Efficient Linear Attention for Fast and Accurate Keypoint Matching

Suwichaya Suwanwimolkul
KDDI Research, Inc.
Japan
xsu-suwanui@kddi.com

Satoshi Komorita
KDDI Research, Inc.
Japan
sa-komorita@kddi.com

Figure 1: Our method versus the bigger SOTAs—SuperGlue and SGMNet—on speed (left), image matching (top), and 3D reconstruction (bottom).

Abstract
Recently Transformers have provided state-of-the-art performance in sparse matching, crucial to realize high-performance 3D vision applications. Yet, these Transformers lack efficiency due to the quadratic computational complexity of their attention mechanism. To solve this problem, we employ an efficient linear attention for the linear computational complexity. Then, we propose a new attentional aggregation that achieves high accuracy by aggregating both the global and local information from sparse keypoints. To further improve the efficiency, we propose the joint learning of the local feature matching and description. Our learning enables simpler and faster matching than Sinkhorn, often used in matching the learned descriptors from Transformers. Our method achieves competitive performance with only 0.84M learnable parameters against the bigger SOTAs, SuperGlue (12M parameters) and SGMNet (30M parameters), on three benchmarks, HPatch, ETH, and Aachen Day-Night.

CCS Concepts
• Computing methodologies → Matching.

Keywords
Sparse matching, learning-based matching, visual localization, SfM.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ACM Reference Format:
Suwichaya Suwanwimolkul and Satoshi Komorita. 2022. Efficient Linear Attention for Fast and Accurate Keypoint Matching. In Proceedings of the 2022 International Conference on Multimedia Retrieval (ICMR ’22), June 27–30, 2022, Newark, NJ, USA. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3512527.3531369

1 Introduction
Local feature matching is a fundamental step to achieve high performance in vision applications, such as visual localization [39], Structure from Motion (SfM) [41], and 3D reconstruction [17]. Classical local feature matching starts from extracting feature descriptors and keypoints that are robust against various transformations. The local feature matching relies both on the descriptive power of the descriptors and the geometrical consistency of keypoints. The similarity of descriptors is crucial in finding the nearest neighbors in feature space. Recent studies [22, 34, 36, 44, 55] focused on using deep learning techniques to boost the descriptive power of the descriptors. Transformers have become the core technology to realize state-of-the-art performance in sparse matching [5, 36]. Specifically, the Transformers originated from [49] were extended to learn the descriptiveness of sparse keypoints through self-attention and cross-attention [5, 36]. Self-attention encodes the descriptiveness by aggregating information within an image; cross-attention aggregates the information between the pair.

Nevertheless, the efficiency of these Transformers [5, 36, 49] remains a critical issue when the number of keypoints is large. The major cause of the lack of efficiency is the quadratic computational complexity of softmax attention in these Transformers. Although Chen, et al. [5] attempted to improve the complexity of [36] by

“In image matching, green & red lines indicate the correct and incorrect matches.
using seeds to represent groups of keypoints in matching, the complexity remains quadratic in the number of seeds: $O(N^2C)$ for $N$ denoting the number of seeds (or keypoints) and $C$ denoting feature dimensions. Nevertheless, another reason for the lack of efficiency is the descriptors matching after encoding by the Transformers. In order to match the encoded descriptors, the existing works \cite{5, 36} formulate the learning as an optimal transport problem where Sinkhorn algorithm \cite{9, 50} is used to match the descriptors. The computational cost of Sinkhorn, however, is very high. In matching 10k keypoints, Sinkhorn increases the runtime by an order of magnitude of the inference runtime of the Transformer \cite{5}.

To address this problem, we resort to using the linear attention \cite{18, 44} that offers linear computational complexity, \textit{i.e.}, $O(NC^2)$. However, it offered a lower or comparable accuracy than the regular softmax attention \cite{8}. Thus, we further improve the accuracy of the linear attention for sparse keypoint matching by proposing a new attentional aggregation, namely \textit{pairwise neighborhood attention}, to aggregate the local information from the neighborhoods of candidate matches in addition to the global information from the self-and cross-attention. Despite the accuracy improvement, the resulting complexity is kept low. Table 1 provides the time complexity of our proposed attention versus the SOTAs. To further improve the efficiency, we propose the joint learning of the description and sparse keypoint matching based on minimizing the feature distance. With the proposed learning, we can employ the feature distance-based matching such as \cite{26}, which is simpler and faster than Sinkhorn. Then, the performance can be improved further with efficient filtering based on the feature distance \cite{4}. This results in competitive performance with a low computational cost against the existing SOTAs, as shown in Fig. 1. Our contributions are:

- Pairwise neighborhood attention to boost the performance of existing linear attention.

- Joint learning of the sparse keypoint matching and description via minimizing feature distance, which improves the feature description and enables the efficient matching.

- Competitive performance while having only 0.84M learnable parameters, against the bigger SOTAs: SuperGlue \cite{36} (12M parameters) and SGMNet \cite{5} (30M parameters) on various benchmarks: HPatch, ETH, Aachen Day-Night.

### 2 Related works

#### 2.1 Learnable local feature matching

**Sparse matching** has recently gained a large improvement over the local feature detection by learning to match the detected keypoints. Notably, SuperGlue \cite{36} employed a Transformer similar to \cite{49} to exchange both visual and geometric information between the pair of images. Nevertheless, the Transformer has quadratic computational complexity in the number of keypoints. Recently SGMNet \cite{5} achieves the lower complexity by projecting $N$ keypoints into $K$ seeds. However, SGMNet still employs the softmax attention to aggregate the messages from seeds, which still results in, yet, a quadratic complexity $O(NKC + K^2C)$.

**Dense matching** \cite{22, 33, 34, 44, 55} aims to match descriptors in a pixel-wise manner. To enumerate all the possible matches, the

| Methods | Comp. Complex. | Attention Type |
|---------|----------------|----------------|
| SuperGlue \cite{36} | $O(N^2C)$ | Softmax attention \cite{50, 49} |
| SGMNet \cite{5} | $O(NKC + K^2C)$ | Seeding + Softmax attention \cite{5, 49} |
| Ours | $O(NC^2 + N_{max}C^2)$ | Linear Attention Eq. \cite{18} \cite{8} + Pairwise Neighborhood Attention Eq. \cite{5} |

$N$ denotes the number of keypoints; $C$ or $C'$ denotes the associated feature dimensions after linear projection; $K$ denotes the number of seeds in \cite{5}; $N_{max}$ denotes the size of the largest neighborhood; $N < N_{max}$.

#### 2.2 Graph matching

Graph matching aims to establish node-to-node correspondences between two or multiple graphs, which are found in various applications \cite{1, 6, 46–48}. Graph matching can be formulated as a Quadratic Assignment Problem (QAP) known to be NP-hard \cite{25, 28}. Early works \cite{15, 16, 21, 21, 45} improved the feasibility of QAP solvers. Recent works \cite{13, 35, 53} leverage the graph matching with deep learning, yet they become less feasible in handling more than hundreds of keypoints \cite{36}. Alternatively, the matching problem can be formulated as the optimal transport problem \cite{50} where the Sinkhorn algorithm can be used to efficiently find the solution \cite{9, 20, 30}. A recent study \cite{9} improved the algorithm to achieve the nearly linear runtime of $O(N^2/\epsilon^3)$, where $\epsilon$ is an error tolerance bound. However, the Sinkhorn algorithm still requires an extremely high time cost in matching thousands of keypoints or more, as evidenced by \cite{5, 36}.

#### 2.3 Efficient Attention with Linear Complexity

Regular Transformers \cite{10, 49} contain the powerful softmax attention. However, the softmax attention has the time complexity and memory scale quadratically with the input sequence length $N$, \textit{i.e.}, $O(N^2 \max(D, C))$ for $D$ and $C$ being the feature dimension of query and key. To solve this, Linear Transformers \cite{18, 43, 51} reduce the computational cost to the linear complexity $O(NDC)$ by computing the attention from the feature maps of dimensionality $C$, instead of the softmax attention. The feature maps offer lower or comparable accuracy than the softmax attention in applications such as speech recognition and image generation \cite{8, 18}; however, it can approximate well \textit{without imposing any constraints}, which is opposed to the previously developed techniques, \textit{e.g.}, restricting attention \cite{29}, employing sparsity prior \cite{9}, pooling-based compression \cite{31}. Others reduced the space complexity by sharing attention weights \cite{19} or allowing one-time activation storage in training \cite{7}. These approximations are insufficient for long-sequence problems.

Our work is inspired by the Linear Transformers such as \cite{18, 43, 51} that offer high efficiency. Meanwhile, the existing sparse
Figure 2: Visualization of our pairwise neighborhood attention vs. self- and cross-attention. Given (a) sets of neighborhoods & matching pairs, one can aggregate information with (b) our pairwise neighborhood attention to collect the pairwise, local neighborhood information. Meanwhile, (c) self-attention and (d) cross-attention focus more on the global information within and between images.

matching, i.e., SuperGlue [36] and SGMNet [5] employ the regular Transformer [10, 49] with quadratic computational complexity. LoFTR [44] also uses Linear Transformer, but for dense matching to match every pixel, which offers the denser and more accurate matches. However, these matches are not suitable for large-scale 3D reconstruction due to the high computational cost caused by the redundant matches [24].

3 Proposed method

Our main proposal is the efficient Linear Transformer for sparse matching, where we employ two different types of attentional aggregation to collect the global and local information. Self- and cross-attention are used to aggregate the global information. Then, our proposed pairwise neighborhood attention is used to aggregate the local information. The visualization of the two attention is in Fig. 2. The formulation is first discussed. Then, we present the proposed Transformer, where we used a local neighborhood selection to extract the local information. Then, we match the extracted features with distance-based matching and filtering in matching. Finally, we confirm our design choice with the time complexity.

3.1 Formulation

We consider the problem of finding the matched pairs between \( N \) and \( M \) keypoints in source and target images, \( I_s \) and \( I_t \). Let \( k^s, k^t \in \mathbb{R}^2 \) denotes the sets of keypoint locations in the 2D images. Our goal is to encode the associated descriptors \( x^s \in \mathbb{R}^{N \times D}, x^t \in \mathbb{R}^{M \times D} \) via a parametric function \( f_{\Phi}(\cdot) \) into new feature space such that it establishes the correct matching. This is formulated as finding the best set of parameters \( \Phi \) for the function \( f_{\Phi}(\cdot) \) via minimizing:

\[
L_{\text{triplet}} = \frac{1}{|M_e|} \sum_{c \in M_e} s_c \cdot R(\hat{x}_c^s, \hat{x}_c^t)
\]

where \( \hat{x}_c^s, \hat{x}_c^t = f_{\Phi}(x^s, x^t | k^c, k^c) \) and \( M_e \) is the set of ground truth correspondence. The subscription in \( \hat{x}_c^s \) denotes the coefficient selection where \( c \) denotes the selected indices. The triplet loss \( L_{\text{triplet}} \) encourages the descriptiveness of the encoded descriptors \( \hat{x}_c^s \) through the ranking loss \( R(\hat{x}_c^s, \hat{x}_c^t) \) by minimizing the distances of matched descriptors while maximizing the unmatched ones [27]:

\[
R(\hat{x}_c^s, \hat{x}_c^t) = \left[ D(\hat{x}_c^s, \hat{x}_c^t) - m_p \right]_+ + \left[ m_n - \min_{k \in c} \min_{k \in c} D(\hat{x}_c^t, \hat{x}_c^t) \right]_+
\]

where \( m_p \) and \( m_n \) are small constants to prevent the negative loss value. As \( L_{\text{triplet}} \) decreases, \( D(\hat{x}_c^s, \hat{x}_c^t) = ||\hat{x}_c^s - \hat{x}_c^t||_2 \) for \( c \in M_e \) will be minimized. Meanwhile, the distance of the wrong matching, i.e., \( \hat{x}_c^s \) vs. \( \hat{x}_k^t \) (or \( \hat{x}_k^s \) vs. \( \hat{x}_k^t \)) for \( k \notin M_e \), will be further enlarged.

Then, we weigh the distance minimization with confidence \( s_c \) for \( c \in M_e \). The confidence \( s_c \) is a scalar product between \( f_c^s \) and \( f_c^t \), where \( f_c^s, f_c^t \) are intermediate outputs from \( f_{\Phi} \), and \( f_c^s, f_c^t \) are column feature vectors:

\[
s_c = f_c^s^T f_c^t.
\]

The higher confidence \( s_c \) will penalize the feature distance more, resulting in higher descriptiveness, and the lower feature distance can lead to the higher similarity between \( f_c^s \) and \( f_c^t \), which encourages the matching between keypoints. The proposed loss aims at minimizing the feature distance, which is different from the loss used in the existing works (SuperGlue, SGMNet, and LoFTR) focusing on establishing as many matches as possible with the optimal transport layer, Sinkhorn. We replace Sinkhorn with feature-distance based matching and filtering (Section 3.5) for better efficiency.

We implement \( f_{\Phi} \) as a Linear Transformer shown in Fig. 4 (Section 3.3) where self- and cross-attention layers collect global information with linear attention [18]. Then, our pairwise neighborhood layers collect the local information from candidate matches. The number of candidate matches is controlled by the global information from the final cross-attention layer in Fig. 4. Thus, \( \hat{f}_c^s \) and \( \hat{f}_c^t \) in Eq. (3) are the output from this layer. Meanwhile, \( \hat{x}_c^s \) and \( \hat{x}_c^t \) are the combinations of global and local information from the final layer.

3.2 Efficient Linear Attention

Our Transformer \( f_{\Phi} \) contains multiple encoders. The function of each encoder is defined by their attention as shown in Fig. 3a. We adopt the architecture of the encoder from [44]. Our Transformer
consists of two types of attentional aggregation: (1) **linear attention** [18] and (2) our **pairwise neighborhood attention**.

**Linear Attention.** At first, we employ the linear attention similar to [18]. The architecture is provided in Fig. 3b. The inputs of attention are vectors resulting from the linear projection of the source and target descriptors with three matrices $W_Q \in \mathbb{R}^{D_K \times D_Q}$, $W_K \in \mathbb{R}^{D_K \times D_K}$, and $W_V \in \mathbb{R}^{D_K \times D_V}$. Let $Q = x_W Q$, $K = x_W K$, $V = x_W V$. Then, the output from the attention $p = \text{PairAtt}(x_s, x_t)$, is:

$$V' = [V']_{i} \in [N] = \frac{\phi(Q_i)^T \sum_{j \in [M]} \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j \in [M]} \phi(K_j)}_{i \in [N]} \quad (4)$$

where $\phi(\cdot) = \text{elu}(\cdot) + 1$. The subscription $i$ on a matrix returns a column vector of the $i$-th row, e.g., $K_j$ is a vector of size $D_K \times 1$.

**Pairwise Neighborhood Attention.** We employ the local information about candidate matches to support the global information captured by Eq. (4). Let $N_s^p$ and $N_t^p$ denote a pair of keypoint neighborhood, where $N_s^p$ is from the source, and $N_t^p$ from the target. Both center around seed points $p_1$, $p_2$ of the matching pair $p = (p_1, p_2)$. Thus, our attention incorporates the positional information of the matching neighborhood $N_s^p$ and $N_t^p$. The output $V_P = \text{PairAtt}(x_s, x_t | N_s^p, N_t^p)$, is:

$$V_P = [V_P]_{i} \in N_p^p = \frac{\phi(Q_i)^T \sum_{j \in N_s^p} \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j \in N_t^p} \phi(K_j)}_{i \in N_s^p} \quad (5)$$

Any element outside $N_s^p$ is filled with zero value, i.e., $V(k)^P = 0$ for $k \notin N_s^p$. If there is more than one pair, the output is a superposition of $V_P$, i.e., $V_P = \sum_{p \in P} V_P$ where $P$ is the set of matching pairs. The architecture is provided in Fig. 3c. An example of the keypoint neighborhood $N_s^p$ and $N_t^p$ of a matching pair $p$ is provided in Fig. 2a. The visualization of the attentional aggregation is provided in Fig. 2b, which results in the collection of local information in the pairwise neighborhood. Furthermore, the dominating cost of $\text{PairAtt}(\cdot)$ is $O(N_{max}C^2)$ which linearly increases with the largest neighborhood size $N_{max}$.

The derivation is in Section 3.6.

### 3.3 Network Architecture

Our network architecture is provided in Fig. 4. Each layer consists of an encoder layer (Fig. 3a) with linear or pairwise neighborhood attention, which results in linear attention layer and pairwise neighborhood layer. We use the linear attention Eq. (4) to perform the self- and cross-attention to collect the global information through intra- and inter-relationship between descriptors. The self-attention layer updates its message by

$$\hat{x}^t = \text{LinAtt}(x^t, x^t), \quad \hat{x}'^t = \text{LinAtt}(x'^t, x'^t) \quad (6)$$

The cross-attention layer updates messages with information collected from the inter-relationship between two descriptors [36]:

$$\hat{x}^t = \text{LinAtt}(x^t, x'), \quad \hat{x}' = \text{LinAtt}(x', x^t) \quad (7)$$

Then, we employ our pairwise neighborhood attention Eq. (5) to form the pairwise neighborhood layer that aggregates the local information around candidate matches. We construct a pairwise neighborhood layer using Eq. (5). Given $(N_s^p, N_t^p)$ extracted by the neighborhood selection (Section 3.4), the message update is

$$\hat{x}^P = \text{PairAtt}(x^t, x^t | N_s^p, N_t^p), \quad \hat{x}'^P = \text{PairAtt}(x', x' | N_s^p, N_t^p) \quad (8)$$

where any element outside $N_s^p$ is filled with zero value, i.e., $\hat{x}^t(k)^P = 0$ for $k \notin N_s^p$ and $\hat{x}'(k)^P = 0$ for $k \notin N_t^p$. Finally, $\hat{x}^t = \sum_{p \in P} \hat{x}^P$ and $\hat{x}' = \sum_{p \in P} \hat{x}'^P$. Then, we perform $L_1$ loop updates between self- and cross-attention layers, and $L_2$ loop updates over the pairwise neighborhood layer. Unlike [5, 36, 44], we did not employ any positional encoder. In addition, our first layer ($l = 1$) has additional linear weights $W_Q', W_K',$ and $W_V'$ to reduce the dimension of input descriptors into the lower dimensions $D_Q, D_K$, and $D_V$, leading to the lower computational cost in the multi-head attention of the subsequent layers [49]. Here, we set $D_Q, D_K, D_V = C'$.

### 3.4 Local Neighborhood Selection

We track the local information from candidate matches for pairwise neighborhood layers as follows. $\bar{f}$ and $\bar{f}'$ from the final cross-attention layer are used to extract the matching pairs. Then, we construct the set of hypothesis matching seeds $\mathcal{P}$ while ensuring that the seeds well spread across images. Finally, we extract the set of neighborhoods compatible with the matching seeds to construct the keypoint neighborhood, i.e., $N_s^p$ and $N_t^p$ for $p \in \mathcal{P}$ for Eq. (5).
Hypothesis Matching Seeds Selection. At first, we establish the set of seed points with high matching confidence and well spread around the image. Let \( \mathcal{M} \) denotes a set containing the matching pair extracted by the distance ratio algorithm \( \text{Dist}(\cdot) \) [26] where \( \theta \) is an appropriate threshold. Let \( \text{distratio}(i, j) \) denotes the distance ratio value corresponding to the match \( (i, j) \). Then, the set of matching seeds is defined as follows:

\[
P := \{ (i, j) \mid \text{distratio}(i, j) > \text{distratio}(i, k), \quad \text{for } k \in \text{Nei}(j|\mathcal{R}), \forall (i, j) \in \mathcal{M} \}
\]  

(9)

where \( \text{Nei}(\cdot) \) denotes the index set of neighboring keypoints within radius \( R \). We follow [4] to employ the seed separation condition where the match index \( (i, j) \) is selected to the set of matching seeds \( \mathcal{P} \), if it has the highest distance ratio among its local neighbors, to ensure that the matching seeds are well spread.

Local Neighborhood Set Selection. To include candidate matches that are geometrically consistent with the matching seed \( p \in \mathcal{P} \), we collect the points that locate in a similar neighborhood, following [4, 37]. Let \( (k_p^p, k_p^g) \) denote the location of the matched keypoints from source to target corresponding to the matching seed \( p \in \mathcal{P} \). The local neighborhood set \( \mathcal{N}_p \) is defined as:

\[
\mathcal{N}_p := \{ (p_1, p_2) \mid \|k_{p_1} - k_p^p\| \leq \lambda R_s \text{ and } \|k_{p_2} - k_p^g\| \leq \lambda R_t, \quad \forall (p_1, p_2) \in \mathcal{M} \}
\]  

(10)

where \( R_s \) and \( R_t \) are the radii to control the coverage of neighboring points around the matching seed \( p \) in \( I_s \) and \( I_t \), respectively. \( \lambda \) is a hyperparameter that regulates the overlapping between neighborhoods. Then, the pair of keypoint neighborhood \( (\mathcal{N}_p^p, \mathcal{N}_p^g) \) is

\[
\mathcal{N}_p^p := \{ i \mid (i, j) \in \mathcal{N}_p \}, \quad \mathcal{N}_p^g := \{ j \mid (i, j) \in \mathcal{N}_p \}
\]  

(11)

The pair of keypoint neighborhood \( \mathcal{N}_p^p, \mathcal{N}_p^g \) will be used to define the aggregation in Eq. (5) to emphasize the area of candidate matches.

3.5 Feature distance-based matching & filtering

Given the descriptors \( \hat{x}_s, \hat{x}_t \) from our Transformer, one can obtain the set of match pairs \( \mathcal{M}_c \) by distance ratio thresholding such as [26]. However, the fixed thresholding value tends to overly restrict the candidates. Thus, we employ the similar procedure to Section 3.4 to include the candidate matches compatible with \( \mathcal{M}_c \):

### Table 2: Complexity of Linear Attention

| Step | Operation | Input | Output | Complexity |
|------|-----------|-------|--------|------------|
| 1. Numerator | \( \sum_{i \in \mathcal{N}_p} \phi(K_x)^T \) \( \phi(Q_y)^T K_{x} \) | two \([C \times 1]\) \( K_{x} = [C \times C] \) | \( K_{y} = [C \times 1] \) | \( O(MC^2) \) |
| 2. Denominator | \( \sum_{i \in \mathcal{N}_p} \phi(K_x)^T \) \( \phi(Q_y)^T K_{x} \) | two \([C \times 1]\) \( K_{x} = [C \times C] \) | \( K_{y} = [C \times 1] \) | \( O(C^2) \) |
| 3. Final | \( Q_{c_{1}} \) \( \text{Den} \) | \([1 \times C], [1 \times 1]\) \( V_{y}^T = [1 \times C] \) | \( V_{y} = [N_c \times C] \) | \( O(N_c C) \) |

(1) Extract hypothesis matching seeds \( \mathcal{P} \), with Eq. (9) where \( \hat{x}_s, \hat{x}_t \) are used to construct the set of matching pairs \( \mathcal{M}_c \).

(2) Extract the set of candidate matches, i.e., \( \mathcal{N}_c \mid c \in \mathcal{P} \) where \( \mathcal{N}_c \) is extracted with Eq. (10).

Filtering. We employ the filtering process of AdaLAM [4] (without refitting) to improve the performance by verifying the local affine consistency in each \( \mathcal{N}_c \) with highly parallel RANSACs [4, 14]. The filtering scales well with the high number of keypoints (>10,000).

The resulting matches \( \{\mathcal{N}_c \mid c \in \mathcal{P} \} \) could contain many wrong matches; however, using our network with such procedure (denoted as distance matching or DM) provides comparable performance to AdaLAM [4] in most cases (see Table 4). The filtering process in AdaLAM (Filter) improves the performance further, yet the performance gain is more obvious with our pairwise neighborhood layer. It can be shown that the runtime cost of the feature distance-based matching and filtering is much lower than Sinkhorn that is used by SuperGlue and SGMNet from Table 6, and using linear transformer with Sinkhorn does not lead to higher matches (see Section D).

3.6 Time Complexity

Time complexity of the two attentional aggregation, linear attention Eq. (4) and our pairwise neighborhood attention Eq. (5), is provided. Our derivation is based on the size of \( \mathcal{Q}, \phi(Q) \in \mathcal{R}^{N \times C} \) and \( K, V, \phi(K) \in \mathcal{R}^{M \times C} \).

Linear Attention. The complexity of Eq. (4) is derived as in Table 2. The analysis starts from the computations in numerator Step 1, denominator Step 2, and final division Step 3. The total complexity is \( O(MC^2 + C^2 + M + C^2 + C^2 + N_c) \approx O(MC^2) \).

Pairwise neighborhood attention. The time complexity of Eq. (5) is derived similarly in Table 3. The only difference is the range of summation operations. The total complexity is \( O(|\mathcal{N}_p^p|C^2 + C^2 + |\mathcal{N}_p^g| + C^2 + |\mathcal{N}_p^p|C^2) \). Let \( N_{\text{max}} \) denote the size of the largest neighborhood among \( \mathcal{N}_p^g \) for \( p \in \mathcal{P} \), i.e., \( N_{\text{max}} = \max_{p \in \mathcal{P}} |\mathcal{N}_p^g| \). Thus, the dominating complexity is \( \approx O(N_{\text{max}}C^2) \).

Total. Combining the two, we obtain \( O(MC^2 + N_{\text{max}}C^2) \approx O(MC^2) \) for \( N_{\text{max}} \ll M \). Table 1 provides the comparison with SOTAs.
4 Experimental Results

We provide the ablation study and the scalability of our work against SOTAs. Then, our method is evaluated on several practical scenarios, i.e., image matching, 3D reconstruction, and visual localization. The implementation and training are provided in suppl. A. Additional numerical and visual results are provided in arXiv version.

Comparative methods. Our work is compared with 1) Sparse matching: SuperGlue [36] and SGMNet [5]. 2) Dense matching: LoFTR [44], Patch2Pix [55], NCNet [33]. 3) Local features: SuperPoint [11], R2D2 [32], D2-Net [12], and ASLFeat [27], where the standard matching, e.g., MNM-matching or Lowe’s Thresholding, is used for matching local features. 4) Keypoint filtering: AdaLAM [4] and OANet [54]. We report either results from the original papers or derived from the official implementations with default settings unless otherwise specified. In each table, we highlight the top two or top three and underline the best result.

In this paper, we apply our method to match the local features of SuperPoint [11] where keypoints are limited to 2k for image matching, 10k for 3D Reconstruction, and 4k for visual localization.

4.1 Ablation Study

This study uses localization accuracy on Aachen Day-Night [38, 40].

Ablation Study on the Proposed Networks. We provide the ablation study on the impact of each component in our network Fig. 4, i.e., linear attention layer, pairwise neighborhood layer, feature distance-based matching and filtering, and feature dimensions.

From Table 4, our Pair.Neigh. with both linear attention layer (LA) and pairwise neighborhood layer (PN) offers the higher accuracy than Linear, that uses only linear attention layer, in most cases, from 1k to 4k keypoints. Using filtering (Filt.) further improves the accuracy, especially for Pair.Neigh. Next, we compare the model size defined by \( \# \text{dim} \). The large-size model (L) offers the higher robustness, yet our small model (S) offers the better trade-off with the computational cost. Since the goal is to achieve the high efficiency, our small model is used in subsequent comparisons against SOTAs.

Configuration of Local Neighbor Selection. We consider three configurations: • Pair.-w/oSep.-Inp. omits the seed separation in Eq. (9) & uses \( x^s, x^t \), instead of \( f^s, f^t \). in Fig. 4 for Local Neigh. Selection. • Pair.-w/oSep. omits the seed separation in Eq. (9) and uses \( f^s, f^t \) as input for Local Neigh. Selection. • Pair.Neigh. follows all the steps, similar to No.5 in Table 4.

Table 5: Configurations of local neighborhood selection.

| Methods | Config. of Local Neigh. Selection | Accuracy @ 0.25m, 2° |
|---------|----------------------------------|----------------------|
| LA PN DM Filt. \( \# \text{dim} \) size | 1k 2k 3k 4k |
| Pair.-w/oSep.-Inp. \( x^s, x^t \) | ✓ ✓ 256 | 79.6 71.4 79.6 76.5 |
| Pair.-w/oSep. \( f^s, f^t \) | ✓ ✓ S 64 | 78.6 72.4 78.6 80.6 |
| Pair.Neigh. \( f^s, f^t \) | ✓ ✓ S | 78.6 75.5 74.5 77.6 |

Table 5 shows that our Pair.Neigh. and Pair.-w/oSep. offer the highest accuracy when the number of keypoints is high (>2k). Meanwhile, Pair.-w/oSep.-Inp. offers higher robustness when the number of keypoints is low. Notice that all of them offer higher accuracy than using only Linear Attention (No.4). We report the results of three configurations in the next SOTAs comparison. The detailed results across all the error tolerances, i.e., (0.25m, 2°), (0.5m, 5°), and (5m, 10°), and visualization are provided in suppl. E.

4.2 Overall Scalability

We confirm the overall performance of our work on time and memory cost when running inference in Fig. 5. All the reported results are based on the official settings and run in real-time on a Titan RTX. In the official SuperGlue and SGMNet, the Sinkhorn iteration is set to 100. We also compare against SuperGlue-10 and SGMNet-10 where Sinkhorn iteration set to 10. We also report our large-size model (Our.Pair.Neigh.-L), with the same settings as No.6 in Table 4.

Time Cost. From Fig. 5a, our time cost is remarkably lower than SuperGlue and SGMNet and is linear with the number of keypoints (#Kpt.) Specifically, at 16k keypoints, our method is about 28 and 9 times faster than the official SuperGlue and SGMNet.
Figure 6: Image matching. Our method versus SOTAs—local features, dense matching, and sparse matching—on HPatches dataset [3]. We report MMA across error thresholds (1-10 px), the number of matches (#Matches), the number of learnable parameters (#Param.), and Total Time †.

Table 6: Individual runtime: (a) Transformer & (b) Matching.

| Methods                  | Pub.    | #Matches | InLRatio | #Param. | Total Time (ms) |
|--------------------------|---------|----------|----------|---------|-----------------|
| D2-Net [12]              | CVPR'19 | 2.50 × 10^7 | 0.42     | 7.64 × 10^6 | 487.06          |
| RD2 [32]                 | NFSP'19 | 2.05 × 10^7 | 0.74     | 1.04 × 10^6 | 1755.16         |
| SuperPoint [11]          | CVPR'18 | 1.08 × 10^7 | 0.65     | 1.30 × 10^6 | 34.30           |
| NCNet [33]               | CVPR'17 | 1.48 × 10^7 | 0.46     | 21.3 × 10^6 | 297.18          |
| Patch2Pix [55]           | CVPR'21 | 1.26 × 10^7 | 0.76     | 31.6 × 10^6 | 453.07          |
| LoFTR [44]               | CVPR'21 | 4.71 × 10^7 | 0.87     | 11.6 × 10^6 | 212.98          |
| SuperPoint + SuperGlue   | ICCV'21 | 8.32 × 10^6 | 0.84     | 12.0 × 10^6 | 115.14          |
| SuperPoint + SGMMNet     | ICCV'21 | 8.66 × 10^6 | 0.80     | 31.1 × 10^6 | 116.11          |
| SuperPoint + Our Pair-w/o-Sep-Inp | † | – | 7.15 × 10^7 | 0.80 | 0.841 × 10^6 | 68.49 |
| SuperPoint + Our Pair-w/o-Sep | – | – | 6.97 × 10^7 | 0.81 | 0.841 × 10^6 | 68.58 |
| SuperPoint + Our Pair.Neigh. | – | – | 7.11 × 10^7 | 0.80 | 0.841 × 10^6 | 73.52 |

Sink: Sinkhorn, DF: Distance Matching & Filtering.

21 and 3 times faster than SuperGlue-10 and SGMMNet-10. Our large model has higher runtime yet is much faster than the SOTAs.

Memory Cost. In Fig. 5b, we measure the memory cost using the peak of memory consumption similar to [5]. Our method consumes lower memory than SuperGlue and SGMMNet even when the number of keypoints is as low as 1k. When the number of keypoints ≥ 4k, our GPU memory cost is 50% and 20% lower than SuperGlue and SGMMNet, respectively. Our large-size model consumes slightly higher memory, which is the advantage of linear attention [18].

Accuracy vs. Keypoints. Fig. 5c demonstrates the impact on visual localization accuracy (0.25m, °) as the number of keypoints increases. For our work, the impact on visual localization accuracy is more obvious as the keypoints increase. Meanwhile, SuperGlue and SGMMNet only slightly improve with the number of keypoints. Our work outperforms both when the number of keypoints is ≥ 3k.

Runtime of Individual Parts. Table 6 provides the time cost of the individual parts: (a) Transformer and (b) Matching. Our runtime increases with a much lower rate for both parts. Our large-size model (L) behaves similarly. This confirms the superior efficiency of our linear attention against the regular softmax attention of the SOTAs, as well as the faster speed of our distance-based matching and filtering over Sinkhorn used in SuperGlue and SGMMNet.

4.3 Image Matching

This section we compare the image matching performance between our method against the SOTA local features, dense matching, and sparse matching on HPatches [3] following the protocol of [12].

Local Features. In Fig. 6, our MMA curves surpass all the SOTA local features D2-Net, RD2, and SuperPoint with MNM-matching (baseline). Notice that our Total Time also includes the feature extraction runtime of SuperPoint.

Dense Matching. In overall and viewpoint changes, our methods achieve similar MMA curves to LoFTR. LoFTR also offers the highest InLRatio and #Matches. However, Total Time of LoFTR is about three times that of our method. Patch2Pix also provides high MMA, but the Total Time is about six times of ours.

Sparse Matching. Our methods offer slightly lower MMA and #Matches, and a similar InLRatio compared to SuperGlue and SGMMNet. However, we can realize this performance with 1/10 #Param. and half Total Time. Our study on scalability (Section 4.2) reveals the efficiency improvement by our methods.

4.4 3D Reconstruction

Evaluation. 3D reconstruction is a keypoint-consuming application; thus, we report the matching runtime (Match. Time) to indicate the efficiency alongside other indicators. We follow the ETH evaluation [42] where sparse and dense reconstruction are performed by the SM and MVS from COLMAP [41]. The dense points are from the dense reconstruction. Our method is compared against the official SuperGlue and SGMMNet and SuperGlue-10 and SGMMNet-10 in Table 7. Because the official implementations take too much runtime on the medium-size datasets, we compare our method against SuperGlue-10 and SGMMNet-10 in Table 8.

Results on Small-size ETH. From Table 7, our methods provide the longest Track. Len., with lower Reproj. Error, and comparable Dense Points to SuperGlue, SuperGlue-10, SGMMNet, and SGMMNet-10. Our Match. Time is about 10 times and 3 times faster than SuperGlue-10 and SGMMNet-10, respectively. Compared with SuperGlue and SGMMNet with the official settings, the efficiency gap of our work becomes larger. Our Match. Time is at least 20 times and about 8 times faster than SuperGlue and SGMMNet, respectively.

Results on Medium-size ETH. From Table 8, our method provides the longest Track. Len. and low Reproj. Error in most cases. Our method offers moderate Dense Points with lower runtime than
Table 7: Small-size ETH. Our methods versus the official Superglue and SGMNet, Superglue-10 and SGMNet-10.

| Datasets       | Methods                          | Track. Len. | Reproj. Error | Sparse Points | Dense Points | Match. Time (sec) |
|----------------|----------------------------------|-------------|---------------|---------------|--------------|-------------------|
| Herzjew 8 images | SuperGlue [36]†                  | 4.45        | 0.821         | 8.5k          | 1.14M        | 4.71 × 10^3       |
|                | SuperGlue-10 [36]                | 4.44        | 0.821         | 8.5k          | 1.15M        | 2.68 × 10^3       |
|                | SGMNet [5]                      | 4.16        | 0.860         | 9.6k          | 1.14M        | 1.43 × 10^3       |
|                | SGMNet-10 [5]                   | 4.14        | 0.955         | 9.8k          | 1.14M        | 0.66 × 10^2       |
|                | Our Pair.-w/oSep-Ins.           | 5.54        | 0.881         | 7.3k          | 1.14M        | 0.24 × 10^2       |
|                | Our Pair.-w/oSep.               | 5.54        | 0.872         | 7.2k          | 1.15M        | 0.24 × 10^2       |
|                | Our Pair.Neigh.                 | 4.53        | 0.873         | 7.3k          | 1.14M        | 0.23 × 10^2       |
| Fountain 11 images | SuperGlue [36]†                | 5.14        | 0.961         | 11.4k         | 1.84M        | 7.90 × 10^2       |
|                | SuperGlue-10 [36]               | 5.14        | 0.960         | 11.4k         | 1.83M        | 4.42 × 10^2       |
|                | SGMNet [5]                      | 4.93        | 0.959         | 11.9k         | 1.84M        | 2.38 × 10^2       |
|                | SGMNet-10 [5]                   | 4.92        | 0.966         | 11.9k         | 1.84M        | 1.12 × 10^2       |
|                | Our Pair.-w/oSep-Ins.           | 5.17        | 0.909         | 10.0k         | 1.83M        | 0.41 × 10^2       |
|                | Our Pair.-w/oSep.               | 5.16        | 0.905         | 10.0k         | 1.83M        | 0.42 × 10^2       |
|                | Our Pair.Neigh.                 | 5.14        | 0.903         | 10.0k         | 1.83M        | 0.41 × 10^2       |
| South-Building 128 images | SuperGlue [36]†               | 7.90        | 0.947         | 114.4k        | 12.53M       | 402.23 × 10^2     |
|                | SuperGlue-10 [36]               | 7.88        | 0.949         | 114.8k        | 12.51M       | 228.43 × 10^2     |
|                | SGMNet [5]                      | 6.95        | 0.979         | 132.2k        | 12.93M       | 108.67 × 10^2     |
|                | SGMNet-10 [5]                   | 6.97        | 0.981         | 131.2k        | 12.33M       | 48.79 × 10^2      |
|                | Our Pair.-w/oSep-Ins.           | 6.31        | 0.837         | 94.8k         | 12.40M       | 14.83 × 10^2      |
|                | Our Pair.-w/oSep.               | 6.31        | 0.832         | 94.2k         | 12.42M       | 13.86 × 10^2      |
|                | Our Pair.Neigh.                 | 6.27        | 0.836         | 95.1k         | 12.45M       | 13.29 × 10^2      |

† SuperGlue with its official setting (Sinkhorn iter. = 100).
‡ SGMNet with its official setting (Sinkhorn iter. = 100, num. seeds = 128).

SuperGlue-10 and SGMNet-10. The baseline [26] provides the lowest reprojection error. However, our methods provide longer tracking length and higher #Reg. Img. to AdaLAM and the baseline in most cases. Our Dense Points is also higher than these two approaches and is comparable with SuperGlue-10 and SGMNet-10, suggesting the similar visual quality of the 3D reconstruction. Our Match. Time is about 3 times and twice faster than SuperGlue-10 and SGMNet-10, due to the lower detected keypoints by SuperPoint.

4.5 Visual Localization

Evaluation. We employ the Aachen Day-Night [38, 40] to demonstrate the effect on visual localization. We follow the evaluation protocols of Visual Localization Benchmark and report the percent of successfully localized images.

Results. From Table 9, our method gives the competitive accuracy at 0.25m, 2°: our Pair.Neigh,§ gives the highest accuracy among the methods that employ SuperPoint as input features, i.e., the sparse matching (SuperGlue, SGMNet) and the keypoint filtering (OANet, AdaLAM). Meanwhile, our Pair.-w/oSep. offers higher accuracy than SGMNet but lower than SuperGlue. Our performance drops as the error threshold becomes less restrictive and is on par with AdaLAM. This suggests that our method is more accurate but less robust, as our works tend to provide less matches than SuperGlue and SGMNet. Nevertheless, our methods can achieve this with a much lower #Param. and #dim. Compared to the SOTA local features, we use only 4k keypoints but give the closest performance to ASLFeat.

5 Summary

To improve the efficiency of existing SOTA Transformers in sparse matching applications, we propose efficient attention that offers linear time complexity and high accuracy by aggregating the local and global formation. To keep the high efficiency, we proposed to train the Transformer with the joint learning of the sparse matching and description optimized based on the feature distance. This enables the use of feature distance-based matching and filtering that is simpler and faster than Sinkhorn, which results in high accuracy and extremely low runtime. Extensive experiments indicate a significant improvement in efficiency against the bigger SOTAs.
of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 4950–4960. https://doi.org/10.1109/CVPR.2017.526

[46] Lorenzo Torresani, Vladimir Koltunovor, and Carsten Rother. 2008. Feature Correspondence via Graph Matching: Models and Global Optimization. In Proceedings of the European Conference on Computer Vision (ECCV). Springer Berlin Heidelberg, 596–609.

[47] Lorenzo Torresani, Vladimir Koltunovor, and Carsten Rother. 2013. A Dual Decomposition Approach to Feature Correspondence. IEEE Transactions on Pattern Analysis and Machine Intelligence 35, 2 (2013), 259–271. https://doi.org/10.1109/TPAMI.2012.105

[48] Nikolai Uler and Bjorn Ommer. 2017. Deep Semantic Feature Matching. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 5929–5938. https://doi.org/10.1109/CVPR.2017.628

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Illia Polosukhin. 2017. Attention is All you Need. In Advances in Neural Information Processing Systems. I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ce243547de9e19db0536c1ec4a85aa-Paper.pdf

[50] Cédric Villani. 2008. Optimal Transport: Old and New.

[51] Sinong Wang, Belinda Z. Li, Madihan Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-Attention with Linear Complexity. arXiv:2006.04768

[52] Zihao Wang, Xueyi Li, and Zhen Li. 2021. Local Representation is Not Enough: Guided Pixel-Level Correspondences. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2684–2693. https://doi.org/10.1109/CVPR.2020.00284

[53] Andrei Zanfir and Cristian Sminchisescu. 2018. Deep Learning of Graph Matching. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2684–2693. https://doi.org/10.1109/CVPR.2018.00284

[54] Jialui Zhang, Dexei Sun, Zixin Luo, Anbang Yao, Lei Zhou, Tianwen Shen, Yurong Chen, Long Quan, and Hongen Liao. 2019. Learning Two-View Correspondences and Geometry Using Order-Aware Network. In International Conference on Computer Vision (ICCV).

[55] Qunjie Zhou, Torsten Sattler, and Laura Leal-Taixe. 2021. Patch2Pix: Epipolar-Guided Pixel-Level Correspondences. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

A Parameter settings

In the first layer, we set \( W^Q_x \), \( W^K_y \), and \( W^V_z \) to linearly project from high to low dimensional space. Given that the dimensionality of SuperPoint is 256, the linear projection maps from 256 → 64, and for any subsequent layer, \( D \), \( D_Q \), \( D_K \), \( D_V \) = 64. The encoded descriptors with 64 dimensions are reshaped to \( 8 \times 8 \) for multi-head attention (the number of heads = 8). For the local neighborhood selection, we set \( \theta = 1.0 \) for Lowes’ Thresholding. For Eq. (10), we use \( \lambda = 2 \) for image matching and 3D reconstruction and \( \lambda = 3 \) for localization, where \( R_x \), \( R_y \), \( R_z \) = \( \sqrt{\frac{H \times W}{1024}} \).

B Training datasets

We train the proposed model with Megadepth [23] datasets using the same image scenes as [36]. For each epoch, we sample 100 pairs per scene and select the pair with overlapping scores in range [0.5,1]. Given an image pair, we extract the local features using SuperPoint [11] and sample 1024 keypoints per image. To generate the ground truth correspondence, we use the camera poses with depth maps corresponding to the two images to project the keypoints. The reprojection distances of the keypoints is used to determine ground truth matches and unmatchable points. Following [36], a pair of keypoints are considered ground truth matches if they are mutual nearest with a reprojection distance lower than 3 pixels; otherwise, it is labeled as unmatchable. We further filter out pairs if the ground truth matches are fewer than 50. Our data generation produces around 200k training pairs in total.

Table 10: Sinkhorn vs. distance matching and filtering on #matches and inlier ratio.

| Methods | #Matches | Inl.Ratio |
|---------|---------|----------|
| SuperPoint + SuperGlue [36] | 8.32 × 10² | 0.84 |
| SuperPoint + Our Linear. + Sink-h100 | 6.67 × 10² | 0.76 |
| SuperPoint + Our Linear. + Dist.Match. + Filt. | 6.75 × 10² | 0.82 |

Figure 7: Comparison on (a) time and (b) memory cost.

Learning. We use Adam optimizer with learning rate of \( 10^{-3} \) with exponential decay rate of 0.99992. We train for 10 epochs.

C Evaluation protocols & settings

C.1 3D Reconstruction

Exhaustive matching that matches the global information between all possible images is used to retrieve images for the small datasets, Herzs jes and Fountain. Meanwhile, NetVLAD [2] is used to retrieve the top 20 nearby images from South-Building, Madrid Metropolis, Gendarmenmarkt, and Tower of London. Sparse and dense reconstruction are performed by the SM and MVS from COLMAP [41].

C.2 Visual Localization

According to the protocols of Visual Localization Benchmark\(^*\), we provided the costumed features and performed image registration with COLMAP [41]; then, the localization is performed. We use the Aachen Day-Night datasets [38, 40] whose goal is to match images with extreme day-night changes for 98 queries.

D Sinkhorn vs. Distance Matching & Filtering

Table 10 provides the comparison between using Sinkhorn versus distance matching & filtering with the linear transformer. Following [36], we have trained the linear transformer with Sinkhorn with optimal transport loss (similar settings to Section B). Using Sinkhorn does not provide higher #matches nor inlier ratios, yet Sinkhorn requires much higher time cost in Fig. 7.

E Additional Ablation Studies

In this section, we provide the additional results to confirm our conclusion in Section 4.1. We provide the results of the localization accuracy across all the three error tolerances, i.e., (0.25m, 2\(^°\)), (0.5m, 5\(^°\)), (5m, 10\(^°\)) on Aachen Day-Night [38, 40].

E.1 Components in the Proposed Network

Table 11 demonstrates the impact of components in the proposed network (Fig. 4) on the localization accuracy across all the three error tolerances. Our Pair.Neigh. (No Filt.), without any filtering

\(^*\) https://www.visuallocalization.net/
Table 11: Impact of components in the proposed network (Fig. 4) on localization accuracy.

| No. | Methods          | Network Architecture | Accuracy @ 0.25m, 2° | Accuracy @ 0.5m, 5° | Accuracy @ 5m, 10° |
|-----|------------------|----------------------|----------------------|---------------------|---------------------|
|     |                  |                      | 1k 2k 3k 4k          | 1k 2k 3k 4k         | 1k 2k 3k 4k         |
| 1   | AdaLAM [4]       | - - ✓ ✓              | 41.8 71.4 79.6 76.5  | 52.0 80.6 85.7 86.7 | 61.2 87.8 92.9 95.9 |
| 2   | Our Linear (No Filt.) | L₁=10 ☒ ☒ ☒ ✓       | 61.2 72.4 73.5 76.5  | 71.4 81.6 81.6 87.8 | 75.5 87.8 91.8 95.9 |
| 3   | Our Pair.Neigh. (No Filt.) | L₁=8 L₂=2 ☒ ☒ ☒ ✓   | 66.3 74.5 73.5 76.5  | 74.5 83.7 83.7 83.7 | 82.7 91.8 93.9 93.9 |
| 4   | Our Linear       | L₁=10 ☒ ☒ ☒ ✓       | 48.0 73.5 76.5 77.6  | 59.2 82.7 84.7 84.7 | 64.3 90.8 92.9 98.9 |
| 5   | Our Pair.Neigh. | L₁=10 L₂=2 ☒ ☒ ☒ ✓  | 58.2 72.4 78.6 80.6  | 65.3 85.7 86.7 86.7 | 73.5 93.9 95.9 95.9 |
| 6   | Our Pair.Neigh.-L | L₁=8 L₂=2 ☒ ☒ ☒ ✓   | 63.3 73.5 74.5 78.6  | 70.4 85.7 87.8 87.8 | 76.5 94.9 95.9 96.9 |

LA: Linear Attention layer, PN: Pairwise Neighborhood Attention layer, DM: Distance Matching, Filt: Filtering process, #dim: Encoded feature dimension, size: Network size, large (L) or small (S).

Our Linear (No Filt.), Our Pair.Neigh. (No Filt.), Our Linear, and Our Pair.Neigh. (No Filt.) can match more and cover more areas than Linear and Linear (No Filt.) that uses only linear attention layers.

Figure 8: The impact of pairwise neighborhood attention and linear attention layers on keypoint matching. We provide the keypoint matching samples by our Linear, Linear (No Filt.), Pair.Neigh., and Pair.Neigh. (No Filt.). Our Pair.Neigh. and Pair.Neigh. (No Filt.) can match more and cover more areas than Linear and Linear (No Filt.) that uses only linear attention layers.

process, offers higher accuracy than Linear when the number of keypoints is low. This could be due to the combination of pairwise neighborhood (PN) and linear attention (LA). Meanwhile, our Linear uses the linear attention layers only. Employing the filtering process (Filt) can improve the performance further, yet the performance gain is more obvious with Pair.Neigh. The large-size model (Pair.Neigh.-L) provides the highest accuracy in most cases. Fig. 8 demonstrates the combination of pairwise neighborhood and linear attention layers versus using linear attention layers only on the keypoint matching. We provide the output samples of keypoint matching resulted from our Linear (No Filt.), Pair.Neigh. (No Filt.), and Linear and Pair.Neigh. The matched keypoints are highlighted.
### Table 12: Impact of configurations in Local Neighborhood Selection (Section 3.4) on Localization Accuracy.

| No. | Methods                  | Local Neigh. Selection | Accuracy @ 0.25m, 2° | Accuracy @ 0.5m, 5° | Accuracy @ 5m, 10° |
|-----|--------------------------|------------------------|----------------------|---------------------|---------------------|
|     |                          | Input                  | 1k 2k 3k 4k          | 1k 2k 3k 4k         | 1k 2k 3k 4k         |
| 1   | Our Pair.-w/oSep-Inp.    | $x^s, x^t$             | ✓ 59.2 75.5 74.7 77.6| ✓ 69.4 84.7 84.7 84.7| ✓ 75.5 93.9 94.9 94.9|
| 2   | Our Pair.-w/oSep.        | $\hat{f}^s, \hat{f}^t$| ✓ 57.1 71.4 74.5 78.6| ✓ 65.3 82.7 86.7 86.7| ✓ 71.4 93.9 94.9 95.9|
| 3   | Our Pair.Neigh.         | $\hat{f}^s, \hat{f}^t$| ✓ 58.2 72.4 78.6 80.6| ✓ 65.3 85.7 86.7 86.7| ✓ 73.5 93.9 95.9 95.9|

Figure 9: The output samples from Local Neighborhood Selection (Section 3.4) configured according to our Pair.-w/oSep-Inp., Pair.-w/oSep., and Pair.Neigh. The local neighborhood selection of Pair.-w/oSep-Inp. depends on the input descriptors $x^s, x^t$. However, the local neighborhood selection of our Pair.-w/oSep. and Pair.Neigh. depends on $\hat{f}^s, \hat{f}^t$. Nevertheless, Pair.-w/oSep. ignores the separation condition; thus, it can collect more matching seeds than Pair.Neigh.

#### E.2 Local Neighborhood Selection.

Table 12 provides the impact on localization accuracy due to the configuration of Local Neighborhood Selection (Section 3.4). Our Pair.-w/oSep-Inp. offers higher robustness when the number of keypoints are low. Meanwhile, Pair.Neigh. and Pair.-w/oSep. offer the highest accuracy when the number of keypoints are high, as $\hat{f}^s, \hat{f}^t$ are resulted from the aggregation of information. Using both pairwise neighborhood attention and linear attention layers offers higher accuracy than using only linear attention layer in most cases.

Fig. 9 demonstrates the output samples from the Local Neighborhood Selection by our Pair.-w/oSep-Inp., Pair.-w/oSep. and Pair.Neigh. at 2k keypoints. The local neighborhood selection of Pair.-w/oSep-Inp. depends on the input descriptors $x^s, x^t$. Meanwhile, the local neighborhood selection of Pair.-w/oSep. and Pair.Neigh. depends on $\hat{f}^s, \hat{f}^t$. Since Pair.-w/oSep. ignores the separation condition, it can collect more matching seeds than Pair.Neigh. Despite having less matching seeds, Pair.Neigh. employs the seed separation condition which enforces the matching seeds to be spreading across the images. The spreading of matching seeds shows to be an important factor in gaining high accuracy in localization according to Table 12.

#### E.3 Additional Results.

Additional numerical and visual results for image matching, 3D reconstruction, and visual localization are provided in the appendix of arXiv version.