Supplementary Material: LoFTR: Detector-Free Local Feature Matching with Transformers

1. Formulation Details

Positional Encoding. We use the 2D extension of the absolute sinusoidal positional encoding following [1]:

$$\mathcal{PE}^{i}_{x,y} = f(x,y)^{i} := \begin{cases} \sin\left(\omega_k \cdot x\right), & i = 4k \\ \cos\left(\omega_k \cdot x\right), & i = 4k+1 \\ \sin\left(\omega_k \cdot y\right), & i = 4k+2 \\ \cos\left(\omega_k \cdot y\right), & i = 4k+3 \end{cases},$$

where $\omega_k = \frac{1}{10000^{2k/d}}$, d is the number of channels of features which positional encoding applied on, i is a index for feature channels.

2. Timing

We test LoFTR's speed on 100 randomly sampled ScanNet image pairs, with a size of 640×480, and report the averaged results. A RTX 2080Ti GPU is used for timing. For the full model, both the optimal transport and the dual-softmax variants are reported. As illustrated in Tab. 1, LoFTR-DS runs faster on average than LoFTR-OT. The difference comes mainly from the matching module. When using a smaller model with $1/16$ of the original resolution at the coarse-level and $1/4$ at the fine-level (i.e., the second ablation experiment in the main text), the speed is about 20% faster. Note that performance may vary over different image pairs and different matching layers, mainly caused by the different number of matches for the coarse-to-fine model to refine. We notice that a large fraction of time is spent on the Local Feature CNN, which is caused by the use of FPN and a design flaw in the widths of the FPN backbone (we use a width of 128 for the ResNet stem and the first stage). We expect to fix this problem in our future released models.

Process	Timing(ms)	
	Full model (OT / DS)	Small model (OT)
Local Feature CNN	48.2	37.4
Coarse-Level LoFTR	33.6	33.5
Matching Module	21.0 / 12.0	9.1
Coarse-to-Fine Module	17.2 / 19.7	13.1
Overall	121 / 116	96

Table 1: **Timing measurements.** Time cost for an image pair of 640×480. Both the full model of LoFTR-OT and LoFTR-DS and the small model of LoFTR-OT are tested.

3. Implementation Details

3.1. Architecture

Local Feature CNN. The Local Feature CNN is built with a modified ResNet-18 with FPN. Different from the original ResNet-18, we use a width of 128 for the stem and widths of [128, 196, 256] for the subsequent three stages. We build the FPN with levels P_1 through P_3 and take features from P_3 as the input features $\tilde{F}_{tr}$ for coarse-level LoFTR, features from P_1 as the input features $\hat{F}_{tr}$ for fine-level LoFTR. Unlike the original FPN, we use various widths for different levels, which are the same as the corresponding stages in the bottom-up ResNet.

LoFTR Encoder Layer. The original Transformer [10] is composed of a Transformer encoder and a Transformer decoder, both of which consist of several homogeneous encoder or decoder layers. Our LoFTR module is an encoder-only architecture consists of cascaded LoFTR encoder layers. The LoFTR encoder layer inspired by [8, 7] is a variant of the Post-LN encoder layer originally used in [10]. Fig. 1 illustrates their differences. In our experiments, the Post-LN layer(Fig. 1(b)) requires careful tuning of the learning rate and its scheduling and suffers from slow convergence. Our variant has fast convergence without extensive hyperparameter tuning. For normalization layers, we use the layer normalization, while batch normalization also fits into our encoder layer and performs comparatively. As mentioned in the main paper, we use 4 encoder layers in the coarse-level and 1 encoder layer in the fine-level. We use 8 heads in all multi-head attention layers. Input features have 256 and 128 channels respectively for the coarse- and fine-level LoFTR modules, both from the FPN backbone.

Differentiable Matching with Optimal Transport. LoFTR can utilize the optimal transport layer proposed in [8] to extract matches in coarse-level. Similar to [8], we express the pairwise score as the similarity of coarse-level LoFTR features $\tilde{F}^{A}_{tr}$ and $\tilde{F}^{B}_{tr}$:

$$\mathcal{S}\left(i,j\right) = \frac{1}{D} \cdot \langle \tilde{F}^{A}_{tr}(i), \tilde{F}^{B}_{tr}(j) \rangle,$$

where D is the feature dimension of $\tilde{F}_{tr}$. Under the partial assignment setup, we augment the score matrix $\mathcal{S}$ to $\overline{\mathcal{S}}$ with

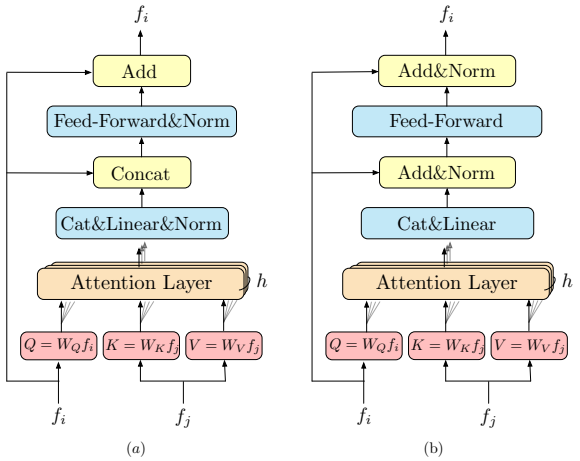

Figure 1: **Encoder layer in LoFTR and the Post-LN variant used in [10].**

a learnable slack variable (referred to as the dustbin in [8]), in order to solve the assignment solution with sinkhorn algorithm:

$$\overline{\mathcal{S}}(i, N^B) = \overline{\mathcal{S}}(N^A, j) = \overline{\mathcal{S}}(N^A, N^B) = z \in \mathbb{R},$$

where N^A and N^B are the number of features in $\tilde{F}^A_{tr}(i)$ and $\tilde{F}^B_{tr}(j)$. We empirically find that three sinkhorn iterations gives a good balance between speed and the overall performance.

3.2. Dataset Preparation

MegaDepth. We use the MegaDepth [6] dataset to train and validate LoFTR on outdoor pose estimation and homography estimation. We remove scenes used as the test set in the Image Matching Challenge, as well as scenes with low-quality depth maps pointed out by [3]. Among scenes kept, we enumerate all image pairs with covisible scores in a range of $[0.1, 0.7]$ and further split each scene into sub-scenes with covisible scores in ranges $[0.1, 0.3], [0.3, 0.5], [0.5, 0.7]$ respectively. Sub-scenes used as the validation set in the Image Matching challenge are also grouped as a validation set, from with we sample 1500 images with each sub-scene evenly sampled. The remaining sub-scenes are used for training, leading to a training set composed of 368 sub-scenes in total.

ScanNet. We follow exactly [8] for the construction of ScanNet dataset. About 230M image pairs from 1513 scenes with covisible scores in the range $[0.4, 0.8]$ form the training set. We refer readers to the supplementary of [8] for more details.

3.3. Supervision

We elaborate on the generation of coarse- and fine-level supervisions in this section.

Coarse-level Supervision. Since the coarse-level LoFTR module is applied on features at 1/8 of the original image dimension, each of which represents a grid of pixels in original images, there might exist one-to-many matches. This makes it difficult to accurately determine the ground-truth coarse matches. We take the mutual nearest neighbors between the central localtions of 1/8 grids of the input images as a approximated ground-truth. In implementation, we take the central location of a 1/8 grid in left image, project it to the the same scale as the depth map, and index its depth. Based on its depth value and the known camera pose, we warp the grid center to the right image, and take its nearest neighbor as a matching candidate. The same process is repeated again from right to the left image. Based on two sets of one directional nearest neighbor matches, we only keep the mutual nearest ones as the final groud-truth. Note that we do not enforce a depth consistency check or thresholding the reprojection distances. We empirically find that further constraints on the approximated ground-truth matches leads to models with worse performance.

Fine-level Supervision. The fine-level LoFTR operates on $w \times w$ windows centering around the established coarse-level matches. For a pair of coarse-level matches, we warp the central location of the left grid to the right image, calculate its distance to its nearest neighbor, and check if it locates within the corresponding fine-level window of the right grid. We only keep those coarse matches for fine-level supervision, whose central locations of the left grids are matchable within the fine-level windows of the right grid.

3.4. Training Details

We empirically find that LoFTR is not sensitive to hyperparameters, thus we didn't extensively tune the hyperparameters for training. However, we elaborate on training details for the results exhibited in the main paper here.

MegaDepth. As mentioned in Section 3.2, we split the MegaDepth dataset into sub-scenes with covisible scores in different ranges. Based on the data sampling scheme of [2], we randomly sample 100 pairs from each sub-scene during each epoch of training, resulting in 36800 samples per epoch. The sampling scheme also balance the sample difficulties and scene variaties of each iteration, which facilitate convergence of networks. We train LoFTR for 30 epochs in total. Under a batch size of 8, the initial learning rate is set to 1×10^{-3}, with a linear learning rate warm-up in 3 epochs from 0.1 of the initial learning rate. All hyperparameters scale with batch size following the linear scaling rule [4]. We decay the learning rate by 0.5 every 8 epochs starting from the 8th epoch. The training takes one day on 16 V100 GPUs.

ScanNet. Training on ScanNet follows a similar scheme as MegaDepth. We sample 200 image pairs per scene at each

epoch and balance scene varianties for each iteration. This leads to 302400 samples for each epoch. We train LoFTR for 30 epochs, with a batch size of 64. The initial learning rate is set to 6×10^{-3}, with a linear warm up for 4800 iterations, both of which scale with the linear scaling rule. We decay the learning rate by 0.5 every 3 epochs starting from the 3rd epoch. The training takes one day on 64 1080 Ti GPUs. Note that our training scheme is much shorter, looking at about 9M samples in the entire training, than SuperGlue [8], which looks at about 360M samples.

Sampling for Fine-level Training. LoFTR is trained end-to-end from the FPN backbone until the fine-level LoFTR module. To reduce and stabilize the use of GPU memory during training, we limit the number of training samples for fine-level LoFTR to 30% of the maximum number of coarse-level LoFTR matches possible. If fewer matches are produced in the coarse-level, we pad coarse-level matches by randomly sampling from ground-truth matches. To increase the number of usable samples and ease network convergence, we make sure there is at least 200 ground-truth coarse-level matches for fine-level training, which is ignored if not much ground-truth matches exists.

Shared Training Setups. All setups not mentioned are shared for training on ScanNet and MegaDepth. For example, we use the Adam-W optimizer with weight decay factor equals to 0.1. We use gradient clipping equals 0.5 to avoid exploding gradients. We don't use dropout for regularization as usually used when training Transformers.

4. On the Naming of This Paper

During the review period, we received some suggestions on the naming of the paper. Reviewer 1 kindly suggested that the idea of "local feature" is not fully accurate and conflicts with the title "detector-free", and we should name our paper as a semi-dense matching method. After careful considerations, we decide to keep the original name. We elaborate the thinking behind this decision in the hope of communicating with the reviewer.

We consider the concept of "local feature" lies in that the information each feature vector describes is from a local region on the image. Indeed, most previous works uses the term "local feature" in the context of the detector-based feature extraction pipelines. However, we don't think "local feature" can only refer to features around the detected keypoints. Due to this reason, we believe that it's reasonable to extract *dense* local features with a CNN, which has limited receptive field and the extracted information is only related to a *local* region. In our paper, we later use Transformer to match the two sets of local features and extend the receptive field to the entire image pair. We hope that these explanations will resolve the questions regarding the naming of this paper, and we welcome Reviewer 1 to further discuss this topic with us (whether anonymously or not).

5. More Qualitative Results

More comparison of LoFTR and baselines on ScanNet and MegaDepth datasets is illustrated in Fig. 2. We also present more qualitative results on the InLoc benchmark in Fig. 3. We finally give more examples of the attention weight visualization in self- and cross-attention, as well as the transformed feature visualization through PCA in Fig. 4.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

[2] 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. *CVPR*, 2019.

[3] Mihai Dusmanu, Johannes L. Schönberger, and Marc Pollefeys. Multi-View Optimization of Local Feature Geometry. In *Proceedings of the European Conference on Computer Vision*, 2020.

[4] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[5] Xinghui Li, Kai Han, Shuda Li, and Victor Prisacariu. Dual-resolution correspondence networks. *NeurIPS*, 2020.

[6] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *CVPR*, 2018.

[7] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. 2019.

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

[9] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. InLoc: Indoor visual localization with dense matching and view synthesis. In *CVPR*, 2018.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.

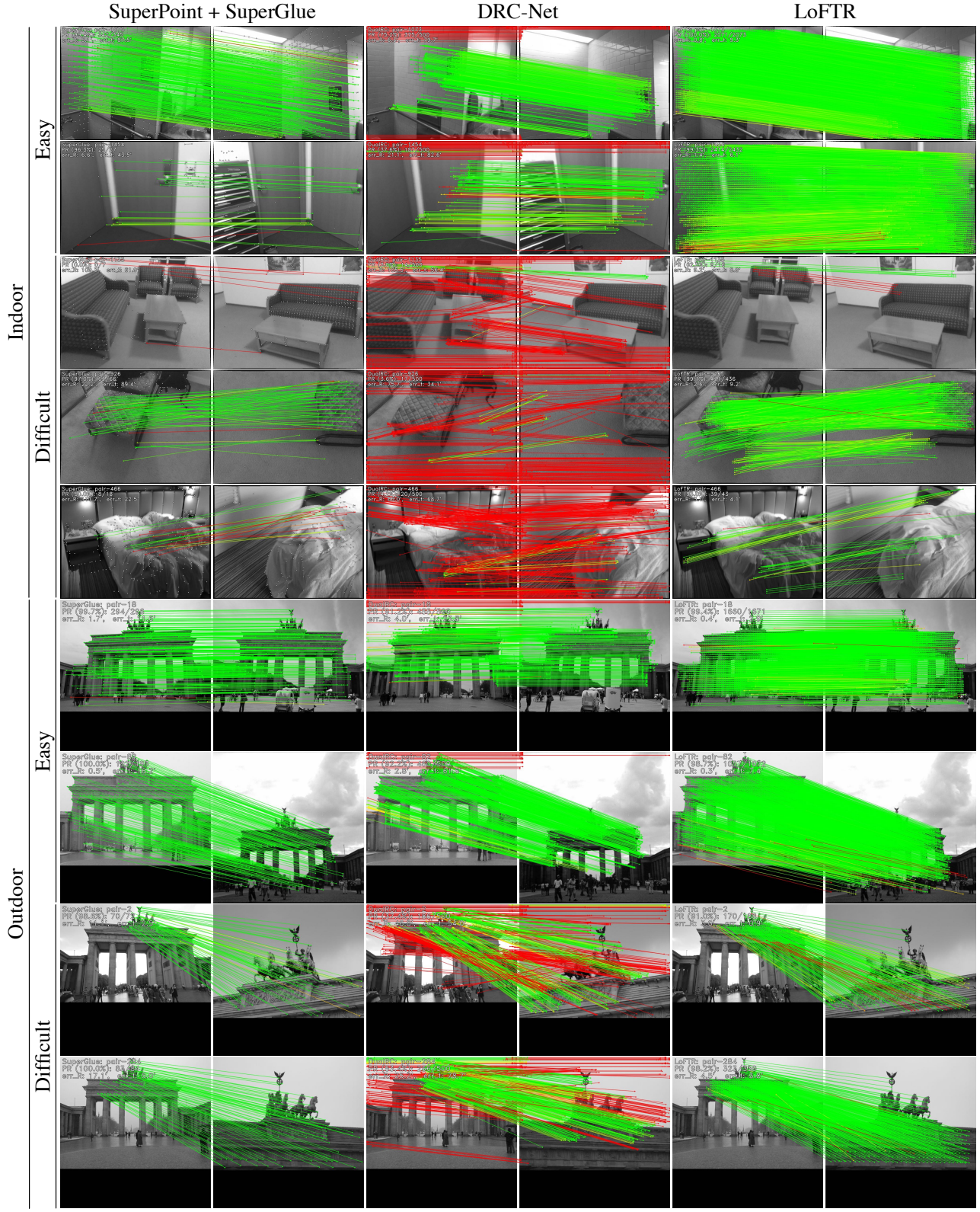

Figure 2: **Qualitative image matches on ScanNet and MegaDepth.** We report more results that compare SuperGlue [8], DRC-Net [5] and LoFTR in indoor and outdoor environments. The matching precision and pose error are printed in the upper left corner. The red color indicates a match error larger than five pixels.

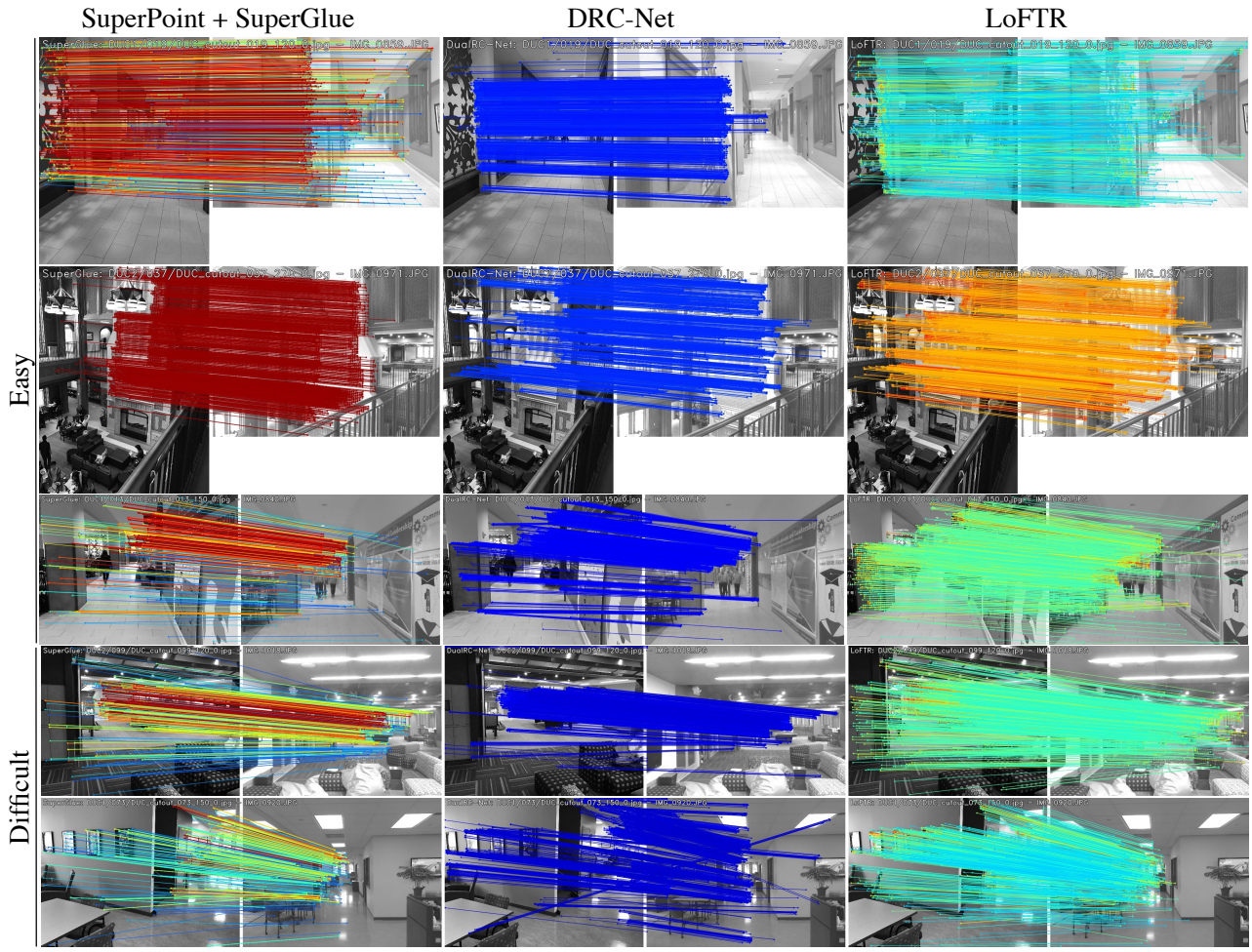

Figure 3: **Qualitative image matches on InLoc benchmark.** We report comparison results on InLoc [9] benchmark. Since no ground-truth pose is available, we color the match with predicted confidence. Red indicates higher confidence and blue for the opposite.

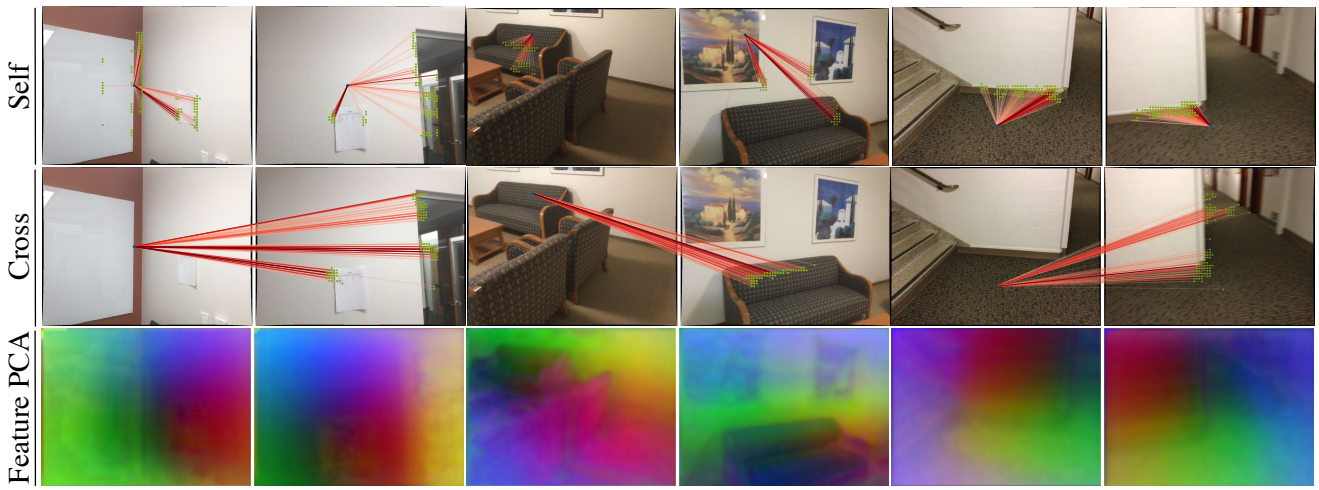

Figure 4: **Visualization of self- and cross-attention weights and the transformed features.** We visualize more self- and cross-attention results and PCA of feature representation.