

Supplementary Material

DH3D: Deep Hierarchical 3D Descriptors for Robust Large-Scale 6DoF Relocalization

Juan Du^{1*}, Rui Wang^{1,2*}, and Daniel Cremers^{1,2}

¹ Technical University of Munich

² Artisense

{duj, wangr, cremers}@in.tum.de

1 Overview

Despite the significant progresses in deep learning, one issue we are commonly facing is that reproducibility is not guaranteed by all the papers. In this respect, in addition to releasing our code, we provide in this supplementary document the remaining technical aspects and some insights we have gathered during the development. We hope this can help the other researchers in this field.

In the first part, we show more detailed results of our keypoint repeatability on the ETH dataset and on the point clouds generated by Stereo DSO on Oxford RobotCar (referred to as StereoDSO-Oxford), which we omitted in the main paper due to space limitation. We also provide more qualitative registration results on Oxford RobotCar and StereoDSO-Oxford, including some failure cases. The second part provides results revealing the influence of using different operators for global aggregation, where we see that although there exist several ways, attention based NetVLAD layer is so far the best choice. Some qualitative results on global retrieval are attached afterwards. The third part is dedicated to presenting all the technical details left over from the main paper, including (1) structures of sub-networks, (2) training data preparation, (3) two-phase training of the local and global networks, (4) generation of the Stereo DSO point clouds, and (5) details on the weak supervision used in the ablation study in the main paper.

2 Additional Results on Local Feature

2.1 Keypoint Repeatability

We use the same method as in the main paper to evaluate keypoint repeatability on ETH and StereoDSO-Oxford, except that now we choose 0.3m as the threshold distance for the ETH dataset to determine whether a point is repeatable. For all the detectors, 512 and 1024 keypoints are respectively extracted for StereoDSO-Oxford and ETH. The results are shown in Fig. 1.

* Authors contributed equally.

On StereoDSO-Oxford, the results hold a similar pattern as those on the Oxford RobotCar LiDAR points shown in the main paper. Our detector outperforms other methods except USIP which is a pure detector that dedicatedly designed for feature repeatability. All the handcrafted detectors and 3DFeatNet have repeatability lower than 0.2. On the ETH dataset, our detector has the highest repeatability while the performance of USIP degrades significantly. When USIP is trained, a relatively fixed receptive field is defined for extracting each feature location (by pre-setting the parameter M and K , please refer to [4] for details). In other words, the performance of the learned network is highly correlated to certain point densities and scales in the receptive field. Therefore, when applied to point clouds with very different spatial distribution as the training data, the network suffers to generalize well.

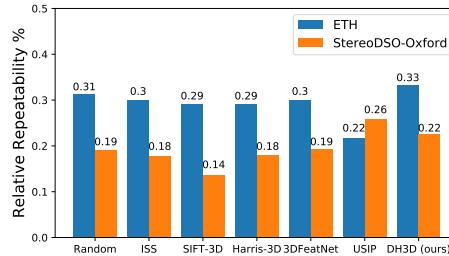


Fig. 1: Relative repeatability on ETH dataset and StereoDSO-Oxford testing set.

2.2 Qualitative Results on Oxford RobotCar

We show additional qualitative registration results by our method in Fig. 2, including two failure cases due to the domination of local features from the facade of buildings which are repeated and contain limited local textures.

2.3 Qualitative Results on Stereo DSO Points

Fig. 3 shows more registration results of StereoDSO-Oxford, which covers different conditions including night, dusk, direct sun, snow, rain and roadworks. Compared to LiDAR scanner, due to the nature of tending to reconstructed 3D points corresponding to pixels with high image gradients, visual SLAM methods usually generate point clouds with evidently different spatial densities and distributions under different lighting conditions. Such variations become even significant when further combined with different scene layouts. This can be clearly observed in the figure. Still, our keypoint detector and local descriptors trained on LiDAR points are able to achieve fairly good matchings without fine-tuning.

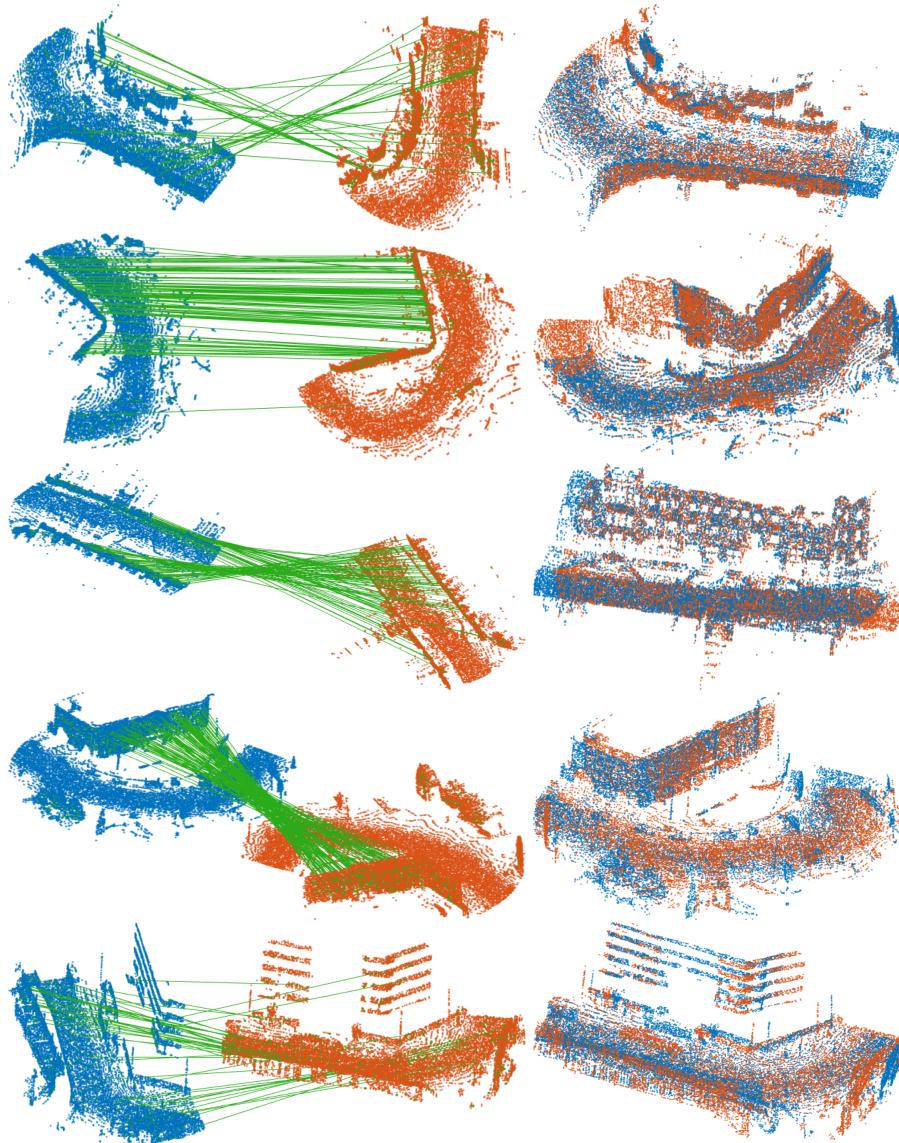


Fig. 2: More results of registration of Oxford RobotCar. The first column shows the matched features (after RANSAC) and the second shows the point clouds after alignment. The last two rows show failure cases caused by the dominated repetitive vertical structures.

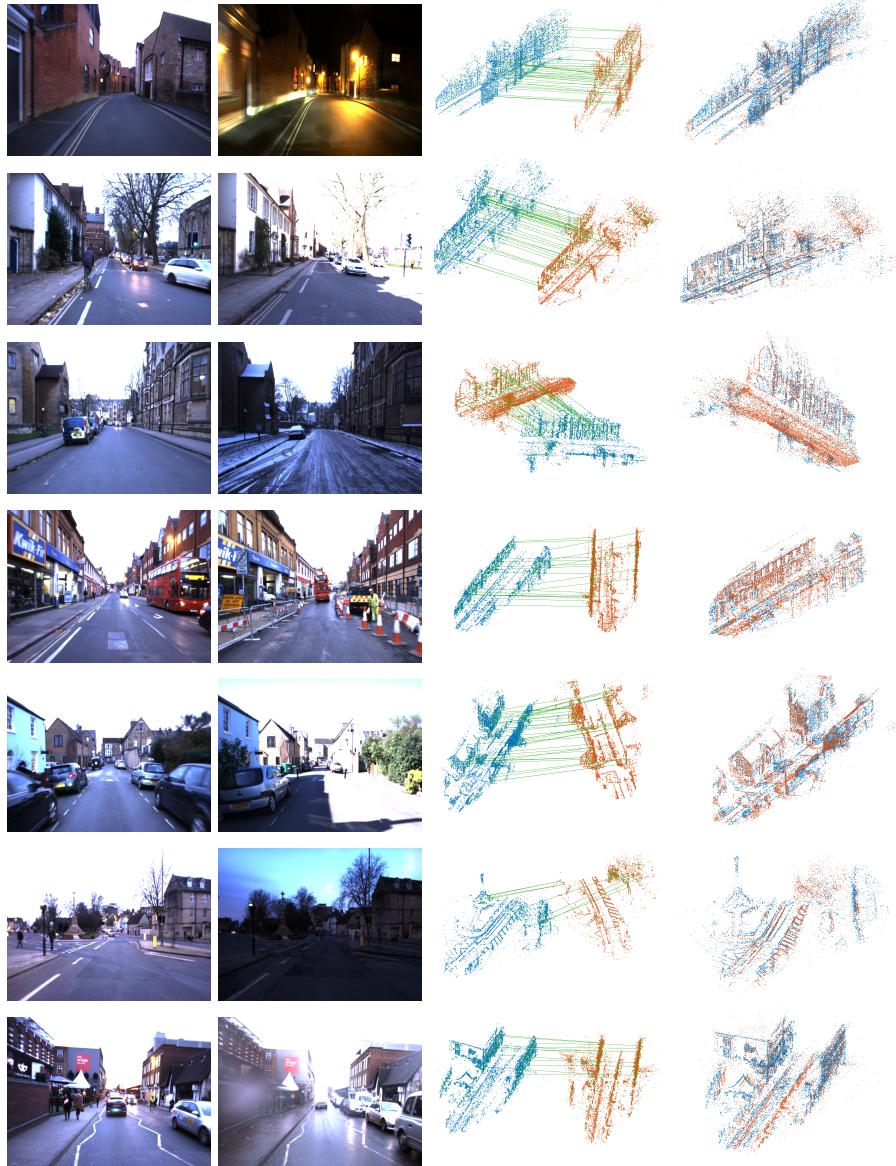


Fig. 3: More results of registration of StereoDSO-Oxford. The first two columns display frames from the reference and the query sequences. The last two columns show the matched features (after RANSAC) and the point clouds after alignment. Note that under different lighting and weather conditions, the point clouds generated by Stereo DSO have very different spatial densities and distributions.

3 Additional Results on Global Descriptor

3.1 Comparison of Different Global Aggregators

As mentioned in the main paper, there exist many ways to aggregate the extracted local descriptors into a global one. Apart from the attention based NetVLAD model, which is adopted as our global assembler, we tested four other models, i.e. PointNet++ [7], Dynamic Graph CNN (DGCNN) [9], PointCNN [5], and FlexConv [2]. Max- and average-pooling (Max-Pool, Avg-Pool) are tested additionally as two baselines. We first explain the details on each model and discuss the results presented in Fig. 4 afterwards.

PoinNet++: To construct a PoinNet++ style global assembler, we use a combination of Multi-Scale Grouping (MSG) layer and a Set Abstraction(SA) layer proposed in [7]. The architecture is as follows: SA(1024, [4.0, 8.0], [16, 32], [[128, 128, 256], [128, 128, 256]]) → SA([256, 512, 1024]) → FC(512, 0.5) → FC(256, 0.5), where $\text{SA}(K, r, [l_1, \dots, l_d])$ is a SA level with K local regions of ball radius r and use a PointNet of d fully connected layers with width $l_i (i = 1, \dots, d)$. Note that all the models discussed in this section include a final fully connected layer to transform the global descriptor to a fixed dimension of 256.

DGCNN: We use two consecutive EdgeConv blocks (MLP(128, 128) and MLP(256, 256)) to process the local descriptors, where $\text{MLP}(a_1, \dots, a_n)$ is a multi-layer perceptron with the number of layer neurons defined as (a_1, a_2, \dots, a_n) . The output of these two layers are concated which is followed by another MLP(1024) layer to generate a tensor of shape $N \times 1024$. Then, a global max pooling is used to get the global descriptor.

PointCNN: Inspired by PointCNN, we also test the idea of incorporating χ -convolution to extract a global descriptor. A χ -conv(N, C, K, D) layer takes N points as input and outputs a tensor of shape $N \times C$. It is defined in a local region which constructed by sampling K input points from $K \times D$ neighboring points, where D is the dilation rate. In our settings, we add four χ -conv layers to transform the local features as follows: $\chi\text{-conv}(2048, 256, 16, 2) \rightarrow \chi\text{-conv}(768, 512, 16, 2) \rightarrow \chi\text{-conv}(384, 512, 16, 2) \rightarrow \chi\text{-conv}(128, 1024, 16, 2)$.

FlexConv: We add four additional FlexConv layers as follows, FlexConv(256, 8, 512) → FlexConv(256, 8, 128) → FlexConv(512, 8, 32) → FlexConv(1024, 8, 32), where $\text{FlexConv}(D, k, N)$ denotes a Flex-Convolution operation on N input points with the neighborhood size as k and D is the dimension of the output. Then, we apply a Flex-Convolution at the center point $(0, 0, 0)$ with all the remaining points as neighbors to generate a global descriptor.

Pooling: The Max-Pool and Avg-Pool take the extracted local feature map as input and perform a simple max/avg pooling to produce a single global embedding.

Fig.4 shows the average recall curves within the top 25 retrievals on the Oxford RobotCar dataset. We draw the overall conclusion from these results that the

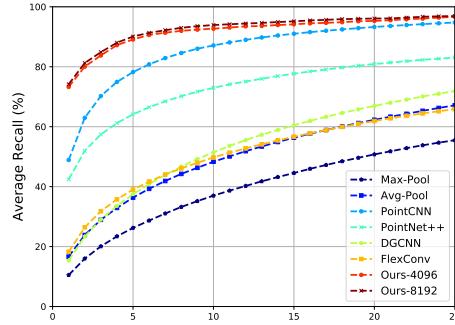


Fig. 4: Average recall of the top 25 retrievals of different approaches for assembling global descriptors on the Oxford RobotCar dataset.

attention based NetVLAD global aggregation is so far the best option among the multiple tested models. This validates our choice on the global aggregator in the main paper. In addition, we can observe that among all the baseline methods, PointCNN far outperforms others. One possible explanation is that PointCNN estimates a transformation on the input points which sufficiently exploits the 3D position information compared to other methods. This can also account for the second highest recall obtained by PointNet++ which takes advantage of the 3D position implicitly by using an MSG layer. FlexConv, DGCNN and Avg-Pool produce similar results while Max-Pool gives the worst performance, most likely because a simple max pooling causes a large amount of loss of information.

3.2 Quanlitative Results on Point Cloud Retrieval

In addition to the quantitative results of our global descriptors (Ours-8192) on point cloud retrieval reported in Sec. 4.3 of the main paper, we provide some qualitative examples in Fig. 5. To this end, we first randomly select a full traversal (Sequence 2014-12-10-18-10-50) as the reference map. Then three query point clouds from three other randomly selected traversals in the remaining 32 sequences are chosen, each representing one sample submap from individual testing areas as shown in the right sub-figure of Fig. 8. For each example, we display the query point cloud and the top5 retrieved results. In addition, the location of each point cloud is plotted in the reference map on the right. We use different colors to indicate the L2-distances between the global descriptors of the query point clouds and those of the submaps contained in the reference map. For each query, the best match (represented by the blue square) correctly overlaps with the target location (indicated by the red circle), which demonstrates that our global descriptors have good robustness to rotation and occlusion. It is also worth noting that most of the un-retrieved submaps in the reference map have relatively high distances in the descriptor field, demonstrating the good discriminativeness of our global descriptor.

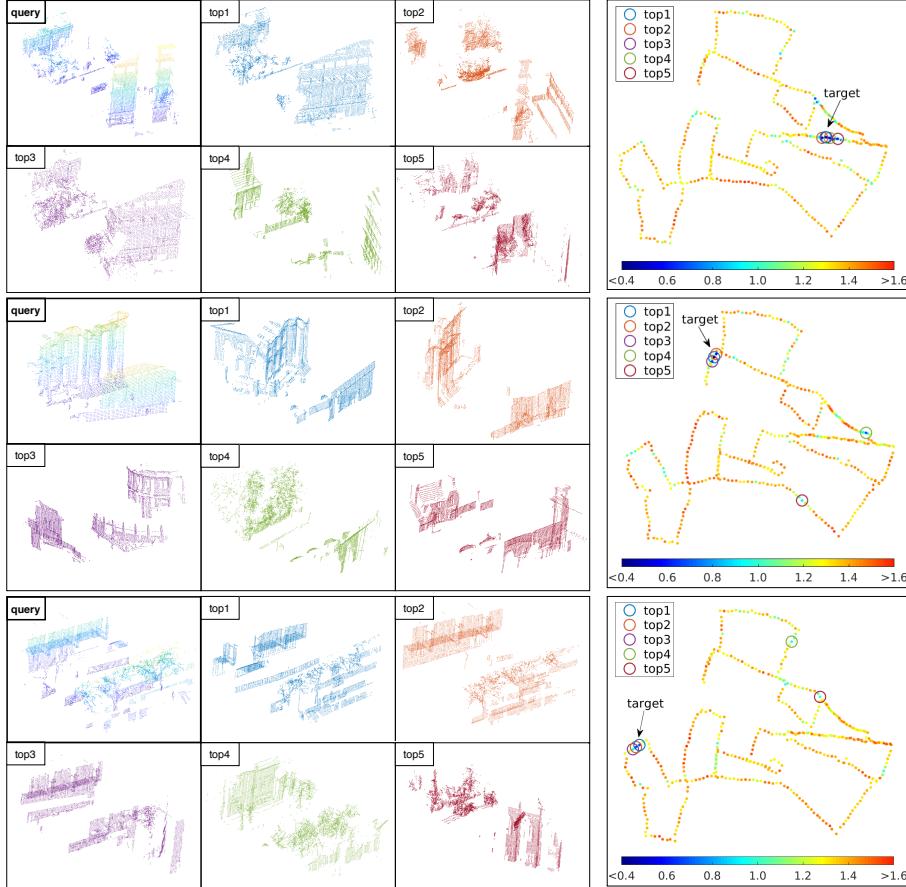


Fig. 5: Visualizations of example retrieval results of our network on the Oxford Robot-Car dataset. For each retrieval, we display the query point cloud and the top5 matches returned by our method. We also indicate the 3D locations of these point clouds in the associated reference map. Colors in each query point cloud indicate different heights above ground while colors of the retrieved point clouds correspond to the markers drawn in the reference maps. The L2-distances between the global descriptor of the query point cloud and those of all the submaps are color-coded.

4 Technical Details

4.1 Additional Network Details

The architectures of our local feature encoder and global descriptor assembler are illustrated in the main paper. Here we provide the structural details of the other two sub-networks, i.e., the keypoint detector and the attention map predictor.

3D Keypoint detector. Fig. 6 depicts the architecture of our 3D keypoint detector. As explained in the main paper, it consists of 4 1×1 convolution layers and a sigmoid activation function at the end. All the convolution layers are followed by ReLU non-linear activation and BN (BatchNorm). The 1×1 convolutions essentially transform each point independently and are invariant to point ordering. The sigmoid activation function is used to restrict the individual output value to $[0, 1]$, which represents the matching reliability of each local descriptor.

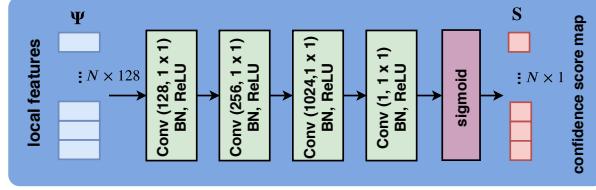


Fig. 6: The architecture of the local feature detector in Fig. 2 of the main paper.

Attention map predictor. We use a sub-network which has similar structure as the detector to predict attentive weight for each local descriptor. The main component is composed of 3 1×1 convolution layers with ReLU activation and BN. A softmax operation is applied to produce the final attention distribution that sums up to 1. Recall that in our case, the inputs to the attention predictor are the learned local descriptors, which have already encoded multi-level contextual information. Thus our attention predictor can benefit from this and have a relatively simple yet effective model structure.

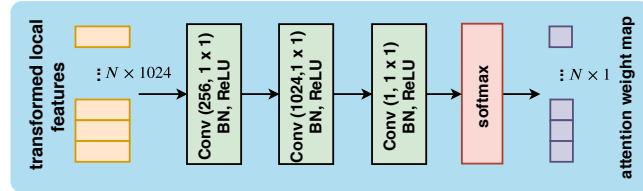


Fig. 7: The architecture of the attention predictor in Fig. 3 of the main paper.

4.2 Training Data Preparation

Data splitting. We use the LiDAR points from the Oxford RobotCar dataset [6] to learn both local and global descriptors. For fair comparisons with other methods, we split the dataset differently for these two tasks when constructing the training and testing data. More specifically, for the keypoint detection and local descriptor extraction, we follow the practice of 3DFeatNet [3] and make use of 40 full traversals with each traversal split into two disjoint sets for training and testing. The training set is constructed by the point clouds accumulated from the training region of the first 35 sequences. The point clouds collected from the testing region of the remaining 5 sequences are used to generate the testing set. For global descriptor learning, we adopt the split used in PointNetVLAD [1], where a set of 44 traversals are used, both full and partial. Each run is then geographically split into 70% and 30% for training and testing, respectively. A comparison of the above two data splittings is illustrated in Fig. 8.

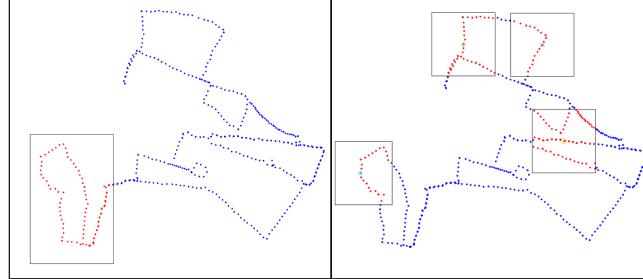


Fig. 8: Different data splitting strategies for learning local descriptors and keypoint detector (left) and global descriptors (right). Blue points represent submaps in the training set and red points represent those in the testing set.

Data preprocessing. For each traversal, we create a 3D point cloud submap with a 20m radius for every 10m interval whenever good GPS/INS poses are available. According to the widely adopted convention, the ground planes are removed in all submaps since they are repetitive structures and non-informative. The resulting submaps are then downsampled using a VoxelGrid filter with a grid size of 0.2m. For each point cloud, 8192 points are randomly chosen on the fly during training. Adopting the data splitting strategies as explained above, we obtain 20,731 point clouds for learning local descriptor and detector and 21,026 submaps for training global descriptor. As mentioned in the main paper, due to the lack of accurate point-to-point correspondences between different sequences, we generate training samples for local descriptor and detector learning with synthetic transformations (arbitrary rotations around the upright axis and Gaussian noise with $\sigma_{noise} = 0.02m$). The correspondence matrix $\mathbf{M} \in \mathbb{R}^{N \times N}$ used in Eq. (2) and Eq. (3) in the main paper can be computed as :

$$\mathbf{M}(i, j) = \mathbf{1}(\|\mathbf{p}_i - \mathbb{T}\mathbf{p}'_j\|_2 < \tau), \quad (1)$$

where $\mathbf{1}$ is the indicator function, \mathbf{p}_i and \mathbf{p}'_j are the 3D centers of the i th and j th local patches from the anchor point cloud \mathbf{P} and the positive point cloud \mathbf{P}' , respectively. \mathbb{T} is the applied synthetic transformation. τ is set to 0.5 in our experiments. Different from learning the local descriptors, the training of the global descriptor assembler can make use of the GPS/INS readings provided by the dataset as the supervision information. To this end, each downsampled submap is tagged with a UTM coordinate at its respective centroid. Similar point clouds are defined to be at most 10m apart and those dissimilar to be at least 50m apart.

4.3 Two-Phase Training

In practice, we train our model in two phases to improve stability. We first focus on the local descriptor matching task. The input is a pair of point clouds and the output is two sets of matching descriptors associated with their matching confidence. The loss function used is formulated as:

$$L = L_{desc} + \lambda L_{det}. \quad (2)$$

We use a batch size of 6 pairs of point clouds and for each pair of training samples, we randomly choose 512 points from the anchor point cloud as centers of the local patches (the points are chosen correspondingly in the transformed point clouds). This results in 6×512^2 combinations for the network per batch. We use the Adam optimizer with a learning rate of 10^{-4} , $\beta_1 = 0.9$, $\beta_2 = 0.999$. The learning rate is successively halved every 5 epochs. In the second phase, we freeze the local feature encoder so that the networks are enforced to train later feature extracting layers which aims to model higher-level semantic information, as well as the attention map and the NetVLAD layer which is for aggregating local features into discriminative global features. We use the same lazy quadruplet loss as used in PointNetVlad [1] and PCAN [10] which is formulated as:

$$\begin{aligned} L_{lazyQuad}(\mathcal{T}) = & \max([\alpha + \delta_{pos} - \delta_{neg}]_+) \\ & + \max([\beta + \delta_{pos} - \delta_{neg*}]_+), \end{aligned} \quad (3)$$

where \mathcal{T} refer to a tuple of point cloud samples in one training iteration, which includes an anchor point cloud \mathbf{P}_{anc} , a set of positive and a negative point clouds to the anchor $\{\mathbf{P}_{pos}\}, \{\mathbf{P}_{neg}\}$ as well as a random sample \mathbf{P}_{neg*} that is dissimilar to all the former samples. We use $\delta_{pos/neg}$ to denote the L2-distance between the global descriptor vectors of \mathbf{P}_{anc} and $\mathbf{P}_{pos/neg}$, while δ_{neg*} measures the distance between \mathbf{P}_{neg*} and \mathbf{P}_{neg} . The max operator of the first term selects the best positive in $\{\mathbf{P}_{pos}\}$, and the hardest negative in $\{\mathbf{P}_{neg}\}$. Similarly, the hardest negative in $\{\mathbf{P}_{neg}\}$ that gives the smallest δ_{neg*} value is selected by the second loss term. α and β are two different constant parameters giving the margins which are set as 0.5 and 0.2 respectively in our experiments. In this stage, we train our model with a single batch of data consisting 1 \mathbf{P}_{anc} , 1 \mathbf{P}_{neg*} , 2 \mathbf{P}_{pos} and 8 \mathbf{P}_{neg} . The Adam optimizer is used with $\beta_1 = 0.9, \beta_2 = 0.999$. The

initial learning rate is set as 5×10^{-4} and exponentially decayed after every 10 epochs until 10^{-5} . The dimension of the output global descriptor and the number of clusters in the NetVLAD layer are set to be same as in [1, 10], i.e., 256 and 64, respectively.

4.4 Testing Set Generation by Stereo DSO

In Sec. 4.4 of the main paper, we use the point clouds generated by Stereo DSO [8] to evaluate the generalization capability of our method. Here we provide more details on the testing set used in our experiments. The Bumblebee XB3 images under the wide-baseline configuration from the Oxford RobotCar dataset are taken as input to Stereo DSO, which estimates the camera poses and pixel depths by minimizing a photometric error obtained by direct image alignment. To extract a local point cloud, Stereo DSO is first run on a sub-sequence of images and all the valid 3D points associated with each keyframe are collected. These points are then projected into one common coordinate system utilizing the estimated relative camera poses.

To cover a wide range of day-time and weather conditions, eight sequences are chosen as listed in Tab. 1. One sequence in the overcast condition is set as the reference traversal, where the Stereo DSO is run on the full sequence. For each of the other seven sequences, 50 different images are uniformly selected as the starting frames and the corresponding local point clouds are extracted using the method described above. After filtering out invalid ones due to failures of Stereo DSO under some challenging situations, we obtain 332 point clouds as shown in Tab. 1. To generate the ground truth transformations, the corresponding point clouds accumulated from LiDAR scans are also collected to calculate the accurate relative poses by performing ICP between overlapped point clouds. This gives us 318 pairwise poses for testing. We crop each extracted point cloud with a fixed radius of 30m around its centroid followed by a downsampling step with a VoxelGrid filter with the fixed grid size of 0.2m. Lastly, we randomly rotate each point cloud around the vertical axis to evaluate the rotational invariance. We thank the authors of Stereo DSO again for their great help on generating the point clouds for us. These point clouds as well as the related data splits explained in Sec. 4.2 will be released to the community upon the publication of this paper.

4.5 Weak Supervision Used in the Ablation Study

In the ablation study, we compare our local descriptors with those trained using a weak supervised manner proposed by 3DFeatNet [3]. The weak supervision information is provided at the submap-level which is based on the ground truth coarse registrations of the point clouds in the training set. More specifically, the input to the network in each iteration is an anchor point cloud \mathbf{P}_{anc} with a positive point cloud \mathbf{P}_{pos} and a negative point cloud \mathbf{P}_{neg} to the anchor. The model is trained using the triplet loss computed by :

Sequence	Conditions	Valid point clouds
2014-12-02-15-30-08 (ref)	overcast	107
2014-11-14-16-34-33	night	8
2014-12-16-18-44-24	night	17
2014-11-25-09-18-32	rain	22
2015-02-03-08-45-10	snow	39
2015-03-24-13-47-33	sunny	43
2015-02-20-16-34-06	dusk, roadworks	47
2015-11-10-14-15-57	overcast, roadworks	49

Table 1: Number of point clouds generated by Stereo DSO for point cloud registration evaluation.

$$L = \sum_{n=1}^N \left[\min_{\mathbf{x}_i \in \mathcal{X}_{pos}} \|\mathbf{x}_n - \mathbf{x}_i\|_2 - \min_{\mathbf{x}_j \in \mathcal{X}_{neg}} \|\mathbf{x}_n - \mathbf{x}_j\|_2 + \gamma \right]_+, \quad (4)$$

where N is the number of local clusters extracted in each point cloud. Each local descriptor \mathbf{x}_n in \mathcal{P}_{anc} can find its closest descriptor in both \mathcal{P}_{pos} and \mathcal{P}_{neg} . The former is considered as the correct local match and the latter one is considered as the hardest negative. γ is the margin which is set as 0.2 in our experiments.

References

1. Angelina Uy, M., Hee Lee, G.: PointNetVLAD: Deep point cloud based retrieval for large-scale place recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4470–4479 (2018)
2. Groh, F., Wieschollek, P., Lensch, H.P.: Flex-convolution. In: Asian Conference on Computer Vision. pp. 105–122. Springer (2018)
3. Jian Yew, Z., Hee Lee, G.: 3DFeat-Net: Weakly supervised local 3D features for point cloud registration. In: Proceedings of the European Conference on Computer Vision. pp. 607–623 (2018)
4. Li, J., Lee, G.H.: USIP: Unsupervised stable interest point detection from 3D point clouds. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 361–370 (2019)
5. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: PointCNN: Convolution on x -transformed points. In: Advances in Neural Information Processing Systems. pp. 820–830 (2018)
6. Maddern, W., Pascoe, G., Linegar, C., Newman, P.: 1 year, 1000 km: The Oxford RobotCar dataset. The International Journal of Robotics Research **36**(1), 3–15 (2017)
7. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in Neural Information Processing Systems. pp. 5099–5108 (2017)
8. Wang, R., Schworer, M., Cremers, D.: Stereo DSO: Large-scale direct sparse visual odometry with stereo cameras. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3903–3911 (2017)
9. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph CNN for learning on point clouds. ACM Transactions on Graphics (TOG) **38**(5), 1–12 (2019)
10. Zhang, W., Xiao, C.: PCAN: 3D attention map learning using contextual information for point cloud based retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 12436–12445 (2019)