Supplementary Appendix

A Implementation Details

We provide in this part details about our implementations as well as experiment settings.

A.1 Training Details

Training Data. We use GL3D dataset to generate training data. GL3D dataset is originally based on 3D reconstruction of 543 different scenes, including landmarks and small objects, while in its latest version additional 713 sequences of internet tourism photos are added.

We sample 1000 pairs for each sequence and filter out pairs that are either too hard or too easy for training. More specifically, we use the common track ratio provided by original dataset and rotation angles between cameras to determine pair difficulty, pairs with common track ratio in range $[0.1, 0.5]$ and rotation angle in range $[6°, 60°]$ are kept.

We reproject keypoints between images with depth maps and use reprojection distances to determine ground truth matches and unmatchable points. More specifically, a keypoint is labeled as unmatchable if its reprojection distances with all keypoints in the other image are larger than 10 pixels, while a pair of keypoints that are mutual nearest after reprojection and with a reprojection distance lower than 3 pixels are considered ground truth matches. We further filter out pairs with ground matches fewer than 50.

Our data generation protocol yields around 400k training pairs in total.

Training Parameters. We use Adam optimizer for training optimization with learning rate of 10^{-4}. We apply learning rate decay after 300k iterations with decay rate of 0.999996 until 900k iterations. We block the gradient flow between initial/refinement stages for the first 140k iterations.

A.2 Network Details

As is done in SuperGlue [3], we apply multi-head attention mechanism for all weighted attention operations in our network with head number of 4.

For initial Seeding Module, we apply additional mutual nearest neighbour check and ratio test to seed correspondences. NMS is also employed in seeding. The radius for NMS is $\frac{\theta}{|\alpha|(|\alpha|-1)} \sum_{(i,j)\in\alpha\times\alpha, i\neq j} d_{ij}$, where α is index set for all keypoints, d_{ij} is distance between keypoint i, j, $|.|$ denotes set size, θ is a hyper parameter, which we set as 10^{-2} for all experiment.

For Reseeding Module, we apply sinkhorn algorithm with 10 iterations on correlation matrix to obtain assignment matrix. Correspondences with top-k scores on both dimensions are sampled as seeds. NMS is also applied in reseeding module.

For the inlier likelihood predictor CN in Seeded GNN, we use the $PointCN$ structure illustrated in Fig 2 (down).

We set iterations number of Sinkhorn algorithm to 100.

A.3 Experiment Settings

We use OpenCV implementation for SIFT and official implementation of SuperPoint and ContextDesc. For ContextDesc, we use the latest public model (denoted as *ContextDesc++upright* in the official GitHub repository).

YFCC100M. We extract SIFT and ContextDesc with images in original resolution and resize images so that the longest dimension is 1600 to extract SuperPoint. We use OpenCV *findEssentialMat* and *recoverPose* functions to recover relative poses with the embedded RANSAC, of which the threhold is set as 1 pixel under resolution of resized images. For matching score and precision, we use epipolar distance of 5×10^{-3} to determine inlier matches.

ScanNet. We resize images to [640,480] resolution to extract keypoints and use same protocol to recover relative poses, mathing scores and precision as in YFCC100M.

FM-Bench. The original evaluation pipeline of FM-Bench is based on Matlab and we reimplement it with Python. The parameter for evaluation is consistent with the original implementation. We use OpenCV $findFundamental$ function for fundamental matrix estimation and set the threshold of embedded RANSAC to 1 pixel. Compared with original implementation, our evaluation pipeline tends to give out higher accuracy, especially for wide baseline datasets. We believe it is due to better performance of OpenCV $findFundamental$ function and is beneficial for a more precise evaluation.

Aachen Day-Night. We use the official pipeline and default parameters for evaluation. We extract upright feature for both RootSIFT and ContextDesc.

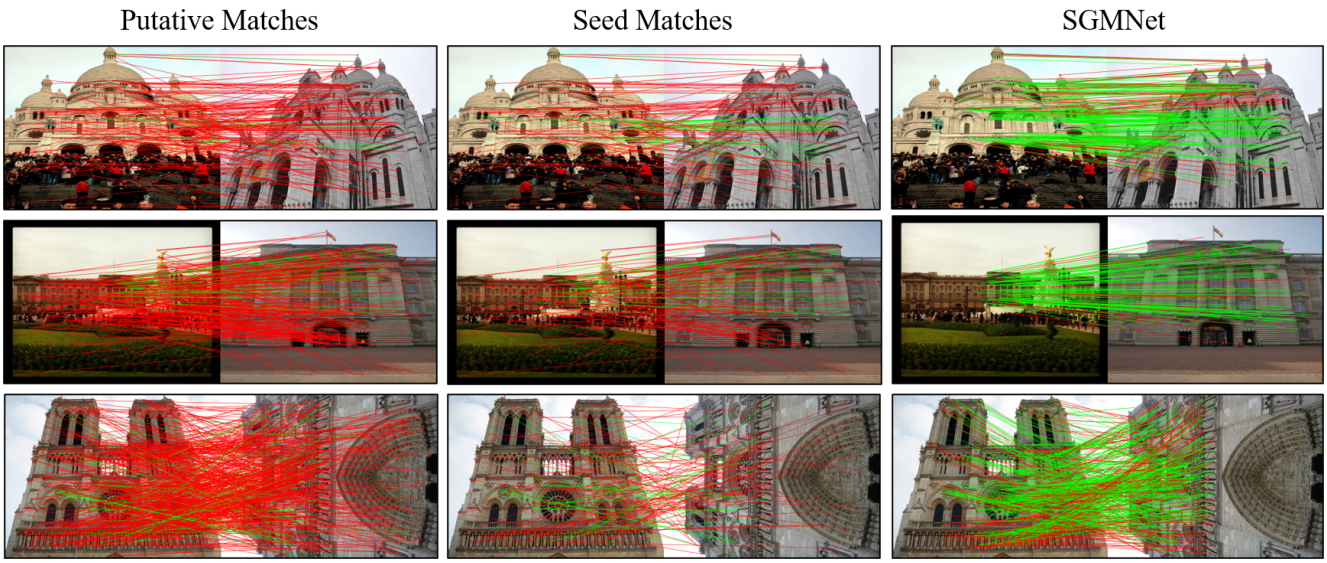

Figure 1. Visualizations of raw putative matches(left), seed correspondences(middle) and matches obtained by SGMNet(right). Note that even with heavily noisy seeds, SGMNet is capable of discovering underlying patterns, which are leveraged to guided message pass across keypoints for robust and accurate matching.

B Designs of SGMNet

Despite the effectiveness of SGMNet, we provide our experiment results and analysis on some other potential designs in this part.

Learned Seeding. After obtaining initial nearest neighbour correspondences, we employ a light-weight permutation invariant Network (architecture illustrated in Fig 2) to determine each correspondence's inlier likelihood score, as is done in previous works [7, 5]. We thus sample correspondences with top-k inlier-score instead of ratio scores as seeding correspondences.

Results. We report results in Table 1. Although applying light-weight pointCN block for inlier seeds prediction slightly increases seeding precision, it's not enough to bring meaningful impacts on the level of pose estimation.

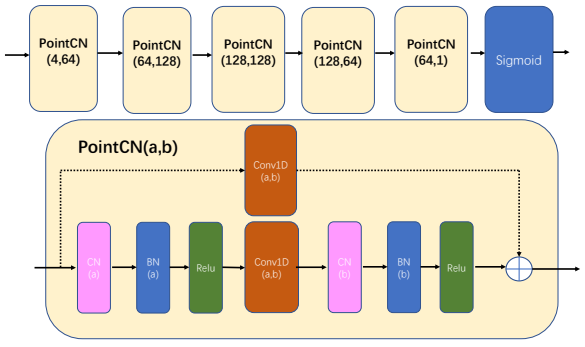

Figure 2. PointCN structure(down) we use to construct learned seeding module(top). $CN(a)$/$BN(a)$ denotes context normalization [5]/batch normalization [1] with input of a channels.

Designs	AUC @5°	AUC @10°	AUC @20°	M.S.	Prec.	Prec.(S)
Learned Seeding	62.80	72.55	81.09	17.25	85.15	51.22
DiffPool	50.85	60.50	69.68	10.18	61.52	-
ISA	52.11	61.44	70.03	10.65	63.24	-
SGMNet	62.72	72.52	81.48	17.08	86.08	39.24
NN+RT	49.07	58.76	68.58	10.05	56.38	-

Table 1. Evaluation Results on YFCC100M using RootSIFT with different designs. **Prec.(S)** denotes precision of seed correspondences.

In general, we are open for the possibility to increase matching quality by introducing more complicated seeding strategies. However, targeting at efficient matching, our seeding method achieves good balance between performance and cost.

The critical component in our method for efficiency message passing is essentially pooling of original keypoints. In this part, we apply two well-studied pooling designs either in GNN or transformer architecture, namely DiffPool [6] and Induced Set Attention [2], to image matching task and evaluate their performance.

DiffPool. As a pooling operation in GNN, DiffPool predicts assignment matrix based on each node's embedding in graph, which is designed to build hierarchical and sparse graph representation [6]. In OANet [7], DiffUnpool, the counter part of DiffPool, is proposed to recover node clusters to original size. As an experiment, we apply DiffPool/Unpool to keypoint graph in each image as a substitution for our proposed attentional pooling. Cross/self attention [3] will be performed on the pooled clusters for mes-

sage exchange.

Induced Set Attention. Induced set attention(ISA) is first proposed in Set Transformer [2]. Different from using seed features as attention bottleneck, ISA adopts a set of learned fixed features(induced point) as attention pass between set elements and is only verified on self-attention for sparse input. We subsitute our seed-based attention with ISA. More specifically, we let the network learned induced points for both sides and let induced points attend to original keypoints(both cross/self) to perform message pass.

Results. We report evaluation result on YFCC100M. For all pooling method, we set pooling number to 128 and extract up to $2k$ keypoints. As illustrated in Table 1, both DiffPool and ISA shows only marginal improvements over baselines, which indicates that applying pooling methods designed for generic GNN/efficient transformer is not necessarily effective for image matching tasks, and further prove that our seed based attentional pooling/unpooling operation is critical for the success of our method.

C Fast Convergence

The compactness of SGMNet not only contributes to the cut-down on computation/memory complexity but also leads to a faster convergence for training. In Fig. 3 we plot the training curve for both SGMNet and SuperGlue. As illustrated, SGMNet takes fewer iterations to reach convergence.

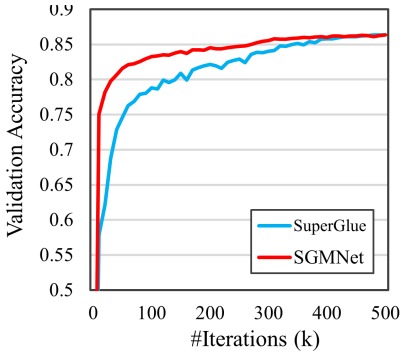

Figure 3. Convergence curve.

D Additional Experiment Results

D.1 Impact of Seeding Number

SGMNet requires seeding a set of seed correspondences, the number of which not only influences our method's efficiency but also accuracy. Therefore, it is important to investigate the impact of seeding number. We carefully conduct grid search on YFCC100M with different keypoint and seeding number.

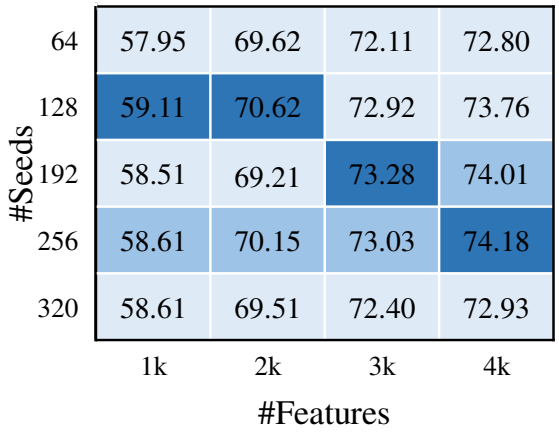

Figure 4. The effect of seed number when varing the keypoint number. Numbers in grids are Exact AUC@$20°$ using RootSIFT.

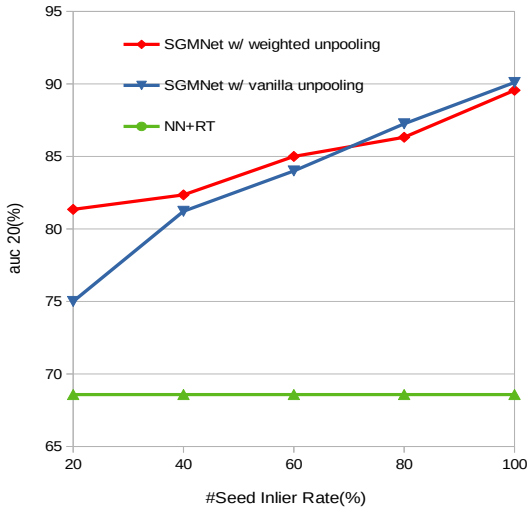

Figure 5. Relationship between pose estimation accuracy and seed precision. Note SGMNet maintain a high matching quality even with seed precision of only 20%.

As is illustrated in Fig 4, an approximate proportional relationship between keypoint/seed numbers yields best performance, as too many seeds may deliver less reliable guidance while seeding too few correspondences results in severe information lost.

D.2 Robustness to Seed Noise

To evaluate the robustness of our method w.r.t. potential false seeds, we conduct experiment on YFCC100M. More specifically, for each pair we select a set of inlier matches, which is determined using ground truth, and pad them with random sampled noise to construct seed correspondences with different precision. We feed the pre-selected seed correspondences to Seeded GNN instead of applying Seeding Module.

As is shown in Fig 5, SGMNet maintains high match-

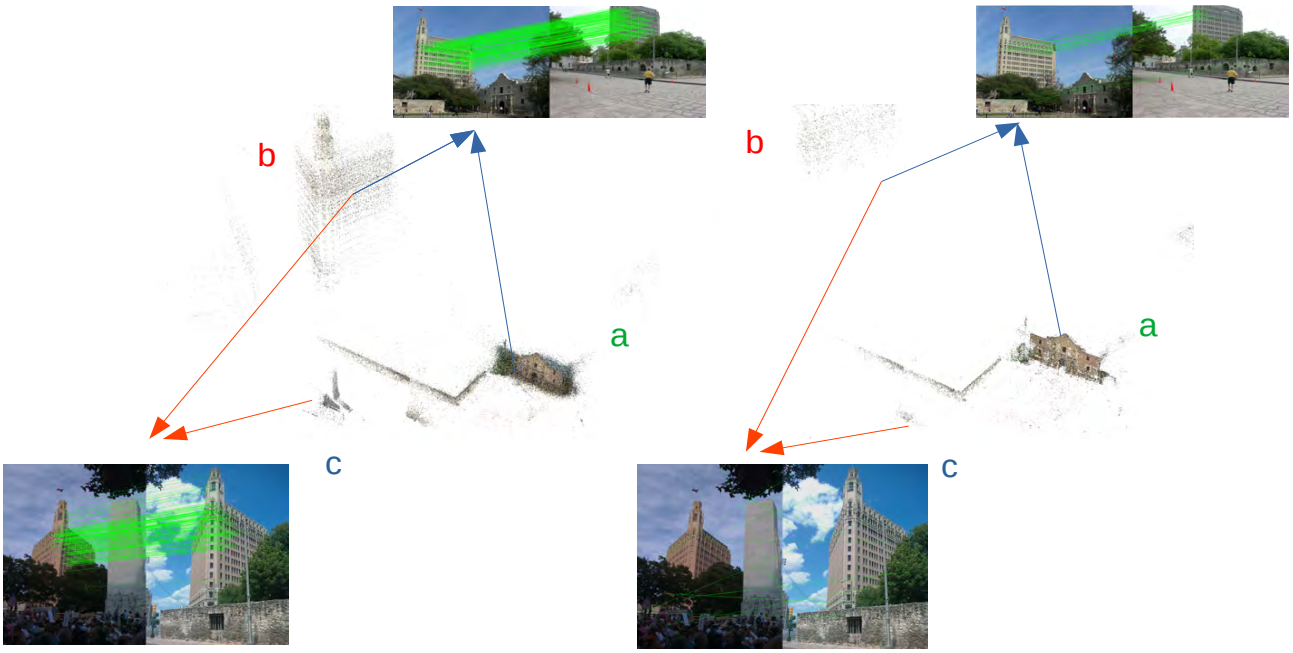

Figure 6. Reconstruction results of vanilla nearest neighbour matching(left) and SGMNet(right). The completeness of reconstruction is determined by matching quality between some critical frames. In this case, NN matching fails to generate descent correspondences between tall building(b)/statue(c) and tall building(b)/remains(a), which results in incomplete reconstruction, while SGMNet registered theses critical frames successfully.

Methods	#Registered Images	#Sparse Points	Mean Rro. Error	Mean Track Len.	Matching Time	Total Time
NN+Ratio+Mutual Check	799	132265	**0.58px**	**11.27**	1h 13min	2h 22min
SuperGlue	916	223950	0.95px	10.93	42h 34min	44h 06min
SGMNet	**943**	**276240**	1.10px	10.73	5h 37min	7h 56min

Table 2. SfM results for Alamo scene in 1DSFM dataset.

ing quality even with heavily noisy seed correspondences. It's noteworthy that for SGMNet without weighted unpooling, the pose estimation accuracy degenerate more rapidly as seed precision decreases, which indicates lower robustness to seed noise. More visualizations related to noisy seed and matching results can be seen in Fig 1.

D.3 SfM Experiment

Typical Structure from Motion(SfM) pipeline usually involves extracting keypoints in large number(e.g. 8k) and matching among hundreds/thousands of images to obtain ultra accurate poses and more complete reconstructions. In this section, we embed different matching methods into COLMAP SfM pipeline for comparison. We reconstruct challenging Alamo scene from 1DSFM [4], which involves 2915 images taken under very different illumination conditions. 8k RootSIFT features are extracted for each image and we set sinkhorn iterations to 10 for both SGMNet and SuperGlue. We use a GTX 1080 GPU to preform matching sequentially. Apart from common statistics for reconstruction, we also report time consumption for matching and the whole SfM pipeline.

As shown in Tab 2, both SuperGlue and SGMNet produces much more complete reconstruction compared with vanilla NN matching and heuristic pruning. However, SuperGlue largely lengthen the time for whole SfM pipeline, while our method retains the matching time to a feasible level.

E More Visualizations

See Fig. 7 and Fig. 1.

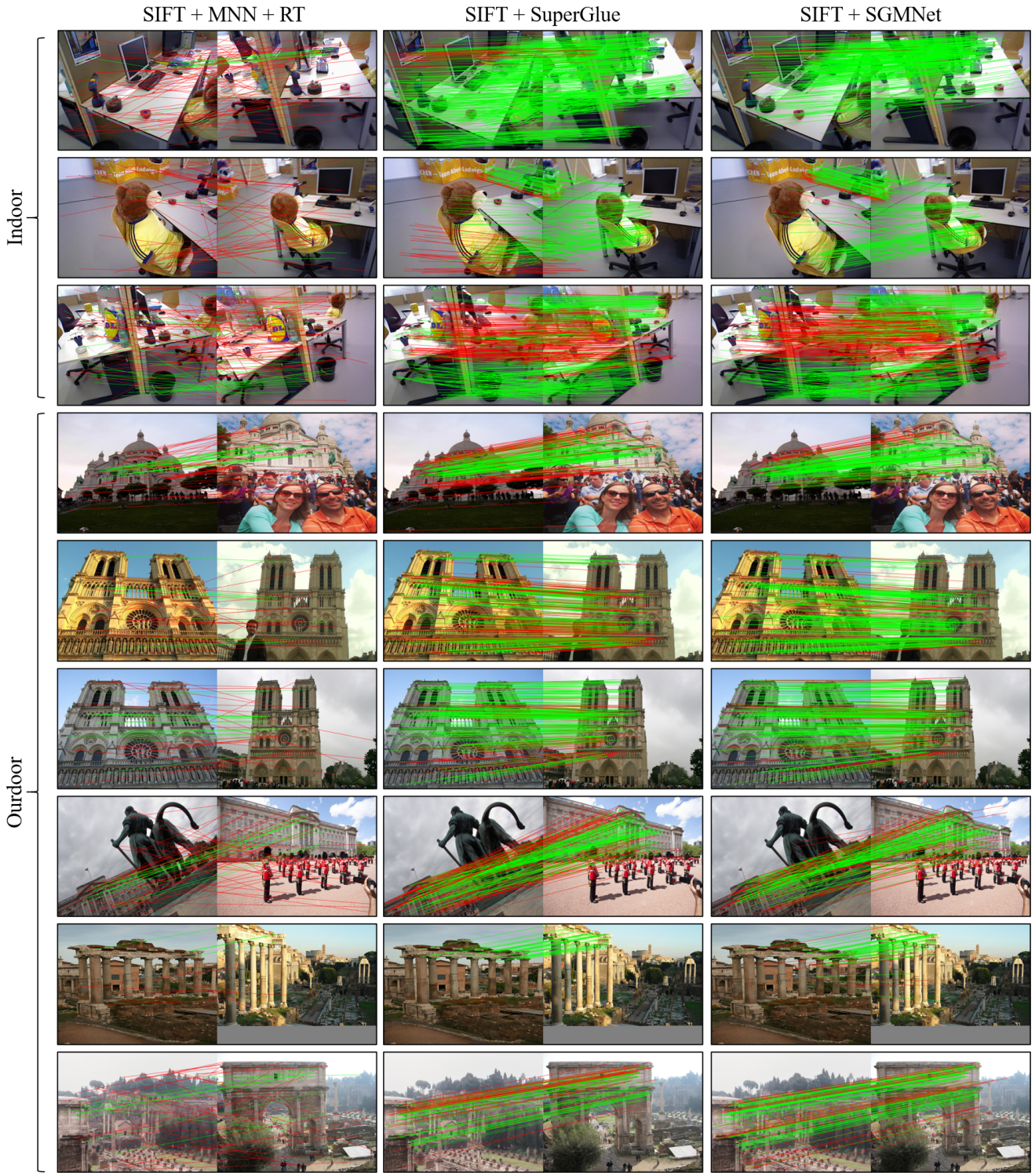

Figure 7. More visualizations.

References

[1] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift., 2015. 2

[2] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A frame-

work for attention-based permutation-invariant neural networks. In *ICML*, 2019. 2, 3

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

[4] Kyle Wilson and Noah Snavely. Robust global translations with 1dsfm. In *ECCV*, 2014. 4

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

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

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