$ . The positional encoding is not applied to the value  $\mathbf{v}_j$  and thus does not spill into the state  $\mathbf{x}_i$ . The encoding is identical for all layers and is thus computed once and cached.

**Cross-attention:** Each point in  $I$  attends to all points of the other image  $S$ . We compute a key  $\mathbf{k}_i$  for each element but no query. This allows to express the score as

$$a_{ij}^{IS} = \mathbf{k}_i^{I^\top} \mathbf{k}_j^S \stackrel{!}{=} a_{ji}^{SI}. \quad (5)$$

We thus need to compute the similarity only once for both  $I \leftarrow S$  and  $S \leftarrow I$  messages. This trick has been previously referred to as *bidirectional* attention [77]. Since this step is expensive, with a complexity of  $O(NMd)$ , it saves a significant factor of 2. We do not add any positional information as relative positions are not meaningful across images.

### 3.2. Correspondence prediction

We design a lightweight head that predicts an assignment given the updated state at any layer.

**Assignment scores:** We first compute a pairwise score matrix  $\mathbf{S} \in \mathbb{R}^{M \times N}$  between the points of both images:

$$\mathbf{S}_{ij} = \text{Linear}(\mathbf{x}_i^A)^\top \text{Linear}(\mathbf{x}_j^B) \quad \forall (i, j) \in \mathcal{A} \times \mathcal{B}, \quad (6)$$

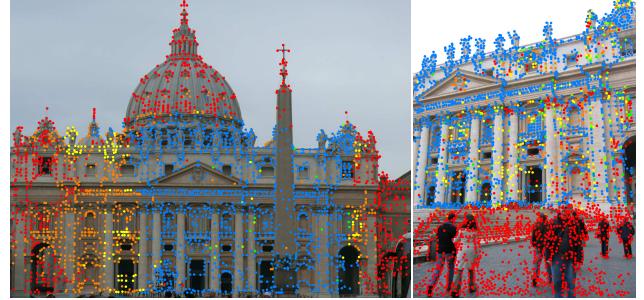
where  $\text{Linear}(\cdot)$  is a learned linear transformation with bias. This score encodes the affinity of each pair of points to be in correspondence, *i.e.* 2D projections of the same 3D point. We also compute, for each point, a matchability score as

$$\sigma_i = \text{Sigmoid}(\text{Linear}(\mathbf{x}_i)) \in [0, 1]. \quad (7)$$

This score encodes the likelihood of  $i$  to have a corresponding point. A point that is not detected in the other image, *e.g.* when occluded, is not matchable and thus has  $\sigma_i \rightarrow 0$ .

**Correspondences:** We combine both similarity and matchability scores into a soft partial assignment matrix  $\mathbf{P}$  as

$$\mathbf{P}_{ij} = \sigma_i^A \sigma_j^B \underset{k \in \mathcal{A}}{\text{Softmax}(\mathbf{S}_{kj})_i} \underset{k \in \mathcal{B}}{\text{Softmax}(\mathbf{S}_{ik})_j}. \quad (8)$$



**Figure 4. Point pruning.** As LightGlue aggregates context, it can find out early that some points (●) are unmatchable and thus exclude them from subsequent layers. Other, non-repeatable points are excluded in later layers: ● → ○ → ●. This reduces the inference time and the search space (●) to ultimately find good matches fast.

A pair of points  $(i, j)$  yields a correspondence when both points are predicted as matchable and when their similarity is higher than any other point in both images. We select pairs for which  $\mathbf{P}_{ij}$  is larger than a threshold  $\tau$  and than any other element along both its row and column.

### 3.3. Adaptive depth and width

We add two mechanisms that avoid unnecessary computations and save inference time: i) we reduce the number of layers depending on the difficulty of the input image pair; ii) we prune out points that are confidently rejected early.

**Confidence classifier:** The backbone of LightGlue augments input visual descriptors with context. These are often reliable if the image pair is easy, *i.e.* has high visual overlap and little appearance changes. In such case, predictions from early layers are confident and identical to those of late layers. We can then output these predictions and halt the inference.

At the end of each layer, LightGlue infers the confidence of the predicted assignment of each point:

$$c_i = \text{Sigmoid}(\text{MLP}(\mathbf{x}_i)) \in [0, 1]. \quad (9)$$

A higher value indicates that the representation of  $i$  is reliable and final – it is confidently either matched or unmatchable. This is inspired by multiple works that successfully apply this strategy to language and vision tasks [62, 20, 71, 80, 40]. The compact MLP adds only 2% of inference time in the worst case but most often saves much more.

**Exit criterion:** For a given layer  $\ell$ , a point is deemed confident if  $c_i > \lambda_\ell$ . We halt the inference if a sufficient ratio  $\alpha$  of all points is confident:

$$\text{exit} = \left( \frac{1}{N+M} \sum_{I \in \{A, B\}} \sum_{i \in \mathcal{I}} [\![c_i^I > \lambda_\ell]\!] \right) > \alpha. \quad (10)$$

We observe, as in [62], that the classifier itself is less confident in early layers. We thus decay  $\lambda_\ell$  throughout the layers based on the validation accuracy of each classifier. The exit

threshold  $\alpha$  directly controls the trade-off between accuracy and inference time.

**Point pruning:** When the exit criterion is not met, points that are predicted as both confident and unmatchable are unlikely to aid the matching of other points in subsequent layers. Such points are for example in areas that are clearly not covisible across the images. We therefore discard them at each layer and feed only the remaining points to the next one. This significantly reduces computation, given the quadratic complexity of attention, and does not impact the accuracy.

### 3.4. Supervision

We train LightGlue in two stages: we first train it to predict correspondences and only after train the confidence classifier. The latter thus does not impact the accuracy at the final layer or the convergence of the training.

**Correspondences:** We supervise the assignment matrix  $\mathbf{P}$  with ground truth labels estimated from two-view transformations. Given a homography or pixel-wise depth and a relative pose, we wrap points from  $A$  to  $B$  and conversely. Ground truth matches  $\mathcal{M}$  are pairs of points with a low re-projection error in both images and a consistent depth. Some points  $\bar{\mathcal{A}} \subseteq \mathcal{A}$  and  $\bar{\mathcal{B}} \subseteq \mathcal{B}$  are labeled as unmatchable when their reprojection or depth errors are sufficiently large with all other points. We then minimize the log-likelihood of the assignment predicted at each layer  $\ell$ , pushing LightGlue to predict correct correspondences early:

$$\text{loss} = -\frac{1}{L} \sum_{\ell} \left( \frac{1}{|\mathcal{M}|} \sum_{(i,j) \in \mathcal{M}} \log {}^{\ell}\mathbf{P}_{ij} + \frac{1}{2|\bar{\mathcal{A}}|} \sum_{i \in \bar{\mathcal{A}}} \log (1 - {}^{\ell}\sigma_i^A) + \frac{1}{2|\bar{\mathcal{B}}|} \sum_{j \in \bar{\mathcal{B}}} \log (1 - {}^{\ell}\sigma_j^B) \right). \quad (11)$$

The loss is balanced between positive and negative labels.

**Confidence classifier:** We then train the MLP of Eq. (9) to predict whether the prediction of each layer is identical to the final one. Let  ${}^{\ell}m_i^A \in \mathcal{B} \cup \{\bullet\}$  be the index of the point in  $B$  matched to  $i$  at layer  $\ell$ , with  ${}^{\ell}m_i^A = \bullet$  if  $i$  is unmatchable. The ground truth binary label of each point is  $[{}^{\ell}m_i^A = {}^Lm_i^A]$  and identically for  $B$ . We then minimize the binary cross-entropy of the classifiers of layers  $\ell \in \{1, \dots, L-1\}$ .

### 3.5. Comparison with SuperGlue

LightGlue is inspired by SuperGlue but differs in aspects critical to its accuracy, efficiency, and ease of training.

**Positional encoding:** SuperGlue encodes the absolute point positions with an MLP and fuses them early with the descriptors. We observed that the model tends to forget this

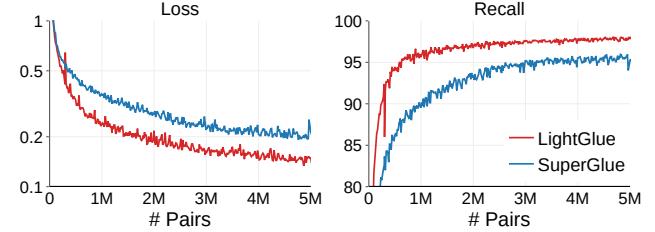


Figure 5. **Ease of training.** The LightGlue architecture vastly improves the speed of convergence of the pre-training on synthetic homographies. After 5M image pairs (only 2 GPU-days), LightGlue achieves -33% loss at the final layer and +4% match recall. SuperGlue requires over 7 days of training to reach a similar accuracy.

positional information throughout the layers. LightGlue instead relies on a relative encoding that is better comparable across images and is added in each self-attention unit. This makes it easier to leverage the positions and improves the accuracy of deeper layers.

**Prediction head:** SuperGlue predicts an assignment by solving a differentiable optimal transport problem using the Sinkhorn algorithm [66, 48]. It consists in many iterations of row-wise and column-wise normalization, which is expensive in terms of both compute and memory. SuperGlue adds a dustbin to reject unmatchable points. We found that the dustbin entangles the similarity score of all points and thus yields suboptimal training dynamics. LightGlue disentangles similarity and matchability, which are much more efficient to predict. This also yields cleaner gradients.

**Deep supervision:** Because of how expensive Sinkhorn is, SuperGlue cannot make predictions after each layer and is supervised only at the last one. The lighter head of LightGlue makes it possible to predict an assignment at each layer and to supervise it. This speeds up the convergence and enables exiting the inference after any layer, which is key to the efficiency gains of LightGlue.

## 4. Details that matter

**Recipe:** LightGlue follows the supervised training setup of SuperGlue. We first pre-train the model with synthetic homographies sampled from 1M images [50]. Such augmentations provide full and noise-free supervision but require careful tuning. LightGlue is then fine-tuned with the MegaDepth dataset [38], which includes 1M crowd-sourced images depicting 196 tourism landmarks, with camera calibration and poses recovered by SfM and dense depth by multi-view stereo. Because large models easily overfit to such distinctive scenes, the pre-training is critical to the generalization of the model but was omitted in recent follow-ups [8, 65].

**Training tricks:** While the LightGlue architecture improves the training speed, stability, and accuracy, we found that some details have a large impact too. Figure 5 shows that

this reduces the resources required to train a model compared to SuperGlue. This lowers the cost of training and makes deep matchers more accessible to the broader community.

Since the depth maps of MegaDepth are often incomplete, we also label points with a large epipolar error as unmatchable. Carefully tuning and annealing the learning rate boosts the accuracy. Training with more points also does: we use 2k per image instead of 1k. The batch size matters: we use gradient checkpointing [10] and mixed-precision to fit 32 image pairs on a single GPU with 24GB VRAM.

**Implementation details:** LightGlue has  $L=9$  layers. Each attention unit has 4 heads. All representations have dimension  $d=256$ . Throughout the paper, run-time numbers labeled as *optimized* use an efficient implementation of self-attention [14]. More details are given in the Appendix.

We train LightGlue with both SuperPoint [16] and SIFT [41] local features but it is compatible with any other type. When fine-tuning the model on MegaDepth [38], we use the data splits of Sun *et al.* [68] to avoid training on scenes included in the Image Matching Challenge [31].

## 5. Experiments

We evaluate LightGlue for the tasks of homography estimation, relative pose estimation, and visual localization. We also analyze the impacts of our design decisions.

### 5.1. Homography estimation

We evaluate the quality of correspondences estimated by LightGlue on planar scenes of the HPatches [2] dataset. This dataset is composed of sequences of 5 image pairs, each under either illumination or viewpoint changes.

**Setup:** Following SuperGlue [56], we report the precision and recall compared to GT matches at a reprojection error of 3px. We also evaluate the accuracy of homographies estimated from the correspondences using robust and non-robust solvers: RANSAC [22] and the weighted DLT [24]. For each image pair, we compute the mean reprojection error of the four image corners and report the area under the cumulative error curve (AUC) up to values of 1px and 5px. Following best practices in benchmarking [31] and unlike past works [56, 68], we use a state-of-the-art robust estimator [3] and extensively tune the inlier threshold for each method separately. We then report the highest scoring results.

**Baselines:** We follow the setup of [68] and resize all images such that their smaller dimension is equal to 480 pixels. We evaluate sparse matchers with 1024 local features extracted by SuperPoint [16]. We compare LightGlue against nearest-neighbor matching with mutual check and the deep matchers SuperGlue [56] and SGMNet [8]. We use the official models trained on outdoor datasets [38, 64]. For reference, we also evaluate the dense matcher LoFTR [68], selecting only the top 1024 predicted matches for the sake of fairness.

	features + matcher	R	P	AUC - RANSAC		AUC - DLT	
				@1px	@5px	@1px	@5px
dense	LoFTR	-	92.7	41.5	78.8	38.5	70.6
	NN+mutual	72.7	67.2	35.0	75.3	0.0	2.0
	SuperGlue	94.9	87.4	38.3	79.3	33.8	76.7
	SGMNet	<b>95.5</b>	83.0	<b>38.6</b>	79.0	31.7	76.0
SuperPoint	<b>LightGlue</b>	94.3	<b>88.9</b>	38.3	<b>79.6</b>	<b>35.9</b>	<b>78.6</b>

Table 1. **Homography estimation on HPatches.** LightGlue yields better correspondences than sparse matchers, with the highest precision (P) and a high recall (R). This results in accurate homographies when estimated by RANSAC or even a faster least-squares solver (DLT). LightGlue is competitive with dense matchers like LoFTR.

**Results:** Table 1 shows that LightGlue yields correspondences with higher precision than and similar recall to SuperGlue and SGMNet. When estimating homographies with DLT, this results in much more accurate estimates than with other matchers. LightGlue thus makes DLT, a simple solver, competitive with the expensive and slower MAGSAC [3]. At a coarse threshold of 5px, LightGlue is also more accurate than LoFTR despite being constrained by sparse keypoints.

### 5.2. Relative pose estimation

We evaluate LightGlue for pose estimation in outdoor scenes that exhibit strong occlusion and challenging lighting and structural changes.

**Setup:** We use image pairs from the MegaDepth-1500 test set following the evaluation of [68]. The test set contains 1500 image pairs from two popular phototourism destinations: St. Peters Square and Reichstag. The data was collected in a way that the difficulty is balanced based on visual overlap. We evaluate our method on the downstream task of relative pose estimation.

We estimate an essential matrix both with vanilla RANSAC and LO-RANSAC [34], respectively, and decompose them into a rotation and a translation. The inlier threshold is tuned for each approach on the test data – we think that this makes the comparison more fair as we do not evaluate RANSAC itself. We compute the pose error as the maximum angular error in rotation and translation and we report its AUC at 5°, 10°, and 20°.

**Baselines:** We extract 2048 local features per images, each resized such that its larger dimension is 1600 pixels. With SuperPoint [16] features, we compare LightGlue to nearest-neighbor matching with mutual check and to the official implementations of SuperGlue [56] and SGMNet [8]. With DISK [73] we only evaluate against its own strong baseline, as no other trained matcher with DISK is publicly available.

We also evaluate the recent, dense deep matchers LoFTR [68], MatchFormer [78], and ASpanFormer [9]. We carefully follow their respective evaluation setups and resize the input images such that their largest dimension is 840 pixels (LoFTR, MatchFormer) or 1152 pixels (ASpanFormer).

dense	features + matcher	RANSAC AUC	LO-RANSAC AUC	time (ms)
		5° / 10° / 20°		
dense	LoFTR	52.8 / 69.2 / 81.2	66.4 / 78.6 / 86.5	181
	MatchFormer	53.3 / 69.7 / 81.8	66.5 / 78.9 / 87.5	388
	ASpanFormer	<b>55.3 / 71.5 / 83.1</b>	<b>69.4 / 81.1 / 88.9</b>	369
DISK	NN+ratio	38.1 / 55.4 / 69.6	57.2 / 69.5 / 78.6	7.4
	<b>LightGlue</b>	<b>43.5 / 61.0 / 75.3</b>	<b>61.3 / 74.3 / 83.8</b>	44.5
SuperPoint	NN+mutual	31.7 / 46.8 / 60.1	51.0 / 54.1 / 73.6	5.7
	SuperGlue	49.7 / 67.1 / <b>80.6</b>	65.8 / 78.7 / 87.5	70.0
	SGMNet	43.2 / 61.6 / 75.6	59.8 / 74.1 / 83.9	73.8
	<b>LightGlue</b>	<b>49.9 / 67.0 / 80.1</b>	<b>66.7 / 79.3 / 87.9</b>	44.2
	↪ adaptive	49.4 / <b>67.2</b> / 80.1	66.3 / 79.0 / <b>87.9</b>	31.4

Table 2. **Relative pose estimation.** On the MegaDepth1500 dataset, LightGlue predicts more precise correspondences with higher pose accuracy (AUC), and speed than existing sparse matchers. It is competitive with dense matchers for a fraction of the inference time, and even outperforms LoFTR and MatchFormer with the superior LO-RANSAC estimator. The adaptive scheme greatly reduces the run time for only a minor loss of accuracy.

Larger images would improve their accuracy, as with sparse features, but would incur prohibitive and unpractical run time and memory requirements.

**Results:** Table 2 shows that LightGlue largely outperforms the existing approaches SuperGlue and SGMNet on SuperPoint features, and can greatly improve the matching accuracy over DISK local features. It yields better correspondences and more accurate relative poses and reduces the inference time by 30%. LightGlue typically predicts slightly fewer matches than SuperGlue but those are more accurate. By detecting confident predictions early in the model, the adaptive variant is over 2× faster than SuperGlue and SGMNet and still more accurate. With a carefully tuned LO-RANSAC [34], LightGlue can achieve higher accuracy than some popular dense matcher which are between 5 and 11 times slower. Among the evaluated dense matchers, ASPANFormer is the most accurate. Considering trade-off between accuracy and speed, LightGlue outperforms all approaches by a large margin.

### 5.3. Outdoor visual localization

**Setup:** We evaluate long-term visual localization in challenging conditions using the large-scale Aachen Day-Night benchmark [59]. We follow the Hierarchical Localization framework with the hloc toolbox [55]. We first triangulate a sparse 3D point cloud from the 4328 daytime reference images, with known poses and calibration, using COLMAP [60]. For each of the 824 daytime and 98 nighttime queries, we retrieve 50 images with NetVLAD [1], match each of them, and estimate a camera pose with RANSAC and a Perspective-n-Point solver. We report the pose recall at multiple thresholds and the average throughput of the matching step during both mapping and localization.

SuperPoint + matcher	Day	Night	pairs per second
	(0.25m,2°) / (0.5m,5°) / (1.0m,10°)		
SuperGlue	88.2 / <b>95.5 / 98.7</b>	86.7 / 92.9 / <b>100</b>	6.5
SGMNet	86.8 / 94.2 / 97.7	83.7 / 91.8 / 99.0	10.2
ClusterGNN	<b>89.4 / 95.5 / 98.5</b>	81.6 / <b>93.9 / 100</b>	13*
<b>LightGlue</b>	89.2 / 95.4 / 98.5	<b>87.8 / 93.9 / 100</b>	<b>17.2 / 26.1</b>

Table 3. **Outdoor visual localization.** On the Aachen Day-Night dataset, LightGlue performs on par with SuperGlue but runs 2.5× faster, 4× when *optimized*. SGMNet and ClusterGNN are both slower and less robust on night-time images (\*approximation).

**Baselines:** We extract up to 4096 features with SuperPoint and match them with SuperGlue, SGMNet [8], ClusterGNN [65], and LightGlue with adaptive depth and width. Since the implementation of ClusterGNN is not publicly available, we report the accuracy found in the original paper and the time estimates kindly provided by the authors.

**Results:** Table 3 shows that LightGlue reaches a similar accuracy as SuperGlue but at a 2.5× higher throughput. The *optimized* variant, which leverages an efficient self-attention [14], increases the throughput by 4×. LightGlue thus matches up to 4096 keypoints in real time.

### 5.4. Insights

**Ablation study:** We validate our design decisions by evaluating LightGlue after its pre-training on the challenging synthetic homography dataset with extreme photometric augmentations. We train different variants with SuperPoint features and 5M samples, all within 4 GPU-days. We create a test set from the same augmentations applied to images unseen during training. We extract 512 keypoints from each. We also compare against SuperGlue, which we train with the same setup. More details are provided in the Appendix.

We report the ablation results in Table 4. Compared to SuperGlue, LightGlue converges significantly faster, and achieves +4% recall and +12% precision. Note that SuperGlue can achieve similar accuracies as LightGlue with a long-enough training, but the improved convergence makes it much more practical to train on new data.

Without the matchability classifier, the network loses its ability to discriminate between good and bad matches, as shown in Figure 6. Intuitively, the similarity matrix proposes many likely matches while the matchability filters incorrect proposals. Thus, our partial assignment can be viewed as an elegant fusion of mutual nearest neighbor search and a learned inlier classifier [44, 82]. This is significantly faster than solving the optimal transport problem of SuperGlue.

Replacing learned absolute positional encoding with rotary embeddings improves the accuracy, with a minor penalty on run time from rotating queries and keys at each self-attention layer. Using relative positions, LightGlue learns to match geometric patterns across images. Reminding the network about positions at each layer improves the robustness

architecture	precision	recall	time (ms)
SuperGlue	74.6	90.5	29.1
<b>LightGlue</b> (full)	<b>86.8</b>	96.3	19.4
↳ a) no matchability	67.4	<b>97.0</b>	18.9
↳ b) absolute positions	84.2	94.7	18.7
↳ c) full cross-attention	86.6	96.1	22.8
↳ d) early layer (#5/9)	78.1	92.7	<b>11.9</b>

Table 4. **Ablation study on synthetic homographies.** a-b) Both matchability and positional encoding improve the accuracy without impact on the time. c) The bidirectional cross-attention is faster without drop of accuracy. d) Thanks to the deep supervision, early layers yield good predictions on pairs with low difficulty.



Figure 6. **Benefit of the matchability.** The matchability helps filter out outliers (red) that are visually similar, retaining only inlier correspondences (green).

of the network, resulting in +2% precision.

Bidirectional cross-attention is equally accurate as standard cross-attention, but saves 20% run time by only computing the similarity matrix once. Currently, the bottleneck is computing the softmax along two dimensions. With a dedicated bidirectional softmax kernel, plenty of redundant computations could be avoided.

Using deep supervision, also intermediate layers have meaningful outputs. Already after 5 layers, the network can predict robust matches, achieving > 90% recall. In the final layers, the network focuses on rejecting outliers, thus improving the match precision.

**Adaptivity:** By predicting matchability scores and confidences, we can adaptively reduce the computations during a forward-pass on a case-by-case basis. Table 5 studies the effectiveness of the two pruning mechanisms – adaptive depth and width – on MegaDepth image pairs for different ranges of visual overlap. For easy samples, such as the successive frames of a video, the network quickly converges and exits after a few layers, resulting in a 1.86× speedup. In cases of low visual overlap, e.g. loop closure, the network requires more layers to converge. It however rejects confident and

metric	difficulty			average
	easy	medium	hard	
average index of stopping layer ↓	4.7	5.5	6.9	5.7
ratio of unmatchable points (%) ↑	19.8	23.4	27.9	23.7
speedup over non-adaptive ↑	1.86	1.33	1.16	1.45

Table 5. **Impact of adaptive depth and width.** Early stopping helps most on smaller scenes, where the network stops after just half the layers. On harder scenes, the network requires more layers to converge, but smaller view overlap between image pairs allows the network to more aggressively prune the width of the network. Overall, adaptive depth- and width- pruning reduces the run time by 33% and is particularly effective on easy pairs.

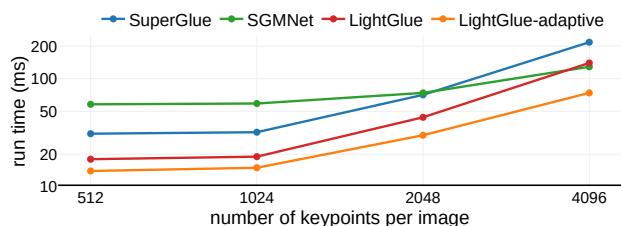


Figure 7. **Run time vs number of keypoints.** The full LightGlue model is 35% faster than SuperGlue and the adaptive depth and width make it even faster. SGMNet is comparably fast only for 4k keypoints and above but is much slower for standard input sizes.

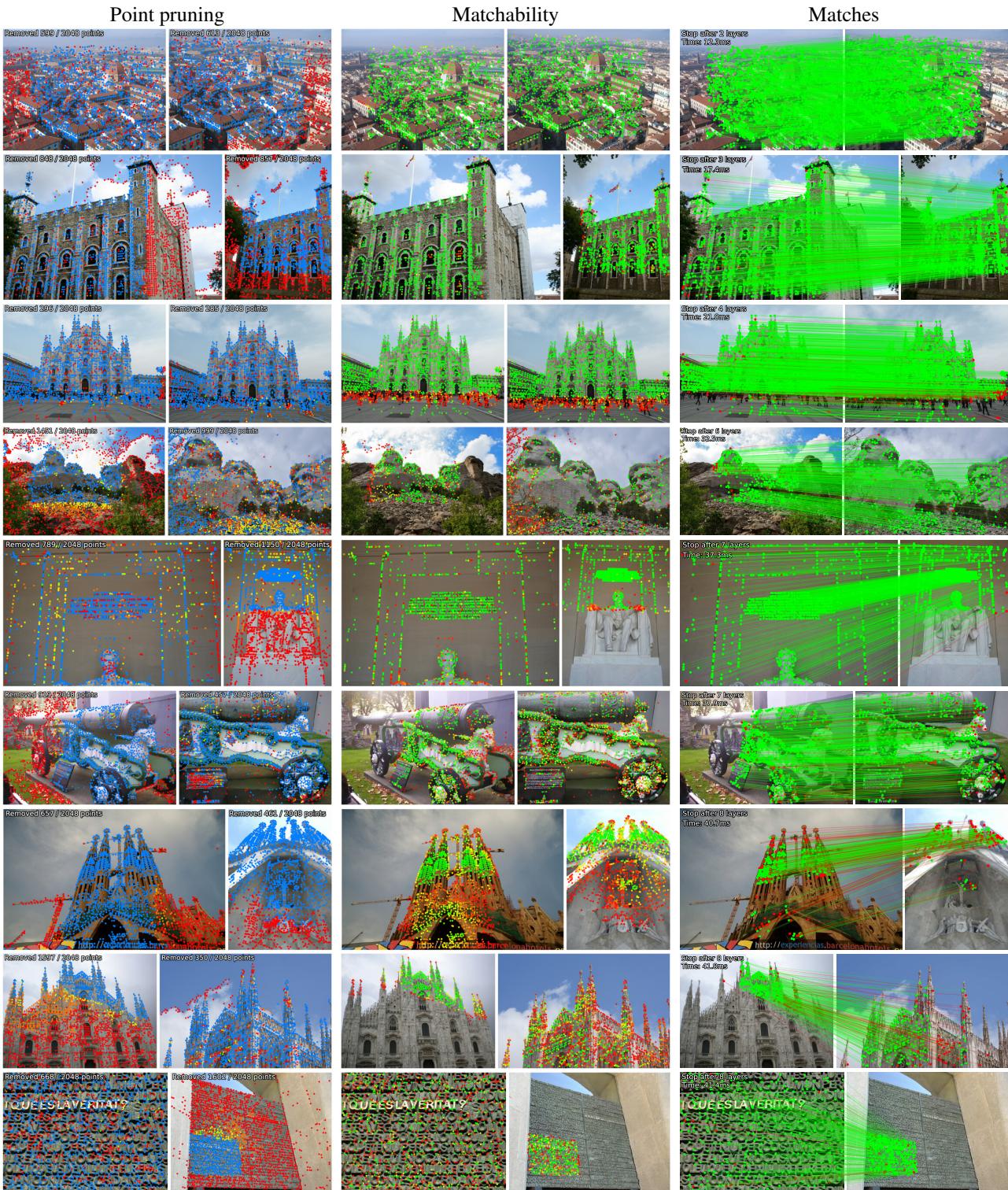
unmatchable points early and leaves them out of the inputs to subsequent layers, thus avoiding unnecessary computations.

**Efficiency:** Figure 7 shows run times for different numbers of input keypoints. For up to 2K keypoints per image, which is a common setting for visual localization, LightGlue is faster than both SuperGlue [56] and SGMNet [8]. Adaptive pruning further reduces the run time for any input size.

## 6. Conclusion

This paper introduces LightGlue, a deep neural network trained to match sparse local features across images. Building on the success of SuperGlue, we combine the power of attention mechanisms with insights about the matching problem and with recent innovations in Transformer. We give this model the ability to introspect the confidence of its own predictions. This yields an elegant scheme that adapts the amount of computation to the difficulty of each image pair. Both its depth and width are adaptive: 1) the inference can stop at an early layer if all predictions are ready, and 2) points that are deemed not matchable are discarded early from further steps. The resulting model, LightGlue, is finally faster, more accurate, and easier to train than the long-unrivaled SuperGlue. In summary, LightGlue is a drop-in replacement with only benefits. The code will be released publicly for the benefit of the community.

**Acknowledgments:** We thank Mihai Dusmanu, Rémi Pautrat, and Shaohui Liu for their helpful feedback.



**Figure 8. Visualization of adaptive depth and width.** From top to bottom, we show three easy, medium and difficult image pairs. The left column shows how LightGlue reduces its width: it finds out early that some points (●) are unmatchable (mostly by visual overlap) and discards non-repeatable points in later layers: ● → ○ → ●. This is very effective on difficult pairs. LightGlue looks for matches only in the reduced search space (●). The matchability scores (middle column, from non-matchable ● to likely matchable ○), help find accurate correspondences and are almost binary. On the right we visualize predicted matches as epipolar in- or outliers. We report the run time and stopping layer for each pair. On easy samples, LightGlue stops after only 2-3 layers, running with close to 100 FPS.

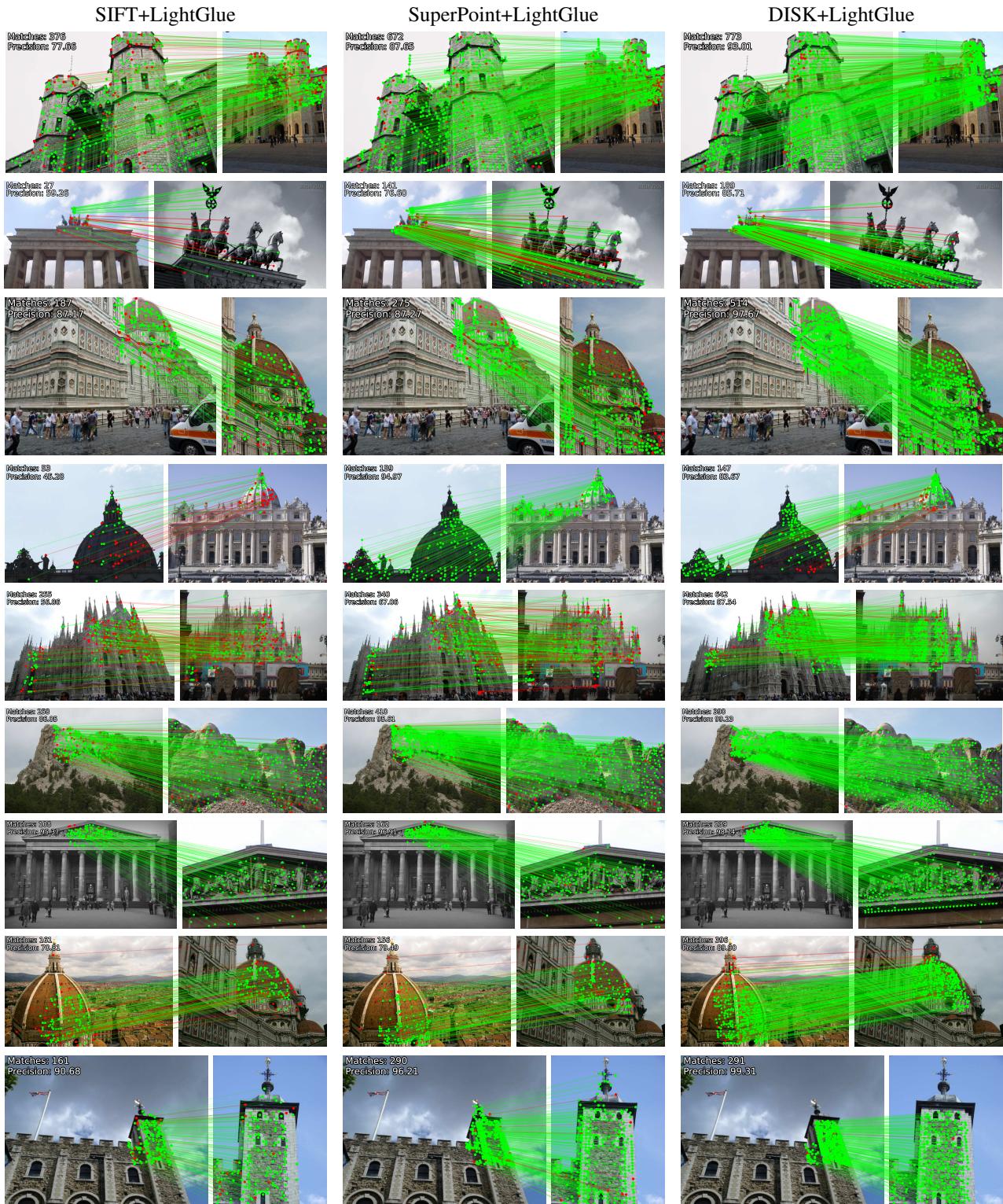


Figure 9. **Comparison of features produced by LightGlue for different local features.** We compare the outputs of SIFT+LightGlue (left), SuperPoint+LightGlue (middle) and DISK+LightGlue (right).

## Appendix

### A. Image Matching Challenge

In this section, we present results obtained on the PhotoTourism dataset of the Image Matching Challenge 2020 (IMC) [26] in both stereo and multi-view tracks. The data is very similar to the MegaDepth [38] evaluation, exhibits similar statistics but different scenes. We follow the standardized matching pipeline of IMC with the setup and hyperparameters of SuperGlue [56]. We run the evaluation on the 3 validation scenes from the PhotoTourism dataset with LightGlue trained with two kinds of local features.

**SuperPoint:** For SuperPoint+SuperGlue and SuperPoint+LightGlue, we extract a maximum of 2048 keypoints and use DEGENSAC [11, 12, 43] with a threshold on the detection confidence of 1.1 in the stereo track (as suggested by SuperGlue). We do not perform any parameter tuning and reuse our model from the outdoor experiments with adaptive depth- and width, and use efficient self-attention [14] and mixed-precision during evaluation.

**DISK:** We also train LightGlue with DISK local features [73], a previous winner of the Image Matching Challenge. We follow the same training setup as for SuperPoint. For evaluation, we follow the guidelines from the authors for the restricted keypoint scenario (max 2048 features per image) and use mutual nearest neighbor matching with a ratio test of 0.95 as a baseline. We again use DEGENSAC for relative pose estimation with a threshold of 0.75.

**Results:** Table 6 reports the evaluation results. We also report the average matching speed over all 3 validation scenes. LightGlue is competitive with SuperGlue both in the stereo and multi-view track, while running  $2.5\times$  faster. Most of these run time improvements are due to the adaptive-depth, which largely reduces the run time for easy image pairs.

LightGlue trained with DISK [73] largely outperforms both the nearest-neighbor matching baseline with ratio test but also SuperPoint+LightGlue. On the smaller thresholds, DISK+LightGlue achieves +8%/+5% AUC in the stereo and multi-view tasks compared to our SuperPoint equivalent. With DISK, our model predicts 30% more matches than SP+LightGlue with an even higher epipolar precision.

**Image Matching Challenge 2021:** We evaluate the phototourism subset of the IMC 2021 [27] benchmark, both in the stereo- and multiview track. We compare our baseline on SuperPoint [16] and DISK [73] with their respective baselines in a clean setting and in a restricted keypoint setting (max 2048 detections). Furthermore, we compare our best scoring method on IMC 2020, DISK+LightGlue, with tuned versions of DISK [73], SuperPoint+SuperGlue [16, 56] as well as the SfM implementation of the dense matcher LoFTR [68]. Table 7 reports the experiment. LightGlue outperforms all

SfM features (2048 keypoints)	Task 1: Stereo		Task 2: Multiview			Pairs per second	
	AUC@K°		AUC@5°@N				
	5°	10°	5	10	25		
SP+SuperGlue	58.64	71.07	61.88	78.97	86.75	16.2	
<b>SP+LightGlue</b>	<b>59.03</b>	<b>71.13</b>	<b>62.87</b>	<b>79.36</b>	<b>86.98</b>	<b>43.4</b>	
DISK+NN+ratio	57.76	68.73	59.91	78.95	87.54	<b>196.7</b>	
<b>DISK+LightGlue</b>	<b>67.02</b>	<b>77.82</b>	<b>67.91</b>	<b>80.58</b>	<b>88.35</b>	44.5	

Table 6. **Structure-from-Motion** with the Image Matching Challenge 2020. We evaluate the stereo track, at multiple error thresholds, and the multi-view track, for various numbers of images  $N$ . LightGlue yields better poses than SuperGlue on the multi-view track and significantly reduces the matching time. In combination with DISK, LightGlue improves over SuperPoint+SuperGlue and DISK+NN+ratio in both tracks by a large margin.

features + matcher	Task 1: Stereo		Task 2: Multiview		Average
	AUC 5° / 10°	AUC 5° / 10°	AUC 5° / 10°	AUC 5° / 10°	
SP+SGMNet	29.6 / 43.0		60.2 / 71.6		44.9 / 57.3
SP+SuperGlue	36.5 / 50.5		63.3 / 73.8		49.9 / 62.2
<b>SP+LightGlue</b>	<b>36.7 / 50.7</b>		<b>63.6 / 74.4</b>		<b>50.2 / 62.6</b>
DISK+NN+ratio	36.3 / 48.5		61.5 / 71.6		48.9 / 60.1
<b>DISK+LightGlue</b>	<b>43.1 / 56.6</b>		<b>66.2 / 76.2</b>		<b>54.7 / 66.4</b>
DISK (8K) +NN+ratio*	44.6 / 56.2		65.0 / 74.4		54.8 / 65.3
SP+SuperGlue*	44.6 / 58.6		66.8 / 77.1		55.7 / 67.9
LoFTR-SfM	48.4 / 60.9		66.4 / 76.1		57.4 / 68.5
DISK (8K)+LightGlue	<b>48.7 / 61.8</b>		<b>68.9 / 78.2</b>		<b>58.8 / 70.0</b>

Table 7. **IMC 2021 – Phototourism.** \*DISK+NN and SP+SG use test-time augmentation while LightGlue does not. To compete with these tuned baselines, we just increase the number of keypoints, e.g. DISK (8K). LoFTR-SfM clusters dense matches with SuperPoint detections. LightGlue outperforms other sparse baselines both in the stereo and multiview task, and even surpasses tuned baselines from the public leaderboard by a large margin.

approaches with a fair margin.

**Image Matching Challenge 2023:** We compete in the IMC 2023 [28], which evaluates end-to-end Structure-from-Motion in terms of camera pose accuracy, averaged over multiple thresholds, with a diverse set of scenes beyond phototourism. We use the default reconstruction pipeline of hloc [55] and retrieve 50 pairs per image using NetVLAD [1]. We average the results over 3 runs to reduce the impact of randomness in the reconstruction pipeline. On the public / private leaderboards, respectively, SuperPoint+SuperGlue achieves a score of 36.1 / 43.8 (%), while **SuperPoint+LightGlue** reaches **38.4 / 46.1**, which is a **+2.3%** improvement.

### B. Additional results

#### Relative pose estimation:

Results reported in Section 5.2 were computed with a sub-set of the MegaDepth dataset [38] as introduced by previous

	features + matcher	#matches	P	pose estimation AUC			time (ms)
				@5°	@10°	@20°	
dense	LoFTR	2231	89.8	66.4	79.1	87.6	181
	MatchFormer	2416	91.2	65.2	78.1	87.4	388
	ASpanFormer	4299	94.7	68.0	80.4	88.7	239
SIFT	NN+ratio	160	82.3	48.3	62.2	73.2	5.7
	SGMNet	405	82.5	50.7	66.6	76.5	71.7
	<b>LightGlue</b>	383	84.1	57.0	71.3	81.8	44.3
SuperPoint	NN+mutual	697	49.4	37.7	50.9	62.3	5.6
	SuperGlue	712	93.0	64.8	77.5	86.6	70.0
	SGMNet	<b>725</b>	89.8	61.7	74.3	83.4	74.0
	<b>LightGlue</b>	709	<b>94.5</b>	<b>65.5</b>	<b>77.8</b>	<b>86.9</b>	44.2

Table 8. **Relative pose estimation on Megadepth-1800.** This split is different from Table 2. In contrast to the split used by previous works [38, 68], this set of test images avoids training overlap with SuperGlue [56]. LightGlue predicts a similar amount of correspondences but with higher precision (P), pose accuracy (AUC), and speed than existing sparse matchers. It is competitive with dense matchers for a fraction of the inference time.

works [9, 68, 78]. However, the images therein overlap with the training set of SuperGlue [56], the state-of-the-art sparse feature matcher and thus our main competitor.

For a more fair evaluation, we perform an extensive outdoor experiment on the test scenes of our MegaDepth [38] split, which covers 4 unique phototourism landmarks that SuperGlue was not trained with: Sagrada Familia, Lincoln Memorial Statue, London Castle, and the British Museum. To balance the difficulty of image pairs, we bin pairs into three categories based on their visual overlap score [19, 56], with intervals [10, 30]%, [30, 50]%, and [50, 70]%. We sample 150 image pairs per bin per scene, totaling 1800 image pairs. We carefully rerun the experiment with the same setup that was used in Table 2. We report the precision as the ratio of matches with an epipolar error below 3px. With SIFT [41], we evaluate the ratio test and SGMNet [8] only, as the original SuperGlue model is not publicly available.

Table 8 confirms that LightGlue predicts more accurate correspondences than existing sparse matchers, at a fraction of the time. Detector-free feature matchers like LoFTR remain state-of-the-art on this task, although by a mere 2% AUC@5° with LO-RANSAC.

**Outdoor visual localization:** For completeness, we also report results on the Aachen v1.1 dataset [59] and compare our method to recent sparse and dense baselines. Table 9 shows that all methods perform similarly on this dataset, which is largely saturated, with insignificant variations in the results. LightGlue is however far faster than all approaches.

**Indoor visual localization:** We report results for InLoc in Table 10. We use hloc and run SuperGlue again for fairness. For LoFTR and ASpanFormer, report existing results as no code is available. LightGlue is competitive with SuperGlue and more accurate at (0.25m,10°). Differences of <2% are

features + matcher	Day	Night	pairs per second
	(0.25m,2°) / (0.5m,5°) / (1.0m,10°)		
LoFTR	88.7 / 95.6 / 99.0	<b>78.5</b> / 90.6 / 99.0	-
ASpanFormer	89.4 / 95.6 / 99.0	77.5 / <b>91.6</b> / 99.5	-
SP+SuperGlue	89.8 / <b>96.1</b> / <b>99.4</b>	77.0 / 90.6 / <b>100</b>	6.4
<b>SP+LightGlue</b>	<b>90.2</b> / 96.0 / <b>99.4</b>	77.0 / 91.1 / <b>100</b>	<b>17.3</b>

Table 9. **Outdoor visual localization on Aachen v1.1.** LightGlue achieves similar accuracy with higher throughput.

features + matcher	DUC1	DUC2
	(0.25m,10°) / (0.5m,10°) / (1.0m,10°)	
LoFTR	47.5 / 72.2 / 84.8	54.2 / 74.8 / <b>85.5</b>
MatchFormer	46.5 / 73.2 / 85.9	<b>55.7</b> / 71.8 / 81.7
ASpanFormer	<b>51.5</b> / <b>73.7</b> / <b>86.4</b>	55.0 / 74.0 / 81.7
SP+SuperGlue	47.0 / 69.2 / 79.8	53.4 / <b>77.1</b> / 80.9
SP+LightGlue	49.0 / 68.2 / 79.3	55.0 / 74.8 / 79.4

Table 10. **Indoor visual localization on InLoc.** LightGlue performs similarly to SuperGlue (within the variability of the dataset).

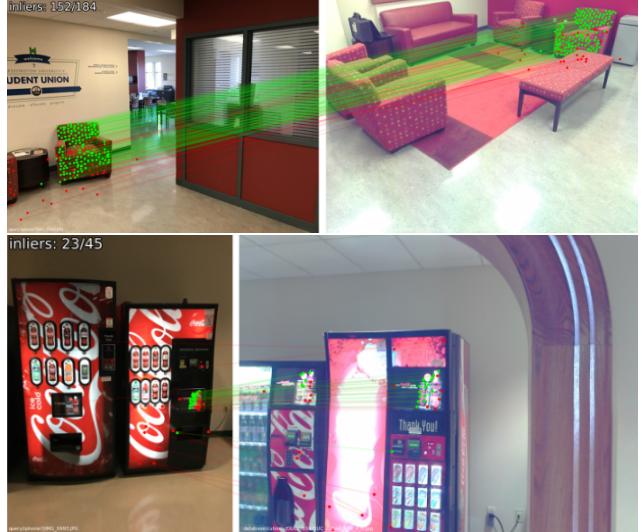


Figure 10. **Failure cases on InLoc [70].** LightGlue sometimes matches repeated objects in the scene with strong texture, instead of the geometric structure.

insignificant because each split only has 205/151 queries (1.5% of difference  $\equiv$  3 queries). Failures of LightGlue over SuperGlue (6/356 images @1m) are due to more matches on repeated objects (like trash cans), *i.e.* to better matching and weak retrieval – we show an example in Figure 10.

## C. Implementation details

### C.1. Architecture

**Positional Encoding.** 2D image coordinates are normalized to a range [-1, 1] while retaining the image aspect ratio. We then project 2D coordinates into frequencies with a linear projection  $\mathbf{W}_p \in \mathbb{R}^{2d/2h}$ , where  $h$  is the number of attention

heads. We cache the result for all layers. We follow the efficient scheme of Roformer [67] to apply the rotations to query and key embeddings during self-attention, avoiding quadratic complexity to compute relative positional bias. We do not apply any positional encoding during cross-attention, but let the network learn spatial patterns by aggregating context within each image.

**Graph Neural Network:** The graph neural network consists of 9 transformer layers with both a self- and cross-attention unit. The update MLP (Eq. 1) has a single hidden layer of dimension  $d_h = 2d$  followed by LayerNorm, GeLU activation and a linear projection  $(2d, d)$  with bias.

Each attention unit has three projection matrices for query, key and value, plus an additional linear projection that merges the multi-head output. In bidirectional cross attention, the projections for query and key are shared. In practice we use an efficient self-attention [14] which optimizes IO complexity of the attention aggregation. This could also be extended for bidirectional cross attention. While training we use gradient checkpointing to significantly reduce the required VRAM.

**Correspondences:** The linear layers (Eq. 6) map from  $d$  to  $d$  and are not shared across layers. For all experiments we use the mutual check and a filter threshold  $\tau = 0.1$ .

**Confidence classifier:** The classifier predicts the confidence with a linear layer followed by a sigmoid activation. Confidences are predicted for each keypoint and only at layers  $1, \dots, L - 1$ , since, by definition, the confidences of the final layer  $L$  are 1. Each prediction is supervised with a binary cross-entropy loss and its gradients are not propagated into the states to avoid impacting the matching accuracy. The state already encodes sufficient information since it is also supervised for matchability prediction.

**Exit criterion and point pruning:** During training we observed that the confidence predictions are less accurate in earlier layers. We therefore exponentially decay the confidence threshold:

$$\lambda_l = 0.8 + 0.1e^{-4\ell/L} . \quad (12)$$

A state is deemed confident if  $c_i^\ell > \lambda_\ell$ . During inference, we halt the network if  $\alpha=95\%$  of states are deemed confident.

For point pruning, a point is deemed unmatchable when its predicted confidence is high and its matchability is low:

$$\text{unmatchable}(i) = c_i^\ell > \lambda_\ell \text{ } \& \text{ } \sigma_i^\ell < \beta \quad (13)$$

We report an ablation on the exit confidence  $\alpha$  in Table 11 for relative pose estimation on MegaDepth. Lowering  $\alpha$  to 80% reduces the inference time by almost 50% compared to our full model, while maintaining competitive accuracy compared to SuperGlue on this task. Reducing the confidence threshold is far more effective in terms of run time -

Method	#matches	P	pose estimation AUC			time (%)
			@5°	@10°	@20°	
<b>SP+LightGlue</b>	<b>613</b>	<b>96.2</b>	<b>66.7</b>	<b>79.3</b>	<b>87.9</b>	100.0
↳ layer 7/9	705	96.0	66.2	79.1	88.0	82.4
↳ layer 5/9	702	94.5	65.0	77.8	87.0	60.0
↳ layer 3/9	687	90.0	64.0	76.7	85.8	<b>41.9</b>
↳ confidence 98%	<b>610</b>	<b>96.2</b>	66.6	79.3	88.0	80.5
↳ <b>confidence 95%</b>	608	95.4	66.3	79.0	87.9	70.6
↳ confidence 90%	607	94.5	65.9	78.5	87.2	61.5
↳ confidence 80%	605	92.6	65.2	77.8	86.7	48.4

Table 11. **Evaluation of early-stopping on MegaDepth.** Matches predicted by deeper layers are more accurate but require more computations with a higher inference time. Modeling confidences adaptively selects the model depth that yields a sufficient accuracy. A more conservative stopping, with a higher threshold  $\alpha$ , yields a higher accuracy at the cost of higher inference time.  $\alpha=95\%$  yields the best trade-off.

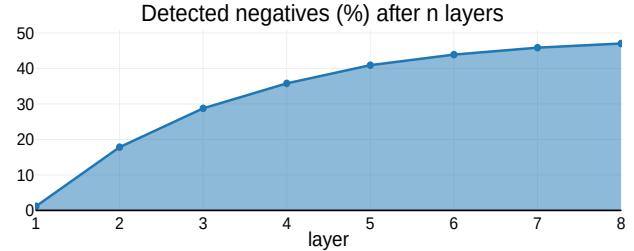


Figure 11. **Continuous detection of unmatchable points.** After just a few layers the network detects many points which are unmatchable, and we exclude them from context aggregation.

accuracy tradeoff than trimming the model to fewer layers. Stopping the network early mainly sacrifices precision. For our experiments we chose 95% confidence, which yields on average 25% run time reduction with hardly any loss of accuracy on downstream tasks.

Here,  $\beta = 0.01$  is a threshold on how matchable a point is. If Eq. 13 holds, we exclude the point from context aggregation in the following layers. This adds an overhead of gather and scatter per layer, but pruning becomes increasingly effective with more keypoints.

In Figure 11 we report the fraction of keypoints excluded in each layer. After just a few layers of context aggregation, LightGlue is confident to exclude  $> 30\%$  of keypoints early on. Since the number of keypoints have a quadratic impact on run time, as shown in Fig. 7, this can largely reduce the number of computations in a forward pass and thus significantly reduce inference time.

## C.2. Local features

We train LightGlue with three popular local feature detectors and descriptors: SuperPoint [16], SIFT [41] and DISK [73]. During training and evaluation, we discard the detection threshold for all methods and use the top-k keypoints according to the detection score. During training, if

there are less than  $k$  detections available, we append random detections and descriptors. For SIFT [41] and DISK [73], we add a linear layer to project descriptors to  $d=256$  before feeding them to the Transformer backbone.

**SuperPoint:** SuperPoint is a popular feature detector which produces highly repeatable points located at distinctive regions. We use the official, open-sourced version of SuperPoint from MagicLeap [16]. The detections are pixel-accurate, i.e. the keypoint localization accuracy depends on the image resolution.

**SIFT:** We use the excellent implementation of SIFT from vlfeat [75] when training on MegaDepth, and SIFTGPU from COLMAP [60] for fast feature extraction when pre-training on homographies. We observed that these implementations are largely equivalent during training and can be exchanged freely. Also, SIFT features from OpenCV can be used without retraining. Orientation and scale are not used in positional encoding.

**DISK:** DISK learns detection and description with a reinforcement learning objective. Its descriptors are more powerful than SIFT and SuperPoint and its detections are more repeatable, especially under large viewpoint and illumination changes.

### C.3. Homography pre-training

Following Sarlin *et al.* [56], we first pre-train LightGlue on synthetic homographies of real-images.

**Dataset:** We use 170k images from the Oxford-Paris 1M distractors dataset [50], and split them into 150k/10k/10k images for training/validation/test.

**Homography sampling:** We generate homographies by randomly sampling four image corners. We split the image into four quarters, and sample a random point in each quarter. To avoid degenerates, we enforce that the enclosed area is convex. After, we apply random rotations and translations to the corners s.t. the corners remain inside the image. With this process, we can generate extreme perspective changes while avoiding border artifacts. This process is repeated twice, resulting in two largely skewed homographies. In interpolation, we then enforce the extracted images to be of size 640x480.

**Photometric augmentation:** The color images are then forwarded through a sequence of strong photometric augmentations, including blur, hue, saturation, sharpness, illumination, gamma and noise. Furthermore, we add random additive shades into the image to simulate occlusions and non-uniform illumination changes.

**Supervision:** Correspondences with 3px symmetric reprojection error are deemed inliers, and points without any correspondence under this threshold are outliers.



Figure 12. **Examples of synthetic homographies.** We show the original images (left) and two augmented examples (center and right) resulting from strong perspective transformations and extreme photometric augmentations.

**Training details:** We extract 512/1024/1024 keypoints for SuperPoint/SIFT/DISK, and a batch size of 64. The initial learning rate is 0.0001, and we multiply the learning rate by 0.8 each epoch after 20 epochs. We stop the training after 40 epochs (6M image pairs), or 2 days with 2 Nvidia RTX 3090 (for SuperPoint). Our network achieves > 99% recall and > 90% precision on the validation and test set. We also observed that, for fine-tuning, one can stop the pre-training after just one day with only minor losses.

We also experimented with sampling images from MegaDepth [38] for homography pre-training, and could not observe major differences. Strong photometric augmentations and perspective changes are crucial for training a robust model.

### C.4. Finetuning on MegaDepth

We fine-tune our model on phototourism images with pseudo ground-truth camera poses and depth images.

**Dataset:** We use the MegaDepth dataset [38], which contains dense reconstructions of a large variety of popular landmarks all around the globe, obtained through COLMAP+MVS [60, 61]. Following Sun *et al.* [68], we bin each pair by its covisibility score [19], into ranges [0.1, 0.3], [0.3, 0.5] and [0.5, 0.7]. Scenes which are part of the validation and test set in the image matching challenge [26] are also excluded from training, resulting in 368/5/24 scenes for training/validation/test. At the beginning of each epoch, we sample 100 image pairs per scene.

Images are resized s.t. their larger edge is of size 1024, and zero-pad images to  $1024 \times 1024$  resolution.

**Supervision:** Following SuperGlue [56], we reproject points using camera poses and depth to the other image. Correspondences with a maximum reprojection error of 3 pixels and which are mutually closest are labelled as inliers. A point where the closest correspondence has a reprojection error larger than 5px is labelled as outlier. Furthermore, we also declare points without depth and no correspondence with a Sampson Error smaller than 3 px outliers.

**Training details:** Weights are initialized from the pre-trained model on homographies, Training starts with a learn-

ing rate of  $1e-5$  and we exponentially decay it by 0.95 in each epoch after 10 epochs, and stop training after 50 epochs (2 days on 2 RTX 3090). The top 2048 keypoints are extracted per image, and we use a batch size of 32. To speed-up training, we cache detections and descriptors per image, requiring around 200 GB of disk space.

## C.5. Homography estimation

We validate the models capabilities on real homographies on the Hpatches dataset [2]. We follow the setup introduced in LoFTR [68] and resize images to a maximum edge length of 480.

For SuperPoint we extract the top 1024 keypoints with the highest detection score, and report precision (fraction of matches within 3px homography error) and recall (fraction of recovered mutual nearest-neighbour matches within 3px homography error). For LoFTR we only report epipolar precision. Furthermore, we evaluate the models in the downstream task of homography matrix estimation. Following SuperGlue [56], we report pose estimation results from robust estimation using RANSAC/MAGSAC [3] and the least squares solution with the weighted DLT algorithm. We evaluate the accuracy of estimated homography by their mean absolute corner distance towards the ground-truth homography.

We use OpenCV with USAC\\_MAGSAC for robust homography estimation, and tune the threshold for each method separately. Our reasoning behind this decision, which is in contrast to previous works in feature matching [56, 68] which fix the RANSAC parameters, is that we mainly use RANSAC as a tool to evaluate the low-level matches on a downstream task, and we want to minimize the variations introduced by its hyperparameters in order to obtain fair and representative evaluations. Different matches typically require different RANSAC thresholds, and thus a fixed threshold is suboptimal for comparison. For example on outdoor relative pose estimation, tuning the RANSAC threshold yields +7% AUC@ $5^\circ$  on SuperGlue, skewing the reported numbers.

## D. Timings

All experiments were conducted on a single RTX 3080 with 10GB VRAM. We report the timings of the matching process only, excluding sparse feature extraction (which is linear in the number of images) and robust pose estimation. We report the average over the respective datasets.

In Figure 13 we benchmark self-/cross-attention and solving the partial assignment problem against the respective counterparts in SuperGlue [56]. Bidirectional cross-attention reduces the run-time by 33% by only computing the similarity matrix once. However, the main bottleneck remains computing the softmax over both directions.

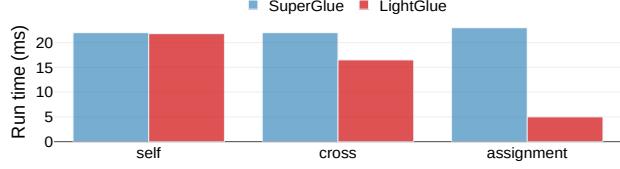


Figure 13. **Run time breakdown.** We evaluate the runtime of self-, cross- and partial assignment layers on 1024 keypoints for SuperGlue and LightGlue. Most of LightGlue’s default inference time improvements stem from a significantly faster partial assignment layer and reuse of computations in bidirectional cross-attention.

Our cheap double-softmax and the unary matchability predictions are significantly faster than solving it using optimal transport [66, 48], where 100 iterations are required during training to maintain stability.

In practice, we also use efficient self-attention [14] and mixed-precision to significantly reduce run time and memory requirements. However, for a fair comparison, we exclude these performance improvements from all experiments except where explicitly stated otherwise.

## E. Qualitative Results

Figure 8 shows how LightGlue discards unmatched points and its early stopping mechanism on easy/medium/hard pairs. Figure 9 illustrates the matching output for LightGlue with SIFT [41], SuperPoint [16] and DISK [73] on some qualitative examples.

## References

- [1] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. In *CVPR*, 2016. 7, 11
- [2] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *CVPR*, 2017. 6, 15
- [3] Daniel Barath, Jiri Matas, and Jana Noskova. Magsac: marginalizing sample consensus. In *CVPR*, 2019. 6, 15
- [4] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *ECCV*, 2006. 2
- [5] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *TRO*, 32(6):1309–1332, 2016. 2
- [6] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging Properties in Self-Supervised Vision Transformers. In *ICCV*, 2021. 1
- [7] Luca Cavalli, Viktor Larsson, Martin Ralf Oswald, Torsten Sattler, and Marc Pollefeys. Handcrafted outlier detection revisited. In *ECCV*, 2020. 2
- [8] 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. *ICCV*, 2021. 1, 2, 5, 6, 7, 8, 12
- [9] Hongkai Chen, Zixin Luo, Lei Zhou, Yurun Tian, Mingmin Zhen, Tian Fang, David McKinnon, Yanghai Tsin, and Long Quan. ASpanFormer: Detector-Free Image Matching with Adaptive Span Transformer. In *ECCV*, 2022. 2, 6, 12
- [10] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training Deep Nets with Sublinear Memory Cost. *arXiv:1604.06174*, 2016. 6
- [11] Ondřej Chum, Jiří Matas, and Josef Kittler. Locally optimized RANSAC. In *Joint Pattern Recognition Symposium*, pages 236–243. Springer, 2003. 11
- [12] Ondřej Chum, Tomas Werner, and Jiri Matas. Two-view geometry estimation unaffected by a dominant plane. In *CVPR*, 2005. 11
- [13] Aaron Daniel Cohen, Adam Roberts, Alejandra Molina, Alena Butryna, Alicia Jin, Apoorv Kulshreshtha, Ben Hutchinson, Ben Zevenbergen, Blaise Hilary Aguera-Arcas, Chung ching Chang, Claire Cui, Cosmo Du, Daniel De Freitas Adwardana, Dehao Chen, Dmitry (Dima) Lepikhin, Ed H. Chi, Erin Hoffman-John, Heng-Tze Cheng, Hongrae Lee, Igor Krivokon, James Qin, Jamie Hall, Joe Fenton, Johnny Soraker, Kathy Meier-Hellstern, Kristen Olson, Lora Mois Aroyo, Maarten Paul Bosma, Marc Joseph Pickett, Marcelo Amorim Menegali, Marian Croak, Mark Díaz, Matthew Lamm, Maxim Krikun, Meredith Ringel Morris, Noam Shazeer, Quoc V. Le, Rachel Bernstein, Ravi Rajakumar, Ray Kurzweil, Romal Thoppilan, Steven Zheng, Taylor Bos, Toju Duke, Tulsee Doshi, Vincent Y. Zhao, Vinodkumar Prabhakaran, Will Rusch, YaGuang Li, Yanping Huang, Yanqi Zhou, Yuanzhong Xu, and Zhifeng Chen. LaMDA: Language Models for Dialog Applications. *arXiv:2201.08239*, 2022. 1
- [14] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *NeurIPS*, 2022. 2, 6, 7, 11, 13, 15
- [15] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal Transformers. In *ICLR*, 2019. 2
- [16] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabovich. SuperPoint: Self-supervised interest point detection and description. In *CVPR Workshop on Deep Learning for Visual SLAM*, 2018. 2, 6, 11, 13, 14, 15
- [17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019. 1, 3
- [18] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*, 2021. 1
- [19] 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, 12, 14
- [20] Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-Adaptive Transformer. In *ICLR*, 2020. 2, 4
- [21] Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially Adaptive Computation Time for Residual Networks. In *CVPR*, 2017. 3
- [22] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 2, 6
- [23] Christopher G Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, 1988. 2
- [24] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 6
- [25] Jared Heinly, Johannes L Schonberger, Enrique Dunn, and Jan-Michael Frahm. Reconstructing the World\* in Six Days \*(as Captured by the Yahoo 100 Million Image Dataset). In *CVPR*, 2015. 2
- [26] CVPR 2020 Image Matching Challenge. <https://www.cs.ubc.ca/research/image-matching-challenge/2020/>. Accessed June 15, 2023. 11, 14
- [27] CVPR 2021 Image Matching Challenge. <https://www.cs.ubc.ca/research/image-matching-challenge/>. Accessed June 15, 2023. 11
- [28] CVPR 2023 Image Matching Challenge. <https://www.kaggle.com/competitions/image-matching-challenge-2023/overview>. Accessed June 15, 2023. 11
- [29] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J Henaff, Matthew Botvinick, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver IO: A general architecture for structured inputs & outputs. In *ICLR*, 2022. 1
- [30] Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver: General Perception with Iterative Attention. In *ICML*, 2021. 2
- [31] Yuhe Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. Image Matching across Wide Baselines: From Paper to Practice. *IJCV*, 2020. 6
- [32] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020. 2
- [33] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The Efficient Transformer. In *ICLR*, 2020. 2
- [34] Viktor Larsson. PoseLib - Minimal Solvers for Camera Pose Estimation, 2020. 6, 7
- [35] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In *ICML*, 2019. 2

- [36] Xiaoxiao Li, Ziwei Liu, Ping Luo, Chen Change Loy, and Xiaoou Tang. Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade. In *CVPR*, 2017. 3
- [37] Yang Li, Si Si, Gang Li, Cho-Jui Hsieh, and Samy Bengio. Learnable Fourier Features for Multi-dimensional Spatial Positional Encoding. In *NeurIPS*, 2021. 3
- [38] Zhengqi Li and Noah Snavely. MegaDepth: Learning single-view depth prediction from internet photos. In *CVPR*, 2018. 5, 6, 11, 12, 14
- [39] Philipp Lindenberger, Paul-Edouard Sarlin, Viktor Larsson, and Marc Pollefeys. Pixel-Perfect Structure-from-Motion with Featuremetric Refinement. In *ICCV*, 2021. 2
- [40] Zhuang Liu, Zhiqiu Xu, Hung-Ju Wang, Trevor Darrell, and Evan Shelhamer. Anytime Dense Prediction with Confidence Adaptivity. In *ICLR*, 2022. 3, 4
- [41] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 2, 6, 12, 13, 14, 15
- [42] Anastasiya Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Working hard to know your neighbor’s margins: Local descriptor learning loss. In *NeurIPS*, 2017. 2
- [43] Dmytro Mishkin, Jiri Matas, and Michal Perdoch. Mods: Fast and robust method for two-view matching. *Computer Vision and Image Understanding*, 2015. 11
- [44] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *CVPR*, 2018. 2, 7
- [45] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *TRO*, 31(5):1147–1163, 2015. 2
- [46] Rémi Pautrat, Viktor Larsson, Martin R. Oswald, and Marc Pollefeys. Online invariance selection for local feature descriptors. In *ECCV*, 2020. 2
- [47] Malte Pedersen, Joakim Bruslund Haurum, Thomas B Moeslund, and Marianne Nyegaard. Re-identification of giant sunfish using keypoint matching. In *Northern Lights Deep Learning Workshop*, 2022. 1
- [48] Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019. 2, 5, 15
- [49] Markus N. Rabe and Charles Staats. Self-attention Does Not Need  $O(n^2)$  Memory. *arXiv:2112.05682*, 2021. 2
- [50] Filip Radenović, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondřej Chum. Revisiting Oxford and Paris: Large-scale image retrieval benchmarking. In *CVPR*, 2018. 5, 14
- [51] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018. 1, 3
- [52] Jerome Revaud, Philippe Weinzaepfel, César De Souza, Noe Pion, Gabriela Csurka, Yohann Cabon, and Martin Humenberger. R2D2: Repeatable and reliable detector and descriptor. In *NeurIPS*, 2019. 2
- [53] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *ECCV*, 2006. 2
- [54] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R Bradski. ORB: An efficient alternative to SIFT or SURF. In *ICCV*, 2011. 2
- [55] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, 2019. 1, 7, 11
- [56] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 1, 2, 6, 8, 11, 12, 14, 15
- [57] Paul-Edouard Sarlin, Mihai Dusmanu, Johannes L. Schönberger, Pablo Speciale, Lukas Gruber, Viktor Larsson, Ondrej Miksik, and Marc Pollefeys. LaMAR: Benchmarking Localization and Mapping for Augmented Reality. In *ECCV*, 2022. 1, 2
- [58] Paul-Edouard Sarlin, Ajaykumar Unagar, Måns Larsson, Hugo Germain, Carl Toft, Viktor Larsson, Marc Pollefeys, Vincent Lepetit, Lars Hammarstrand, Fredrik Kahl, and Torsten Sattler. Back to the Feature: Learning robust camera localization from pixels to pose. In *CVPR*, 2021. 1
- [59] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, Fredrik Kahl, and Tomas Pajdla. Benchmarking 6DOF outdoor visual localization in changing conditions. In *CVPR*, 2018. 1, 7, 12
- [60] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 2, 7, 14
- [61] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, 2016. 14
- [62] Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident Adaptive Language Modeling. In *NeurIPS*, 2022. 2, 4
- [63] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-Attention with Relative Position Representations. In *NAACL-HTL*, 2018. 3
- [64] Tianwei Shen, Zixin Luo, Lei Zhou, Runze Zhang, Siyu Zhu, Tian Fang, and Long Quan. Matchable image retrieval by learning from surface reconstruction. In *ACCV*, 2018. 6
- [65] 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 *CVPR*, 2022. 1, 2, 5, 7
- [66] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 1967. 5, 15
- [67] Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced Transformer with Rotary Position Embedding. *arXiv:2104.09864*, 2021. 3, 4, 13
- [68] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with Transformers. *CVPR*, 2021. 2, 6, 11, 12, 14, 15
- [69] Jiaming Sun, Zihao Wang, Siyu Zhang, Xingyi He, Hongcheng Zhao, Guofeng Zhang, and Xiaowei Zhou. OnePose: One-shot object pose estimation without CAD models. In *CVPR*, 2022. 1
- [70] 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. *TPAMI*, 2019. 12

- [71] Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. BranchyNet: Fast inference via early exiting from deep neural networks. *ICPR*, 2016. 3, 4
- [72] 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
- [73] Michał J Tyszkiewicz, Pascal Fua, and Eduard Trulls. DISK: Learning local features with policy gradient. In *NeurIPS*, 2020. 2, 6, 11, 13, 14, 15
- [74] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017. 1, 2, 3
- [75] Andrea Vedaldi and Brian Fulkerson. VLFeat: An open and portable library of computer vision algorithms. In *ACM international conference on Multimedia*, 2010. 14
- [76] Thomas Verelst and Tinne Tuytelaars. Dynamic convolutions: Exploiting spatial sparsity for faster inference. In *CVPR*, 2020. 3
- [77] Phil Wang. Bidirectional cross attention. <https://github.com/lucidrains/bidirectional-cross-attention>. 4
- [78] Qing Wang, Jiaming Zhang, Kailun Yang, Kunyu Peng, and Rainer Stiefelhagen. MatchFormer: Interleaving Attention in Transformers for Feature Matching. In *ACCV*, 2022. 2, 6, 12
- [79] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-Attention with Linear Complexity. *arXiv:2006.04768*, 2020. 2
- [80] Yan Wang, Zihang Lai, Gao Huang, Brian H. Wang, Laurens van der Maaten, Mark E. Campbell, and Kilian Q. Weinberger. Anytime Stereo Image Depth Estimation on Mobile Devices. *ICRA*, 2018. 3, 4
- [81] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. LIFT: Learned invariant feature transform. In *ECCV*, 2016. 2
- [82] Jiahui Zhang, Dawei Sun, Zixin Luo, Anbang Yao, Lei Zhou, Tianwei Shen, Yurong Chen, Long Quan, and Hongen Liao. Learning two-view correspondences and geometry using order-aware network. In *ICCV*, 2019. 2, 7
- [83] Lulin Zhang, Ewelina Rupnik, and Marc Pierrot-Deseilligny. Feature matching for multi-epoch historical aerial images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 182:176–189, 2021. 1