Learning Two-View Correspondences and Geometry Using Order-Aware Network

Jiahui Zhang[13*†] Dawei Sun[2*†] Zixin Luo[3‡] Anbang Yao [2§] Lei Zhou[3‡] Tianwei Shen[4]
Yurong Chen[2] Long Quan[3] Hongen Liao[1§]
[1]Tsinghua University [2]Intel Labs China [3]Hong Kong University of Science and Technology
[4]Everest Innovation Technology (Altizure)
{jiahui-z15@mails.,liao@}tsinghua.edu.cn {dawei.sun,anbang.yao,yurong.chen}@intel.com
{zluoag,lzhouai,quan}@cse.ust.hk tianwei@altizure.com

Abstract

Establishing correspondences between two images requires both local and global spatial context. Given putative correspondences of feature points in two views, in this paper, we propose Order-Aware Network, which infers the probabilities of correspondences being inliers and regresses the relative pose encoded by the essential matrix. Specifically, this proposed network is built hierarchically and comprises three novel operations. First, to capture the local context of sparse correspondences, the network clusters unordered input correspondences by learning a soft assignment matrix. These clusters are in a canonical order and invariant to input permutations. Next, the clusters are spatially correlated to form the global context of correspondences. After that, the context-encoded clusters are recovered back to the original size through a proposed upsampling operator. We intensively experiment on both outdoor and indoor datasets. The accuracy of the two-view geometry and correspondences are significantly improved over the state-of-the-arts.

1. Introduction

Two-view geometry estimation is a fundamental problem in computer vision, which plays an important role in Structure from Motion (SfM) [37, 33] and visual Simultaneous Localization and Mapping (SLAM) [21]. Current state-of-the-art SfM [37, 33] and SLAM [21] pipelines commonly start from local feature extraction and matching. Outlier rejection algorithm is then applied which is necessary for accurate relative pose estimation. After that, the relative pose can be recovered from inliers.

Until recently, great efforts have been spent on applying deep learning techniques to geometric matching pipeline, and most of them focus on learning local feature detectors and descriptors [40, 4]. More interestingly, learning-based outlier rejection has also been revisited [20, 29] and achieves appealing results. Our work also applies learning-based outlier rejection as the core component for two-view geometry estimation. We exploit a neural network to infer the probability of each correspondence as an inlier, then recover the relative pose by regressing the essential matrix through a closed-form and differentiable computation. The overview of the workflow is illustrated in Fig. 1.

Previous works [20, 29] exploited PointNet-like architecture [26] and Context Normalization [20, 35] to classify putative correspondences, which we refer to as PointCN. It has following drawbacks: (1) PointNet-like architecture applies Multi Layer Perceptrons (MLPs) on each point individually. Hence it cannot capture the local context [27], *e.g.*, similar motion shared by neighboring pixels [1], which has been shown to be beneficial for outlier rejection [1, 44]. (2) PointCN relies on Context Normalization to encode the global context. Such a simple operation normalizes the feature maps by their mean and variance, which overlooks the underlying complex relations among different points and may hinder the overall performance.

One of the challenges in mitigating the above limitations is exploiting neighbors to encoding local context. Unlike 3D point clouds, sparse matches have no well-defined neighbors, where this issue is previously tackled in bilateral domain [14] (2D spatial domain and 2D motion domain) or by a graphical model [44]. Besides, another challenge is modeling the relation between correspondences since they are unordered and have no stable relations to be captured.

To address the above two problems, we draw inspiration from hierarchical representations of Graph Neural Net-

*indicates equal contributions.
†interns at Intel Labs China.
‡interns at Shenzhen Zhuke Innovation Technology (Altizure).
§indicates corresponding authors.

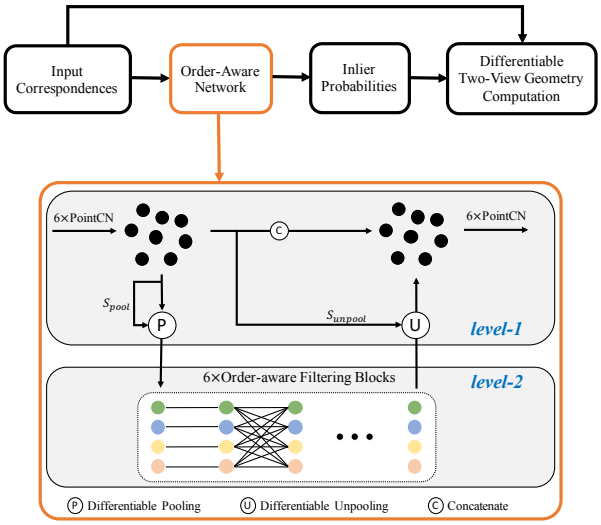


Figure 1. The Order-Aware Network to learn two-view correspondences and geometry. PointCN blocks are used in level-1 to process unordered input. Besides, we introduce three novel operations to exploit the local and global context: (1) The DiffPool layer (left), which maps unordered nodes to a set of clusters in a canonical order to caputre local context; (2) the Order-Aware DiffUnpool layer (right), which upsamples the clusters using the spatial information of input nodes to build a hierarchical architecture; (3) the Order-Aware Filtering block in level-2, which correlates the clusters thus allows the network to better model the global context.

work (GNN). In particular, we generalize the Differentiable Pooling (DiffPool) [41] operator, which is permutation-invariant and originally designed for GNN, into a PointNet-like framework to capture the local context. Specifically, as shown in Fig. 1, DiffPool maps input nodes to a set of clusters by learning a soft assignment matrix, instead of using pre-defined heuristic neighbors. Meanwhile, the permutation-invariant DiffPool essentially yields a canonical order for the resulting clusters, which eschews the need of heuristic sorting such as [22, 43]. Moreover, being in a canonical order further enables us to exploit the cluster relation with effective spatially-correlated operators, *i.e.* the proposed Order-Aware Filtering block, to capture more complex global context. Finally, to assign per-correspondence predictions, we develop a novel Differentiable Unpooling (DiffUnpool) layer to upsample these clusters to the original size. It is noteworthy that the proposed DiffUnpool operator is specially designed to be order-aware so as to precisely align the upsampled features with the original input correspondences.

The proposed method is extensively evaluated on both large-scale indoor and outdoor datasets with diverse scenes and achieves significant accuracy improvements on relative pose estimation over the-state-of-the-arts.

Our main contributions are threefold:

- We introduce the DiffPool and DiffUnpool layers to capture the local context of unordered sparse correspondences in a learnable manner.
- By the collaborative use of DiffPool operator, we propose Order-Aware Filtering block which exploits the complex global context of sparse correspondences.
- Our work significantly improves the relative pose estimation accuracy on both outdoor and indoor datasets.

2. Related Work

2.1. Learning based Matching

With the emergence of deep learning, many works attempted to employ learning-based methods to solve geometric matching tasks, including both dense methods [36, 45, 13, 30] and sparse methods [40, 23, 4, 3, 18, 17]. For these sparse methods, most of them focused on interest point extraction and description with convolutional neural network (CNN) to replace handcrafted features such as SIFT [16]. Meanwhile, some works [2, 20, 29] also attempted to solve the outlier rejection problem with learning-based methods to improve the accuracy of relative pose estimation, which is the topic of this work.

2.2. Outlier Rejection

Typically, putative correspondences established by handcrafted or learned features contain many outliers, *e.g.* in the wide baseline case. So outlier rejection is necessary to improve relative pose estimation accuracy. RANSAC [6] is the standard and still the most popular outlier rejection method. USAC [28] provided a universal framework for RANSAC variants. BF [14] utilized a piecewise smoothness constraint on the bilateral domain to filter outliers. GMS [1] simplified the idea of smoothness constraints as a statistical formulation. RMBP [44] defined a graphical model which describes the spatial organization of matches to reject outliers.

In the deep learning era, DSAC [2] mimicked the behavior of RANSAC and proposed a differentiable counterpart using probabilistic selection. PointCN [20] reformulated the outlier rejection task as an inlier/outlier classification problem and an essential matrix regression problem. It exploited PointNet-like architecture to label input correspondences as either inliers or outliers and introduced a weighted eight-point algorithm to directly regress essential matrix. Context Normalization was proposed which can drastically improve the performance. A concurrent work DFE [29] also used PointNet-like architecture and Context Normalization but adopted a different loss function and an iterative network. N^3Net [25] inserted soft k-nearest neighbors (KNN) layer to augment PointCN. Our work is also built on PointCN but puts effort on improving the local and global contexts by borrowing ideas from Geometric Deep Learning.

2.3. Geometric Deep Learning

Geometric Deep Learning deals with data on non-Euclidean domains, such as graphs [11, 19, 8, 5] and manifolds [26, 39, 12, 7, 42]. PointNet-like architecture can be regarded as a special case of Graph Neural Network which processes graphs without edges. Different from 3D point clouds, sparse correspondences have no well-defined neighbors. This is also a difficulty faced by many tasks on graphs [41]. Instead of defining heuristic neighbors for correspondences as done in previous works [14, 44], we exploit Differentiable Pooling [41] to cluster nodes in a learnable manner and capture the local context. However, the original DiffPool Network is not applicable in our case because it does not give a full size prediction. Hence, we propose a novel DiffUnpool layer to upsample the coarsened feature maps and build a hierarchical architecture. Moreover, we introduce an Order-Aware Filtering block with spatial connections to capture the global context.

3. Order-Aware Network

We will present Order-Aware Network for learning two-view correspondences and geometry, which contains three novel operations: Differentiable Pooling layer, Order-Aware Differentiable Unpooling layer, and Order-Aware Filtering block. The formulation of our problem is first introduced, and then these submodules successively.

3.1. Problem Formulation

Given image pairs, the goal of our task is to remove outliers from putative correspondences and recover the relative pose. More specifically, after extracting keypoints and their descriptors in each image using handcrafted features [16, 32] or learned features [40, 4], putative correspondences can be established by finding their nearest neighbors in the other image. Then outlier rejection method is applied to establish geometrically consistent correspondences. Finally, an essential matrix can be recovered from the inlier correspondences by a closed-form solution [15, 20].

The input to the outlier rejection process is a set of putative correspondences:

$$\mathbf{C} = [c_1; c_2; ...; c_N] \in \mathcal{R}^{N\times 4}, c_i = (x_1^i, y_1^i, x_2^i, y_2^i), \quad (1)$$

where c_i is a correspondence and (x_1^i, y_1^i), (x_2^i, y_2^i) are the coordinates of keypoints in these two images. The coordinates are normalized using camera intrinsics [20].

Following [20], we formulate the two-view geometry estimation task as an inlier/outlier classification problem and an essential matrix regression problem. We use a neural network to predict the probability of each correspondence to be an inlier and then apply the weighted eight-point algorithm [20] to directly regress the essential matrix. The architecture can be written as:

$$\mathbf{z} = f_\phi(\mathbf{C}), \quad (2)$$

$$\mathbf{w} = \tanh(\mathrm{ReLU}(\mathbf{z})), \quad (3)$$

$$\hat{\mathbf{E}} = g(\mathbf{w}, \mathbf{C}), \quad (4)$$

where $\mathbf{z}$ is the logit values for classification. $f_\phi(\cdot)$ is a permutation-equivariant neural network and ϕ denotes the network parameters. $\mathbf{w}$ is the weights of correspondences. For each weight $w_i \in [0, 1)$, $w_i = 0$ means an outlier. tanh and ReLU are applied to easily remove outliers [20]. $g(\cdot, \cdot)$ in Eq. 4 is the weighted eight-point algorithm. $\hat{\mathbf{E}}$ is the regressed essential matrix. $g(\cdot, \cdot)$ takes more than eight correspondences and their weights to compute essential matrix via self-adjoint eigendecomposition. The weighted eight-point algorithm can be more robust to outliers than traditional eight-point algorithm [9] because it has considered the contribution of each correspondence. Besides, it is differentiable with respect to $\mathbf{w}$ which makes it possible to regress the essential matrix in an end-to-end manner.

The optimization objective of this neural network is to minimize a classification loss and an essential matrix loss as follows:

$$loss = l_{cls}(\mathbf{z}, \mathbf{s}) + \alpha l_{ess}(\hat{\mathbf{E}}, \mathbf{E}), \quad (5)$$

where l_{ess} is the essential matrix loss between the predicted essential matrix $\hat{\mathbf{E}}$ and the ground truth essential matrix $\mathbf{E}$. It can be a $L2$ loss [20]

$$loss_{L2} = \min\{\|\hat{\mathbf{E}} \pm \mathbf{E}\|\} \quad (6)$$

or a geometry loss [29, 9]

$$loss_{geo} = \frac{\left(\mathbf{p_2^T}\hat{\mathbf{E}}\mathbf{p_1}\right)^2}{\|\mathbf{E}\mathbf{p_1}\|_{[1]}^2 + \|\mathbf{E}\mathbf{p_1}\|_{[2]}^2 + \|\mathbf{E^T}\mathbf{p_2}\|_{[1]}^2 + \|\mathbf{E^T}\mathbf{p_2}\|_{[2]}^2}, \quad (7)$$

where $\mathbf{p_1}, \mathbf{p_2}$ are correspondences and $\mathbf{t}_{[i]}$ denotes the ith element of vector $\mathbf{t}$. l_{cls} is a binary cross entropy loss for the classification term. $\mathbf{s}$ denotes weakly supervised labels for correspondences, which are also derived using the above geometric error, and a threshold of 10^{-4} is used to determine valid correspondences. α is the weight to balance these two losses.

3.2. Differentiable Pooling Layer

The unordered input correspondences require network $f_\phi(\cdot)$ to be permutation-equivariant. So PointNet-like architecture was used [20, 29]. Each block in the PointNet-like [20] architecture comprises one Context Normalization layer, one Batch Normalization layer with ReLU, and one shared Perceptron layer. This so called PointCN block is shown in Fig. 2. The proposed Context Normalization layer

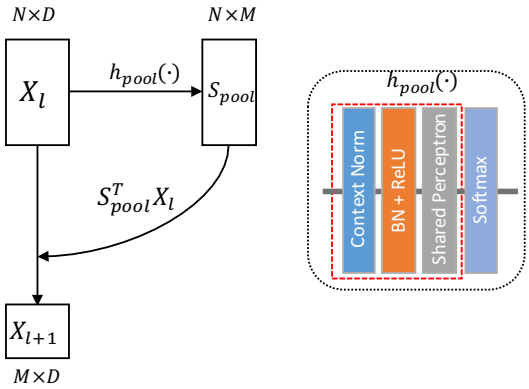
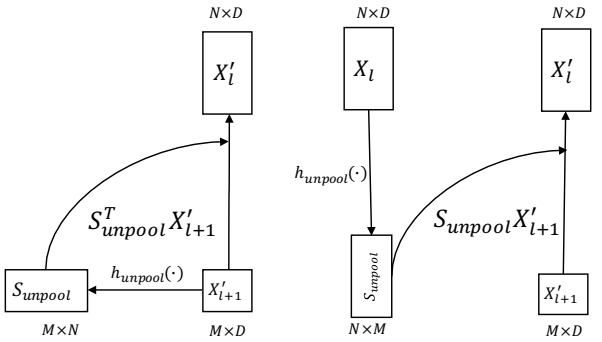


Figure 2. Differentiable Pooling layer. DiffPool maps nodes to clusters in a soft assignment manner. The soft assignment matrix is learned by $h_{pool}(\cdot)$ which contains one PointCN block (in dashed red box) and one softmax layer.



(a) Plain DiffUnpool (b) Order-Aware DiffUnpool

Figure 3. Designs of Differentiable Unpooling layer. (a) Plain DiffUnpool layer. It learns a soft assignment matrix using features at level $l+1$. (b) Order-Aware DiffUnpool layer. It learns a soft assignment matrix using features at level l which can encode the order information of nodes at level l.

[20] normalizes features of each sample using their statistics and can largely boost the performance.

However, PointNet-like architecture has the drawback in capturing the local context because there is no direct interaction between points. In order to capture the local context for sparse correspondences, we draw the idea from DiffPool layer [41] to learn to cluster nodes to a coarser representation, as shown in Fig. 2. The DiffPool layer is analogous to Pooling layer in CNN which assigns nodes to different clusters. Rather than employing a hard assignment for each node, the DiffPool layer learns a soft assignment matrix. Denoting the assignment matrix as $\mathbf{S}_{pool} \in \mathcal{R}^{N\times M}$, DiffPool layer maps N nodes to M clusters:

$$\mathbf{X}_{l+1} = \mathbf{S}_{pool}^T \mathbf{X}_l, \tag{8}$$

where $\mathbf{X}_l \in \mathcal{R}^{N\times D}$ and $\mathbf{X}_{l+1} \in \mathcal{R}^{M\times D}$ are the features at level l and level $l+1$ respectively. D is the dimension of features, and typically $M < N$, *e.g.* $N = 2000, M = 500$.

As we have mentioned before, the assignment matrix is learned rather than pre-defined. More specifically, taking the features at level l, we directly generate the assignment matrix using a permutation-equivariant network as follows:

$$\mathbf{S}_{pool} = \text{softmax}(h_{pool}(\mathbf{X}_l)), \tag{9}$$

where the permutation-equivariant function $h_{pool}(\cdot)$ is one PointCN block here. It maps features from $N\times D$ to $N\times M$. Softmax layer is applied to normalize the assignment matrix along the row dimension. These clusters can be viewed as weighted average results of nodes in the previous level.

Permutation-invariance. DiffPool is a permutation-invariant[1] operation [41], which will play a crucial role in our design. Assuming permuting $\mathbf{X}_l$ with a permutation matrix $\mathbf{P} \in \{0,1\}^{N\times N}$, Eq. 9 becomes

$$\tilde{\mathbf{S}}_{pool} = \text{softmax}(h_{pool}(\mathbf{P}\mathbf{X}_l)) = \mathbf{P}\mathbf{S}_{pool}, \tag{10}$$

because both $h_{pool}(\cdot)$ and softmax are permutation-equivariant functions. So, according to Eq. 8, features at level $l+1$ become

$$\mathbf{X}_{l+1} = \tilde{\mathbf{S}}_{pool}^T \mathbf{P}\mathbf{X}_l = \mathbf{S}_{pool}^T \mathbf{P}^T \mathbf{P}\mathbf{X}_l = \mathbf{S}_{pool}^T \mathbf{X}_l, \tag{11}$$

since $\mathbf{P}^T\mathbf{P} = \mathbf{I}$ holds for every permutation matrix. Eq. 11 and Eq. 8 prove the permutation-invariance property of DiffPool layer.

The permutation-invariance property also means that, once the network is learned, no matter how the input are permuted, they will be mapped into clusters in a particular ***learned canonical order*** by the DiffPool layer. This canonical order is determined by the parameters of $h_{pool}(\cdot)$.

3.3. Differentiable Unpooling Layer

DiffPool Network was used to predict the label for an entire graph [41]. However, it is not applicable for our sparse matching problem, since we need to give predictions for all correspondences. So, we develop a Differentiable Unpooling layer inspired by the DiffPool layer to upsample the coarse representation and build a hierarchical architecture.

A straightforward way to implement the DiffUnpool layer is reversing the behavior of DiffPool layer, as shown in Fig. 3a. More specifically, similar to Eq. 8 and Eq. 9, an unpooling assignment matrix $\mathbf{S}_{unpool} \in \mathcal{R}^{M\times N}$ is first predicted taking features $\mathbf{X}'_{l+1}$ through:

$$\mathbf{S}_{unpool} = \text{softmax}(h_{unpool}(\mathbf{X}'_{l+1})), \tag{12}$$

where $\mathbf{X}'_{l+1} \in \mathcal{R}^{M\times D}$ denotes new features at the same level of $\mathbf{X}_{l+1}$, and it is computed from $\mathbf{X}_{l+1}$. We then map

[1] Equivariance means applying a transformation to input equals to applying the same transformation to output, while invariance means applying a transformation to input will not affect the output.

features $\mathbf{X}^{'}_{l+1}$ to a new embedding $\mathbf{X}^{'}_{l} \in \mathcal{R}^{N\times D}$ at level l as follows:

$$\mathbf{X}^{'}_{l} = \mathbf{S}^{T}_{unpool}\mathbf{X}^{'}_{l+1}. \tag{13}$$

However, we find the above implementation is not optimal because it cannot align the unpooled features $\mathbf{X}^{'}_{l}$ with features $\mathbf{X}_l$ in the previous stage (see section 4.4). The point is that DiffPool is a permutation-invariant operation, which means one $\mathbf{X}_{l+1}$ can correspond to various input $\mathbf{X}_l$. In the other words, features $\mathbf{X}_{l+1}$ and $\mathbf{X}^{'}_{l+1}$ at level $l+1$ have lost the spatial order information of features $\mathbf{X}_l$ at level l. We cannot expect the learned assignment matrix as in Eq. 12 can recover the original spatial order of $\mathbf{X}_l$ or generate features which can be precisely aligned with $\mathbf{X}_l$, since $\mathbf{S}_{unpool}$ in Eq. 12 only utilizes information at level $l+1$.

Keeping this in mind, we propose an Order-Aware DiffUnpool layer as shown in Fig. 3b, which can be aware of the particular order (position) of nodes in the previous level. Different from the above implementation, the assignment matrix for unpooling is learned from features at level l which has stored the input order information as follows:

$$\mathbf{S}_{unpool} = \text{softmax}(h_{unpool}(\mathbf{X}_l)). \tag{14}$$

With this unpooling assignment matrix $\mathbf{S}_{unpool} \in \mathcal{R}^{N\times M}$, we can map features at level $l+1$ to level l by:

$$\mathbf{X}^{'}_{l} = \mathbf{S}_{unpool}\mathbf{X}^{'}_{l+1}. \tag{15}$$

Since each row in this $\mathbf{S}_{unpool} \in \mathcal{R}^{N\times M}$ corresponds to one node in $\mathbf{X}_l$, it has already encoded the particular order information of $\mathbf{X}_l$ and ensures the unpooled features can well aligned to the previous stage. The mapping in Eq. 15 also requires the learned assignment matrix to be aware of the order of $\mathbf{X}^{'}_{l+1}$. But it is much easier for the network this time since the feature $\mathbf{X}^{'}_{l+1}$ is in a canonical order. $h_{unpool}(\cdot)$ in Eq. 14 is also a PointCN block and it maps features from $N\times D$ to $N\times M$. We apply the softmax along the column dimension this time[2], so the unpooled features can be viewed as weighted average results of different clusters. $\mathbf{X}^{'}_{l}$ is then concatenated with $\mathbf{X}_l$ to fuse shallow features.

Another advantage of the proposed Order-Aware DiffUnpool layer is that it does not require a fixed size input. When there are less than or more than 2000 keypoints in images, we can still pool nodes to fixed 500 clusters and then upsample clusters back to the same size. This is useful in practice.

3.4. Order-Aware Filtering Block

With the DiffPool and DiffUnpool layers, we can build a multiscale network which is a common practice in CNN.

[2]Actually we find changing the normalization directions in Eq. 9 and Eq. 14 only has little influence on results. They do not even need to be orthogonal.

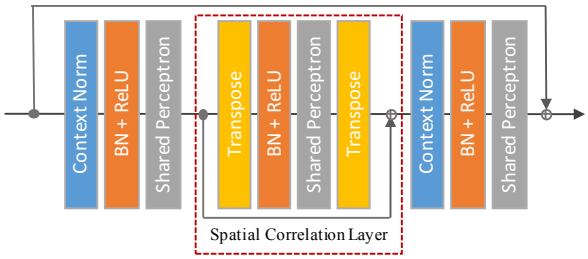


Figure 4. Order-Aware Filtering block. We insert the Spatial Correlation layer to PointCN ResNet block. This layer is complementary to PointCN and can help capture the global context effectively.

We can apply PointCN blocks repeatedly to process these newly generated clusters. However, as we have discussed above, PointCN may have weakness in modeling the complex global context because it ignores the relation between nodes. Here we propose a simple but more effective operation than PointCN block, which is called Spatial Correlation layer to explicitly model relation between different nodes and capture the complex global context.

As we have shown above, the pooled features are in a canonical order after the DiffPool layer. This is a useful property but PointNet-like architecture cannot make full use of it. Our Spatial Correlation layer applies weight-sharing perceptrons directly on the spatial dimension to establish connections between nodes. Note this operation is different from the fully connected layer because the weights are shared along the channel dimension, which can help to prevent overfitting. The Spatial Correlation layer is orthogonal to PointCN, since one is along the spatial dimension and the other is along the channel dimension. These two operations are complementary, so we assemble them into one block to better capture the global context as shown in Fig. 4.

Spatial Correlation layer is implemented by transposing the spatial and channel dimensions of features. After the weight-sharing perceptrons layer, we transpose features back. Residual connection and batch normalization with ReLU are also used. We insert the Spatial Correlation layer to the middle of PointCN ResNet block and call this composite module Order-Aware Filtering block which can process data in a canonical order. Note that before the DiffPool layer, we cannot apply the Spatial Correlation layer on the feature maps as the input data is unordered and there is no stable spatial relation to be captured. So we apply this simple block only at the level after the DiffPool layer and find it can significantly boost the performance.

4. Experiments

We conducted experiments on outdoor YFCC100M [34] dataset and indoor SUN3D [38] dataset. Experiment results and network interpretation are as follows.

threshold	S	L	mAP5(%)	mAP10(%)	mAP20(%)
0.01	✓		17.53/12.50	27.61/21.15	42.06/34.21
0.001	✓		44.50/12.50	54.50/21.15	65.27/34.21
		✓	47.98/23.55	58.13/36.58	68.67/53.08

Table 1. Performances of baseline network [20] on YFCC100M unknown sequences. Results **with/without** RANSAC under error thresholds of 5°, 10° and 20° are all reported. Changing the inlier threshold in RANSAC and using more data can significantly boost the performance. **S**: using only sequences 'Saint Peter's' and 'brown_bm_3_05' as [20]. **L**: using 68 sequences.

PointCN	UnA	UnB	OF	L3	Geo	Iter	Known	Unknown
✓							34.36/13.93	47.98/23.55
✓	✓						34.38/14.04	47.93/24.10
✓		✓					36.33/17.88	49.65/28.78
✓		✓	✓				40.78/25.94	51.63/32.55
✓		✓	✓	✓			39.69/26.04	50.70/30.48
✓		✓	✓		✓		40.79/28.39	51.10/33.68
✓		✓	✓		✓	✓	**42.46**/33.06	**52.18**/39.33

Table 2. Ablation study on YFCC100M. mAP (%) on both known and unknown scenes are reported **with/without** RANSAC post-processing. **UnA**: the plain DiffUnpool layer. **UnB**: the Order-Aware DiffUnpool layer. **OF**: using the Order-Aware Filtering blocks rather than PointCN blocks in the second level. **L3**: a larger model with three levels. **Geo**: using geometry loss rather than $L2$ loss. **Iter**: using the iterative network.

4.1. Datasets

Outdoor scenes. We use the Yahoo's YFCC100M dataset [34], which contains 100 million photos from internet. The authors of [10] later generated 72 3D reconstructions of tourist landmarks from a subset of the collections. We use four sequences [20] as unknown scenes to test generalization ability. For training sequences, different from PointCN, we use the remaining 68 sequences for training, while [20] uses only two sequences. Our setting is not prone to overfitting on known sequences and has better generalization ability as shown in Tab. 1. To have a fair comparison, we re-train all models on the same data.

Minimum visual overlap is required if pairs are selected into the dataset. For outdoor scenes, the overlap is the number of sparse 3D points in the reconstructed model which can be both seen by the image pairs. We use the camera poses and sparse models provided by [10] to generate ground-truth.

Indoor scenes. We use the SUN3D dataset [38] for indoor scenes, which is an RGBD video dataset with camera poses computed by generalized bundle adjustment. Following [36] we split the dataset into 253 scenes for training and 15 as unknown scenes for testing. This splitting can ensure there is no spatial overlap between training and testing datasets. We find some sequences in the training set do not provide camera poses, so we drop these sequences and finally get 239 sequences for training. We subsample videos every 10 frames. The visual overlap for indoor scenes is computed by projecting the depth map to the other image.

Following [20], we test on both known scenes and unknown scenes. The known scenes are the training sequences. We split them into disjoint subsets for training (60%), validation (20%) and testing (20%). The unknown sequences are the test sequences described above.

4.2. Evaluation Metrics

We use the angular differences between ground truth and predicted vectors for both rotation and translation as the error metric. mAP results with and without RANSAC post-processing are reported. We find the inlier threshold of OpenCV function `findEssentialMat()` used in [20] is not optimal. Changing the threshold from 0.01 to 0.001 will largely improve results with RANSAC, as shown in Tab. 1. We will use mAP under 5° as the default metric since it is more usable in 3D reconstruction context.

4.3. Implementation Details

The baseline network [20] has 12 PointCN ResNet blocks. Based on this network, we add one DiffPool layer and one DiffUnpool layer. Another 6 Order-Aware Filtering blocks at the second level are used as shown in Fig. 1. The channel dimensions are all 128 in these blocks. The inputs to the network are $N \times 4$ putative correspondences established using SIFT feature, typically $N = 2000$. After DiffPool layer, the number of nodes are reduced to fixed 500 which gives best performance. Besides, we also use an iterative network as [29] which takes residuals and weights of previous stage as additional inputs. This can futher improve the performance. Our network is implemented with Pytorch [24]. We use Adam solver with a learning rate of 10^{-4} and batch size 32. Weight α is 0 during the first 20k iterations and then 0.1 in the rest 480k iterations as in [20].

4.4. Ablation Studies

In this section, we will give ablation studies about the proposed operations, loss functions and network architecture on YFCC100M dataset.

DiffUnpool layer design. To demonstrate the efficacy of DiffUnpool layer, we add DiffPool and DiffUnpool layers to the baseline PointCN model. Both plain DiffUnpool and Order-Aware DiffUnpool described in section 3.3 are tested. After the DiffPool layer, another six PointCN ResNet blocks are used. Features after DiffUnpool layer are concatenated to the previous stage. As shown in Tab. 2, our Order-Aware DiffUnpool (PointCN + UnB) achieves an improvement of 5.23% over the baseline on unknown scenes when without RANSAC, while the plain DiffPool (PointCN + UnA) gives a negligible improvement over the baseline.

Plain PointCN block *vs.* Order-Aware Filtering

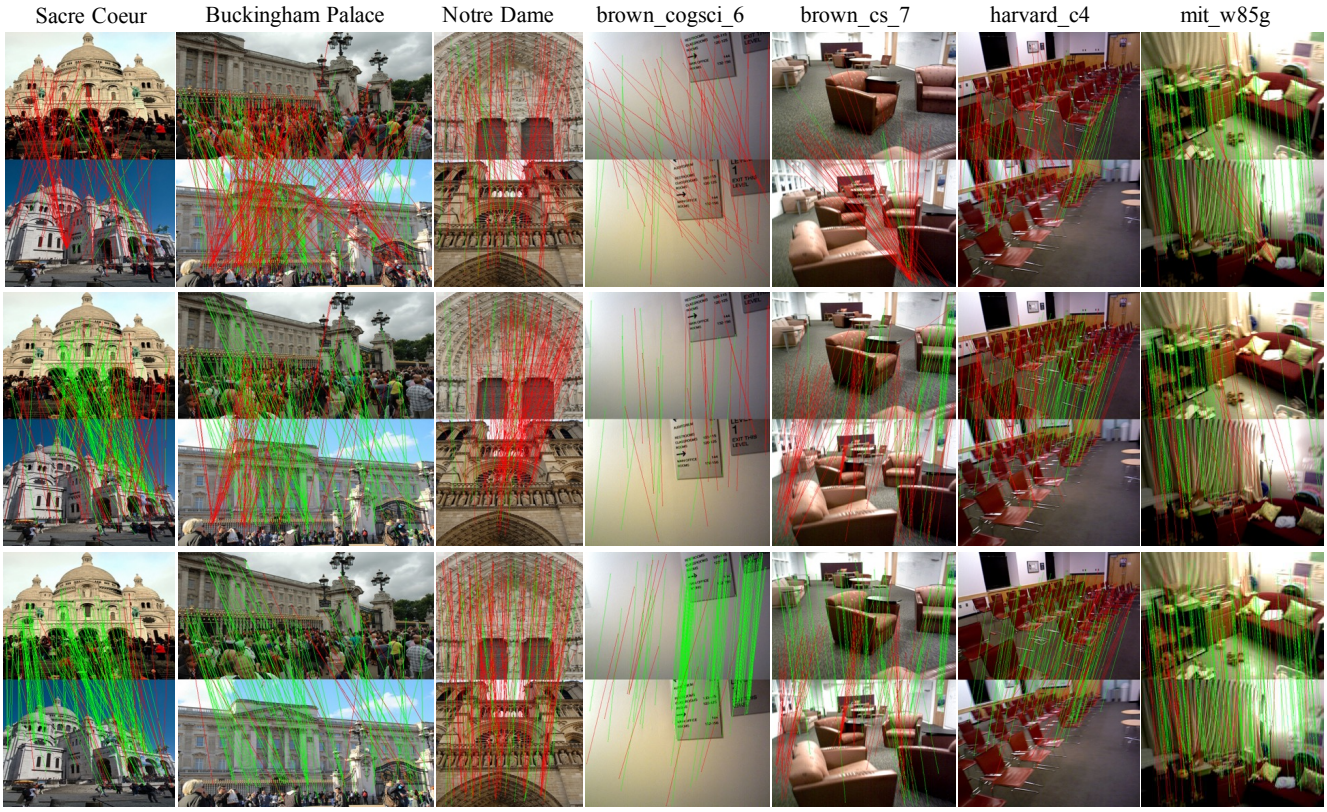


Figure 5. Matching results using RANSAC (top), PointCN [20] (middle) and our method (bottom). Images are taken from test set of YFCC100M and SUN3D datasets. Correpondences are in green if they conform the ground truth essential matrix (true positives), and in red otherwise (false positives). **Best viewed in color**.

block. We replace the PointCN blocks at the second level with Order-Aware Filtering blocks described in section 3.4, which can better exploit the spatial relationships within the clusters. As shown in Tab. 2, the proposed block (PointCN + UnB + OF) can significantly boost the performance over simple PointCN block (PointCN + UnB), achieving an improvement of 3.77% on unknown scenes without RANSAC.

Does a larger model help? We train a larger model which is a U-Net [31] with three levels. 12 PointCN ResNet blocks are used at the first level, 12 and 6 Order-Aware Filtering blocks are used at the second and third level. The number of nodes in the second and third level is 500 and 125 respectively. However, we find this larger model even drops on unknown scenes, as shown in Tab. 2. This might show that the representational ability of Order-Aware Filtering block suffices to capture the global context. So we use the two-level network in our rest experiments.

Essential matrix loss. $L2$ loss is used as the essential matrix loss in previous experiments. However, the $L2$ loss is not geometric meaningful. So we replace the $L2$ loss with the Gold Standard geometry loss [29, 9]. α is set to 0.5. Clamping the geometry losses to 0.1 works best in our case. Using the geometry loss helps a little for both known and unknown scenes as shown in Tab. 2.

Iterative network. Iterative network shares similarity [29] with traditional guided matching method. Residuals and weights are passed to next stage iteratively to guide the estimation. Here we use one initialization network and one refinement network. Each network has 6 PointCN ResNet blocks and 3 Order-Aware Filtering blocks to keep almost the same amount of parameters. We find it is really necessary to detach the gradients from latter stage. Tab. 2 shows that the iterative network can largely improve the mAP from 33.68% to 39.33% without RANSAC on unknown scenes.

4.5. Comparison to Other Baselines

We compare our network with other state-of-the-art models from [20, 25, 26, 29] on both outdoor and indoor datasets. All these models are trained under the same settings. For N^3Net [25], we use the official implementation. We find N^3Net is unstable during training, so we run it for three times and give the best results here. PointNet++ [26] is an extension of PointNet which also aims to improve the capability in capturing local context of point sets. As we have discussed before, it may not be optimal for our sparse matching problem because correspondences have no well-

	Outdoor(%)		Indoor(%)	
	Known	Unknown	Known	Unknown
RANSAC	5.82/-	9.08/-	4.38/-	2.86/-
PointCN[20]	34.36/13.93	47.98/23.55	20.44/11.28	15.98/9.36
PointNet++[26]	34.15/9.28	46.23/14.04	20.28/7.15	15.61/5.59
N^3Net[25]	34.18/12.49	49.13/23.18	20.31/7.95	15.38/7.13
DFE[29]	36.87/18.40	49.45/29.70	20.97/14.09	16.45/12.45
Ours	40.78/25.94	51.63/32.55	21.82/16.09	16.51/12.54
Ours++	**42.46**/33.06	**52.18**/39.33	**22.50**/21.44	**17.50**/16.39
RANSAC*	15.21/-	21.95/-	18.17/-	14.58/-
PointCN*[20]	30.48/13.82	43.18/24.83	23.66/12.04	18.52/10.21
Ours*	**33.42**/23.85	**46.28**/32.18	**24.31**/14.81	**19.04**/12.12

Table 3. Comparision with other baselines on YFCC100M and SUN3D. mAP (%) (**with/without** RANSAC post-processing) on are reported. **Ours++** uses the geometry loss and iterative network while **Ours** not use. Methods with * means using SuperPoint [4], otherwise using SIFT.

defined neighbors. Here we implement a 4D-version PointNet++ which exploits the 4D Euclidean space as the underlying metric space. DFE [29] is a concurrent work with [20] and has similar core designs. We implement [29] based on [20] by adopting their loss formulation and iterative network with the authors' help.

Results are shown in Tab. 3, our method achieves best results under all settings, showing improvements of 15.78% and 7.03% over PointCN [20] on both outdoor and indoor unknown scenes without RANSAC and still works well with strong RANSAC post-processing. We also provide the precision (inlier ratio), recall and F-score of each method in supplementary material. Fig. 5 shows the visualization results of our method and other baselines. It can be found that our method can give better results on several difficult scenes such as wide baselines, textureless objects, repetitive structures, and large illumination changes.

We also evaluate learned features such as SuperPoint [4] as shown in Tab. 3. It is surprising to find SuperPoint gives worse results in outdoor scenes than SIFT when using learned outlier rejection methods. Although it performs much better than SIFT when only using RANSAC. It might demonstrate that SuperPoint has better descriptors but less accurate keypoints. It can give putative correspondences with higher inlier ratio thus has better performance when only using RANSAC. But the bottleneck may become keypoint accuracy when inlier ratio is largely improved, in which situation, SuperPoint performs worse.

4.6. Network Visualization

In order to understand the mechanism of the proposed Order-Aware Network, we visualize the assignment matrix $S_{unpool} \in \mathcal{R}^{N \times M}$ of DiffUnpool layer which reflects the spatial relationships between different nodes in the first level. More specifically, we visualize the top k responses in each column of S_{unpool}. Each column in S_{unpool} represents one cluster and each row corresponds to one putative correspondence. These top k correspondences are "clustered" together because they all have a strong response to the same cluster. We find DiffUnpool can capture meaningful context for sparse matching. Fig. 6 shows that different clusters might correspond to different local motions. Moreover, we find the corresponding motion of a particular cluster are roughly consistent in different pairs and even in different scenes as shown in Fig. 7, which supports that the pooled features are in a canonical order.

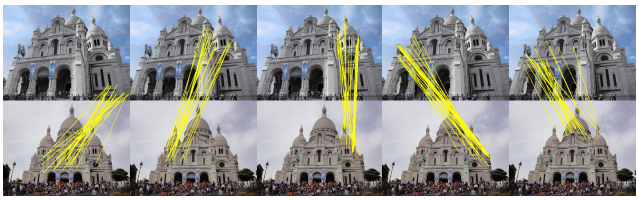


Figure 6. DiffUnpool layer visualization. Top 15 responses in different columns of S_{unpool} are visualized in the same image pair. Different clusters might correspond to different motions in different areas. **Best viewed in color with** 200% **zoom in.**

Figure 7. DiffUnpool layer visualization. Top 20 responses in the same column of S_{unpool} are visualized in different image pairs. Motions in different pairs are roughly consistent. **Best viewed in color with** 200% **zoom in.**

5. Conclusion

In this work, we proposed the Order-Aware Network for learning two-view correspondences and geometry. The introduced DiffPool layer and Order-Aware DiffUnpool layer can learn to cluster meaningful nodes to capture local context. Besides, we develop Order-Aware Filtering blocks to capture the global context. These operations can significantly improve relative pose estimation accuracy on both outdoor and indoor datasets.

Acknowledgment

This work was supported by National Natural Science Foundation of China (81427803, 81771940), National Key Research and Development Program of China (2017YFC0108000), Beijing Municipal Natural Science Foundation (7172122, L172003) and Soochow-Tsinghua Innovation Project (2016SZ0206). Part of work was done when Jiahui Zhang was an intern at ILC and visiting HKUST. We also thanks Vladlen Koltun, René Ranftl and David Hafner for helping us reimplement their work.

References

[1] JiaWang Bian, Wen-Yan Lin, Yasuyuki Matsushita, Sai-Kit Yeung, Tan-Dat Nguyen, and Ming-Ming Cheng. Gms: Grid-based motion statistics for fast, ultra-robust feature correspondence. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2

[2] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. Dsac differentiable ransac for camera localization. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[3] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Self-improving visual odometry. *arXiv preprint arXiv:1812.03245*, 2018. 2

[4] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018. 1, 2, 3, 8

[5] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 3

[6] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981. 2

[7] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 3

[8] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 3

[9] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 3, 7

[10] Jared Heinly, Johannes L Schonberger, Enrique Dunn, and Jan-Michael Frahm. Reconstructing the world* in six days*(as captured by the yahoo 100 million image dataset). In *Computer Vision and Pattern Recognition (CVPR)*, 2015. 6

[11] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. 2017. 3

[12] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *International Conference on Computer Vision (ICCV)*, 2017. 3

[13] Ruihao Li, Sen Wang, Zhiqiang Long, and Dongbing Gu. Undeepvo: Monocular visual odometry through unsupervised deep learning. In *International Conference on Robotics and Automation (ICRA)*, 2018. 2

[14] Wen-Yan Daniel Lin, Ming-Ming Cheng, Jiangbo Lu, Hongsheng Yang, Minh N Do, and Philip Torr. Bilateral functions for global motion modeling. In *European Conference on Computer Vision (ECCV)*, 2014. 1, 2, 3

[15] H Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 1981. 3

[16] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 2004. 2, 3

[17] 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 *Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[18] 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 *European Conference on Computer Vision (ECCV)*, 2018. 2

[19] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[20] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 3, 4, 6, 7, 8

[21] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 2015. 1

[22] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International Conference on Machine Learning (ICML)*, 2016. 2

[23] Yuki Ono, Eduard Trulls, Pascal Fua, and Kwang Moo Yi. Lf-net: learning local features from images. In *Advances in Neural Information Processing Systems (NIPS)*, 2018. 2

[24] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 6

[25] Tobias Plötz and Stefan Roth. Neural nearest neighbors networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2018. 2, 7, 8

[26] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 3, 7, 8

[27] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 1

[28] Rahul Raguram, Ondrej Chum, Marc Pollefeys, Jiri Matas, and Jan-Michael Frahm. Usac: a universal framework for random sample consensus. *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)*, 2013. 2

[29] René Ranftl and Vladlen Koltun. Deep fundamental matrix estimation. In *European Conference on Computer Vision (ECCV)*, 2018. 1, 2, 3, 6, 7, 8

[30] Ignacio Rocco, Relja Arandjelovic, and Josef Sivic. Convolutional neural network architecture for geometric matching.

In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[31] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 2015. 7

[32] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *International Conference on Computer Vision (ICCV)*, 2011. 3

[33] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[34] 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

[35] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, 2016. 1

[36] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 6

[37] Changchang Wu et al. Visualsfm: A visual structure from motion system. 2011. 1

[38] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *Computer Vision and Pattern Recognition (CVPR)*, 2013. 5, 6

[39] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *European Conference on Computer Vision (ECCV)*, 2018. 3

[40] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In *European Conference on Computer Vision (ECCV)*, 2016. 1, 2, 3

[41] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems (NIPS)*, 2018. 2, 3, 4

[42] Jiahui Zhang, Hao Zhao, Anbang Yao, Yurong Chen, Li Zhang, and Hongen Liao. Efficient semantic scene completion network with spatial group convolution. In *European Conference on Computer Vision (ECCV)*, 2018. 3

[43] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 2

[44] Lei Zhou, Siyu Zhu, Zixin Luo, Tianwei Shen, Runze Zhang, Mingmin Zhen, Tian Fang, and Long Quan. Learning and matching multi-view descriptors for registration of point clouds. In *European Conference on Computer Vision (ECCV)*, 2018. 1, 2, 3

[45] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 2