Supplementary Material for Multi-scale Local Implicit Keypoint Descriptor for Keypoint Matching

JongMin Lee
Seoul National University
sdrjseka96@naver.com

Eunhyeok Park
POSTECH
canusglow@gmail.com

Sungjoo Yoo
Seoul National University
sungjoo.yoo@gmail.com

1. Architecture Details

	Layer Description	K	Output Tensor Dim.
#0	Input RGB image		3 × H × W
	Encoder		
#1	Conv2d + BatchNorm + LReLU	3	32 × H × W
#2	Conv2d + BatchNorm + LReLU + Dropout	3	32 × H × W
#3	Max. Pooling (× 1/2)		32 × H/2 × W/2
#4	Conv2d + BatchNorm + LReLU	3	64 × H/2 × W/2
#5	Conv2d + BatchNorm + LReLU + Dropout	3	64 × H/2 × W/2
#6	Max. Pooling (× 1/2)		64 × H/4 × W/4
#7	Conv2d + BatchNorm + LReLU	3	128 × H/4 × W/4
#8	Conv2d + BatchNorm + LReLU + Dropout	3	128 × H/4 × W/4
#9	Max. Pooling (× 1/2)		128 × H/8 × W/8
#10	Conv2d + BatchNorm + LReLU	3	256 × H/8 × W/8
#11	Conv2d + BatchNorm + LReLU + Dropout	3	256 × H/8 × W/8
#12	Max. Pooling (× 1/2)		256 × H/16 × W/16
	Score Head		
#13	Conv2d + BatchNorm + LReLU (#11)	3	256 × H/8 × W/8
#14	Conv2d + BatchNorm	3	256 × H/8 × W/8
#15	Pixel Shuffle (× 2)		64 × H/4 × W/4
#16	Conv2d + BatchNorm + LReLU (#8 ⊕ #15)	3	256 × H/4 × W/4
#17	Conv2d + Sigmoid	3	1 × H/4 × W/4
	Location Head		
#18	Conv2d + BatchNorm + LReLU (#11)	3	256 × H/8 × W/8
#19	Conv2d + BatchNorm	3	256 × H/8 × W/8
#20	Pixel Shuffle (× 2)		64 × H/4 × W/4
#21	Conv2d + BatchNorm + LReLU (#8 ⊕ #20)	3	256 × H/4 × W/4
#22	Conv2d + Tan. Hyperbolic	3	2 × H/4 × W/4
	Decoder		
#23	Conv2d + BatchNorm + ReLU (#12)	3	64 × H/16 × W/16
#24	Conv2d + BatchNorm + ReLU	3	64 × H/16 × W/16
#25	Bilinear Upsampling (×2)		64 × H/8 × W/8
#26	Conv2d + BatchNorm + ReLU (⊕ #11)	3	64 × H/8 × W/8
#27	Conv2d + BatchNorm + ReLU	3	64 × H/8 × W/8
#28	Bilinear Upsampling (×2)		64 × H/4 × W/4
#29	Conv2d + BatchNorm + ReLU (⊕ #8)	3	64 × H/4 × W/4
#30	Conv2d + BatchNorm + ReLU	3	64 × H/4 × W/4

Table 1. Description of our model's CNN parts, composed of an encoder, decoder and two detector heads. The network receives as input an RGB image and returns scores, locations and descriptor's feature maps. Numbers in parentheses indicate input layers and ⊕ denotes feature concatenation.

Tables 1 and 2 show the details of CNN part and LIKD module, respectively. As mentioned in Figure 2 and Section 4.4 in the paper, the final descriptor is obtained from LIKD module by feeding descriptor feature map (#24 & #30) and keypoint's coordinate (#22).

	Layer Description	Output Tensor Dim.
	Each corner's feature (f_θ in Eqn. 7)	
#0	Input relative coordinate & feature vector	N × (64 × 9 + 2)
#1	Linear + ReLU	N × 512
#2	Linear + ReLU	N × 256
#3	Linear	N × 128
	Each corner's weight (g_ϕ in Eqn. 7)	
#4	Input relative coordinate & feature vector	(4 × N) × (64 + 2)
#5	Linear	(4 × N) × 64
#6	Linear + ReLU	N × 64
#7	Linear	N × 4
#8	Softmax	N × 4

Table 2. Description of LIKD. Each network receives feature vector and relative coordinate of keypoints from corners as input. N denotes the number of detected keypoints.

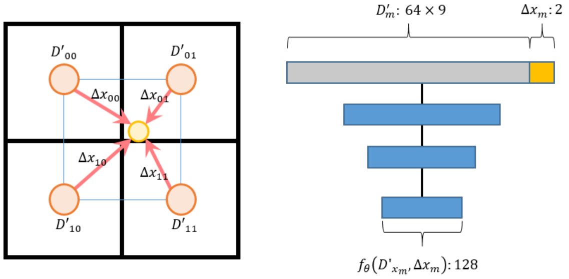


Figure 1. Description of f_θ in LIKD.

Eqn. 7 in the paper is shown below for a better reference.

$$d'_x = \sum_m g_\phi(D_{x_{all}}, \Delta x_{all}, m) \times f_\theta(D'_{x_m}, \Delta x_m) \quad (1)$$

Figures 1 and 2 gives the details of each function f_θ and g_ϕ, respectively. We run f_θ four times (for four different corners of the keypoint) for each keypoint. Figure 1 shows

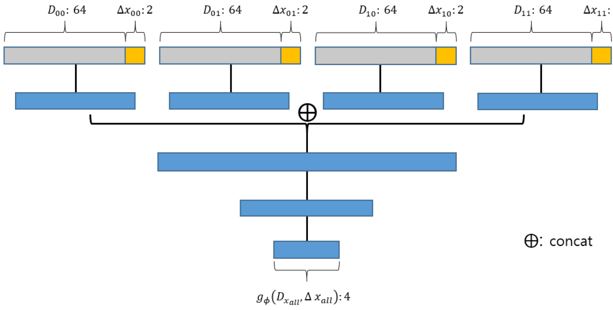


Figure 2. Description of g_ϕ in LIKD.

that for each corner $m \in \{00, 01, 10, 11\}$, f_θ takes as input D'_m and Δx_m and produces a 128-dim vector. Note that, as shown below (Eqn. 5 in the paper),

$$D'_{x_{ij}} = Concat(\{D_{x_{i+l,j+k}}\}_{l,k \in \{-1,0,1\}})$$

D'_m is derived from feature map D computed by CNN parts, where the feature vectors in a window of size 3 are concatenated to give a vector of 64×9 dimensions (unfolding in PyTorch).

g_ϕ is executed to produce four weights of four corners. As Figure 2 shows, we execute g_ϕ only once to obtain four weights for the four corners. Given the four concatenated D_m's and x_m's, g_ϕ applies a linear layer and concatenates the results, and then applies two linear layers finally to produce a weight vector of size 4.

The computation cost of CNN and LIKD parts, in Tables 1 and 2, amounts to about 36.5GFLOPS (CNN), and 3.7GFLOPS (LIKD), respectively, for obtaining 1,000 keypoints and descriptors from a 480×480 image.

2. Training Details

We first pretrain our model using the COCO 2017 dataset [2] according to KeypointNet [6]'s training pipeline. The training pipeline uses self-supervised learning with randomly generated homography, and the dataset consists of about 118,000 images. We pretrain the model with 5 epochs with COCO dataset, and train the model with GL3D [3,5,7] and [4] for 3 epochs.

When we train our model's descriptor from GL3D dataset, the number of negative samples for triplet margin loss and circle loss is determined by the number of keypoints detected by location head. Since the training dataset's image resolution is 480×480 and the cell size is 4, the maximum number of detected keypoints is 120×120. We found that this requires huge memory for calculating loss in training, so we randomly sample 4,000 keypoints for negative samples.

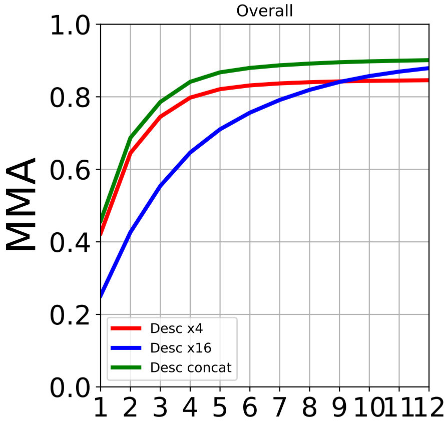
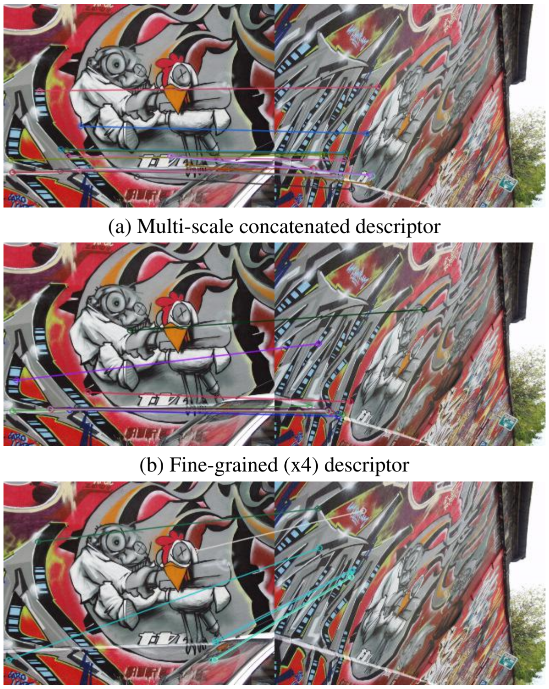


Figure 3. Comparison of Mean Matching Accuracy (MMA) on HPatches dataset [1]. We evaluate each descriptor with varying pixel error thresholds.



(a) Multi-scale concatenated descriptor

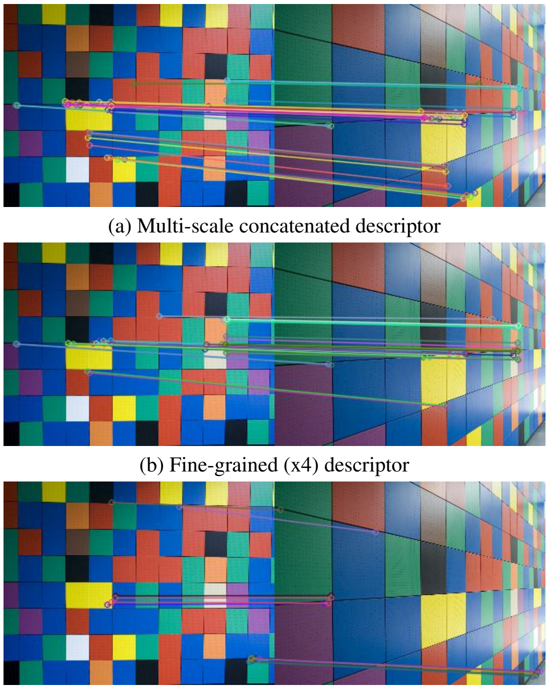
(b) Fine-grained (x4) descriptor

(c) Coarse-grained (x16) descriptor

Figure 4. Qualitative comparison of descriptors on the "v_graffiti" subset of HPatches.

3. Analysis of Descriptors

Figure 3 compares coarse-grained (x16), fine-grained (x4) and concatenated descriptors on MMA for HPatches [1] dataset. The figure shows that Desc x4 outperforms Desc x16 when the threshold is small (less than 9px). However, the performance of concatenated descriptor shows that Desc x16 can offer additional performance gain even when the threshold is small. As the threshold gets larger, Desc



(a) Multi-scale concatenated descriptor

(b) Fine-grained (x4) descriptor

(c) Coarse-grained (x16) descriptor

Figure 5. Qualitative comparison of descriptors on the "v_colors" subset of HPatches.

x16 becomes dominant. Moreover, Desc x16 continues to improve performance as the threshold gets larger, which contributes to the monotonically increasing performance of the concatenated descriptor. Overall, Figure 3 shows both coarse- and fine-grained descriptors contribute, in a rather complementary manner, to the performance of concatenated descriptor. The qualitative comparisons are provided in Figures from 4 to 5.

Methods	**MMA@1**	**MMA@3**	**MMA@5**	**MMA@10**
Desc x4 (before)	0.426	0.737	0.813	0.835
Desc x4 (after)	0.423	0.745	0.821	0.843
Desc x16 (before)	0.244	0.540	0.700	0.848
Desc x16 (after)	0.252	0.554	0.710	0.857

Table 3. MMA comparison on each descriptor before (V4 in Table 1 of the paper) and after (V5) negative sampling.

Table 3 demonstrates the effect of negative sampling. Negative sampling consistently improves MMA on both coarse- and fine-grained descriptors across all the thresholds except one case (of 1px for Desc x4).

4. Qualitative Results

We show examples of successful matching under strong illuminations, rotations and perspective transformations with HPatches dataset in Figure 6. We show only the matching cases which are filtered by inlier mask obtained from *findHomography* of OPENCV.

We also show examples of successful matching in indoor and outdoor image pairs with Scannet and Megadepth dataset in Figures 7 and 8. We show only the matching cases which are filtered by inlier mask obtained from *findEssentialMat* of OPENCV.

Finally, we show examples of successful matching in Aachen day-night dataset in Figure 9. We show only the matching cases which are filtered by inlier mask obtained from *findFundamentalMat* of OPENCV.

References

[1] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors, 2017. 2

[2] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. 2

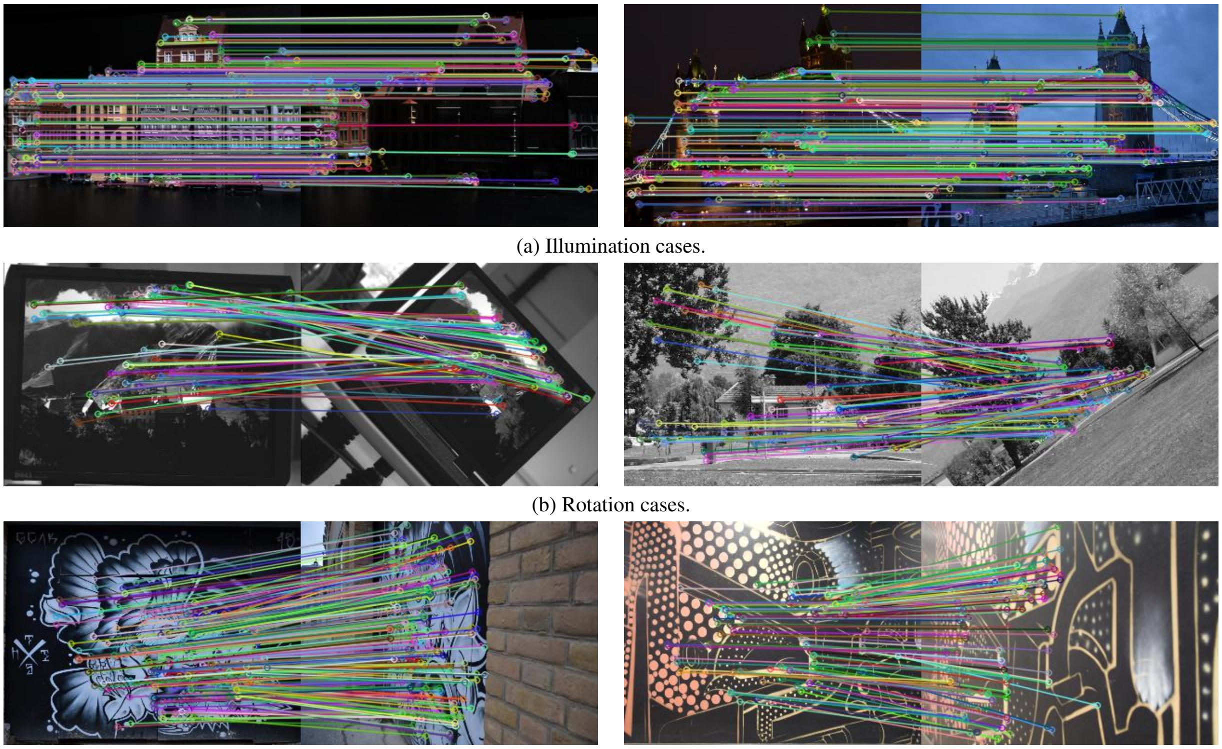
[3] Zixin Luo, Tianwei Shen, Lei Zhou, Siyu Zhu, Runze Zhang, Yao Yao, Tian Fang, and Long Quan. Geodesc: Learning local descriptors by integrating geometry constraints. In *European Conference on Computer Vision (ECCV)*, 2018. 2

[4] Filip Radenovic, Giorgos Tolias, and Ondrej Chum. CNN image retrieval learns from bow: Unsupervised fine-tuning with hard examples. *CoRR*, abs/1604.02426, 2016. 2

[5] Tianwei Shen, Zixin Luo, Lei Zhou, Runze Zhang, Siyu Zhu, Tian Fang, and Long Quan. Matchable image retrieval by learning from surface reconstruction. In *The Asian Conference on Computer Vision (ACCV*, 2018. 2

[6] Jiexiong Tang, Hanme Kim, Vitor Guizilini, Sudeep Pillai, and Rares Ambrus. Neural outlier rejection for self-supervised keypoint learning, 2019. 2

[7] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. 2



(a) Illumination cases.

(b) Rotation cases.

(c) Perspective transformation cases.

Figure 6. Qualitative results of our method on images pairs of the HPatches dataset.

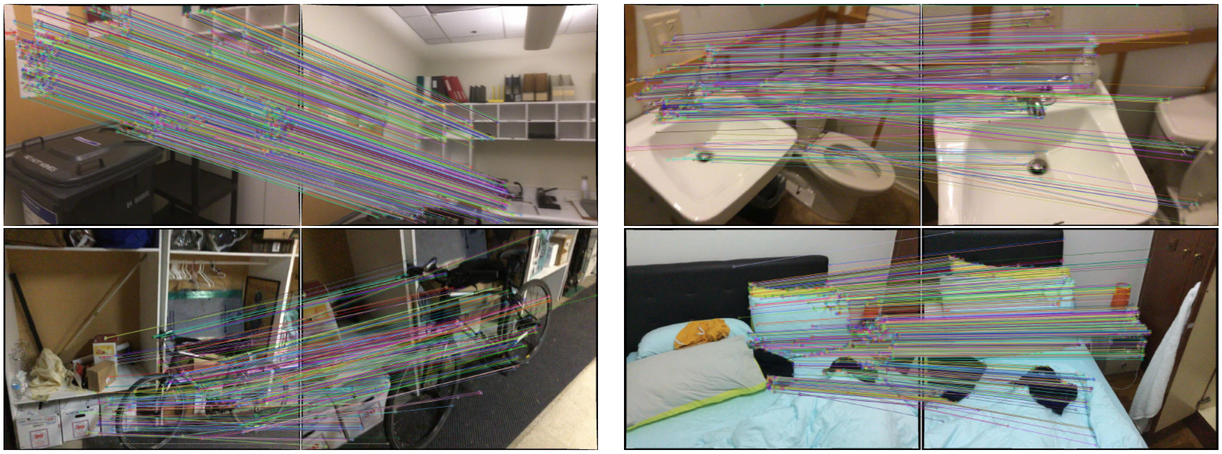


Figure 7. Qualitative results of our method on images pairs of the Scannet dataset.

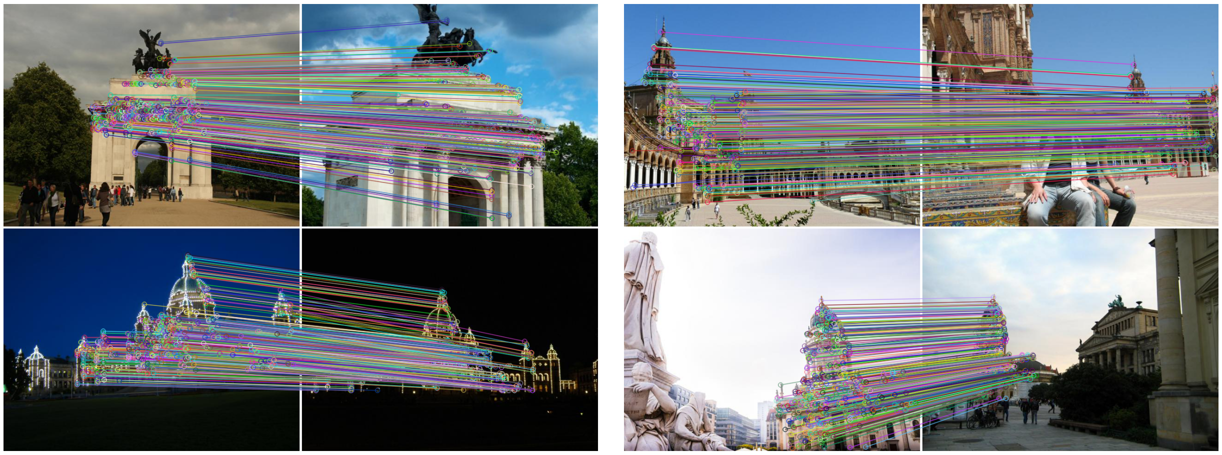


Figure 8. Qualitative results of our method on images pairs of the Megadepth dataset.

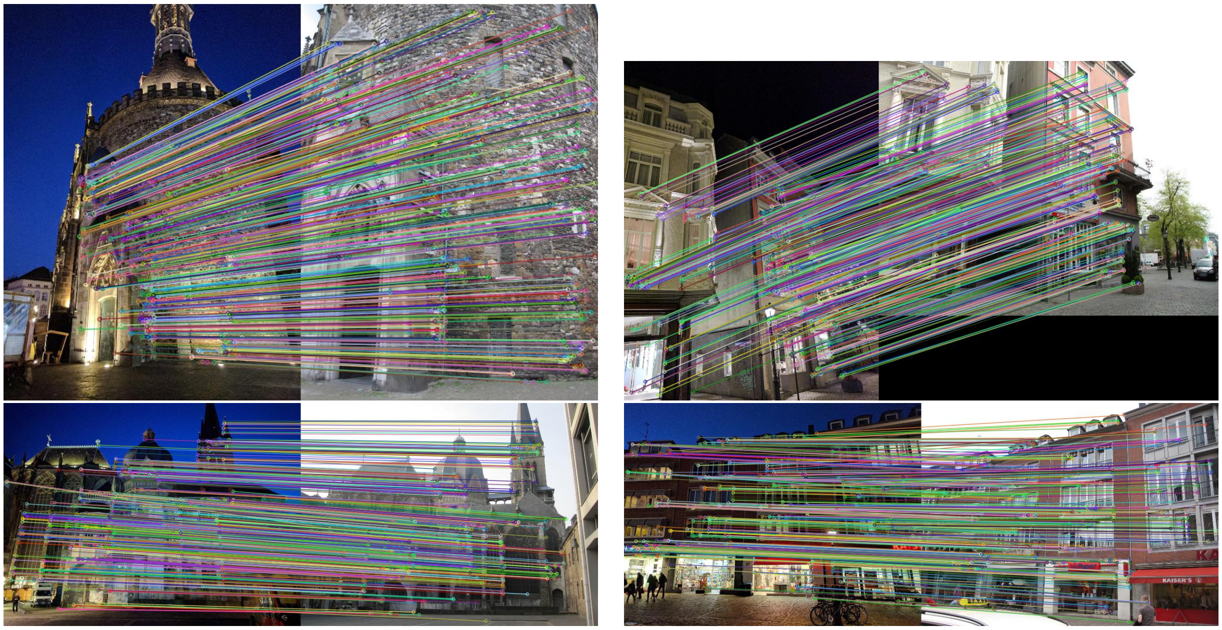


Figure 9. Qualitative results of our method on images pairs of the Aachen day-night dataset.