Learning to Match Features with Seeded Graph Matching Network

Hongkai Chen[1] Zixin Luo[1] Jiahui Zhang[2] Lei Zhou[1] Xuyang Bai[1] Zeyu Hu[1]
Chiew-Lan Tai[1] Long Quan[1]
[1]Hong Kong University of Science and Technology [2] Tsinghua University
{hchencf,zluoag,lzhouai,xbaiad,zhuam,taicl,quan}@cse.ust.hk jiahui-z15@mails.tsinghua.edu.cn

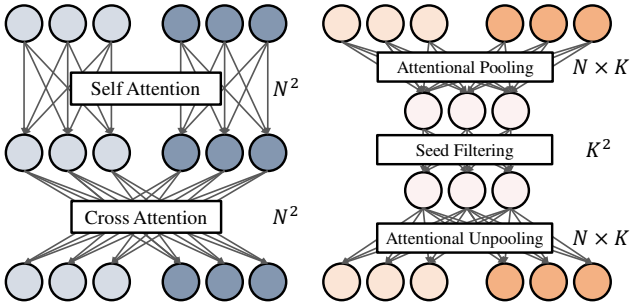
Abstract

Matching local features across images is a fundamental problem in computer vision. Targeting towards high accuracy and efficiency, we propose Seeded Graph Matching Network, a graph neural network with sparse structure to reduce redundant connectivity and learn compact representation. The network consists of 1) Seeding Module, which initializes the matching by generating a small set of reliable matches as seeds. 2) Seeded Graph Neural Network, which utilizes seed matches to pass messages within/across images and predicts assignment costs. Three novel operations are proposed as basic elements for message passing: 1) Attentional Pooling, which aggregates keypoint features within the image to seed matches. 2) Seed Filtering, which enhances seed features and exchanges messages across images. 3) Attentional Unpooling, which propagates seed features back to original keypoints. Experiments show that our method reduces computational and memory complexity significantly compared with typical attention-based networks while competitive or higher performance is achieved.

1. Introduction

Establishing reliable correspondences across images is an essential step to recover relative camera pose and scene structure in many computer vision tasks, sush as Structure-from-Motion (SfM) [37], Multiview Stereo(MVS) [17] and Simultaneous Localization and Mapping (SLAM) [31]. In classical pipelines, correspondences are obtained by nearest neighbour search (*NN*) of local feature descriptors and are usually further pruned by heuristic tricks, such as mutual nearest neighbour check (*MNN*) and ratio test (*RT*) [24].

In the past few years, great efforts have been made on designing learnable matching strategy. Early works [60, 43, 62] in this direction utilize PointNet [32]-like networks to reject outliers of putative correspondences. In these works, the correspondence coordinates are fed into permutation-equivariant networks, then inlier likelihood scores are predicted for each correspondence. Despite showing exciting results, these methods are limited in two aspects: 1) They operate on pre-matched correspondences, whereas finding more matches than vanilla nearest-neighbour matching is impossible. 2) They only reason about the geometric distribution of putative correspondences, neglecting the critical information of original local visual descriptors.



(a) SuperGlue (b) SGMNet

Figure 1. The designs of message passing layer. (a) SuperGlue densely connects every node in the graph, resulting in $\mathcal{O}(N^2)$ computational complexity. (b) Instead, the proposed network, SGMNet, adopts pooling/unpooling operations, reducing the complexity to $\mathcal{O}(NK)$, where $K \ll N$ in general.

Another thread of methods cast feature matching as a graph matching problem [35, 5, 49], which mitigates the limit of vanilla nearest neighbour correspondences. The representative work, SuperGlue [35], constructs densely-connected graphs between image keypoints to exchange messages about both visual and geometric context. However, the superior performance comes along with high computation and memory cost, especially when being applied to keypoints of larger number (e.g. up to $10k$). As illustrated in Fig. 1(a), the message passing layer of SuperGlue first calculates similarity scores exhaustively between every two nodes, then gathers features to pass messages densely in the graph. This results in computational complexity of $\mathcal{O}(N^2C)$ for matrix multiplications, and memory occupation of $\mathcal{O}(N^2)$ to hold the attention matrix, supposing that the keypoint number is N and feature channel is C. The complexity increases even drastically for deeper graph networks. Given this, exploring more efficient and compact message passing operation is of practical significance.

Besides the major efficiency bottleneck, it is debatable

if such densely-connected graph introduces too much redundancy or insignificant message exchange that may hinder the representation ability, especially in the context of feature matching where the match set is highly outlier-contaminated and a large portion of keypoints are unrepeatable. As a result, most graph edges from SuperGlue [35] tend to have zero strength, as reported in its original paper and also observed in our experiments. This phenomenon indicates that even a sparse graph is largely sufficient and less distracted from unnecessary message exchanges.

In this paper, we propose Seeded Graph Matching Network (SGMNet) to mitigate above limitations from two aspects. First, inspired by guided matching approaches [10, 42, 29], we design a *Seeding Module* that initializes the matching from a small set of reliable matches so as to more effectively identify inlier compatibility. Second, we draw inspiration from graph pooling operations [61, 55], and construct a *Seeded Graph Neural Network* whose graph structure is largely sparsified to lower the computation and reduce the redundancy. Specifically, three operations are proposed to construct our message passing blocks. As illustrated in Fig. 1(b), instead of densely attending to all features within/across images, original keypoint features are first pooled by 1) *Attentional Pooling* through a small set of seed nodes, of which the features will be further enhanced by 2) *Seed Filtering*, and finally recovered back to original keypoints through 3) *Attentional Unpooling*.

By using seeds as attention bottleneck between images, the computational complexity for attention is reduced from $\mathcal{O}(N^2C)$ to $\mathcal{O}(NKC)$, where K is the number of seeds. When $K \ll N$, for example, $8k$ features are pooled into 512 seeds, the actual computation will be significantly cut down. We evaluate SGMNet under different tasks to demonstrate both its efficiency and effectiveness, and summarize our contributions threefold:

- A seeding mechanism is introduced in graph matching framework to effectively identify inlier compatibility.
- A greatly sparsified graph neural network is designed that enables more efficient and clean message passing.
- Competitive or higher accuracy is reported with remarkably improved efficiency over dense attentional GNN. As an example, when matching $10k$ features, SGMNet runs 7 times faster and consumes 50% less GPU memory than SuperGlue.

2. Related Works

Learnable image matching. Integrating deep learning techniques into geometry-based computer vision task, such as MVS [57, 58, 63, 64] and Visual Localization [66, 45], has received inspiring success during the past few years. As the front-end component for geometry estimation, learnable image matching has also been proven effective, where works in this area can be roughly divided into two categories. The first one focuses on improving local descriptors [59, 40, 26, 25, 30, 48] and keypoints [52, 12, 27, 34, 11] with convolutional neural networks, while methods in the second category attempt to embed learning techniques into matching strategy, which involves learnable outlier rejection [60, 43, 62] and robust estimator [4].

Recently, a new framework, SuperGlue [35], is proposed to integrate feature matching and outlier rejection into a single graph neural network (GNN). Though exhibiting promising results in different tasks, SuperGlue still suffers from the excessive computational cost of fully connected self/cross attention operation, especially when used to match features in high number.

Compared with SuperGlue, our method shares the same advantages, that is, feature matching and refinement are integrated into one single network and allows for end-to-end training. However, our network significantly reduces the computational and memory cost due to its efficient attention block, which is specially designed for image matching.

Efficient transformer architectures. Transformer [51] architectures have gained intensive interest during the past few years. Specially, in the contexts of graph convolution, the attention mechanism in transformer can be used to pass messages across nodes in graph structure [13, 53]. Despite its effectiveness in a wide range of tasks, one major concern about transformer is its quadratic complexity w.r.t. the input size, which hinders its application under large query/key elements number.

Recently, many efforts have been made to address the attention efficiency problem. In [23, 7] predefined sparse attention pattern are adopted to cut down memory/computation cost. In [46, 18], attention span is pruned by using learnable partitioning or grouping on input elements. In [54, 20], pooling operation is utilized to reduce elements number. Despite the inspiring progress, works in this area generally focus on self-attention, where keys and queries are derived from the same element set, while their effectiveness in cross-attention, where keys and queries come from two unaligned sets, remains unstudied.

We draw inspiration from induced set attention (ISA) [20], where a set of learned but fixed nodes are utilized as bottleneck for efficient self attention. To be compatible with cross attention in graph matching, we establish attention between seed matches and original point sets. The selected reliable correspondences align features in both sides and pass message in a low cost way.

Graph matching. Graph matching, which aims to generating node correspondences across graphs, is a widely used model for feature matching in both 2D [49, 5] and 3D [21, 2] domain. Mathematically formulated as a quadratic assignment problem (QAP) [50], graph matching is NP-hard in its most general form and requires infeasible expensive

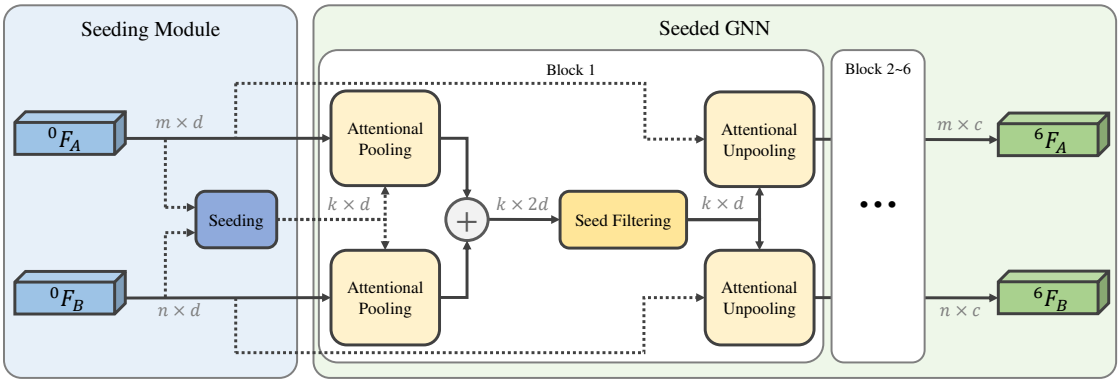


Figure 2. The network architecture of SGMNet, which takes the local features as input, then generates seed matches from *seeding module*, and finally extracts correspondence features from *Seeded GNN* with multiple attentional blocks. In practice, updated features will be fed into a reseeding module and a 3-layer seeded GNN for refinement, while we omit this procedure here for simplicity.

solver for precise solutions. Despite the intractable nature of general graph matching, some methods [14, 15, 28] leverage partially pre-matched correspondences, also called seeds, to help matching, which are referred to as Seeded Graph Matching(SGM). Inspired by SGM, our network integrate seeds into a GNN framework for compact message passing and robust matching.

3. Methodology

We present Seeded Graph Matching Network, shorten as SGMNet, for learning correspondences between two sets of keypoints and their associated visual descriptors. As illustrated in Fig. 2, our network produces matches in two stages: 1) *Seeding Module* generates seeds to guide compact message passing, and 2) *Seeded Graph Neural Network* leverages seeds as message bottleneck to update per-node feature. In the following parts, an overview of our network architecture will be introduced first, followed by detailed description of each module.

3.1. Overview

Given a pair of images $\mathbf{A}$ and $\mathbf{B}$, with n and m keypoints and associated visual descriptors respectively, indexed by $\alpha := \{1, ..., n\}, \beta := \{1, ..., m\}$, our objective is to establish reliable and robust keypoint matches across two images.

We formulate the keypoint matching task as a graph matching problem, where the nodes are the keypoints of each image. Instead of apply fully connected graph, we generate a set of keypoint correspondences, which we refer to as *seed matches*, to guide message pass across nodes in two graphs for subsequent matching. This critical difference allows for processing large number of keypoints with significantly lower memory and computation cost.

The input to our network are keypoints $\mathbf{K_A}$, $\mathbf{K_B}$ in two images, $\mathbf{K}_\mathbf{I}^i = (\mathbf{p}_\mathbf{I}^i, \mathbf{d}_\mathbf{I}^i)$, where $\mathbf{I} \in \{\mathbf{A}, \mathbf{B}\}$ and $\mathbf{p}_\mathbf{I}^i = (x_\mathbf{I}^i, y_\mathbf{I}^i)$ is the coordinate of keypoint i in image $\mathbf{I}$. $\mathbf{d}_\mathbf{I}^i \in \mathbb{R}^d$ is associated d dimensional visual descriptor.

Positions of keypoints are embedded into high dimensional feature space and combined with descriptors by element-wise summation for initial representation $^0\mathbf{F_A}$, $^0\mathbf{F_B}$.

$$^0\mathbf{F}_\mathbf{I}^i = \mathbf{d}_\mathbf{I}^i + \text{MLP}(\mathbf{p}_\mathbf{I}^i), \mathbf{I} \in \{\mathbf{A}, \mathbf{B}\}, \tag{1}$$

A *Seeding Module* follows to construct a set of seed matches $\mathbf{S}$. $^0\mathbf{F_A}$, $^0\mathbf{F_B}$ and $\mathbf{S}$ are then fed into our *Seeded Graph Neural Network* which reasons about visual appearance similarity, neighbourhood consensus as well as the guidance provided by seed matches jointly to update keypoint features.

Inspired by the cascade refinement structure in OANet [62], a second seeding module, or reseeding module, is introduced to generate more accurate seeds based on the updated features, which helps to further refine matches with another Seeded GNN. Final matches are then generated by formulating assignment matrix.

3.2. Seeding Module

Proposing a set of seed matches lays the foundation for subsequent matching. For initial seeding, we adopt a simple yet effective strategy: we generate putative matches by nearest neighbour matching and use the inverse of distance ratio, i.e., the ratio of distance to first and second nearest neighbours [24], as reliability scores. We adopt Non-Maximum Suppression (NMS) for a better spatial coverage of seeds. More details of seeding module can be found in Appendix A.2. Despite potential noise in initial seeds, our network maintains robustness with proposed weighted unpooling operation and reseeding strategy, which will be discussed later.

The seeding module outputs $\mathbf{S} = (\mathbf{S_A}, \mathbf{S_B})$, where $\mathbf{S_A}$, $\mathbf{S_B}$ are index lists for seed matches in each image.

3.3. Seeded Graph Neural Network (Seeded GNN)

Seeded GNN takes initial position-embedded features ${}^0\mathbf{F_A}$, ${}^0\mathbf{F_B}$ and leverages seed matches $\mathbf{S}$ as attention bottlenecks for message passing. To this end, we adopt a pooling-processing-unpooling strategy in each processing unit: seed features first gather information from full point sets on each side through **Attentional Pooling**, then processed by **Seed Filtering** operation and finally be recoverd back to original size by **Attentional Unpooling**. Our Seeded GNN is constructed by stacking 6(3) such processing units for initial(refinement) stages.

Weighted attentional aggregation. We first introduce a weighted version of attentional aggregation, which allows for sharper and cleaner data-dependent message passing.

In a d-dimensional feature space, for m vectors to be updated: $\mathbf{X} \in \mathbb{R}^{m\times d}$, n vectors to be attended to: $\mathbf{Y} \in \mathbb{R}^{n\times d}$ and a weight vector: $\mathbf{w} \in \mathbb{R}^n$, the weighted attentional aggregation **Att** is defined as,

$$\mathbf{X^r} = \mathbf{Att}(\mathbf{X}, \mathbf{Y}, \mathbf{w}) = \mathbf{X} + \mathrm{MLP}(\mathbf{X}||\mathbf{\Delta}), \tag{2}$$

where

$$\Delta = \theta(\mathbf{Q}\mathbf{K}^T)\mathbf{W}\mathbf{V}, \mathbf{W} = Diag(\mathbf{w}), \mathbf{W} \in \mathbb{R}^{n\times n} \tag{3}$$

$\theta(\cdot)$ means row-wise softmax. $\mathbf{Q}$ is linear projection of $\mathbf{X}$ and $\mathbf{K}, \mathbf{V}$ are linear projections of $\mathbf{Y}$. $\mathbf{X^r}$ is renewed representation for $\mathbf{X}$.

By attentional aggregation, elements in $\mathbf{X}$ retrieve and aggregate information from elements in $\mathbf{Y}$. A weighting vector $\mathbf{w}$ is applied on $\mathbf{V}$ to adjust the importance for each element in $\mathbf{Y}$.

Attentional pooling. As the first step in message passing, seed matches retrieves contexts from full keypoint set through attentional aggregration,

For input features ${}^t\mathbf{F_A}$, ${}^t\mathbf{F_B}$ in layer t, features for seed matches are first retrieved by indices $(\mathbf{S_A}, \mathbf{S_B})$

$${}^t\mathbf{S}^1_\mathbf{I} = {}^t\mathbf{F_I}[\mathbf{S_I}], \mathbf{I} \in \{\mathbf{A}, \mathbf{B}\}, \tag{4}$$

where $[\,]$ is the indexing operation. Seed matches are then updated by retrieving context from nodes in each graph

$${}^t\mathbf{S}^2_\mathbf{I} = \mathbf{Att}({}^t\mathbf{S}^1_\mathbf{I}, {}^t\mathbf{F_I}, \mathbf{1}),\ \mathbf{I} \in \{\mathbf{A}, \mathbf{B}\}, \tag{5}$$

where $\mathbf{1}$ is all one vector, which means no weights are applied.

A mutilayer perceptron follows to fuse the seed features,

$$[{}^t\mathbf{S}^3_\mathbf{A}||{}^t\mathbf{S}^3_\mathbf{B}] = \mathrm{MLP}({}^t\mathbf{S}^2_\mathbf{A}||{}^t\mathbf{S}^2_\mathbf{B}) \tag{6}$$

where $||$ means concatenation along row dimension.

Figure 3. Visualization of attentional pooling/unpooling. In attentional pooling, one pair of seed match aggregates context from the other keypoints, while in attentional unpooling each original keypoint retrieves renewed message from seed matches.

The outputs ${}^t\mathbf{S}^3_\mathbf{A}$, ${}^t\mathbf{S}^3_\mathbf{B}$, which encode both visual and position context for each graph and information from seed matches themselves, are fed into subsequent operations.

Seed filtering. We propose seed filtering operation to (1) conduct intra/inter-graph communication between seed matches and (2) suppress the influence of outlier seed matches. More specifically, intra/inter-graph attentional aggregation is applied to the input seed correspondence features ${}^t\mathbf{S}^3_\mathbf{I} \in \mathbb{R}^{k\times d}$, $\mathbf{I} \in \{\mathbf{A}, \mathbf{B}\}$.

$${}^t\mathbf{S}^4_\mathbf{I} = \mathbf{Att}({}^t\mathbf{S}^3_\mathbf{I}, {}^t\mathbf{S}^3_\mathbf{I}, \mathbf{1}), \tag{7}$$

$${}^t\mathbf{S}^5_\mathbf{I} = \mathbf{Att}({}^t\mathbf{S}^4_\mathbf{I}, {}^t\mathbf{S}^4_\mathbf{J}, \mathbf{1}),\ \mathbf{I}, \mathbf{J} \in \{\mathbf{A}, \mathbf{B}\}, \mathbf{I} \neq \mathbf{J}, \tag{8}$$

In addition, a context normalization [60] branch is used to predict inlier likelihood scores γ for each seed correspondence, which will be used as weighting score for seed features in later unpooling stage.

$${}^t\gamma = \mathrm{CN}({}^t\mathbf{S}^5_\mathbf{A}||{}^t\mathbf{S}^5_\mathbf{B}), \tag{9}$$

where CN is a lightweight stacked context normalization [60] blocks. Detailed structures of CN brach can be found in Appendix A.2.

The outputs of seed filtering are the filtered features ${}^t\mathbf{S}^5_\mathbf{A}$, ${}^t\mathbf{S}^5_\mathbf{B}$ and inlier scores $\gamma \in [0, 1]^k$ of seed matches.

Attentional unpooling. After the message exchange between seed matches and inlier score prediction, an inlier-score-weighted attentional aggregation is adopted to broadcast the pooled contexts to every keypoint in each graph, which we refer to as **attentional unpooling**.

Taking ${}^t\mathbf{S}^5_\mathbf{A}$, ${}^t\mathbf{S}^5_\mathbf{B}$, inlier score γ and ${}^t\mathbf{F_A}$, ${}^t\mathbf{F_B}$ as input, attentional unpooling outputs the updated keypoint features ${}^{t+1}\mathbf{F_A}$, ${}^{t+1}\mathbf{F_B}$.

$${}^{t+1}\mathbf{F_I} = \mathbf{Att}({}^t\mathbf{F_I}, {}^t\mathbf{S}^5_\mathbf{I}, {}^t\gamma),\ \mathbf{I} \in \{\mathbf{A}, \mathbf{B}\} \tag{10}$$

Applying inlier score to the aggregation process suppresses information broadcast from false seed matches and results in cleaner feature update, which contributes to the robustness of our network w.r.t. seeding noise (Appendix D.2, Appendix Fig 10).

Assignment matrix formulation. After all processing units, the updated features are used to construct the assignment matrix. Sinkhorn [8] algorithm is applied on the correlation matrix of features with a dustbin channel to produce final assignment matrix $\mathbf{M}$.

Given the keypoint features ${}^{N}\mathbf{F_A} \in \mathbb{R}^{n\times d}$, ${}^{N}\mathbf{F_B} \in \mathbb{R}^{m\times d}$ after N processing blocks, we compute the assignment matrix by

$$\mathbf{M} = \text{Sinkhorn}(\hat{\mathbf{C}}) \tag{11}$$

$$\hat{C}_{i,j} = \begin{cases} C_{i,j}, \text{ for } i \leqslant n, j \leqslant m \\ z, \text{ otherwise} \end{cases}, \hat{\mathbf{C}} \in \mathbb{R}^{(n+1)\times(m+1)} \tag{12}$$

where $\mathbf{C} = {}^{N}\mathbf{F_A}\,{}^{N}\mathbf{F_B}^{\mathbf{T}}$, z is a learnable parameter for dustbin. We derive final matches from the assignment matrix $\mathbf{M}$ with a confidence threshold to remove outliers.

3.4. Reseeding

Although Seeded GNN based on initial seeding exhibits strong capability to identify underlying matches, a second seeding module using updated feature provides even cleaner and richer seeds to further improve the performance. Thus, we adopt a reseeding module. Different from initial seeding where NN matches and ratio scores of raw descriptors are used, reseeding module employs assignment matrix $\mathbf{M}$ of updated features to regenerate seeds. More specifically, matches with highest score in both rows and columns are selected as candidates, where top-k matches are selected as new seeds and are fed into a second Seeded GNN for refinement. More details can be found in Appendix A.2.

3.5. Loss

Seeding module only outputs seed indices that requires no gradient back-propagation, thus our network is fully differentiable and can be trained end-to-end with supervision from indices of ground truth matches $\mathbf{l_m} = \{(i,j)\} \in \alpha\times\beta$ and unmatchable points $\mathbf{l_{uA}}, \mathbf{l_{uB}}$, where a point is regarded as unmatchable if there are no matchable points in the other image. From assignment matrix $\mathbf{M}_r$ for reseeding, final assignment matrix $\mathbf{M}_f$, and inlier scores ${}^{t}\gamma, t \in \{1,2,...,L\}$ for L processing units, we formulate our loss as two parts,

$$L = L_{assign} + \delta\Sigma_{t\in\{1,2,...,L\}} L^{t}_{weight} \tag{13}$$

where

$$L_{assign} = -\sum_{\mathbf{M}\in\{\mathbf{M}_r,\mathbf{M}_f\}} [\sum_{(i,j)\in\mathbf{l_m}} \log(M_{i,j}) + \sum_{i\in\mathbf{l_{uA}}} \log(M_{i,m+1}) + \sum_{j\in\mathbf{l_{uB}}} \log(M_{n+1,j})] \tag{14}$$

L^{t}_{weight} is cross entropy loss for inlier/outlier binary classification in t-th processing unit, a seed correspondence is labeled as inlier if its epipolar distance is less than a threshold. δ is a weight to balance the two loss terms.

3.6. Implementation Details

We train our network on GL3D dataset [38], which covers both indoor/outdoor scenes, to obtain general purpose model. We sample $1k$ keypoints and 128 seeds during training. We use Adam optimizer with learning rate of 10^{-4} in optimization and inlier score weight δ in loss is set to 250. We use 6/3 processing blocks for initial/refine stage and the gradients flow between the two stages are blocked in early iterations(140k iterations). We use 4-head attention in both attentional pooling/unpooling operations. For all experiments, we use a confidence threshold of 0.2 to retain matches and seeding number of $\frac{128\#keypoint}{2000}$, where $\#keypoint$ is the number of keypoints. More details including training data generation and hyper-parameters can be found in Appendix A.

4. Experiments

In the following sessions, we provide experiments results of our methods under a wide range of tasks, as well as further analysis of its computation and memory efficiency.

4.1. Image Matching

Datasets. The performance of our method is first evaluated on image matching tasks and three benchmarks in two-view pose estimation, YFCC100M [47], FM-Bench [3] and ScanNet [9] dataset, are used for demonstration.

For YFCC100M [47], we follow the setting in OANet [62] and choose 4 sequences for testing. FM-Bench [3] comprises four subsets in different scenarios: KITTI [16] for driving settings, TUM [41] for indoor SLAM settings, Tanks and Temples (T&T) [19] and CPC [56] for wide-baseline reconstruction tasks. ScanNet [9] is a widely used indoor reconstruction dataset. Following SuperGlue [35], we use 1500 pairs in test set for evaluation.

Evaluation protocols. On YFCC100M and ScanNet dataset, pose estimation is performed on the correspondences after RANSAC post-processing. We report 1) AUC [35, 62, 60] under different thresholds, computed from

Feature	*Matcher*	CPC		T&T		TUM		KITTI	
		%Recall	*#Corrs(-m)*	*%Recall*	*#Corrs(-m)*	*%Recall*	*#Corrs(-m)*	*%Recall*	*#Corrs(-m)*
RootSIFT [1]	NN+RT	52.9	92 (123)	82.1	208 (287)	61.9	365 (438)	90.6	847 (928)
	OANet [62]	58.6	119 (167)	84.7	219 (306)	62.3	454 (396)	89.0	773 (854)
	SuperGlue [35]	61.1	218 (466)	**86.8**	382 (767)	65.9	655 (1037)	91.0	**1261 (1746)**
	SGMNet	**62.0**	**248 (524)**	85.9	**397 (789)**	**66.6**	**704 (1132)**	**91.2**	1097 (1506)
ContextDesc [25]	NN+RT	62.4	169 (277)	85.5	222 (426)	58.7	456 (625)	90.6	1134 (1416)
	OANet	65.3	187 (288)	86.7	294 (425)	53.2	295 (327)	89.0	791 (907)
	SuperGlue	67.0	260 (579)	89.1	491 (695)	60.1	408(690)	**91.1**	**1401 (1897)**
	SGMNet	**70.8**	**370 (616)**	**89.9**	**514 (705)**	**61.6**	**423 (705)**	90.3	1204 (1724)
SuperPoint [11]	MNN	34.5	152 (421)	72.8	287 (717)	56.7	280 (420)	88.6	848 (1490)
	OANet	62.9	186 (343)	91.2	280 (477)	61.4	332 (473)	82.2	482 (736)
	SuperGlue	68.8	287 (719)	92.9	414 (987)	59.1	512 (1038)	88.7	**957** (1777)
	SuperGlue*	**76.7**	302(712)	**96.6**	431(985)	59.0	121(177)	**89.5**	354(526)
	SGMNet	70.3	**327 (829)**	93.2	**450 (1098)**	**65.8**	**666 (1315)**	86.3	954(**1851**)

Table 1. Evaluation results on FM-Bench [3], where *%Recall* denotes mean recall of all pairs, *#Corrs(-m)* denotes mean number of inlier correspondences after/before RANSAC. SuperGlue* indicates the results obtained from officially released model.

the angular differences between ground truth and estimated vectors for both rotation and translation; 2) Mean matching score (*M.S.*) [35, 11], the ratio of correct matches and total keypoint number; 3) Mean precision (*Prec.*) [35, 11] of the generated matches. We detect up to $2k$ keypoints for all features on YFCC100M, up to $1k$ keypoints for superpoint on ScanNet and up to $2k$ keypoints for other features .

On FM-Bench dataset, we estimate fundamental matrix for each evaluated pair with RANSAC post-processing, and use the normalized symmetric epipolar distance (SGD) [65, 3] as originally defined in FM-Bench paper, to measure the difference between the estimated fundamental matrix and the ground truth. An estimate is considered correct if its normalized SGD to ground truth is lower than a threshold (0.05 is used by default), and up to $4k$ keypoints are detected for each test pair. Following FM-Bench paper [3] , we report: 1) recall (*%Recall*) on fundamental matrix estimation; 2) mean number of correct correspondences (*#Corrs(-m)*) after/before RANSAC.

Comparative methods. We compare our method with heuristic pruning strategy, ratio test [24] or *MNN*, and a various of learning-based matching methods [62, 35, 44, 10, 60, 39]. These methods are applied on both handcrafted descriptors [24, 1] and learning-based local features [25, 11].

For a fair comparison, OANet, SuperGlue and SGMNet are all re-trained using the same sequences of GL3D [38], where 1k keypoints are sampled per image. Noted that the official training code of SuperGlue is not available, and its public model (denoted as SuperGlue*) is trained on MegaDepth [22] and Oxford and Paris dataset [33]. Instead, we retrain SuperGlue on GL3D [38] with similar data selection criteria described in orignal paper. This re-implementation achieves even better results on YFCC100M (Table 3) for RootSIFT than those reported in the original paper. However, there still remains some performance gap when using SuperPoint [11], even though we have carefully tuned the training and enquired the authors about details. Nevertheless, we consider our re-implementation of SuperGlue [35] faithful and thus can be fairly compared. We report results of both official model and our re-

Feature	*Matcher*	AUC			M.S.	Prec.
		@5°	*@10°*	*@20°*		
RootSIFT	NN + RT [24]	49.07	58.76	68.58	8.23	29.79
	AdaLAM(4k) [6]	57.78	68.01	77.38	7.92	83.15
	OANet [62]	58.00	67.80	77.46	5.84	81.80
	SuperGlue* [35]	59.25	70.38	80.44	-	-
	SuperGlue	**63.82**	**73.33**	**82.26**	16.59	81.08
	SGMNet	62.72	72.52	81.48	17.08	**86.08**
ContextDesc	NN + RT	57.90	68.47	78.35	9.39	59.72
	AdaLAM(4k) [6]	60.75	70.91	80.23	9.12	85.45
	OANet	62.28	72.56	81.80	9.33	**88.49**
	SuperGlue	65.98	75.17	83.64	20.38	82.95
	SGMNet	**66.63**	**76.21**	**84.33**	**20.57**	87.34
SuperPoint	MNN	31.05	40.85	52.64	15.12	24.64
	AdaLAM(2k) [6]	40.20	49.03	59.11	10.17	72.57
	OANet	48.80	59.06	70.02	12.48	71.95
	SuperGlue*	**67.10**	**76.18**	**84.37**	21.58	**88.64**
	SuperGlue	60.37	70.51	80.00	19.47	78.74
	SGMNet	61.22	71.02	80.45	**22.36**	85.44
SIFT	PointCN [60]	47.98	58.13	68.67	-	-
	ACNe [44]	-	-	78.00	-	-
	LGLFM [10]	49.60	60.36	71.37	-	-
-	RANSAC-Flow [39]	64.88	73.31	81.56	-	-

Table 2. Results on YFCC100M [47], where AUC evaluates the pose accuracy, $M.S.$ denotes mean matching score and $Prec.$ denotes mean matching precision. SuperGlue* indicates the results obtained from original paper or officially released model.

Feature	*Matcher*	AUC			M.S.	Prec.
		@5°	*@10°*	*@20°*		
RootSIFT	NN + RT [24]	9.08	19.75	32.66	2.28	28.83
	AdaLAM [6]	8.24	18.57	31.01	3.10	47.59
	OANet [62]	10.71	23.10	37.42	3.20	36.93
	SuperGlue	**13.12**	**27.99**	43.92	**8.50**	42.53
	SGMNet	12.82	27.92	**44.55**	8.79	**45.55**
ContextDesc	NN + RT	11.07	23.52	37.66	5.29	28.71
	AdaLAM	8.45	19.81	33.11	6.58	44.08
	OANet	11.95	24.49	40.56	5.12	40.43
	SuperGlue	**15.70**	**31.67**	48.22	**10.75**	42.83
	SGMNet	15.46	31.55	**48.64**	9.99	**48.14**
SuperPoint	MNN	9.44	21.57	36.41	13.27	30.17
	AdaLAM	6.72	15.82	27.37	13.19	44.22
	OANet	10.04	25.09	38.01	10.56	44.61
	SuperGlue	13.95	29.48	46.07	15.82	44.18
	SuperGlue*	**16.19**	**33.82**	**51.86**	**18.50**	47.32
	SGMNet	15.40	32.06	48.32	16.97	**48.01**

Table 3. Results on Scannet [9]. SuperGlue* indicates the results obtained from officially released model.

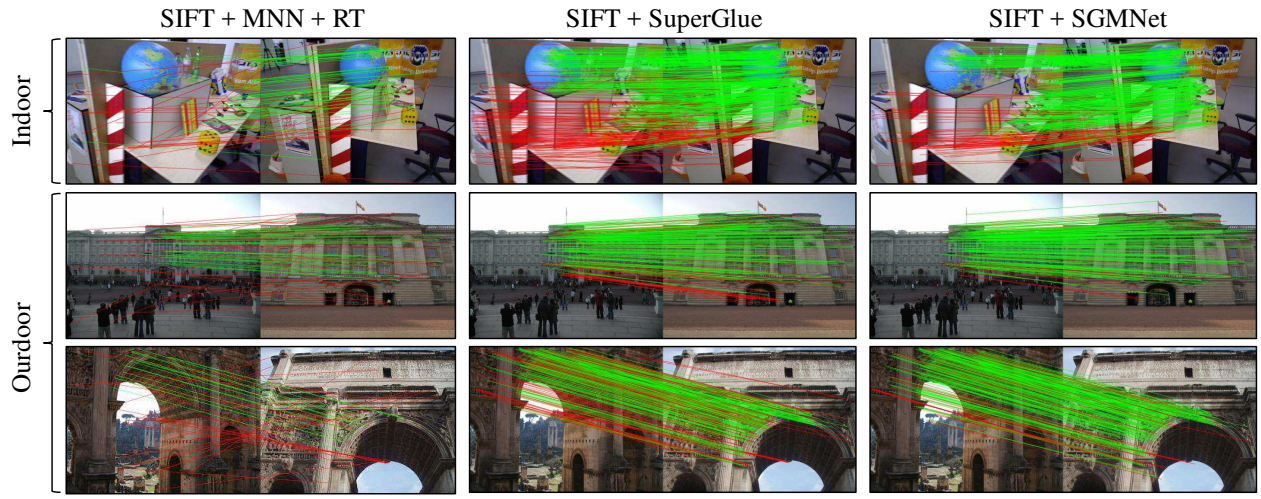


Figure 4. Correspondence visualizations. We showcase with SIFT features, and compare the results of traditional matching (*MNN + RT*), SuperGlue and our method (SGMNet). More visualizations available in Appendix.

implementation.

Results. For YFCC100M, ScanNet and two wide-baseline datasets of FM-Bench (CPC and T&T), our method mostly shows competitive results compared with the state-of-the-arts. For two small-baseline datasets in FM-Bench (TUM and KITTI), the advantages of all learnable methods tend to degenerate due to the reduced matching difficulty. Our method matches most inlier correspondences on almost all dataset regarding *M.S.* on YFCC100M/ScanNet and *Corrs(-m)* on FM-Bench while maintaining a high matching precision, which contributes to the final pose accuracy. Though not specially trained in indoor scenarios, our method generalizes well on indoor setting.

4.2. Visual Localization

To evaluate how our method benefits real downstream applications, we integrate it in a visual localization pipeline and evaluate its performance.

Datasets. We resort to Aachen Day-Night dataset [36] to evaluate the effectiveness of our method on visual localization task. Aachen Day-Night consists of 4328 reference images and 922 (824 daytime, 98 nighttime) query images. All images are taken in urban scenes.

Evaluation protocols. We use the official pipeline of Aachen Day-Night benchmark. Correspondences between reference images are first used to triangulate a 3D reconstruction. Correspondences between each query and its retrieved reference images are then generated to recover the relative pose. Consistent with the official benchmark, we report the pose estimation accuracy under different thresholds. We extract $8k$ keypoints for RootSIFT, ContextDesc and $4k$ keypoints for SuperPoint.

Results. Compared with SuperGlue, our method exhibits better results when using RootSIFT and competitive results when using SuperPoint or ContextDesc. Our method consistently outperforms OANet using all three descriptors. The overall performance proves the generalization ability of our method on real challenging applications.

Feature	*Matcher*	**0.25m, 2°**	**0.5m, 5°**	**5m, 10°**
RootSIFT [1]	MNN	43.9	56.1	65.3
	OANet [62]	69.4	83.7	94.9
	SuperGlue [35]	63.3	80.6	**98.0**
	SGMNet	**70.4**	**85.7**	**98.0**
ContextDesc [25]	MNN	65.3	80.6	90.8
	OANet	74.5	86.7	**99.0**
	SuperGlue	**77.6**	86.7	**99.0**
	SGMNet	75.5	**87.8**	**99.0**
SuperPoint [11]	MNN	71.4	78.6	87.8
	OANet	77.6	86.7	98.0
	SuperGlue*	**79.6**	**90.8**	**100.0**
	SuperGlue	76.5	88.8	99.0
	SGMNet	77.6	88.8	99.0

Table 4. Evaluation results on Aachen Day-Night dataset. We report pose accuracy under different thresholds for challenging night spilt. We include the results of official released model of SuperGlue with SuperPoint (denoted as SuperGlue*).

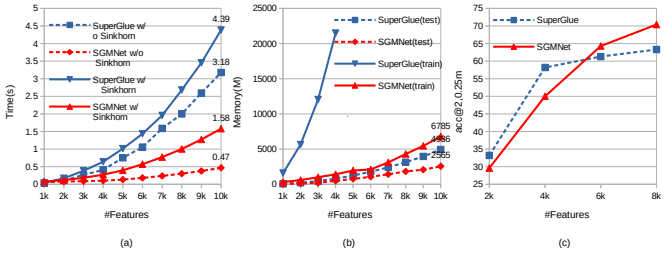


Figure 5. The computation (a) and memory (b) efficiency compared the proposed method with SuperGlue. We report memory occupation averaged by batch size for training. The effect of keypoint number on Aachen Day-Night dataset is illustrated in (c).

4.3. Scalability

In above experiments, the proposed method has shown competitive results against the state-of-the-arts. In this ses-

Matcher	*AUC@20°*	*M.S.*	*Prec.*
NN + RT	68.58	10.05	56.38
SGMNet w Rand. Seed	71.25	12.94	55.57
SGMNet w/o W.U.	78.64	17.07	81.26
SGMNet w/o A.P.	79.35	17.11	82.15
SGMNet w/o Reseeding	80.41	17.12	84.47
SGMNet full	**81.48**	**17.08**	**86.08**

Table 5. Results of ablation study. w/o A.P. stands for w/o attentional pooling, where seed features are directly sent to seed filtering process without attending to original keypoints. w/o W.U. stands for w/o weighted unpooling, where vanilla attention are preformed in unpooling process. Rand. Seed means selecting seed correspondences randomly instead of picking the top-k scores. w/o Reseeding means only use initial seeding.

sion, we demonstrate the major advantage of our method in time/memory efficiency compared with SuperGlue, a closely related method based on GNN.

Time/memory consumption. As shown in Fig. 5(a), the time cost of our method is remarkably lower than SuperGlue. Specifically, we report both run time with and without Sinkhorn iterations on a GTX 1080 GPU, in order to more precisely demonstrate the substantial improvements on GNN design itself. It is noteworthy that on $10k$ keypoints and without Sinkhorn iterations, the proposed method reduces the runtime by one magnitude. Besides, due to the reduction of redundancy, SGMNet also delivers better convergence during training (see Appendix C).

As shown in Fig. 5(b), during test phase, our method consumes half of memory than SuperGlue when keypoint number is larger than $2k$, where the major memory peak of our method is seeding phase and sinkhorn iterations. The advantage becomes even more significant during training. With a batch size of 16 and keypoint number of $1k$, SuperGlue occupies up to 23GB GPU memory for training while SGMNet only consumes less than 9GB memory.

Performance gain when using more keypoints. Within a reasonable range, larger keypoint number generally improves performance of downstream tasks, thus a manageable matching cost is of practical significance to extend the applicability. As a showcase, we vary the keypoint number of RootSIFT when evaluating on Aachen Day-Night Dataset. As can be seen from Fig. 5(c), the accuracy of SGMNet and SuperGlue increases as more keypoints are used. Considering the efficiency advantage of our method, SGMNet delivers better trade-off when increasing the keypoint number. We also provide an experiment of SfM, a typical keypoint-consuming application in Appendix D.3.

5. Discussions

5.1. Ablation Study

To evaluate the effectiveness of different components of our method, we conduct ablation study on YFCC100M dataset using RootSIFT. As shown in Table 5, all different component in our network contributes to the final performance notably. In particular, seeding reliable matches plays an important role, which further proves that seed matches is able to guide message across images for robust matching.

Type	*Matcher*	*AUC*			*Time(ms)*
		@5°	*@10°*	*@20°*	
GNN	SGMNet-10	66.62	76.33	**84.71**	114.81
	SGMNet	**67.42**	**76.63**	84.66	284.66
	SuperGlue-10	65.85	75.35	84.05	458.52
	SuperGlue	66.65	75.41	84.12	604.11
Filter	OANet	63.75	73.60	82.43	21.30
	ACNe	63.37	73.67	82.74	18.26
Handcrafted	AdaLAM	57.78	68.01	77.38	4.59
	GMS	26.15	33.53	42.13	5.44

Table 6. Results on YFCC100M using 4k SIFT features. -10 means setting sinkhorn iterations as 10 instead of 100

5.2. Comparison with Filter-based Methods

For a well-around comparison, we provide in Table 6 more experiment results with filter-based methods (outlier rejection). When using 4k keypoints, SGMNet achieves best performance among all comparative methods while runs 4 times faster than SuperGlue when setting sinkhorn iterations as 10. Despite the fast inference speed of SOTA filter-based methods, a considerable performance gap still remains compared with GNN based methods.

5.3. Designs of SGMNet

We experiment on other designs for SGMNet, including other seeding strategy and pooling operations in GNN/Transformer architecture [51, 54, 61], e.g. diffPool [62, 61] and set transformer [20] . We find 1) learnable seeding acheives limitied improvement over our simple heuristic seeding strategy 2) other general pooling operations, which are often verified on self-attention in GNN/Transformer architecture, are not effective alternatives of our seed-based pooling. Details of our experiments can be found in Appendix B. Hyper-parameter study on seeding number can be found in Appendix D.

6. Conclusion

In this paper, we propose SGMNet, a novel graph neural network for efficient image matching. The new operations we have developed enable message passing with a compact attention pattern. Experiments on different tasks and datasets prove that our method improves the accuracy of feature matching and downstream tasks to competitive or higher level against the state-of-the-arts with a modest computation/memory cost.

Acknowledgment. This work is supported by Hong Kong RGC GRF 16206819, 16203518, T22-603/15N and Guangzhou Okay Information Technology with the project GZETDZ18EG05.

References

[1] Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012. 6, 7

[2] Xuyang Bai, Zixin Luo, Lei Zhou, Hongkai Chen, Lei Li, Zeyu Hu, Hongbo Fu, and Chiew-Lan Tai. Pointdsc: Robust point cloud registration using deep spatial consistency. In *CVPR*, 2021. 2

[3] Jia-Wang Bian, Yu-Huan Wu, Ji Zhao, Yun Liu, Le Zhang, Ming-Ming Cheng, and Ian Reid. An evaluation of feature matchers for fundamental matrix estimation. In *BMVC*, 2019. 5, 6

[4] Eric Brachmann and Carsten Rother. Neural-guided ransac: Learning where to sample model hypotheses. In *ICCV*, 2019. 2

[5] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, and A. J. Smola. Learning graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009. 1, 2

[6] Luca Cavalli, Viktor Larsson, Martin Ralf Oswald, Torsten Sattler, and Marc Pollefeys. Handcrafted outlier detection revisited. In *ECCV*, 2020. 6

[7] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. In *arXiv*, 2019. 2

[8] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NeurIPs*, 2013. 5

[9] Angela Dai, Matthias Nießner, Michael Zollöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface re-integration. *ACM Transactions on Graphics 2017 (TOG)*, 2017. 5, 6

[10] François Darmon, Mathieu Aubry, and Pascal Monasse. Learning to guide local feature matches. 2020. 2, 6

[11] D. DeTone, T. Malisiewicz, and A. Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPRW*, 2018. 2, 6, 7

[12] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint description and detection of local features. In *CVPR*, 2019. 2

[13] Matthias Fey, Jan E. Lenssen, Christopher Morris, Jonathan Masci, and Nils M. Kriege. Deep graph matching consensus. In *ICLR*, 2020. 2

[14] Donniell E. Fishkind, Sancar Adali, Heather G. Patsolic, Lingyao Meng, Digvijay Singh, Vince Lyzinski, and Carey E. Priebe. Seeded graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018. 3

[15] Catherine FRAIKIN and Paul Van Dooren. Graph matching with type constraints on nodes and edges. *Dagstuhl Seminar Proceedings*, 2007. 3

[16] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 5

[17] M. Goesele, B. Curless, and S.M. Seitz. Multi-view stereo revisited. In *CVPR*, 2006. 1

[18] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *ICLR*, 2020. 2

[19] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 2017. 5

[20] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*, 2019. 2, 8

[21] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, 2005. 2

[22] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *CVPR*, 2018. 6

[23] Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. In *ICLR*, 2018. 2

[24] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 1, 3, 6

[25] Zixin Luo, Tianwei Shen, Lei Zhou, Jiahui Zhang, Yao Yao, Shiwei Li, Tian Fang, and Long Quan. Contextdesc: Local descriptor augmentation with cross-modality context. In *CVPR*, 2019. 2, 6, 7

[26] Zixin Luo, Tianwei Shen, Lei Zhou, Siyu Zhu, Runze Zhang, Yao Yao, Tian Fang, and Long Quan. Geodesc: Learning local descriptors by integrating geometry constraints. In *ECCV*, 2018. 2

[27] 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*, 2020. 2

[28] Vince Lyzinski, Daniel L. Sussman, Donniell E. Fishkind, Henry Pao, Li Chen, Joshua T. Vogelstein, Youngser Park, and Carey E. Priebe. Spectral clustering for divide-and-conquer graph matching. *Parallel Computing*, 2015. 3

[29] Josef Maier, Martin Humenberger, Markus Murschitz, Oliver Zendel, and Markus Vincze. Guided matching based on statistical optical flow for fast and robust correspondence analysis. In *ECCV*, 2016. 2

[30] Anastasiia Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. In *NeurIPs*, 2017. 2

[31] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 2015. 1

[32] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 1

[33] Filip Radenović, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondřej Chum. Revisiting oxford and paris: Large-scale image retrieval benchmarking. In *CVPR*, 2018. 6

[34] Jerome Revaud, Philippe Weinzaepfel, César De Souza, Noe Pion, Gabriela Csurka, Yohann Cabon, and Martin Humenberger. R2d2: Repeatable and reliable detector and descriptor. In *NeurIPs*, 2019. 2

[35] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 1, 2, 5, 6, 7

[36] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, Fredrik Kahl, and Tomas Pajdla. Benchmarking 6dof outdoor visual localization in changing conditions. In *CVPR*, 2018. 7

[37] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 1

[38] Tianwei Shen, Zixin Luo, Lei Zhou, Runze Zhang, Siyu Zhu, Tian Fang, and Long Quan. Matchable image retrieval by learning from surface reconstruction. In *ACCV*, 2018. 5, 6

[39] Xi Shen, François Darmon, Alexei A Efros, and Mathieu Aubry. Ransac-flow: generic two-stage image alignment. *ECCV*, 2020. 6

[40] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative Learning of Deep Convolutional Feature Point Descriptors. In *ICCV*, 2015. 2

[41] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *IROS*, 2012. 5

[42] K. Sun, W. Tao, and Y. Qian. Guide to match: Multi-layer feature matching with a hybrid gaussian mixture model. *IEEE Transactions on Multimedia*, 2020. 2

[43] Weiwei Sun, Wei Jiang, Eduard Trulls, Andrea Tagliasacchi, and Kwang Moo Yi. Acne: Attentive context normalization for robust permutation-equivariant learning. In *CVPR*, 2020. 1, 2

[44] Weiwei Sun, Wei Jiang, Eduard Trulls, Andrea Tagliasacchi, and Kwang Moo Yi. Acne: Attentive context normalization for robust permutation-equivariant learning. In *CVPR*, 2020. 6

[45] Shitao Tang, Chengzhou Tang, Rui Huang, Siyu Zhu, and Ping Tan. Learning camera localization via dense scene matching. In *CVPR*, 2021. 2

[46] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. In *ICML*, 2020. 2

[47] 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*, 2016. 5, 6

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

[49] Lorenzo Torresani, Vladimir Kolmogorov, and Carsten Rother. Feature correspondence via graph matching: Models and global optimization. In *ECCV*, 2008. 1, 2

[50] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1988. 2

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPs*. 2017. 2, 8

[52] Yannick Verdie, Kwang Moo Yi, Pascal Fua, and Vincent Lepetit. Tilde: A temporally invariant learned detector. In *CVPR*, 2015. 2

[53] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Learning combinatorial embedding networks for deep graph matching. In *ICCV*, 2019. 2

[54] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. In *arXiv*, 2020. 2, 8

[55] Z. Wang and S. Ji. Second-order pooling for graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 2

[56] Kyle Wilson and Noah Snavely. Robust global translations with 1dsfm. In *ECCV*, 2014. 5

[57] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *ECCV*, 2018. 2

[58] Yao Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. Recurrent mvsnet for high-resolution multi-view stereo depth inference. In *CVPR*, 2019. 2

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

[60] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *CVPR*, 2018. 1, 2, 4, 5, 6

[61] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *NeurIPs*, 2018. 2, 8

[62] Jiahui Zhang, Dawei Sun, Zixin Luo, Anbang Yao, Lei Zhou, Tianwei Shen, Yurong Chen, Long Quan, and Hongen Liao. Learning two-view correspondences and geometry using order-aware network. In *ICCV*, 2019. 1, 2, 3, 5, 6, 7, 8

[63] Jingyang Zhang, Yao Yao, Shiwei Li, Zixin Luo, and Tian Fang. Visibility-aware multi-view stereo network. In *BMVC*, 2020. 2

[64] Jingyang Zhang, Yao Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. Learning stereo matchability in disparity regression networks. In *ICPR*, 2020. 2

[65] Zhengyou Zhang. Determining the epipolar geometry and its uncertainty: A review. *IJCV*, 1998. 6

[66] Lei Zhou, Zixin Luo, Tianwei Shen, Jiahui Zhang, Mingmin Zhen, Yao Yao, Tian Fang, and Long Quan. Kfnet: Learning temporal camera relocalization using kalman filtering. In *CVPR*, 2020. 2