Supplementary Material for Is Geometry Enough for Matching in Visual Localization?

Qunjie Zhou*[1], Sérgio Agostinho*[2], Aljoša Ošep[1], and Laura Leal-Taixé[1]

[1]Technical University of Munich, Germany [2]Universidade de Lisboa, Portugal
[1]{qunjie.zhou,aljosa.osep,leal.taixe}@tum.de
[2]sergio.agostinho@tecnico.ulisboa.pt
https://github.com/dvl-tum/gomatch

In this supplementary material, we provide additional insights and details about certain aspects of the main paper that were not fundamental for its understanding, but that are helpful for interested readers, to fully comprehend and replicate our method. In Section 1, we provide in-depth descriptions of each network component with mathematical definitions as well as architecture diagrams. Next, we present dataset preparation details including dataset processing, query sampling and correspondence ground truth generation in Section 2, followed by the implementation details regarding our training procedure, hyper-parameter choices and an overview of inference process in Section 3. In Section 4, we provide additional details on how to compute the metrics used in the paper, in particular the mean reprojection error Area Under the cumulative Curve (AUC). Finally, in Section 5 we detail how we generate Table 1 in the main paper and provide more insights about the practical challenges of large-scale structure-based localization.

1 Architecture Details

Feature Encoder. We adopt the same encoder architecture as the one presented in [3, 17], which is shown in Fig. 1. The encoder (*left*) consists of a cascade of residual blocks, where each block (*right*) is composed of sequential point-shared fully connected layers, followed by instance normalization and non-linearity (ReLUs). We use two siamese encoders with shared weight parameters to encode bearing vectors of both 2D image keypoints and 3D points, since it has been shown to improve GoMatch's performance (*c.f.* Section 5.3 in the main paper).

Attention. We use a graph neural network to update features from the same modality (*self-attention*) and multi-head attention layers to exchange features across modalities (*cross-attention*). This is similar to PREDATOR [12], although in that work, the attention module is inserted in the bottleneck of the keypoint encoder, *i.e.*, KPConv [25], and thus it is applied to the downsampled keypoints. In our case, the feature encoder does not have a bottleneck and we instead apply

* Equal contribution.

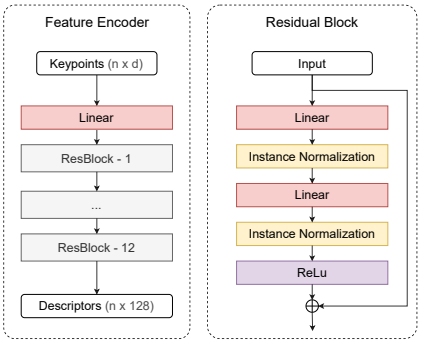

Fig. 1. Feature Encoder (left) & Residual Block (right)

the attention directly to the final output of our encoder. The full attention module contains self → cross → self-attention layers and is applied to both query and database encoded features.

We employ an independent graph neural network [29] to the features coming from each modality. These exchange and refine features of every keypoint type with a fixed number of closest neighbors in coordinate space. Let $\mathbf{f}_i \in \mathbb{R}^d$ represent the feature for keypoint i and $\mathbf{f}_j$ the feature corresponding to one of its neighbors. Let $\mathcal{E}_i$ represent the set of edges formed between keypoint i to its closest K neighbors in coordinate space. We update feature f_i according to

$$\mathbf{f}_i^{(k+1)}\Big|_{k=\{0,1\}} = \max_{j:(i,j)\in\mathcal{E}_i} h_\theta\left(\text{cat}[\mathbf{f}_i^{(k)}, \mathbf{f}_j^{(k)} - \mathbf{f}_i^{(k)}]\right) \tag{1}$$

$$\mathbf{f}_i^{(3)} = h_\theta\left(\text{cat}[\mathbf{f}_i^{(0)}, \mathbf{f}_i^{(1)}, \mathbf{f}_i^{(2)}]\right), \tag{2}$$

where h_θ is a composition of a fully connected layer, instance normalization [27], and a leaky ReLU, and $\mathbf{f}_i^{(0)}$ is the feature produced by the encoder for keypoint i.

To exchange information across modalities we leverage cross-attention [28]. In a cross-attention layer, every keypoint in a modality will interact with keypoints from the other modality through multi-head attention. There are three important concepts that govern this interaction: queries, keys and values. This setup mimics traditional maps or dictionaries, albeit in a continuous form, where every key is paired with a value and these values can be extracted by querying them with the appropriate key. Specifically, given a feature $\mathbf{f}_i$ coming from a keypoint in one modality and features $\mathbf{g}_j$ from an arbitrary keypoint j in the other modality, we form the query, keys and values according to $\mathbf{q}_i = \mathtt{W}_q\mathbf{f}_i$, $\mathbf{k}_j = \mathtt{W}_k\mathbf{g}_j$ and $\mathbf{v}_j = \mathtt{W}_v\mathbf{g}_j$, where $\mathtt{W}_q, \mathtt{W}_k, \mathtt{W}_v \in \mathbb{R}^{d\times d}$ are parameters learned by the network. We

update feature $\mathbf{f}_i$ according to

$$\alpha_{ij} = \text{softmax}(\mathbf{q}_i^\top \mathbf{k}_j / \sqrt{d}) \tag{3}$$

$$\mathbf{m}_i = \sum_j \alpha_{ij} \mathbf{v}_j \tag{4}$$

$$\mathbf{f}_i^{(k+1)} = \mathbf{f}_i^{(k)} + \text{MLP}(\text{cat}[\mathbf{q}_i, \mathbf{m}_i]). \tag{5}$$

Sinkhorn Matching. After getting refined features from the attention module, the Sinkhorn matching stage is responsible for assigning correspondences between query and database keypoints. To that effect, we leverage the Sinkhorn algorithm [4, 23] that has foundations in optimal transport theory, to solve the assignment problem with holistic reasoning. Sinkhorn produces a joint discrete probability distribution of two keypoints being matched. Consider the following features after attention $\mathbf{f}_i^{\text{ATT}}$ and $\mathbf{g}_j^{\text{ATT}}$, corresponding to the i-th query keypoint and the j-th database keypoint. We construct an assignment cost matrix $\mathtt{M} \in \mathbb{R}^{M \times N}$ as

$$m_{ij} = \left\| \frac{\mathbf{f}_i^{\text{ATT}}}{\|\mathbf{f}_i^{\text{ATT}}\|} - \frac{\mathbf{g}_j^{\text{ATT}}}{\|\mathbf{g}_j^{\text{ATT}}\|} \right\|, \tag{6}$$

where m_{ij} is the element in the i-th row and j-th column of $\mathtt{M}$. Sinkhorn provides a solution for the following entropy regularized optimization problem:

$$\mathtt{P}^* = \underset{\mathtt{P} \in \mathcal{U}(\mathbf{r},\mathbf{c})}{\arg\min} \sum_{i=1}^{M} \sum_{j=1}^{N} m_{ij} p_{ij} - \tau p_{ij} \log p_{ij} \tag{7}$$

where $\mathcal{U}(\mathbf{r}, \mathbf{c}) := \{\mathtt{P} \in \mathbb{R}_+^{M \times N}, \mathtt{P}\mathbb{1}_N = \mathbf{c}, \mathtt{P}^\top \mathbb{1}_M = \mathbf{r}\}$, $\mathbf{r} \in R_+^M$, $\mathbf{c} \in R_+^N$, $\sum_i^M r_i = 1$, $\sum_j^N c_j = 1$ and with $\mathbb{1}_M$ denoting a vector of 1s of size M. The vectors $\mathbf{r}$ and $\mathbf{c}$ represent the (marginal) probability vectors of keypoints being matched. In our setup, these marginals are initialized uniformly. The hyperparameter $\tau > 0$ controls the strength of the regularization. This problem is solved iteratively and in a differentiable way through successive steps of row and column-wise normalization, as presented in Cuturi [4].

In a realistic scenario, many of these keypoints will not have a match and in the worst case scenario, it can happen that all points from both sets are unmatched. To handle that, we update the matching cost matrix $\mathtt{M}$ with an extra row and column that act as a "gutter" for unmatched points and denote this new matrix $\tilde{\mathtt{M}} \in \mathbb{R}^{M+1 \times N+1}$. We also add an extra element to both row and column marginals, that is able to "absorb" all all unmatched points in the other set if needed, forming the augmented marginals $\tilde{\mathbf{r}} \in \mathbb{R}_+^{M+1}$ and $\tilde{\mathbf{c}} \in \mathbb{R}_+^{N+1}$. We

present the updated cost matrix and marginals below

$$\tilde{\mathbf{r}} = \begin{bmatrix} \frac{1}{M+N} \\ \vdots \\ \frac{1}{M+N} \\ \frac{N}{M+N} \end{bmatrix} \begin{bmatrix} m_{11} & \dots & m_{1N} & m_u \\ \vdots & \ddots & \vdots & \vdots \\ m_{M1} & \dots & m_{MN} & m_u \\ m_u & \dots & m_u & m_u \end{bmatrix} = \tilde{\mathtt{M}} \tag{8}$$

$$\begin{bmatrix} \frac{1}{M+N} & \cdots & \frac{1}{M+N} & \frac{M}{M+N} \end{bmatrix} = \tilde{\mathbf{c}}, \tag{9}$$

where m_u is a parameter learned by the network that represents the cost of considering points as unmatched. With the cost and marginals as input, Sinkhorn generates the final discrete probability distribution $\tilde{\mathtt{P}} \in \mathbb{R}_{+}^{M+1 \times N+1}$, representing soft correspondences. The pose estimation pipeline requires hard correspondences so we retain only pairs of keypoints that mutually assign to each other.

Outlier Rejection. After Sinkhorn matching, the estimated corresponding pairs may still contain outlier matches. We follow [17] to cast the outlier rejection as a binary classification task, where we use a classifier to predict a confidence score to identify whether a match is an inlier or outlier w.r.t. a specific confidence threshold. As depicted in Fig. 2, given a predicted match referring to a query keypoint $Query\_P_j$ and a database keypoint DB_P_i, we first collect keypoint feature (extracted in the previous stage by the encoder and attention module). We then concatenate the features of its involved keypoints to form a single feature representation of that match and feed it into a classification network. The classifier follows the overall architecture of keypoint encoder described in Fig. 1 and has a final classification layer, *i.e.*, a linear layer and a sigmoid operation, to output a probability score for an input match which is then used to filter a less confident match.

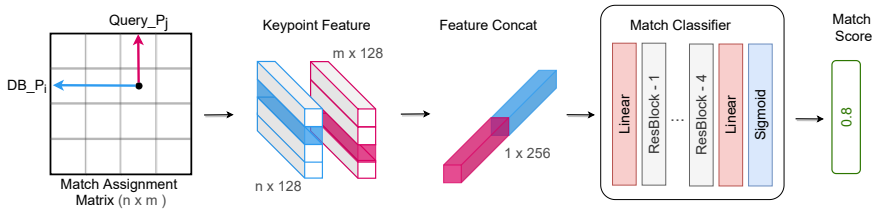

Fig. 2. Outlier rejection through classification. For a match identified by Sinkhorn, we collect the matched keypoint features that are extracted by the encoder and attention module. The feature for a query keypoint and a database keypoint is concatenated into a single feature representing the match. The match classifier then predicts the match confidence (as a probability score) which can be used later to filter a less confident match.

2 Dataset Details

MegaDepth [15]. Our networks are trained on MegaDepth. We first followed the preprocessing steps from [8] to generate the undistorted reconstructions. We adopted the test set published by MegaDepth's authors, composed of 53 sequences. We further split the remaining sequences into training and validation splits, where the training and validation splits ensure a similar distribution in terms of sequence sizes between both. Within each data split, we first sample up to 500 queries per scene. For each query, we collect its k co-visible views that have at least 35% of visual overlapping, where we drop queries with not enough co-visible views, *i.e.*, $k < 3$. The visual overlapping is computed by the number of commonly seen 3D points divided by the total seen 3D points in the query view. Notice, in practice, those co-visible views are supposed to be obtained by image retrieval techniques [1, 19, 26] (as we do for evaluating on the other two benchmark datasets). However, for training and ablation study, we use ground-truth information provided by the dataset to guarantee the co-visibility, which enables stable training and allows us to focus on analysing the geometric-based matching performance with proper retrieval quality.

In total, our training set contains 18881 queries covering 99 scenes, our validation set contains 3146 queries covering 16 scenes and our testing set contains samples 7344 covering 49 scenes.

Cambridge Landmarks [13]. Cambridge Landmarks is an outdoor urban localization dataset, where each frame is annotated with a camera pose label. The video footage was captured through a smartphone camera and includes significant urban clutter from pedestrian and vehicles. The dataset is composed of five scenes and each scene is composed of multiple sequences. We report results in four of these scenes – King's College, Old Hospital, Shop Facade and St. Mary's Church – amounting to a total of 29 sequences, that in our case were all used for testing. To obtain 3D points for our method, we use the publicly available reconstruction generated per-scene with SuperPoint [6] from the training images and ground truth poses. This reconstruction was originally made available by Torsten Sattler and has been used in the recent work PixLoc [21] for localization evaluation. In addition, we also follow PixLoc to use their released top-10 query-retrieval pairs computed by NetVLAD [1]. Both reconstruction and retrieval pairs are hosted here by the authors of PixLoc.

7-Scenes [22]. 7-Scenes is a pose annotated indoor dataset of seven different scenes, captured with an RGB-D camera. Each scene is composed of multiple sequences and every frame in each sequence comes with a color image, a depth image, and the camera pose. The dataset has a total of 46 sequences and we used 18 for testing, following the original test split. We use the top-10 query-retrieval pairs computed by DenseVLAD [26] which were made available here by the authors of PixLoc [21] and used in their experiments. To obtain reference 3D points from database retrievals, we first use a keypoint detector on the color image to generate an original set of candidate 2D keypoints. We then transform their coordinates from the color image space to depth image space and only retain keypoints that have a valid depth measurement, rounding fractional coordinates

to the nearest pixel location. Given a 2D image location and the corresponding depth value, we compute a 3D keypoint in camera space and then transform it to scene's frame of reference. For simplicity and enabled by the fact that we only make use of a sparse subset of points in the original depth image, we don't try to establish 3D point correspondences between different co-visible frames and consider that each depth image observes a set of unique 3D points in each frame. This process allows us to flexibly generate 3D points using different types of keypoint. As shown in Section 5.6, we generated two versions of 3D points that are obtained using SIFT [16] and SuperPoint [5] keypoints to demonstrate the generalization capability of GoMatch.

Keypoint Detection. To be consistent with the 3D models in MegaDepth that are reconstructed with SIFT [16] keypoints, we also use a SIFT detector to extract keypoints from query images. We limit all detectors to extract up to 1024 keypoints per image, this is to fit our model into a 12GB GPU during testing. On 7-Scenes, we are able to apply any keypoint detectors, since we have the depth map to extract the corresponding 3D coordinate. In our experiments, we used a hand-crafted detector – SIFT – and a state-of-the-art learning-based detector – SuperPoint [6] – to extract keypoints. For Cambridge Landmarks, we also used SuperPoint, ensuring consistency with the reconstructed 3D model. All of the keypoints are pre-computed for both datasets and cached locally.

Ground Truth Correspondences. To train our network with the matching loss and to compute reprojection-based AUC metric (*c.f.* Section 4), we need to generate ground truth (gt) correspondence labels between query keypoints and 3D point cloud keypoints (database keypoints). Given a set of database keypoints and a query image with its known camera pose, we project the 3D points into the query image to obtain its 2D projections. Then we perform mutual nearest neighbour search based on the L2 distance between the query keypoints and the projected 3D keypoints and consider them as a gt correspondence if the distance is below a threshold of 0.001 in normalized image coordinates, i.e., distance between bearing vectors.

3 Implementation Details

Architecture. We used 12 residual blocks for keypoint encoders (*c.f.* Fig. 1) and 4 residual blocks for match classifier for outlier rejection (*c.f.* Fig. 2). The features produced by the keypoint encoders and attention module have dimension 128. The graph neural network used for *self attention* establishes a KNN graph with the closest 10 neighbors, in coordinate space. We use 4 parallel heads for multi-head *cross attention*. In total, GoMatch has approximately 1.3M weight parameters.

Training. We train all models using the ADAM [14] optimizer at the learning rate of 0.001. The batch sizes are chosen to allow each model to be trained on a single 48GB NVIDIA Quadro RTX 8000 GPU and vary from 16 to 64 batches (depending on the model size), *e.g.*, we use batch size 16 for GoMatch. We train each model for 50 epochs and determine the best checkpoint based on the

lowest loss value on the validation set. GoMatch requires approximately 20 hours training time for 50 epochs.

Outlier Rate Control. The keypoints involved in gt matches are the inlier keypoints and the remaining keypoints are considered outliers. For training stability, we constrain the keypoint outlier rate to 0.5. Besides our ablation studies shown in Figure 4 (in the main paper), no keypoint outlier control is applied during testing.

Keypoint Number Control. By default, we limit query and database keypoints to a maximum number of 1024, to ensure the testing can be performed on a 12GB GPU. For query keypoints, we enforce the detector to extract at most 1024 keypoints. For database keypoints, we use up to its first 1024 keypoints if exceeding that number. During training, we ignore training samples where the number of query keypoints or database keypoints is less than 100. During inference, we consider a sample as failed if the number of query keypoints or database keypoints is less than 10.

Inference. Given a query image and a 3D point cloud, we first extract 2D keypoints from the query using detectors such as SIFT [16] or SuperPoint [6] (in practice, we load pre-computed keypoints). Next, we identify k retrieval/co-visible reference views to compute 3D points that are visible to query. Correspondences are established between the query keypoints and co-visible 3D points (in bearing vectors) by GoMatch. We run the Sinkhorn algorithm for a maximum of 20 iterations to obtain an initial set of match estimates. Then we remove all matches with classification scores below a threshold 0.5 to filter uncertain matches and finally estimate camera pose of the query.

Pose Estimation. We use the OpenCV [2] library to estimate pose from a set of query to point cloud keypoint correspondences. We first identify an initial set of inlier matches using a minimal P3P [11] solver paired with RANSAC [10] and then estimate the final pose through a Levenberg-Marquardt optimization step acting only on the inliers. We allow RANSAC to perform 1000 iterations with the admissible maximum inlier error threshold for the bearing vectors set to 1e-3.

4 Reprojection Error AUC

For MegaDepth, we reported the mean reprojection error area under the cummulative curve (AUC) with thresholds at 1, 5 and 10 pixels. This metric was inspired by the pose error area under the cummulative curve used in [20, 21, 24]. However, in our case, as the scale unit of MegaDepth is undetermined and might be inconsistent across scenes, we have to compute the AUC based on reprojection error instead of pose errors to ensure consistency. To compute our proposed AUC metric, for each query, we project the inlier 3D points, *i.e.*, the 3D points involved in the ground-truth (gt) correspondences, onto the query image using the gt pose and the estimated pose. The mean reprojection errors are computed as the mean pixel distances between the gt projections and estimated projections of the 3D points. Then we compute the area under the cummulative recall

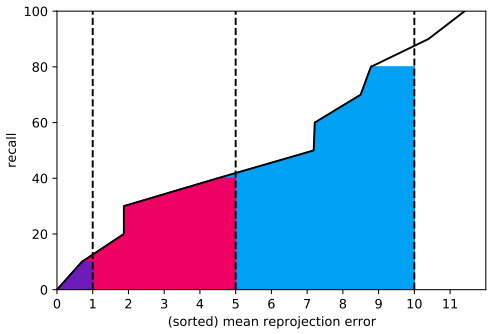

Fig. 3. A hypothetical example of how the mean reprojection error AUC is computed for the thresholds at 1/5/10 pixels. After sorting the mean reprojection error and establishing a bijective relation with the recall, one computes the area under the curve using the trapezoidal rule of integration. Despite not visible, the AUC in cyan for the 10 pixel threshold spans from 0 to 10, and similar to the pink area for the 5 pixels threshold.

curve up to a specific pixel error threshold as illustrated in Fig. 3. Finally, we normalize the area by the error threshold to keep the metric score $\in [0, 100]$.

As described in Section 2, the gt correspondences are computed with a 0.001 normalized threshold, which means the bearing vectors obtained by projecting the inlier 3D points using the gt camera pose have up to a 0.001 distance to the bearing vectors of our labelled gt 2D keypoints. This tolerance is reflected in pixel space, resulting in relatively low AUC scores at 1 pixel threshold (as shown in the main paper Table 1) even using our Oracle matcher. However, this does not affect the function of the AUC metric that is to reveal the performance gap between different methods. We showed that the AUC metric yields similar conclusion as the pose error quantile metric used by BPnPNet [3].

5 Practical Challenges in Large-scale Localization

As introduced in the main paper, the primary motivation of our work is to present a new localization framework that does not suffer from: (i) storage requirements, (ii) descriptor maintenance effort [7], and (iii) privacy concerns [18, 9]. These are challenges that current structure-based localization methods face, especially when it comes to scaling-up for city-level scenes. A considerable amount of literature focuses on making localization methods more accurate, however, the aforementioned practical challenges are much less explored. To study the storage requirements, we perform a detailed analysis on MegaDepth, a dataset that resembles a city-level large-scale environment. While MegaDepth is a collection of landmarks instead of a real city-scale scene, we argue that a city-scale scene consists of a collection of districts (similar to landmarks). As shown in Table 1 in

the main paper, the minimal scene data, *i.e.*, scene coordinates (3D) and camera metadata (Cameras), is always required by a structured-based localization to obtain 2D-3D correspondences, which will take 15.73MB and 3.44GB storage in total for all Megadepth scenes. This is the only data that geometric-based matching (GM) methods need to store. This is in contrast to visual-based matching (VM), as they need to store extra visual descriptors that consume 130/1040GB for SIFT/SuperPoint descriptors. Alternatively, one can extract descriptors on-the-fly from the retrieved raw images, which requires saving all of the raw images with an extra storage of 157GB and leading to more computational burden. **Then**, after addressing the storage issue, one still needs to consider the privacy vulnerability raised during descriptor transmission, since large-scale localization with a 3D map has to be realistically deployed in a server-client mode, where the server stores 3D scene data required to perform localization. This is another motivation driving us towards using geometric information only. **Finally**, as thoroughly covered in [7], with the continuous advance of local features, upgrading descriptors is inevitable in the long-term, and it involves either re-building the map or transforming the descriptors. In contrast, for geometric-based matching, upgrading our localization algorithm does not need an update on the map side unless the scene has changed, a case in which every map would need to be updated.

References

1. Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
2. G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.
3. Dylan Campbell, Liu Liu, and Stephen Gould. Solving the blind perspective-n-point problem end-to-end with robust differentiable geometric optimization. In *European Conference on Computer Vision (ECCV)*, pages 244–261. Springer, 2020.
4. Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.
5. Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
6. Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPR Workshops*, pages 224–236, 2018.
7. Mihai Dusmanu, Ondrej Miksik, Johannes L Schonberger, and Marc Pollefeys. Cross-descriptor visual localization and mapping. In *IEEE International Conference on Computer Vision (ICCV)*, pages 6058–6067, 2021.
8. Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-Net: A Trainable CNN for Joint Detection and Description of Local Features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
9. Mihai Dusmanu, Johannes L Schönberger, Sudipta N Sinha, and Marc Pollefeys. Privacy-preserving image features via adversarial affine subspace embeddings. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
10. Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *CACM*, 24(6):381–395, 1981.
11. Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):930–943, 2003.
12. Shengyu Huang, Zan Gojcic, Mikhail Usvyatsov, Andreas Wieser, and Konrad Schindler. Predator: Registration of 3d point clouds with low overlap. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4267–4276, June 2021.
13. Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.
14. Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.
15. Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
16. David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 2004.
17. Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2666–2674, 2018.

18. Francesco Pittaluga, Sanjeev J Koppal, Sing Bing Kang, and Sudipta N Sinha. Revealing scenes by inverting structure from motion reconstructions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 145–154, 2019.
19. Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation. *TPAM*, 41(7):1655–1668, 2018.
20. Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4938–4947, 2020.
21. Paul-Edouard Sarlin, Ajaykumar Unagar, Mans Larsson, Hugo Germain, Carl Toft, Viktor Larsson, Marc Pollefeys, Vincent Lepetit, Lars Hammarstrand, Fredrik Kahl, et al. Back to the feature: Learning robust camera localization from pixels to pose. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3247–3257, 2021.
22. Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2930–2937, 2013.
23. Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.
24. Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8922–8931, 2021.
25. Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, Francois Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.
26. Akihiko Torii, Relja Arandjelovic, Josef Sivic, Masatoshi Okutomi, and Tomas Pajdla. 24/7 place recognition by view synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1808–1817, 2015.
27. Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization, 2017.
28. 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 I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
29. Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph.*, 38(5), Oct. 2019.