

# ClusterGNN: Cluster-based Coarse-to-Fine Graph Neural Network for Efficient Feature Matching

Yan Shi<sup>1\*</sup> Jun-Xiong Cai<sup>1\*</sup> Yoli Shavit<sup>2</sup> Tai-Jiang Mu<sup>1</sup> Wensen Feng<sup>3†</sup> Kai Zhang<sup>1†</sup>

<sup>1</sup>Tsinghua University      <sup>2</sup>Faculty of Engineering, Bar Ilan University      <sup>3</sup>Huawei Technologies

shi-y20@mails.tsinghua.edu.cn, caijunxiong000@163.com, yolisha@gmail.com

taijiang@tsinghua.edu.cn, fengwensen@huawei.com, zhangkai@sz.tsinghua.edu.cn

## Abstract

Graph Neural Networks (GNNs) with attention have been successfully applied for learning visual feature matching. However, current methods learn with complete graphs, resulting in a quadratic complexity in the number of features. Motivated by a prior observation that self- and cross- attention matrices converge to a sparse representation, we propose ClusterGNN, an attentional GNN architecture which operates on clusters for learning the feature matching task. Using a progressive clustering module we adaptively divide keypoints into different subgraphs to reduce redundant connectivity, and employ a coarse-to-fine paradigm for mitigating miss-classification within images. Our approach yields a 59.7% reduction in runtime and 58.4% reduction in memory consumption for dense detection, compared to current state-of-the-art GNN-based matching, while achieving a competitive performance on various computer vision tasks.

## 1. Introduction

Finding correspondences between images is an essential task for many computer vision applications such as Simultaneous Localization and Mapping (SLAM) [14, 20], Structure-from-Motion (SfM) [32, 43] and camera pose estimation [48]. Given a pair of images, correspondences can be established through point-to-point feature matching. Classical pipelines typically obtain correspondences with a nearest neighbor (NN) search of feature descriptors and reject outliers based on their match score or using a mutual NN check. Such methods focus only on the local similarity between feature descriptors while ignoring geometric information and the global receptive field.

Recent works [5, 30, 35] have proposed to learn the task of feature matching using graph neural networks (GNNs)

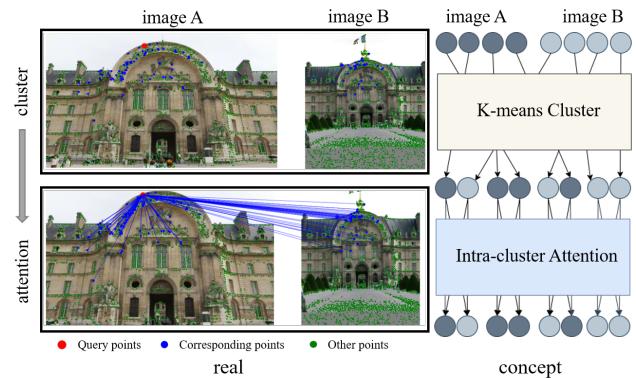


Figure 1. Sparse attention in cluster-based feature matching. Each keypoint interacts only with points within its cluster (proposed method) instead of interacting with all keypoints as in current GNN-based feature matching [30].

and attention. In SuperGlue [30], Transformer [40] based GNNs are applied on the complete graph of keypoints within (intra graph) and between (inter graph) images. Each node is represented with an encoded keypoint descriptor and updated using self- and cross- multi-head attention, while alternating between the intra- and inter- complete graphs, respectively. Learning complete graphs with attention suffers from a computational and memory complexity which is quadratic in N, where N is the number of keypoints. However, keypoints typically show a strong correlation with just a small number of points (sparse adjacency matrix). Furthermore, in the context of feature matching, a large portion of keypoints are non-repeatable and irrelevant for matching. A complete graph representation is thus redundant and results in wasteful attention-based message passing.

Efforts to reduce the quadratic complexity of attention mainly focused on self-attention. For example, reducing the attention dimension by splitting the input sequence into local windows [23] or by approximating attention with kernels [8]. However, these works are less suitable for fea-

\*These authors contributed equally to this work.

†Corresponding authors.

ture matching, where we are required to perform self- and cross- attention on features within and between images, respectively. Inspired by the Routing Transformer [27], we propose a coarse-to-fine cluster-based GNN to learn the feature matching task with a lower redundancy and computational complexity. We extract query and key features of keypoints across images to classify the points with strong correlation into the same cluster and establish local graphs using points from the same category. Each point interacts only with points in the same local graph resulting with a sparser attention computation (Fig. 1). Since clustering keypoints across images may lead to an erroneous grouping within images, we take a coarse-to-fine approach and first divide the points into a small number of major clusters which are gradually divided into multiple smaller clusters. We evaluate our method on multiple tasks, namely: relative pose estimation, homography estimation and visual localization. Our model achieves state-of-the-art accuracy across tasks, with a significant improvement in efficiency for dense detection (59.7% and 58.4% reduction in runtime and memory, respectively).

In summary, our contributions are as follows:

1. We present a learnable, coarse-to-fine clustering method to establish local graphs for feature matching, which reduces the spread of redundant information and makes message passing more effective.
2. We introduce the ClusterGNN architecture, an attentional GNN for learning the feature matching task using gradually forming clusters, approximating attention on complete graphs.
3. The proposed ClusterGNN method achieves state-of-the-art results on various tasks, with a significant reduction in runtime and memory consumption on dense detection (59.7% and 58.4%, respectively) compared to the current leading feature matching method (SuperGlue).

## 2. Related Work

**Visual Feature matching.** Matching features between a pair of images is used in different computer vision applications, such as visual localization and relative pose estimation. In classical pipelines, keypoints are first detected and described using hand-crafted methods such as SIFT [18] and ORB [28]. The nearest neighbor of each descriptor is obtained based on Euclidean distances, followed by filtering of incorrect matches using mutual consistency check, Lowe’s ratio test [18], matching scores and neighborhood consensus.

In recent years, deep learning methods have been employed for improving different aspects of the feature match-

ing pipeline. Learning-based feature detectors and descriptors such as SuperPoint [10], ASLFeat [19] and R2D2 [25], were proposed for learning more robust, discriminative and repeatable representations of keypoints. Other methods focused instead on the matching and filtering tasks [30, 36, 44]. A significant improvement in matching accuracy was recently achieved by SuperGlue [30], an attentional GNN architecture, which establishes complete graphs over keypoints within and between images and update their representations with an attention-based message passing. Albeit, since the runtime and memory complexity of attention is quadratic in the number of nodes, operating on the complete graph does not scale well. This in turns limits usability of SuperGlue, especially when matching a large number of points. In this work, we build on the success of SuperGlue and propose a sparser alternative to reduce the propagation of redundant messages, achieving a significant decrease in runtime and memory while preserving matching performance.

**Efficient Attention.** The attention mechanism was popularized through the Transformer architecture [40], achieving state-of-the-art results across different natural language processing and computer vision tasks [4, 11, 12]. In the context of attentional GNNs, Transformers can be viewed a graph-like model operating on the complete graph of tokens, which are updated using self- and cross- attention [15, 41]. Since Transformers (and attentional GNNs operating on complete graphs) are hampered by the quadratic complexity of attention in the sequence length (number of nodes), different methods were recently proposed to sparsify connections or linearize the attention complexity.

In [5], a small set of reliable nodes are established as seeds to reduce the cost of attention. In [6, 22, 23], data independent or fixed sparsity patterns were proposed to bound temporal dependencies, such as local or strided attention. Drawing inspiration from CNN networks, those works suggested to apply attention within a fixed local neighborhood, which bounds the complexity, but limits the ability to establish long range interactions. In the context of feature matching, such methods may degrade performance since the distribution of keypoints is not regular, and the two matched images may have large scale and viewing angle differences. Other approaches [2, 7, 42] proposed to approximate attention by either lowering the sequence dimension through pooling or lowering the attention matrix dimension using low-rank methods. Such approximations, however, include assumptions which are less appropriate for feature matching.

Inspired by content-based sparse attention [16, 27], which use data-driven methods to cluster tokens and operate within clusters, we propose a learned clustering module, which utilizes k-means to construct sub-graphs in a coarse-to-fine module, and passes messages within each local clus-

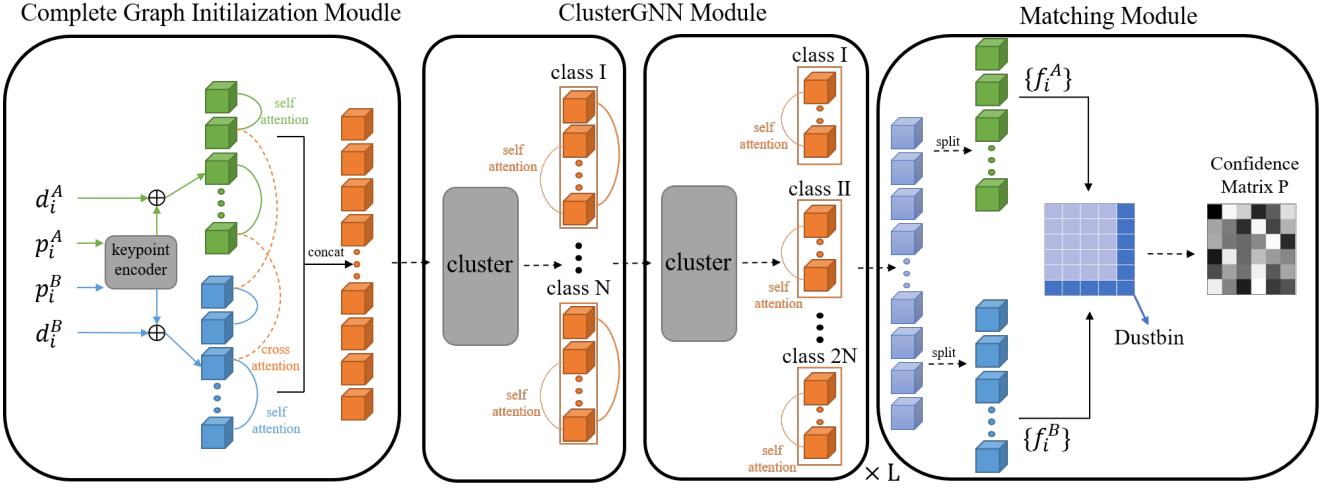


Figure 2. **The ClusterGNN architecture.** Our model is composed on three components. A Complete Graph Initialization Module (Section 3.3.1) first constructs and updates complete inter- and intra- graphs with attentional GNN layers. In order to leverage on the inherent sparsity of keypoint attention maps, a ClusterGNN Module (Section 3.3.2), learns to hierarchically partition the joint complete graph into smaller sub-graphs before applying attention-base updates. Finally, a Matching Module (Section 3.3.3), establishes the matching probability matrix using dot product and the Dual Softmax operator. A learnable dustbin is further appended to account for non-matching keypoints.

ter graph for saving memory and computations.

### 3. Method

#### 3.1. Problem Definition

Given an image pair  $(\mathcal{I}_a, \mathcal{I}_b)$  and extracted keypoints  $(\mathcal{K}_a, \mathcal{K}_b)$  with corresponding confidence scores  $(\mathcal{C}_a, \mathcal{C}_b)$  and descriptors  $(\mathcal{D}_a, \mathcal{D}_b)$ , the feature matching problem is defined as pairing keypoints which match in real-world coordinates:

$$\mathcal{M}_{a,b} = \{(i,j) | \| \mathcal{T}_a(\mathcal{K}_a^{(i)}) - \mathcal{T}_b(\mathcal{K}_b^{(j)}) \| \leq \epsilon\} \quad (1)$$

where  $\mathcal{T}_x$  represent the function which transfer pixel coordinates to world coordinates. In practice,  $\mathcal{T}_x$  is usually obtained by homography matrix estimation based on predicted matches. When faced with large differences between camera views and long-term environmental changes, traditional methods which rely on NN search often produce wrong matches. Recent GNN-based methods apply attention in an iterative manner to remove matching-unrelated environmental noise and learn global geometric distribution information in order to achieve optimal matching and increase matching robustness.

#### 3.2. Motivation

Existing works [30, 35] which learn the feature matching task with attention-based GNNs, use densely connected graphs. However, as shown in Fig.3a, a large portion of keypoints are non-repeatable and irrelevant for feature matching. In addition, the respective self- and cross- attention

matrices tend to converge to sparse matrices (visualized in Fig.3b), where most of the attention values are distributed around zero (Fig.3c). Therefore, it is important to design a sparse structure for efficient feature matching.

In order to learn over sparse graphs and reduce redundancy, we extract query and key features of keypoints to classify strongly correlated points into the same cluster (Fig.3d). The points in each cluster are used to build a small sub-graph. Using attention, we can then pass information and update keypoints representation within each local sub-graph. Due to different descriptor statistics between images, direct classification may cause points from the same image to be wrongly classified into the same cluster. In order to address this problem, we propose a cluster-based coarse-to-fine paradigm and apply attentional GNN layers within clusters for learning feature matching between two sets of feature points and their associated descriptors.

#### 3.3. Network Architecture

Similarly to [30], our method first applies a **Complete Graph Initialization Module** (Section 3.3.1), which constructs complete graphs over encoded keypoints and descriptors within and between images (intra- and inter-graphs respectively), and updates them using attention. Instead of learning multiple attentional GNN layers over these complete graphs, we design a **ClusterGNN module** (Section 3.3.2), which learns to hierarchically partition the complete graph into smaller sub-graphs, and then applies the attention update mechanism within these graphs. Fi-

nally, the matching probability between keypoints is computed with a **Matching Module** (Section 3.3.3), based on the dot product of updated feature representations and the Dual-Softmax operator [26, 35]. An overview of our proposed method is shown in Fig. 2.

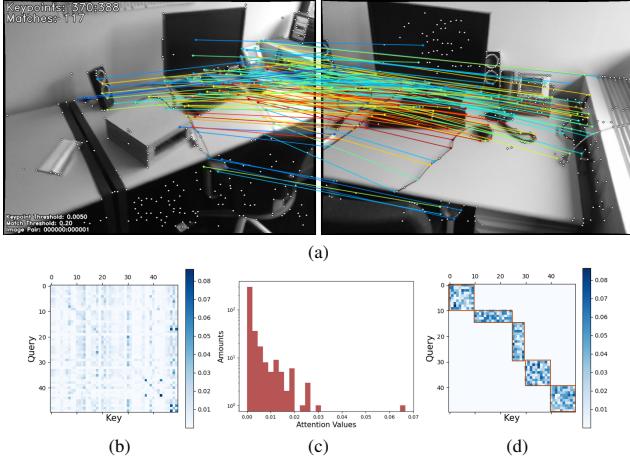


Figure 3. (a) Visualization of feature matching between two images. Keypoints and matches are represented by white points and colored lines, respectively where color varies between green to red according to the matching confidence (higher in red). (b) Cross attention matrix of 50 sampled keypoints from two images. (c) The distribution of attention values in an example row in the attention matrix. (d) The cluster attention matrix. The attention matrix within each cluster is framed with a red box.

### 3.3.1 Complete Graph Initialization Module

Given input keypoints  $\mathcal{K}$ , confidence scores  $\mathcal{C}$  and local descriptors  $\mathcal{D}$ , we generate a joint representation as in [30] by adding descriptors and encoded keypoints and confidence scores. We then construct complete intra (within images) and inter (between images) graphs over the keypoints in  $(\mathcal{I}_a, \mathcal{I}_b)$  and apply multi-head attention as in [40] and [30] for updating node representation. This construction and initialization process can be expressed as follows:

$$\begin{aligned} \mathcal{D}_a^0, \mathcal{D}_b^0 &= CA(SA(KE(\mathcal{I}_a)), SA(KE(\mathcal{I}_b))), \\ \mathcal{D}_a^i, \mathcal{D}_b^i &= CA(SA(\mathcal{D}_a^{i-1}), SA(\mathcal{D}_b^{i-1})), \\ KE(\mathcal{I}_x) &= \mathcal{D}_x + MLP(\mathcal{K}_x \oplus \mathcal{C}_x), \end{aligned} \quad (2)$$

where  $SA/CA$  stands for self/cross attention (introduced below),  $KE$  is the keypoint encoding layer, and  $\oplus$  denotes the concatenation operator. For two input feature sets  $(\mathcal{F}_a, \mathcal{F}_b)$ , we implement attention-based GNNs to pass and

aggregate messages between nodes, as follows:

$$CA(\mathcal{F}_a, \mathcal{F}_b) = \mathcal{F}_{a,b} + MLP(\mathcal{F}_{a,b} \oplus Att(\mathcal{F}_{a,b}, \mathcal{F}_{b,a})),$$

$$Att(\mathcal{F}_a, \mathcal{F}_b) = softMax\left(\frac{Q_a K_b^T}{\sqrt{dim}}\right) V_b,$$

$$(Q_x, K_x, V_x) = Linear_{(Q, K, V)}(\mathcal{F}_x), x \in \{a, b\}, \quad (3)$$

$$SA(\mathcal{F}_x) = CA(\mathcal{F}_x, \mathcal{F}_x), x \in \{a, b\}. \quad (4)$$

In practice, we use multi-head attention [40] to improve the expressivity. In addition, all MLP operators are followed with batch normalization and ReLU before the last layer.

### 3.3.2 Cluster GNN Module

**Cluster-based Sparse Attention** The computational bottleneck of the attentional GNNs (Equation 3) is due to the matrix multiplication operation between the query and the key matrices. However, since keypoints are likely to be correlated to only a small number of points (Fig. 3b), it is desirable to operate over local sub-graphs (where the corresponding matrices are much smaller). The Cluster-GNN module implements a cluster-based sparse attention to approximate self- and cross- attention over complete graphs, as follows:

$$\begin{aligned} \widehat{Att}(\widetilde{\mathcal{F}}) &= M \cdot softMax\left(\frac{\widetilde{Q} \widetilde{K}^T}{\sqrt{dim}}\right) \widetilde{V}, \\ \widetilde{\mathcal{F}} &= \mathcal{F}_a \cup \mathcal{F}_b, \\ \widetilde{Q} &= Q_a \cup Q_b, \widetilde{K} = K_a \cup K_b, \widetilde{V} = V_a \cup V_b \end{aligned} \quad (5)$$

where  $M = \{m_{ij}\}$  is the cluster matrix.  $m_{ij} = 1$  when  $query_i$  (the  $i$ th *query* vector of  $\widetilde{\mathcal{F}}$ ) and  $key_j$  (the  $j$ th *key* vector of  $\widetilde{\mathcal{F}}$ ) belong to the same cluster. Note that with this formulation, we no longer distinguish between features from different images (like the self/cross attention in Section 3.3.1), and directly operate on their union. While other works also apply the K-NN or top-k operators to determine the index matrix  $M$ , they do not apply well to cross attention in matching tasks.

**Learnable Hierarchical Clustering.** The key to our clustering method is the definition of cluster centers and the discrepancy function. We note that the attention map itself provides a means for measuring similarity, where the attention weights between two features are determined according to their *query* and *key* vectors, as follows:

$$\alpha_{i,j} = \frac{\exp(query_i \cdot key_j)}{\sum_k \exp(query_i \cdot key_k)} \quad (6)$$

Since the *query* and *key* vectors are different linear projections derived from the input features (Eq. 3),  $\alpha_{i,j} \neq \alpha_{j,i}$  is satisfied in most cases. Thus, instead of directly clustering

the feature vectors, we propose to cluster the union space of the *query* and *key* vectors in order to get a better sparse approximation.

Given  $k$  known cluster vectors  $\{c_i\}$  and features  $\{f_i | f_i \in Q \cup K\}$ , the discrepancy function is defined as follows:

$$dis(c_i, f_j) = 1 - \frac{c_i \cdot f_j}{\|c_i\| \|f_j\|}, \quad (7)$$

Using this function, we assign each feature with a cluster center index  $c_{id}$  based on its closest cluster and construct the cluster Matrix  $M$  in Equation 5. Note that, in practice,  $M$  does not need to be calculated explicitly.

During the training process, we initialize clusters centers with the k-means algorithm, and update them as follows:

$$c_i = \beta c_i + (1 - \beta) \left( \sum_{cid_j=i} f_j \right) \quad (8)$$

where  $\beta$  is a weight to balance the old and new value of  $c$ , which is set to 0.99. During test time, the trained cluster centers are directly used to cluster the input features, thus avoiding the iterative process used in traditional clustering algorithms.

By using hierarchical clustering, we reduce the theoretical complexity of attention from  $O(n^2)$  to  $O(n^2k^{-2})$ , where  $n$  is the number of features and  $k$  is the number of clusters. Since  $k$  controls the tradeoff between the quality of approximation (better for small values of  $k$ ) and the amount of calculations (smaller for large values of  $k$ ), its value should be set with consideration. In theory, the best balance between clustering semantics and computational complexity can be reached when we set cluster number  $k = \sqrt{n}$  [27]. However, the input features processed in [27] come from the same image and share similar statistics. For the image matching problem, the input features are keypoint descriptors extracted from two images, which may present significant style differences. Directly using the fixed  $\sqrt{n}$  as the number of clusters can thus lead to over-segmentation, so that each cluster only contains features from the same image. In order to mitigate this behavior we apply clusterGNN with an increasing number of clusters, realizing a coarse-to-fine semantic clustering (Fig. 4). We further carry an ablation study to guide the choice of  $k$  and evaluate its effect.

### 3.3.3 Matching Module

We establish a matching confidence matrix  $\mathcal{C}$ , by applying the dot product operator between the features computed with the ClusterGNN module,  $\{\mathcal{F}_a^o(i), i = 1, 2, \dots, n\}$  and  $\{\mathcal{F}_b^o(i), i = 1, 2, \dots, m\}$ :

$$\mathcal{C} = \{\mathcal{C}_{i,j} = \mathcal{F}_a^o(i) \cdot \mathcal{F}_b^o(j)\} \quad (9)$$

We further add an extra learnable dustbin channel on both row and column of the confidence matrix  $\mathcal{C}$  ( $\tilde{\mathcal{C}}$ ) in order to detect keypoints that are not matched. We adopt the

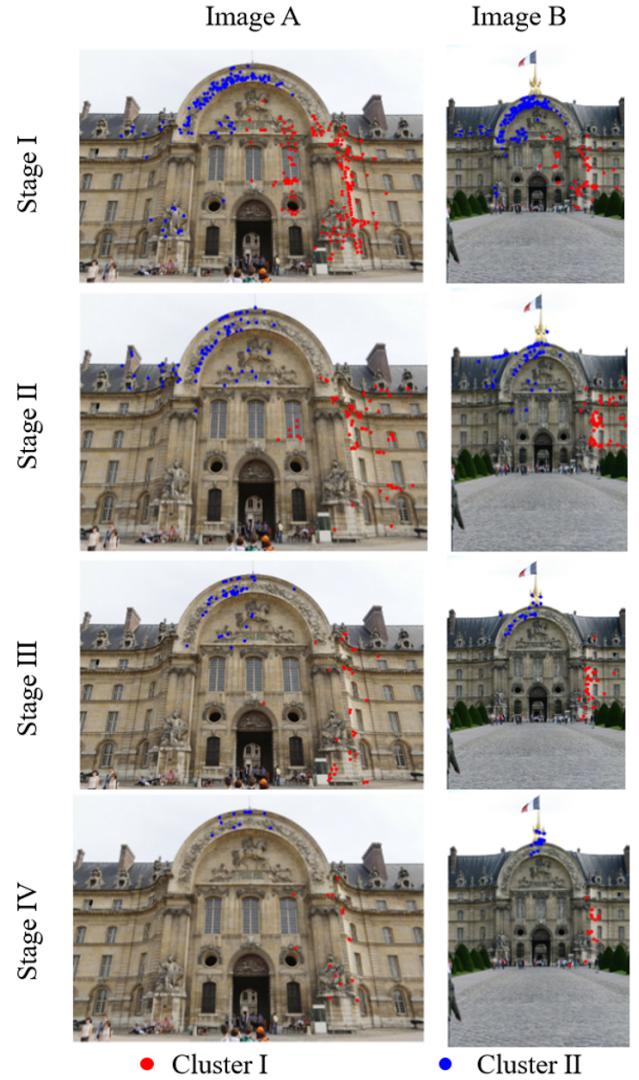


Figure 4. Visualization of hierarchical clustering. We show two clusters (colored in red and blue, respectively) at four different stages of ClusterGNN. The hierarchical clustering enables coarse-to-fine grouping

Dual-softmax operator [26, 35] for computing the matching probability matrix  $\mathcal{P}$ , by applying the log-softmax operator on both the row and column dimensions of  $\tilde{\mathcal{C}}$ , as follows:

$$\mathcal{P}_{i,j} = \logSoftMax(\tilde{\mathcal{C}}_{i,:})_j + \logSoftMax(\tilde{\mathcal{C}}_{:,j})_i. \quad (10)$$

At test time, we predict the matches using the argmax operator and a mutual check mechanism.

### 3.4. Loss

The ground truth matches set  $\mathcal{M}$  and non-matching keypoints set  $(\overline{\mathcal{M}}_a, \overline{\mathcal{M}}_b)$  are supervised with the cross projection error (less than 3 pixels for matches and more than 5

pixels for non-matches). The matching loss  $L_m$  is defined as:

$$\begin{aligned} \mathcal{L}_m = -|\mathcal{M}| \sum_{(i,j) \in \mathcal{M}} \mathcal{P}_{ij} - \\ |\overline{\mathcal{M}}_a| \sum_{i \in \overline{\mathcal{M}}_a} \mathcal{P}_{i,m+1} - |\overline{\mathcal{M}}_b| \sum_{j \in \overline{\mathcal{M}}_b} \mathcal{P}_{n+1,j} \end{aligned} \quad (11)$$

In order to achieve a better clustering effect and enable semi-supervision, we further introduce a clustering loss:  $\mathcal{L}_c$ :

$$\mathcal{L}_c^t = \sum_{i,j} \|c_i^t - f_j^t\|. \quad (12)$$

Our total loss consists of the matching loss  $\mathcal{L}_m$  and the clustering losses  $\mathcal{L}_c^t$  computed across different clustering stages:

$$\mathcal{L} = \mathcal{L}_m + \gamma \sum_{t \in \{1, 2, \dots, L\}} \mathcal{L}_c^t, \quad (13)$$

where  $\gamma$  is set to 0.1 for balancing the two losses,  $t$  indicates the clustering stage and  $L$  is the total number of stages.

### 3.5. Implementation Details

We train our model on the MegaDepth dataset [17], a large outdoor dataset, including 1M internet images from 196 different locations and sparse 3D models computed with COLMAP [32]. It also uses multi-view stereo to generate depth maps. We select training image pairs as in [13, 21], based on the overlap rate from the SfM co-visibility and resize images so that the larger edge is of size 1600.

Our model is optimized using Adam with an initial learning rate of  $1 \times 10^{-4}$ . We implement attention with a four heads multi-head attention throughout. Our Graph Initialization module consists of three stacked self/cross attention GNN layers. In order to reduce the GPU memory cost, we further chunk query vectors into four parts for this module. Our Cluster GNN module involves four stages with a different number of clusters, set as  $\{16, 32, 64, 128\}$ , respectively, where each stage consists of two cluster-based GNN attention layer.

## 4. Experiments

Feature matching is a challenging task due to different factors such as occlusions, illumination and weather changes. We evaluate ClusterGNN on three different tasks, which heavily rely of feature matching, namely: pose estimation, homography estimation and visual localization.

We compare our method with NN-search, SGMNet [5] and SuperGlue [30], using both hand crafted features (SIFT [18]) and learning-based features (ASLFeat [19] and SuperPoint [10]). For SGMNet, we retrain it ASLFeat using the official training code. For SuperGlue, since the official training code is not available and since its public model

(denoted as SuperGlue\*) was trained on MegaDepth [17], Oxford and Paris datasets [24]), we retrain it with different features following the training method described in the SuperGlue paper. We report both our own implementation and the official results reported in the SuperGlue paper. We also provide analysis of computation and memory efficiency. All the reported experiments were run on a Tesla P-100 GPU with 16GB memory.

### 4.1. Pose Estimation

We use the YFCC100M [38] dataset to evaluate the performance of ClusterGNN on the pose estimation task. This dataset provides a sparse reconstruction from SfM [21, 32, 33] and ground truth poses. Following [3, 30, 44, 45], we report the AUC of pose errors at different thresholds ( $5^\circ, 10^\circ, 20^\circ$ ), where the pose errors are computed by the maximum angular differences between estimated results and ground truth in rotation and translation. For pose estimation, we use RANSAC as a post-processing tool to compute the essential matrix from predicted matches.

As shown in Table 1, ClusterGNN outperforms other methods with ASLFeat. When using SuperPoint, our method outperforms our re-implemented SuperGlue and SGMNet, with a slight degradation compared to the official version. When using SIFT, our method outperforms SuperGlue, with a slight degradation compared to SGMNet. These results demonstrate the approximation and denoising abilities of ClusterGNN. Qualitative inspection (Fig. 5), further shows that ClusterGNN detects a larger number of true matches compared to other methods, resulting in an improved pose estimation.

### 4.2. Homography Estimation

We evaluate our method on the homography estimation task using the HPatches dataset [1]. HPatches consist of 52 sequences that exhibit large illumination changes and 56 sequences under significant viewpoint changes. We follow the setup proposed in Patch2Pix [47], and report the percentage of correctly estimated homographies whose average corner error distance is below 1/3/5 pixels. For all compared methods, we apply the OpenCV RANSAC toolbox to estimate homography matrix. The local keypoints of SuperPoint [10] and ASLFeat [19] are both tested in detail. To make a fair comparison, we choose 4k keypoints for all methods and use the same hyper-parameters.

As shown in Table 2, ClusterGNN achieves competitive performance with a slight improvement compared to SuperGlue and SGMNet.

### 4.3. Visual localization

Visual localization is one of the most important applications of feature matching and its performance heavily relies on the matching quality. Given a query image, visual

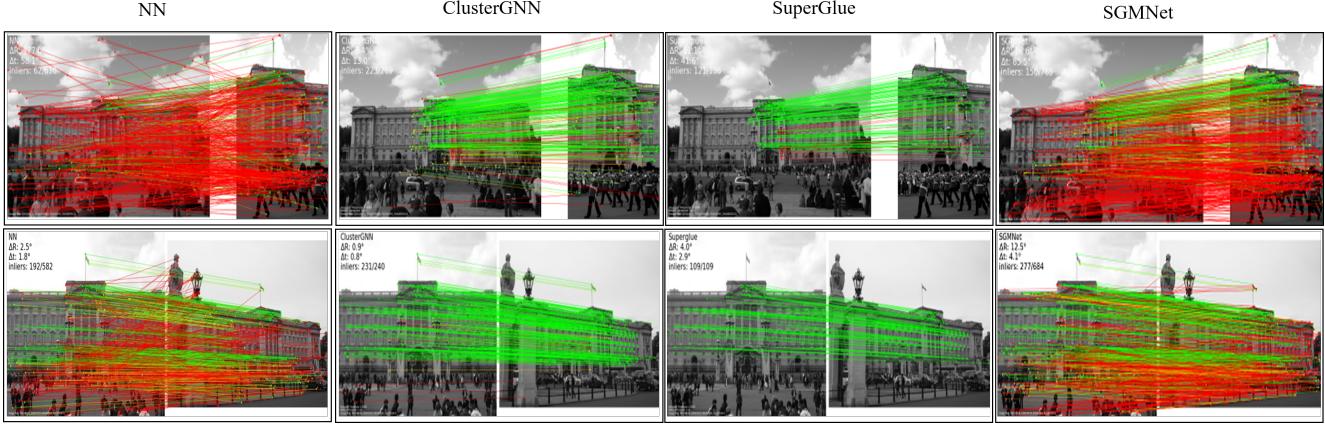


Figure 5. **Qualitative results.** The comparison of NN, Superglue, SGMNet and ClusterGNN on YFCC100M dataset with ASLFeat [19]. The number of keypoints is 2048. The red color indicates outliers and green color indicates inliers. ClusterGNN obtains more correct matches.

Local Features	Matcher	Pose estimation		
		5°	10°	20°
SIFT [18]	NN	15.19	24.72	35.30
	SGMNet	<b>35.63</b>	<b>55.40</b>	<b>71.95</b>
	SuperGlue	30.12	47.25	63.05
	ClusterGNN	32.82	50.25	65.89
SuperPoint [10]	NN	14.12	28.86	44.90
	SGMNet	32.44	52.80	69.90
	SuperGlue*	<b>39.02</b>	<b>59.51</b>	<b>75.72</b>
	SuperGlue	34.93	55.31	72.34
ASLFeat [19]	ClusterGNN	35.31	56.13	73.56
	NN	14.40	27.80	43.20
	SGMNet	32.22	52.53	70.16
	SuperGlue	37.50	58.30	75.07
	ClusterGNN	<b>42.62</b>	<b>61.22</b>	<b>76.75</b>

Table 1. **Pose estimation using the YFCC100M dataset.** We compare different matching methods (NN, SuperGlue, SGMNet and ClusterGNN) across different keypoint extraction methods (SIFT, ASLFeat and SuperPoint). We report the AUC of pose errors at different thresholds (Section 4.1). SuperGlue\* indicates the officially released model. The best performance is highlighted in bold.

localization aims to estimate its 6-DOF position, based on a 3D reconstructed model. We integrate our method into the official HLoc [29] pipeline for visual localization and evaluate it on the Long-Term Visual Localization Benchmark [39]. This benchmark assesses performance under different conditions, such as texture-less scenes of indoor environments and day-and-night changes, thus requiring highly robust matching. Specifically, we use the Aachen Day-Night dataset [31, 46] for evaluating outdoor localization and the InLoc dataset [37] for evaluating indoor localization. The Aachen Day-Night dataset provides 4328 images

Local Features	Matcher	overall			Illumination	viewpoint
		Accuracy(%), $\epsilon < 1/3/5px$				
SuperPoint [10]	NN	0.46 / 0.78 / 0.85	0.57 / 0.92 / 0.97	0.35 / 0.65 / 0.74		
	SGMNet	<b>0.52 / 0.85 / 0.91</b>	0.59 / 0.94 / 0.98	<b>0.46 / 0.74 / 0.84</b>		
	SuperGlue	0.51 / 0.83 / 0.89	<b>0.61 / 0.93 / 0.98</b>	<b>0.45 / 0.73 / 0.83</b>		
	ClusterGNN	<b>0.52 / 0.84 / 0.90</b>	<b>0.61 / 0.93 / 0.98</b>	0.44 / 0.74 / 0.81		
ASLFeat [19]	NN	0.48 / 0.81 / 0.88	0.57 / 0.92 / 0.97	0.34 / 0.68 / 0.78		
	SGMNet	0.49 / 0.82 / 0.89	0.57 / 0.93 / 0.98	0.41 / 0.72 / 0.83		
	SuperGlue	0.49 / 0.83 / 0.89	0.57 / 0.92 / 0.98	0.41 / 0.72 / 0.81		
	ClusterGNN	<b>0.51 / 0.83 / 0.89</b>	<b>0.61 / 0.95 / 0.98</b>	<b>0.42 / 0.72 / 0.82</b>		

Table 2. **Homography Estimation on Hpatches.** We report the percentage of correctly estimated homographies under different corner error distances.

of the Aachen city and 922 query images including 824 daytime images and 98 nighttime images taken by mobile phone cameras. The InLoc dataset [37] offers 9972 reference images and 329 query images which contain significant occlusions and variation in viewpoint and illumination.

We use the official HLoc [29] pipeline for visual localization tasks. Consistent with the official benchmark, we report the pose estimation accuracy under different thresholds. Tables 3 and 4 report the results for indoor and outdoor localization, respectively. ClusterGNN achieves competitive performance compared to SuperGlue on both indoor and outdoor localization tasks, across different features.

#### 4.4. Efficiency

Improving the processing efficiency and GPU memory requirements, while maintaining or improving matching performance is the main motivation for developing ClusterGNN. In this section, we compare the runtime and memory of our method with SGMNet and SuperGlue.

Fig. 6a and 6b, report the time and memory consumption for different numbers of detected keypoints on a Tesla P-100 GPU with 16GB memory. Specifically, we test both run

Local Features	Matcher	DUC1	DUC2
		(0.25m, 10°) / (0.5m, 10°) / (1.0m, 10°)	
SuperPoint [10]	NN	40.4 / 58.1 / 69.7	42.0 / 58.8 / 69.5
	SGMNet	41.9 / 64.1 / 73.7	39.7 / 62.6 / 67.2
	Superglue	<b>49.0</b> / 68.7 / <b>80.8</b>	<b>53.4</b> / <b>77.1</b> / 82.4
	ClusterGNN	47.5 / <b>69.7</b> / 79.8	<b>53.4</b> / <b>77.1</b> / <b>84.7</b>
ASLFeat [19]	NN	39.9 / 59.1 / 71.7	43.5 / 58.8 / 64.9
	SGMNet	43.9 / 62.1 / 68.2	45.0 / 63.4 / 73.3
	Superglue	51.5 / 66.7 / 75.8	53.4 / <b>76.3</b> / <b>84.0</b>
	ClusterGNN	<b>52.5</b> / <b>68.7</b> / <b>76.8</b>	<b>55.0</b> / 76.0 / 82.4

Table 3. Indoor Localization Results (InLoc Dataset). We report the percentage of correctly localized queries under different thresholds. The evaluation metrics are defined according to the leaderboard of Long-Term Visual Localization Benchmark [39].

Local Features	Matcher	Day	Night
		(0.25m, 2°) / (0.5m, 5°) / (1.0m, 10°)	
SuperPoint [10]	NN	85.4 / 93.3 / 97.2	75.5 / 86.7 / 92.9
	SGMNet	86.8 / 94.2 / 97.7	83.7 / 91.8 / 99.0
	Superglue	<b>89.6</b> / 95.4 / <b>98.8</b>	86.7 / <b>93.9</b> / <b>100.0</b>
	ClusterGNN	89.4 / <b>95.5</b> / 98.5	81.6 / <b>93.9</b> / <b>100.0</b>
ASLFeat [19]	NN	82.3 / 89.2 / 92.7	67.3 / 79.6 / 85.7
	SGMNet	86.8 / 93.4 / 97.1	<b>86.7</b> / 94.9 / 98.0
	Superglue	87.9 / 95.4 / 98.3	81.6 / 91.8 / 99.0
	ClusterGNN	<b>88.6</b> / <b>95.5</b> / <b>98.4</b>	85.7 / <b>93.9</b> / <b>99.0</b>

Table 4. Indoor Localization Results (Aachen Day-Night Benchmark(v1.0)). The evaluation metrics are refer to the leaderboard of Long-Term Visual Localization Benchmark [39].

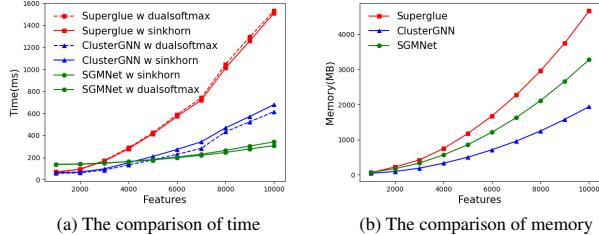


Figure 6. Efficiency Comparison. We report the time and memory consumption with an increasing number of input keypoints.

time with Sinkhorn iterations and dual softmax. It should be noted that memory consumption is consistent in both Sinkhorn and dual softmax. As shown in Fig. 6a, for 10k keypoints(dense detection), ClusterGNN reduces the runtime by 59.7%, compared to Superglue when using dual softmax. Although SGMNet is more time efficient, ClusterGNN can achieve better memory efficiency and performances at the same time. As shown in Fig. 6b, for dense detection, our method the proposed reduces 58.4% of the memory required by Superglue.

#### 4.5. Ablation Study

We conduct ablation studies on the YFCC100M dataset using ASLFeat [19] for detecting keypoints and the same

Methods	Pose estimation		
	5°	10°	20°
Fixed Cluster:16	31.26	48.55	64.50
Fixed Cluster:32	37.05	56.54	72.90
Fixed Cluster:64	37.44	56.58	72.68
Fixed Cluster:128	36.77	55.90	72.08
<b>ClusterGNN</b>	<b>42.62</b>	<b>61.22</b>	<b>76.75</b>

Table 5. The effect of fixing or varying the number of clusters). We report pose accuracy under different thresholds based on YFCC100M using 2K keypoints.

Methods	Pose estimation			Time	Memory
	5°	10°	20°	(ms)	(MB)
ClusterGNN w sinkhorn	38.83	58.87	75.11	68	<b>94</b>
ClusterGNN w dual softmax	<b>42.62</b>	<b>61.22</b>	<b>76.75</b>	<b>61</b>	<b>94</b>

Table 6. The effect of different operators for computing matching probability. We report pose accuracy, processing time and GPU memory for the YFCC100M dataset using 2K keypoints.

experimental setting as in Section 4.1. Our ablation focuses on the effect of fixing or varying the number of clusters and on the operator used for generating the matching probability matrix.

**Fixed vs. Varying Number of Clusters.** Our ClusterGNN strategy gradually decreases the size of clusters while increasing their number due to the tradeoff between over segmentation and efficiency. In this experiment we evaluate the effect of fixing the number of clusters versus our choice to gradually increase it. Table 5 shows the results of fixing the number of cluster to {16, 32, 64, 128} versus the results of our varying strategy. ClusterGNN with a fixed number of cluster suffers from significant drop in performance.

**Sinkhorn vs Dual Softmax.** Superglue [30] uses a differential iterative optimal transport, namely Sinkhorn [9,34], to solve the matching confidence. In our work, we adopt non-iterative dual-softmax [26] operator for efficiency. As shown in Table 6, dual-softmax achieves competitive performance compared with Sinkhorn.

## 5. Conclusion

In this work, we have addressed the quadratic complexity of GNN methods for feature matching. The proposed method, named ClusterGNN, leverages on the inherent sparsity of self- and cross- attention between keypoints in the complete graph and dynamically constructs local subgraphs through a learned coarse-to-fine clustering. Extensive evaluation on several computer vision tasks demonstrates the effectiveness of our approach, achieving a competitive performance while reducing runtime and memory by 59.7% and 58.4%, respectively.

## References

- [1] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *CVPR*, pages 5173–5182, 2017. 6
- [2] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *CoRR*, abs/2004.05150, 2020. 2
- [3] Eric Brachmann and Carsten Rother. Neural-guided ransac: Learning where to sample model hypotheses. In *CVPR*, pages 4322–4331, 2019. 6
- [4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, pages 213–229, 2020. 2
- [5] Hongkai Chen, Zixin Luo, Jiahui Zhang, Lei Zhou, Xuyang Bai, Zeyu Hu, Chiew-Lan Tai, and Long Quan. Learning to match features with seeded graph matching network. In *CVPR*, pages 6301–6310, 2021. 1, 2, 6
- [6] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509, 2019. 2
- [7] Krzysztof Choromanski, Valerii Likhoshesterov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, David Belanger, Lucy Colwell, et al. Masked language modeling for proteins via linearly scalable long-context transformers. *CoRR*, abs/2006.03555, 2020. 2
- [8] Krzysztof Choromanski, Valerii Likhoshesterov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020. 1
- [9] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26:2292–2300, 2013. 8
- [10] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPRW*, pages 224–236, 2018. 2, 6, 7, 8
- [11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, 2019. 2
- [12] 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. 2
- [13] 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*, pages 8092–8101, 2019. 6
- [14] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *TPAMI*, 40(3):611–625, 2017. 1
- [15] Matthias Fey, Jan E Lenssen, Christopher Morris, Jonathan Masci, and Nils M Kriege. Deep graph matching consensus. In *ICLR*, 2020. 2
- [16] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *ICLR*, 2020. 2
- [17] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *CVPR*, pages 2041–2050, 2018. 6
- [18] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 2, 6, 7
- [19] Zixin Luo, Lei Zhou, Xuyang Bai, Hongkai Chen, Jiahui Zhang, Yao Yao, Shiwei Li, Tian Fang, and Long Quan. Aslfeat: Learning local features of accurate shape and localization. In *CVPR*, pages 6589–6598, 2020. 2, 6, 7, 8
- [20] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. 1
- [21] Yuki Ono, Eduard Trulls, Pascal Fua, and Kwang Moo Yi. Lf-net: Learning local features from images. In *NIPS*, pages 6237–6247, 2018. 6
- [22] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *ICML*, pages 4055–4064. PMLR, 2018. 2
- [23] Jiezhong Qiu, Hao Ma, Omer Levy, Scott Wen-tau Yih, Sinong Wang, and Jie Tang. Blockwise self-attention for long document understanding. In *EMNLP*, pages 2555–2565, 2019. 1, 2
- [24] Filip Radenović, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondřej Chum. Revisiting oxford and paris: Large-scale image retrieval benchmarking. In *CVPR*, pages 5706–5715, 2018. 6
- [25] Jerome Revaud, Philippe Weinzaepfel, César De Souza, Noe Pion, Gabriela Csurka, Yohann Cabon, and Martin Humenberger. R2d2: repeatable and reliable detector and descriptor. In *NIPS*, pages 12405–12415, 2019. 2
- [26] Ignacio Rocco, Mircea Cimpoi, Relja Arandjelović, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Neighbourhood consensus networks. In *NIPS*, pages 1658–1669, 2018. 4, 5, 8
- [27] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021. 2, 5
- [28] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *ICCV*, pages 2564–2571. Ieee, 2011. 2
- [29] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, pages 12716–12725, 2019. 7
- [30] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *CVPR*, pages 4938–4947, 2020. 1, 2, 3, 4, 6, 8
- [31] Torsten Sattler, Tobias Weyand, Bastian Leibe, and Leif Kobbelt. Image retrieval for image-based localization revisited. In *BMVC*, volume 1, page 4, 2012. 7
- [32] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, pages 4104–4113, 2016. 1, 6

- [33] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, pages 501–518. Springer, 2016. 6
- [34] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967. 8
- [35] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *CVPR*, pages 8922–8931, 2021. 1, 3, 4, 5
- [36] Weiwei Sun, Wei Jiang, Eduard Trulls, Andrea Tagliasacchi, and Kwang Moo Yi. Acne: Attentive context normalization for robust permutation-equivariant learning. In *CVPR*, pages 11286–11295, 2020. 2
- [37] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. Inloc: Indoor visual localization with dense matching and view synthesis. In *CVPR*, pages 7199–7209, 2018. 7
- [38] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016. 6
- [39] Carl Toft, Will Maddern, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, et al. Long-term visual localization revisited. *TPAMI*, 2020. 7, 8
- [40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017. 1, 2, 4
- [41] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Learning combinatorial embedding networks for deep graph matching. In *CVPR*, pages 3056–3065, 2019. 2
- [42] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *CoRR*, abs/2006.04768, 2020. 2
- [43] Changchang Wu. Towards linear-time incremental structure from motion. In *3DV*, pages 127–134. IEEE, 2013. 1
- [44] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *CVPR*, pages 2666–2674, 2018. 2, 6
- [45] 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 *CVPR*, pages 5845–5854, 2019. 6
- [46] Zichao Zhang, Torsten Sattler, and Davide Scaramuzza. Reference pose generation for visual localization via learned features and view synthesis. *IJCV*, 129(4):821–844, 2021. 7
- [47] Qunjie Zhou, Torsten Sattler, and Laura Leal-Taixé. Patch2pix: Epipolar-guided pixel-level correspondences. In *CVPR*, pages 4669–4678, 2021. 6
- [48] Qunjie Zhou, Torsten Sattler, Marc Pollefeys, and Laura Leal-Taixé. To learn or not to learn: Visual localization from essential matrices. In *ICRA*, pages 3319–3326. IEEE, 2020. 1