A. Supplementary Appendix

A.1 Implementation Details of DCN

Since no native implementation of modulated DCN [45] is currently available in TensorFlow, we implement DCN of our own. To reduce the number of learning weights, only one set of deformation parameters are predicted and then applied along all channels, similar to the setting when `num_deformable_groups=1` in PyTorch implementation[4].

To enforce DCN with affine constraints, we follow the implementation of AffNet [22], and construct a network to predict one bounded scalar to model the scaling factor in Eq. 7, formulated as:

$$\lambda(x) = \exp\left(\tanh\left(x\right)\right). \tag{16}$$

To model the rotation, two scalars are predicted as scaled cosine and sine, which are then used to compute an angle by taking:

$$\theta(x, y) = \text{arctan2}(x, y). \tag{17}$$

To compose the affine shape matrix A', we implement the network to predict the residual shape, and enforce $\det A' = 1$ by:

$$A' = \begin{pmatrix} |1 + a''_{11}| & 0 \\ a''_{21} & |1 + a''_{22}| \end{pmatrix} \begin{pmatrix} \frac{1}{|(1+a''_{11})(1+a''_{22})|} & 0 \\ 0 & 1 \end{pmatrix}, \tag{18}$$

where a''_{11}, a''_{21} and a''_{22} lie in range $(-1, 1)$ through an $\tanh$ activation. In contrast to the observation in AffNet [22], we do not suffer degeneration when joint learning all affine parameters in DCN.

In this paper, we have concluded that the free-form DCN is a preferable choice than other deformation parameterziation subject to geometric constraints, in the context of local feature learning. However, as shown in Tab. 2, this difference is not that obvious. We ascribe this phenomenon to the lack of meaningful supervision for learning complex deformation. As also discussed in AffNet [22], a specialized loss may be needed to guide the local shape estimation, whereas in our current implementation, the same loss is used in both local feature learning and deformation learning (we have tried the loss in AffNet [22], whereas no consistent improvement has been observed). In the future, we will further explore this direction in order to better release the potential of DCN.

A.2 Implementation Details of MulDet

To implement the *multi-scale (pyramid)* variant, we follow D2-Net [7] and R2D2 [27], and feed an image pyramid to the network, which is constructed from the input image sized up to 2048, and downsampled by $\sqrt{2}$ and blurred by a Gaussian kernel factored 0.8 for each scale, until the longest side is smaller than 128 pixels. In each scale, a set of keypoints are identified whose scores are above some threshold, e.g., 0.5, and the final top-K keypoints are selected from the keypoints combined from all scales. The inference time will be doubled when enabling this configuration.

To implement the *multi-scale (in-network)* variant, we follow LF-Net [25], and resize the feature maps from the last convolution, i.e., `conv8`, into multiple scales. Specifically, the resizing is repeated for N times, resulting scales from $1/R$ to R, where $N = 5$ and $R = \sqrt{2}$. Each corresponding score map is generated as Eq. 5, then the final scale-space score map is obtained by merging all the score maps via weighted-summation, where the weight is computed from a softmax function. Since DCN has already handled the in-network scale invariance, we did not find this variant useful when combining with our methods.

To implement the *multi-level (U-Net)* variant, we build skip connections from two levels, i.e., `conv1` and `conv3`, and fuse different levels via feature concatenation. The same training scheme is applied as in the main paper, except that the keypoints are now derived from high-resolution feature maps.

A.3 Additional Experiments

Evaluation on dense reconstruction. In Sec. 4.1, we have used T&T dataset [13] to evaluate the performance in two-view image matching. Here, we resort to its original evaluation protocols defined for evaluation dense reconstruction, and integrate ASLFeat into a dense reconstruction pipeline of our own to further demonstrate its superiority.

Specifically, we use the training set of T&T, including 7 scans with ground-truth scanned models, and use *F-score* defined in [13] to jointly measure the reconstruction accuracy (precision) and reconstruction completeness (recall). For comparison, we choose RootSIFT [46], GeoDesc [19] with SIFT detector [17], and sample the features to 10K for each method. Next, we apply the same matching strategy (mutual check plus a ratio test at 0.8), SfM and dense algorithms to obtain the final dense point clouds. As shown in Tab. 6, ASLFeat delivers consistent improvements on dense reconstruction. Since T&T exhibits less scale difference, ASLFeat without the multi-scale detection yields overall best results.

Application on image retrieval. We use an open-source implementation of VocabTree[5] [47] for evaluating image retrieval performance on the popular Oxford buildings [49] and Paris dataset [48]. For clarity, we do not apply advanced post-processing (e.g., query expansion) or re-ranking meth-

[4] https://github.com/chengdazhi/Deformable-Convolution-V2-PyTorch/blob/master/modules/deform_conv.py

[5] https://github.com/hlzz/libvot

Methods	RootSIFT [46]	GeoDesc [19]	ASLFeat	ASLFeat (MS)
Barn	46.27	50.08	**55.54**	50.27
Caterpillar	50.72	48.87	**51.70**	48.88
Church	42.73	**42.93**	42.66	37.82
Courthouse	43.11	43.96	44.41	**50.39**
Ignatius	66.91	64.45	**67.77**	63.30
Meetingroom	19.89	20.39	**26.59**	25.39
Truck	67.67	67.86	70.43	**71.31**
Mean	48.19	48.36	**51.30**	49.62

Table 6. Evaluation results on T&T dataset [13] for dense reconstruction. The *F-score* is reported to quantify both the reconstruction accuracy and reconstruction completeness.

ods (e.g., spatial verification), and report the mean average precision (mAP) for all comparative methods. For fair comparison, we sample the top-10K keypoints for each method to build the vocabulary tree. As shown in Tab. 7, the proposed feature also performs well in this task, which further extends its usability in real applications.

	RootSIFT [46]	GeoDesc [19]	ContextDesc [18]	ASLFeat	ASLFeat (MS)
Oxford5k	44.94	51.77	65.31	67.01	**73.19**
Paris6k	45.83	48.15	60.79	58.01	**64.96**

Table 7. Evaluation results on Oxford buildings [49] and Paris dataset [48] for image retrieval. The mean average precision (mAP) is reported.

Integration with a learned matcher. In contrast to R2D2 [27] which strengthens the model with additional task-specific training data and data augmentation by style transfer, we explore the usability of equipping a learnable matcher to reject outlier matches for improving the recovery of camera poses. Specifically, we resort to the recent OANet [40], and train a matcher using the authors' public implementation[6] with ASLFeat. Finally, we integrate the resulting matcher into the evaluation pipeline of Aachen Day-Night dataset [30]. As shown in Tab. 8, this integration (*ASLFeat + OANet*) further boosts the localization results.

Methods	0.5m, 2°	1m, 5°	5m, 10°
ASLFeat	45.9	64.3	86.7
ASLFeat + OANet [40]	**48.0**	**67.3**	**88.8**
ASLFeat (MS)	44.9	64.3	85.7
ASLFeat (MS) + OANet	45.9	**67.3**	87.8

Table 8. Evaluation results on Aachen Day-Night dataset [30] for visual localization.

A.4 Discussions

End-to-end learning with DCN. As mentioned in Sec. 3.5, we find that a two-stage training for deformation parameters yields better results, i.e., 72.64 for MMA@3 on HPatches dataset (Tab. 2), while an end-to-end training results in 70.45. Although we have tried different training strategies, e.g., dividing the learning rate of deformation parameters by 10 during end-to-end training, none of them have shown better results than the simple separate training. It is still under exploration whether an end-to-end learning will benefit more to this learning process.

Performance regarding different feature number. In Fig. 5, we again use HPathces dataset [1], and plot the performance change (M.S. and MMA) when limiting different maximum numbers of features.

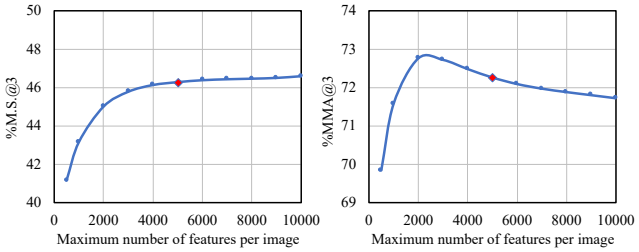

Figure 5. Matching score (M.S.) and mean matching accuracy (MMA) at an error threshold of 3px regarding different maximum numbers of features. We report the results at 5K features (marked in red) for our methods.

A.5 More Visualizations

We provide visualizations in Fig. 6 for comparing the keypoints from different local features, including SIFT, D2-Net and the proposed method.

References

[46] R. Arandjelović, and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012. 11, 12

[47] T. Shen, S. Zhu, T. Fang, R. Zhang, and L. Quan. Graph-based consistent matching for structure-from-motion. In *ECCV*, 2016. 11

[48] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*, 2008. 11, 12

[49] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007. 11, 12

[6] https://github.com/zjhthu/OANet.git

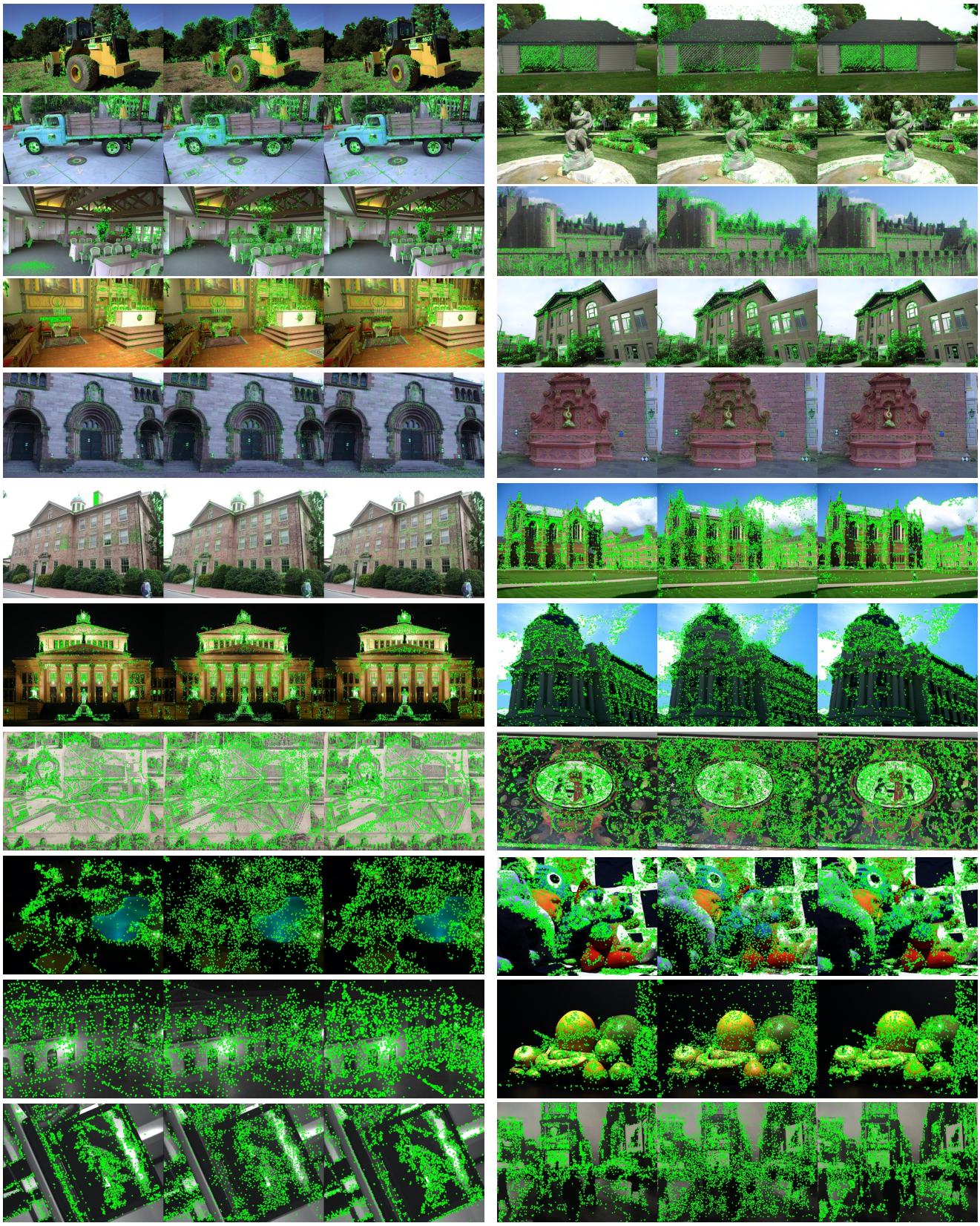

Figure 6. Comparisons of keypoints from different methods.