Learning Feature Descriptors using Camera Pose Supervision –Supplementary Material–

Qianqian Wang[1,2], Xiaowei Zhou[3], Bharath Hariharan[1], and Noah Snavely[1,2]

[1]Cornell University [2]Cornell Tech [3]Zhejiang University

1 Overview

In this supplemental material, we provide additional experimental results, visualizations, and implementation details. In Sec. 2, we demonstrate the performance of our learned descriptors on a standard visual localization benchmark. In Sec. 3, we investigate the robustness of our method to the errors in camera poses. In Sec. 4, we visualize the probabilistic distribution of correspondences for given query points in example image pairs. In Sec. 5, we show more qualitative results for dense feature matching. In Sec. 6, we visualize the sparse point cloud of SfM using our descriptors. Finally, in Sec. 7 we provide additional implementation details, including network architecture and training details.

2 Visual Localization Results

As an additional experiment, we evaluate how our descriptors facilitate the task of visual localization. Following D2-Net [6], we use the Aachen Day-Night dataset [16] and the standard protocol proposed in [15] to evaluate the performance of our descriptors in the context of long-term localization[1]. Specifically, this protocol first does exhaustive feature matching between the day-time images with known poses, and then uses triangulation to obtain 3D scene structure. The resulting 3D models are then used to localize 98 night-time query images in the dataset. For each night-time image, up to 20 relevant day-time images with known camera poses are given as reference images.

Following D2-Net [6], we report the percentage of correctly localized night-time queries under given error thresholds on camera position and orientation. As in the paper, we report the performance of our descriptors combined with both SIFT [9] and SuperPoint [5] keypoints. We compare our method with other state-of-the-art methods and report the results in Tab. 1. It can be seen that combined with SuperPoint [5] keypoints, our descriptors achieve the best overall performance. When combined with SIFT [9] keypoints, our descriptors are also comparable with the baseline methods. Note that our descriptors are learned using only the weak supervision of camera poses.

[1] `https://www.visuallocalization.net/`

Table 1: **Evaluation on the Aachen Day-Night dataset[15,16].** We report the percentage of correctly registered query images within given error thresholds (translation error in meter and rotation error in degree). Our descriptors achieve state-of-the-art performance.

	Correctly localized queries (%)		
Methods	(0.5m, 2°)	(1m, 5°)	(5m, 10°)
SIFT [9]	36.7	54.1	72.5
HardNet [11]	41.8	61.2	84.7
SuperPoint [5]	42.9	57.1	77.6
DELF [12]	38.8	62.2	85.7
D2-Net [6]	45.9	68.4	**88.8**
R2D2 [14]	46.9	66.3	**88.8**
SOSNet [18]	42.9	64.3	85.7
ContextDesc [10]	**48.0**	63.3	**88.8**
Our w/ SIFT kp.	44.9	68.4	87.8
Our w/ SuperPoint kp.	45.9	**69.4**	**88.8**

3 Sensitivity to Camera Pose Error

In the paper, we obtain camera poses from the Structure from Motion (SfM) pipeline. With the global bundle adjustment in the SfM, the camera poses are guaranteed to be fairly accurate. But how does our method perform given noisier and less accurate poses, e.g., poses from visual inertial odometry?

To investigate the robustness of our method to errors in camera poses, we add different levels of noise of uniform distribution to the camera poses and look into how the performance of our method degrades with them. Specifically, we add noise to both relative rotation and translation, with rotation noise measured in degree and translation noise measured in percentage. We follow the same experimental setup as in Fig.7 (a) and report the MMA score at 3-pixel threshold in Tab. 2. It can be seen that the performance of our method is fairly stable even when moderate levels of noise of uniform distribution are added to the camera pose. Thus, we believe we can also train with less accurate poses e.g. from visual inertial odometry.

Table 2: **The performance of our method w.r.t. the noise in camera poses.** We sample the noise randomly from uniform distributions and add them to the relative rotation and translation.

Noise level (Rotation[°], Translation[%])	No noise	(±2.5, ±1)	(±5, ±2.5)
MMA@3px	62.7	61.7	59.4

4 Visualizing Predicted Distributions

Our method obtains the correspondence of a given query point in one image from distributions over pixel locations in the other image at training time. The quality of the distributions reflects the quality of learned descriptors. Therefore, we visualize the distributions at both the coarse and fine levels for test image pairs drawn from the MegaDepth dataset [8] and illustrate the results in Fig. 1. This figure shows that, even under challenging illumination and perspective changes, the peaks of the distributions can still indicate correct correspondences. Moreover, compared with the coarse-level distributions, the fine-level distributions tend to be peakier, demonstrating the discriminability of the fine-level feature descriptors. However, our method can fail when repeating structures are present in the image, as shown in the last row of Fig. 1.

5 More Qualitative Results on Dense Feature Matching

In this section, we present more qualitative results on dense feature matching. To perform dense feature matching, we resize the coarse-level feature descriptors to be the size of fine-level feature descriptors, and concatenate both to form our final dense feature descriptors, whose spatial resolution is 1/4 of the input image. For any given point in the first image, we find its nearest neighbor match in the second image as its correspondence. No post-processing techniques are used. The dense correspondences are visualized in Fig. 2. It is shown that the correspondences are reasonable under various illumination and viewpoint changes.

6 Visualizing SfM Point Clouds

We visualize the sparse point clouds obtained by the protocol of the ETH local feature benchmark [17] using our descriptors. The result is shown in Fig. 3.

7 Implementation Details

In this section, we provide our network architecture and additional techniques that we find useful for improving the robustness and speed of training.

Network Architecture. During training, our system takes a pair of images as input and extracts their features using a network with shared weights. We use ResNet50 [7], as implemented in PyTorch [13], as the backbone of our network. Our network is fully convolutional and accepts input images of any size. We take a single image of size $640 \times 480 \times 3$ as an example input and present a detailed network architecture in Tab. 3. Our code and model will be made available soon.

Curriculum Learning. In general, we find that simply using ImageNet [4] pretrained weights gives the network a good initialization. To help the network converge faster, we can also optionally leverage curriculum learning [2], i.e.,

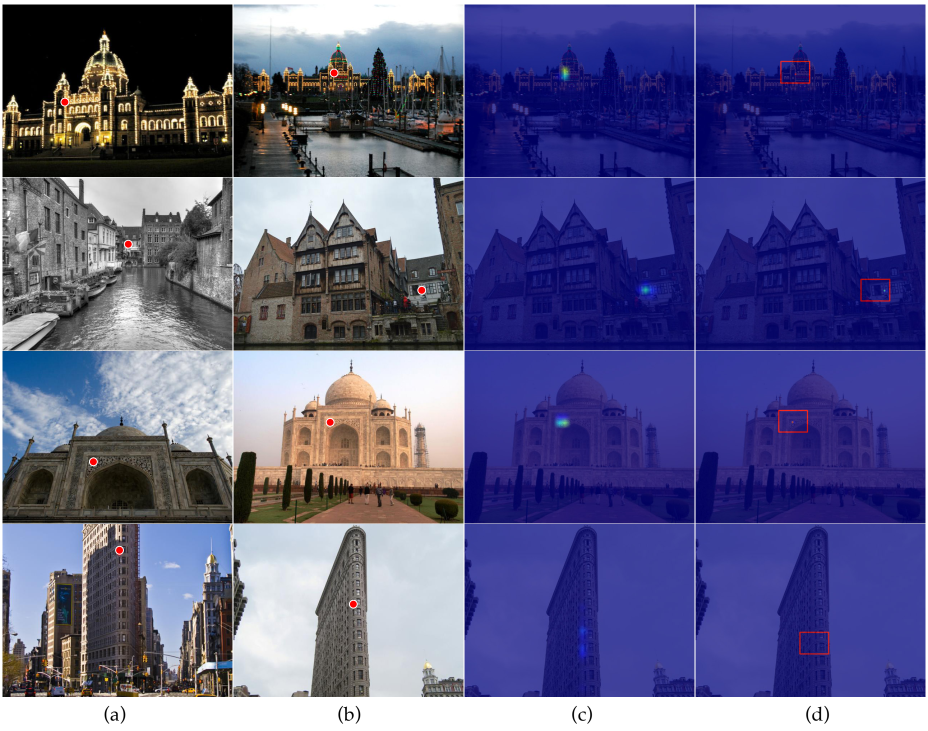


Fig. 1: **Predicted distributions on MegaDepth [8]. (a)**: query image with a single red point indicating the query point, which is randomly sampled from the reconstructed SIFT keypoints provided by MegaDepth [8]. **(b)**: target image with red point indicating the ground-truth match. **(c)**: the predicted coarse-level distribution for the query point, overlaid on the target image. **(d)**: the predicted fine-level distribution for the query point, overlaid on the target image. The red rectangle in (d) is the local window W, whose center location is determined by the highest mode of the coarse-level distribution. The size of W is $1/8\times$ that of the fine-level feature map. Regions outside of the local window W have probability 0. Note that for ease of visualization, we resized all maps to the original image size. The last row shows a failure case due to repetitive patterns.

presenting easier training examples and then harder ones. Specifically, we can sort image pairs by their relative rotation angles, and feed easy pairs into the network at the beginning of training for better initialization.

Training Points Pruning. As mentioned in Sec. 3.2 of the paper, the sampled query points in the first image may not have true correspondences in the second image, and enforcing losses on these points could prohibit training. To alleviate this issue, we introduce the reweighting strategy in the paper. Besides this “soft” filtering, we can also do “hard” filtering to get rid of points that are not likely to have true correspondences. Specifically, we could assume the depth range of the scene to be $[d_{\min}, d_{\max}]$. For each query point in the first image, this range will yield a segment along its epipolar line in the second image. If this line segment does not intersect the image plane of the second image, then this

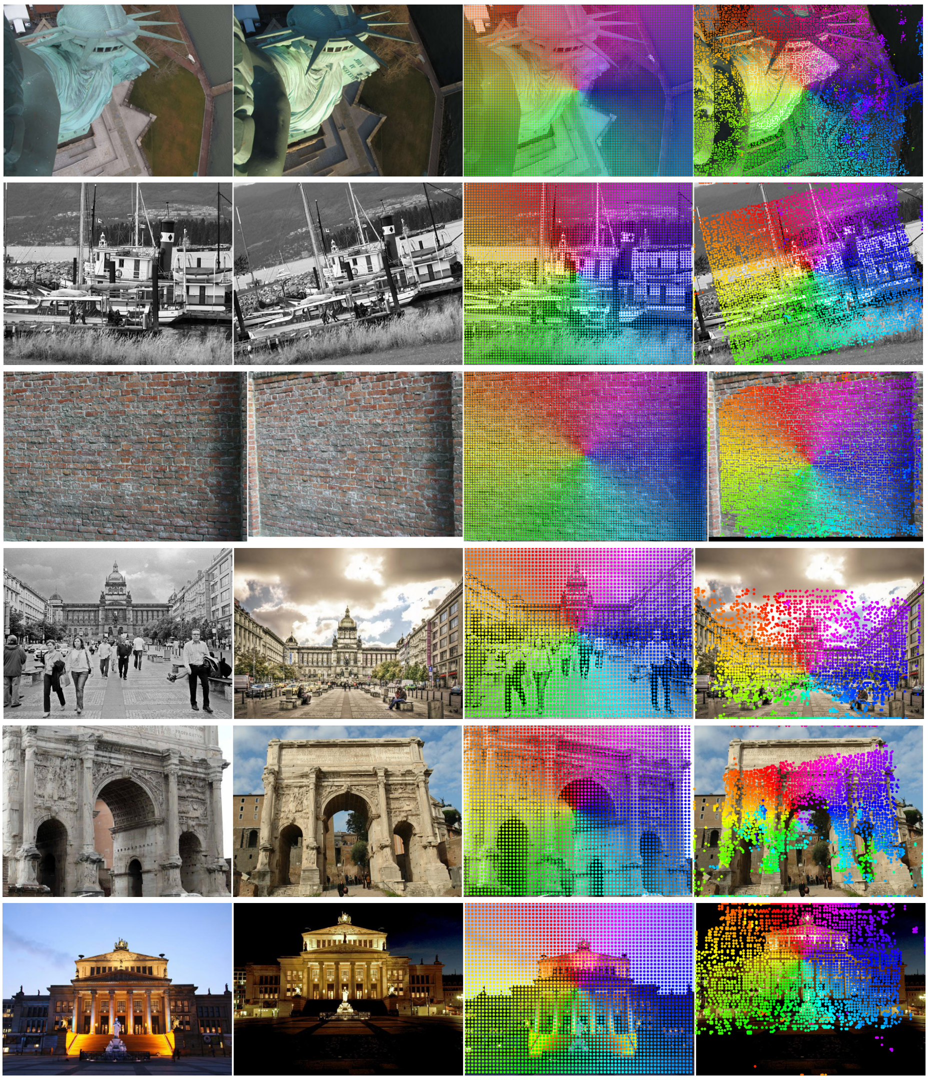
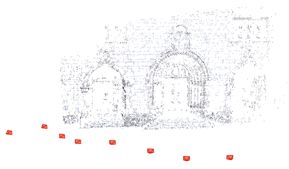


Fig. 2: **Dense Correspondence**. For each row, the first two columns show original images, and the last two columns show the correspondences found by our method. Colors indicate correspondences. The first and second three pairs of images are from HPatches [1] and MegaDepth [8], respectively. For each query point in the first image, we find its correspondence in the second image by nearest neighbor search using our dense descriptors. For the ease of visualization, we show correspondences for query points with an interval of 50 pixels on regular image grids.

point is excluded from training. Since we do not assume known depths, we can only roughly determine the depth range for each image. Even so, this strategy is still found effective in removing a large fraction of unwanted training points. For


(a) Herzjesu

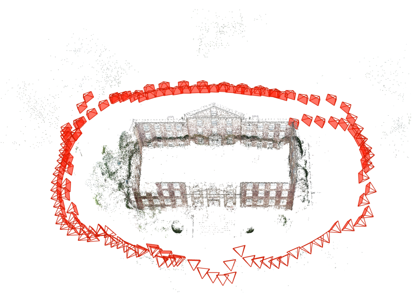

(b) South Building

Fig. 3: **Sparse point cloud visualization.** The two scenes are "Herzjesu" and "South Building", respectively. We follow the ETH local benchmark [17] to obtain the SfM point clouds for both scenes.

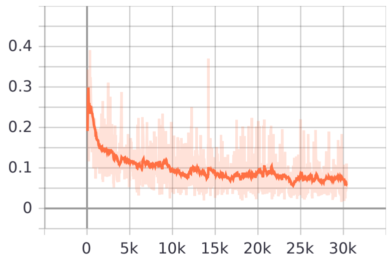
Table 3: **Network architecture**. The first five layers have the same structure as in the original ResNet50 [7] design, and are initialized with ImageNet [4] pretrained weights. "**Coarse**" and "**Fine**" indicate the coarse- and fine-level output feature descriptors, respectively. Note that "Conv" stands for a sequence of operations: convolution, rectified linear units (ReLU) and batch normalization. The default stride is 1. "Upconv" stands for a bilinear upsampling with certain factor, followed by a "Conv" operation with stride 1. "[·, ·]" is the channel-wise concatenation of two feature maps.

Input (**id**: dimension)	Layer	Output (**id**: dimension)
0: $640 \times 480 \times 3$	7×7 Conv, 64, stride 2	**1**: $320 \times 240 \times 64$
1: $320 \times 240 \times 64$	3×3 MaxPool, stride 2	**2**: $160 \times 120 \times 64$
2: $160 \times 120 \times 64$	Residual Block 1	**3**: $160 \times 120 \times 256$
3: $160 \times 120 \times 256$	Residual Block 2	**4**: $80 \times 60 \times 512$
4: $80 \times 60 \times 512$	Residual Block 3	**5**: $40 \times 30 \times 1024$
5: $40 \times 30 \times 1024$	1×1 Conv, 128	**Coarse**: $40 \times 30 \times 128$
5: $40 \times 30 \times 1024$	3×3 Upconv, 512, factor 2	**6**: $80 \times 60 \times 512$
[**4**, **6**]: $80 \times 60 \times 1024$	3×3 Conv, 512	**7**: $80 \times 60 \times 512$
7: $80 \times 60 \times 512$	3×3 Upconv, 256, factor 2	**8**: $160 \times 120 \times 256$
[**3**, **8**]: $160 \times 120 \times 512$	3×3 Conv, 256	**9**: $160 \times 120 \times 256$
9: $160 \times 120 \times 256$	1×1 Conv, 128	**Fine**: $160 \times 120 \times 128$

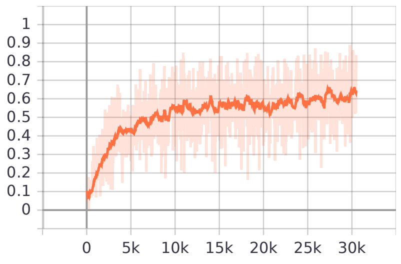
MegaDepth [8], we approximate $d_{\min}$, $d_{\max}$ as 10^{-3} and 1 times the maximum distance between two cameras, respectively, in each reconstructed model.

Training Curves. To give a most straightforward illustration on the effectiveness of our loss function, we show the curves of our training loss as well as the training PCK [3] in Fig. 4. For each training pair, the PCK is computed on sparsely reconstructed keypoints from MegaDepth [8] with a threshold of 5 pixels. Higher PCK means more correct correspondences found on the training data, indicating better descriptors. It is shown in Fig. 4 that as our training loss goes down, the

PCK score increases, which verifies that minimizing our loss terms does lead to the improvement of the feature descriptors.



(a) Training loss



(b) Training PCK

Fig. 4: **Training loss and PCK**. We train our network on MegaDepth [8] dataset, and plot the curves of training loss and PCK [3].

References

1. Balntas, V., Lenc, K., Vedaldi, A., Mikolajczyk, K.: Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In: CVPR (2017)
2. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: ICML (2009)
3. Choy, C.B., Gwak, J., Savarese, S., Chandraker, M.: Universal correspondence network. In: NeurIPS (2016)
4. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)
5. DeTone, D., Malisiewicz, T., Rabinovich, A.: Superpoint: Self-supervised interest point detection and description. In: CVPR Workshops (2018)
6. Dusmanu, M., Rocco, I., Pajdla, T., Pollefeys, M., Sivic, J., Torii, A., Sattler, T.: D2-net: A trainable cnn for joint detection and description of local features. arXiv preprint arXiv:1905.03561 (2019)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
8. Li, Z., Snavely, N.: Megadepth: Learning single-view depth prediction from internet photos. In: CVPR (2018)
9. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV **60**(2), 91–110 (2004)
10. Luo, Z., Shen, T., Zhou, L., Zhang, J., Yao, Y., Li, S., Fang, T., Quan, L.: Contextdesc: Local descriptor augmentation with cross-modality context. In: CVPR (2019)
11. Mishchuk, A., Mishkin, D., Radenovic, F., Matas, J.: Working hard to know your neighbor's margins: Local descriptor learning loss. In: NeurIPS (2017)
12. Noh, H., Araujo, A., Sim, J., Weyand, T., Han, B.: Large-scale image retrieval with attentive deep local features. In: ICCV (2017)

13. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch. In: NIPS Autodiff Workshop (2017)
14. Revaud, J., Weinzaepfel, P., de Souza, C.R., Humenberger, M.: R2D2: repeatable and reliable detector and descriptor. In: NeurIPS (2019)
15. Sattler, T., Maddern, W., Toft, C., Torii, A., Hammarstrand, L., Stenborg, E., Safari, D., Okutomi, M., Pollefeys, M., Sivic, J., et al.: Benchmarking 6dof outdoor visual localization in changing conditions. In: CVPR (2018)
16. Sattler, T., Weyand, T., Leibe, B., Kobbelt, L.: Image retrieval for image-based localization revisited. In: BMVC. p. 4 (2012)
17. Schönberger, J.L., Hardmeier, H., Sattler, T., Pollefeys, M.: Comparative Evaluation of Hand-Crafted and Learned Local Features. In: CVPR (2017)
18. Tian, Y., Yu, X., Fan, B., Wu, F., Heijnen, H., Balntas, V.: Sosnet: Second order similarity regularization for local descriptor learning. In: CVPR (2019)