Supplementary of “Improving Transformer-based Image Matching by Cascaded Capturing Spatially Informative Keypoints”

Chenjie Cao, Yanwei Fu[†]
School of Data Science, Fudan University
{20110980001,yanweifu}@fudan.edu.cn

1. Implementation Details

Feature Encoder. In this section, we provide more details about the implementation of CasMTR. As mentioned in the main paper, we replace partial layers of the feature encoder with pre-trained vision transformer layers from Twins-large [3] for better generalization. Model channels are reduced to balance the computation. We list the detailed encoder architecture in Tab. 1. In particular, since the highest feature resolution of Twins is 1/4, we additionally train a CNN-based ResNet block for 1/2 features. Further, the 1/2 feature is combined to the encoder through the FPN.

Table 1: Model details of feature encoders tackling features with different resolutions (Res.). ‘LSA+GSA’ in our baseline indicate locally-grouped self-attention (LSA) and global sub-sampled attention (GSA) of Twins [3]. Feature channels are listed in brackets.

Feature Res.	Baseline	QuadTree
1/2	ResBlock(64)*2	ResBlock(128)*2
1/4	LSA+GSA(128)*2	ResBlock(196)*2
1/8	LSA+GSA(256)*2	ResBlock(256)*2

Coarse and Cascade Matching Module. We almost follow QuadTree [13] to design our coarse matching module. To relieve the computation from cascade modules, we reduce the attention block number in coarse stage from 8 to 6. As verified in our main paper, our CasMTR can still perform well with a slightly smaller coarse matching module. For cascade modules, we use 4 and 3 attention blocks for 1/4 and 1/2 features respectively. For 1/4 features, cascade modules are interlaced with ‘self-cross-self-cross’ attention blocks, while cascade modules of 1/2 features are interlaced with ‘cross-self-cross’ attention blocks. Since the self-attention is costly in high-resolution features, we tend to learn more cross-view information to make up the self one.

Progressive Training on MegaDepth. CasMTR is trained progressively from scratch on MegaDepth [8]. To ensure that cascade modules can be optimized stably with enough valid matching pairs, *i.e.*, one ground-truth match should appear in the local cascade searching space. We first train CasMTR with only $\frac{1}{8}$ resolution for 8 epochs, which can provide reliable initialization for the subsequent cascade learning. Based on the $\frac{1}{8}$ initialization, CasMTR-4c ($\frac{1}{4}$) is finetuned for 16 epochs while CasMTR-2c ($\frac{1}{4}$, $\frac{1}{2}$) converges faster with 8 epochs. The 8-epoch training of $\frac{1}{8}$ CasMTR costs about 8 hours with batch size 16. The 16-epoch training of CasMTR-4c costs about 1 day with batch size 8, while the 8-epoch training of CasMTR-2c costs about 2 days with batch size 4. All training are based on 4 32GB V100 GPUs.

Incremental Training on ScanNet. We adopt the PMT enhanced incremental tuning for CasMTR on ScanNet [4] based on the off-the-shelf QuadTree [13] ScanNet model. The finetuning of PMT-CasMTR-4c is very efficient with only 2 epochs, which costs just 16 hours for batch size 32 with 4 48GB A6000 GPUs. Note that it would take about 5 days to re-train a competitive matching model on ScanNet in [2]. Moreover, our CasMTR-4c outperforms [2].

2. Self-Attention Modules

Linear Attention (Linear) [7] works fast and efficiently for its quadratic *channel* based complexity. It is also used as the standard attention in LoFTR [11].

Locally-grouped Self-Attention (LSA) [3] is simply learned within non-overlapping local patches while the window size is set in 7 as [3].

Global Sub-sampled Attention (GSA) [14] downsamples keys and values to save the computation. LSA and GSA work complementarily in [3] for both global and local feature learning. We set the downsample rate of GSA in 4.

Simplified Top-k Attention (Top-k). As the resolution raising, the QuadTree top-k attention [13] becomes costly and infeasible. To overcome the heavy computation, here we adopt a simplified Top-k attention. To get the top-k keys and values for self-attention, we utilize the property of two-view matching. First, for each query, we match the top-1 patch from another image. Then, all matched top-1

[†]Corresponding author.

patches search for top-k patches of original images through the coarse attention results, *i.e.*, two-view cycle matching mentioned in the main paper. Therefore, all queries achieve their top-k keys and values without a global self-attention calculation. We set top-64 in this paper.

Large Kernel Attention (LKA) [6] is built with pure convolutions, which comprises depthwise convolution (DW-Conv) and dilated DW-Conv. Compared with vanilla attention, LKA also enjoys large receptive fields, but it is less sensitive on the spatial scale. The effective LKA kernel size is 21.

Patch-based OverLapping Attention (POLA) [15] can be seen as an extension of LSA. POLA magnifies receptive fields by employing overlapping windows with a larger window size for keys and values, while the query windows remain non-overlapping and have a small kernel size. Besides, POLA uses relative position encoding to further improve performance. Kernel sizes for query and key&value are 7 and 21 in POLA respectively.

3. Timing Analysis

We have listed all time and memory costs in the main paper with different input scales. Here we further analyze each component's time cost of our CasMTR and QuadTree [13] with an 832×832 image pair from MegaDepth in Tab. 2. The testing is based on a 32GB V100 GPU. From Tab. 2, the transformer-based backbone in CasMTR just works slightly slower (8.75ms) than the CNN-based backbone in QuadTree. Since we reduce attention blocks in the coarse stage from 8 to 6, CasMTR costs less time for coarse attention. Moreover, our efficient cascade attention takes about 64.95ms and 146.26ms for 1/4 and 1/2 respectively. Compared to the 1/8 coarse attention of QuadTree, our cascade attention modules can tackle high-resolution features with 4 (1/4) and 16 (1/2) times sequence length. Furthermore, we should clarify that CasMTR-4c is good enough, which already achieves significant improvement on various downstream tasks with acceptable computation. Besides, NMS simplifies the final matching pairs; and vanilla attention works more efficiently than the linear one in the patch-wise refinement. Thus our refinement's time cost is still comparable to QuadTree's.

Matching for Large Image Scales. CasMTR would not suffer from prohibitive running time even working for large-scale image matching as verified in Tab. 3. The testing is based on 1536×1536 images on V100 GPU, while QuadTree can be seen as the baseline. As shown in the table, CasMTR is still comparable in high-resolution scenes. Note that the proposed NMS filter could simplify matched pairs to less but more precise ones, which largely reduces the RANSAC time and further narrow the gap. Besides, for extremely high-resolution images, CasMTR-2c is not necessary, while CasMTR-4c (matching in 1/4 resolution) is good enough, such as in-Loc results (most images $>$ 1300pix) in Tab. 7.

Table 2: Timing measurements of CasMTR and QuadTree with 832×832 image pairs. Res. means the resolution of feature maps in this module.

Process	Res.	QuadTree (ms)	CasMTR (ms)
Feature Extraction	–	44.37	53.12
Coarse Attention	1/8	125.62	111.38
Coarse Matching	1/8	23.73	29.05
Cascade Attention	1/4	–	64.95
Cascade Matching	1/4	–	8.67
Cascade Attention	1/2	–	146.26
Cascade Matching	1/2	–	18.91
Refinement	1/2	9.14	10.80
Total	–	202.86	443.14

Table 3: Inference cost (sec/pair) on 1536×1536 image pairs compared with QuadTree [13] and CasMTR.

	QuadTree	CasMTR-4c	CasMTR-2c
Matching	1.25	1.49 (19%↑)	1.82 (46%↑)
+RANSAC	1.37	1.54 (12%↑)	1.87 (36%↑)

4. Supplemental Experiments

4.1. Ablations about Cascade Scales

We compare the CasMTR with different cascade scales on MegaDepth in Tab. 4. The baseline is the first row with only a coarse stage in 1/8. Note that when we extend our model with a more coarse initialization (1/16), CasMTR cannot achieve proper matching results. Although starting from the 1/16 coarse matching is more efficient, matching in 1/16 features is too challenging and causes inevitable errors that corrupt subsequent cascade learning.

Table 4: Ablation studies of the cascade scale without post-processing on MegaDepth.

Coarse	Cascade	Pose Estimation AUC ↑		
		@5°	@10°	@20°
1/8	–	55.77	72.01	83.64
1/8	1/4	56.34	72.11	83.55
1/16	1/8, 1/4	49.26	66.27	79.40
1/8	1/4, 1/2	**56.90**	**72.94**	**84.24**

4.2. Ablations about PMT

We compare the CasMTR with and without the Parameter and Memory-efficient Tuning (PMT) on ScanNet in Tab. 5. While without the PMT, CasMTR-4c directly utilizes frozen FPN features from the off-the-shelf QuadTree matching model [13]. CasMTR based on PMT outperforms the one without it. PMT enjoys both parameter and memory efficiency with only 0.97M trainable parameters, while the whole FPN has 5.91M trainable ones. Note that we dose not try to finetune the QuadTree FPN, because it will disturb the coarse matching initialization; and training the whole QuadTree model with coarse attention modules is very costly and unnecessary for our incremental training.

Table 5: Ablations of CasMTR-4c with/without PMT on ScanNet.

PMT	Pose Estimation AUC ↑		
	@5°	@10°	@20°
✓	**27.1**	**47.0**	**64.4**
×	26.2	46.1	63.5

4.3. Ablations about NMS on HPatches

We further compare CasMTR with different NMS kernel sizes on HPatches [1] for the homography estimation in Tab. 6. First, both CasMTR-4c and CasMTR-2c without NMS outperform QuadTree. Especially, NMS kernels 5 and 9 perform best for CasMTR-4c and CasMTR-2c respectively. And our methods achieve impressive results with only 400 to 500 averaged matches, which are much fewer than QuadTree's. We think that the keypoint location is important for homography estimation, while the proposed NMS detection can work properly for it. Note that the setting of NMS kernel 5 of the relative pose estimation is still competitive on HPatches, which shows a good generalization of NMS.

Table 6: Ablation studies about NMS kernel size (k) on HPatches.

Methods	NMS	Pose Estimation AUC			matches
		@3px	@5px	@10px	
QuadTree	–	66.37	76.23	84.97	2749
CasMTR-4c	–	67.50	77.10	86.25	11439
CasMTR-4c	k=3	67.71	77.45	86.16	923
CasMTR-4c	k=5	**69.71**	**78.81**	86.96	400
CasMTR-4c	k=7	69.70	78.76	**87.01**	212
CasMTR-2c	–	69.06	78.47	86.75	44754
CasMTR-2c	k=3	70.11	79.15	87.30	3520
CasMTR-2c	k=5	70.35	79.60	87.59	1477
CasMTR-2c	k=7	70.89	79.68	87.73	778
CasMTR-2c	k=9	**71.43**	**80.20**	**87.91**	507
CasMTR-2c	k=11	71.19	79.92	87.69	351

4.4. Qualitative Ablations about NMS Kernel

We show some qualitative samples of CasMTR-4c with different NMS kernels in Fig. 1. We find that NMS detection is very effective for two types of image pairs. The first type is image pairs with large displacements (rows 2,3,4 of Fig. 1). These pairs are difficult for matching algorithms to achieve precise matches; and denser predictions usually cause more incorrect matches, which discourage the performance. So detecting local informative keypoints through the NMS is critical for improvement. On the other hand, image pairs with very limited displacements are also challenging (row 1 of Fig. 1), which may cause large translation errors. For these image pairs, NMS detection is also useful to achieve superior performance with more accurate keypoint matches.

Table 7: Visual localization results on InLoc [12]. * means our implementation of LoFTR; note that results of our implementation are better on DUC1 and worse on DUC2 than those reported in [11].

Methods	DUC1	DUC2
	(0.25m,2°) / (0.5m,5°) / (1m,10°)	
HLoc [9]+LoFTR [11]*	49.5/73.7/82.8	**51.9**/69.5/80.9
HLoc [9]+Baseline	47.5/71.7/83.8	48.1/**70.2**/79.4
HLoc [9]+CasMTR-4c	51.0/75.3/84.3	51.1/69.5/**83.2**
HLoc [9]+CasMTR-2c	50.5/74.7/83.8	49.6/**70.2**/**83.2**
HLoc [9]+CasMTR-4c-NMS5	**53.5**/76.8/85.4	**51.9**/**70.2**/**83.2**
HLoc [9]+CasMTR-2c-NMS5	53.0/**77.8**/**86.4**	49.6/**70.2**/82.4

4.5. More Ablation Studies on InLoc

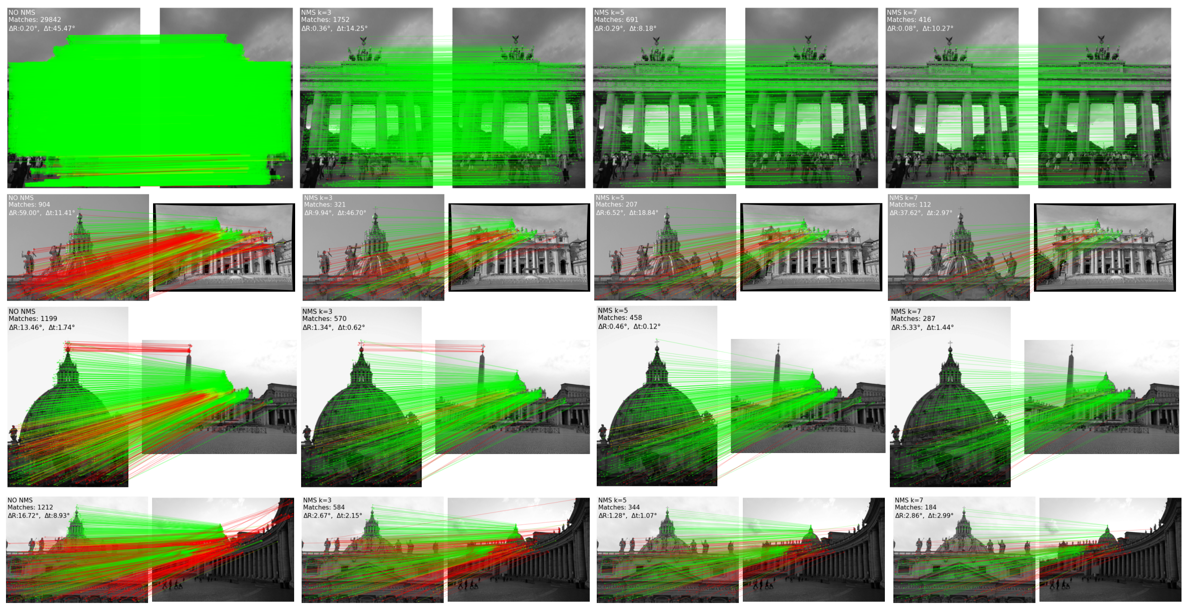
We compare CasMTR-4c and CasMTR-2c with/without NMS (kernel size 5) on InLoc [12] in Tab. 7. And we did not further tune the kernel size of NMS on InLoc to ensure the fairness. CasMTR-2c achieves better results than CasMTR-4c on DUC1, but it performs worse on DUC2. Taking the trade-off of efficiency and performance into consideration, we adopt CasMTR-4c as our final solution in the main paper.

4.6. Matching in Extremely Low Resolutions

Learning the capability for extremely low-resolution matching is interesting because we could not always guarantee access to high-quality images. Results are shown in Tab. 8. We further compare results from SuperPoint [5]+SuperGlue [10]. The performances of both detector-based and detector-free methods are dramatically degraded in extremely low-resolution matching. However, our CasMTRs enjoy better robustness. As the resolution is reduced, the advantages of our algorithm become more apparent, especially for the CasMTR-2c. Note that for 256×256, the matching is very challenging; and our method enjoys about 120%, 82%, and 50% improvements on AUC5°, AUC10°, and AUC20° compared to QuadTree.

4.7. Insights about CasMTR

In this section we further discuss some additional insights about the proposed CasMTR. As shown in Fig. 2, matching in the coarse stage (1/8) usually suffers from some inevitable deviations. In particular, large displacements of viewpoint and occlusions break the rule that a local patch in source view (yellow points in Fig. 2(a)) should be matched to a patch



(a) CasMTR-4c No NMS (b) CasMTR-4c NMS k=3 (c) CasMTR-4c NMS k=5 (d) CasMTR-4c NMS k=7

Figure 1: Qualitative comparisons of CasMTR-4c among different NMS kernels on MegaDepth. Please zoom-in for details.

Table 8: Matching for image pairs with extremely low resolutions.

Methods	640×640			512×512			256×256		
	AUC5	AUC10	AUC20	AUC5	AUC10	AUC20	AUC5	AUC10	AUC20
QuadTree	49.86	66.85	79.43	44.06	61.35	75.15	10.42	22.04	38.22
SuperGlue	27.55	44.43	61.63	18.46	33.60	51.31	1.64	5.11	13.59
CasMTR-4c	51.11	67.76	80.49	47.07	64.21	77.81	12.70	26.77	44.64
CasMTR-2c	**54.98**	**71.48**	**83.11**	**51.36**	**68.08**	**80.78**	**23.38**	**40.24**	**57.11**

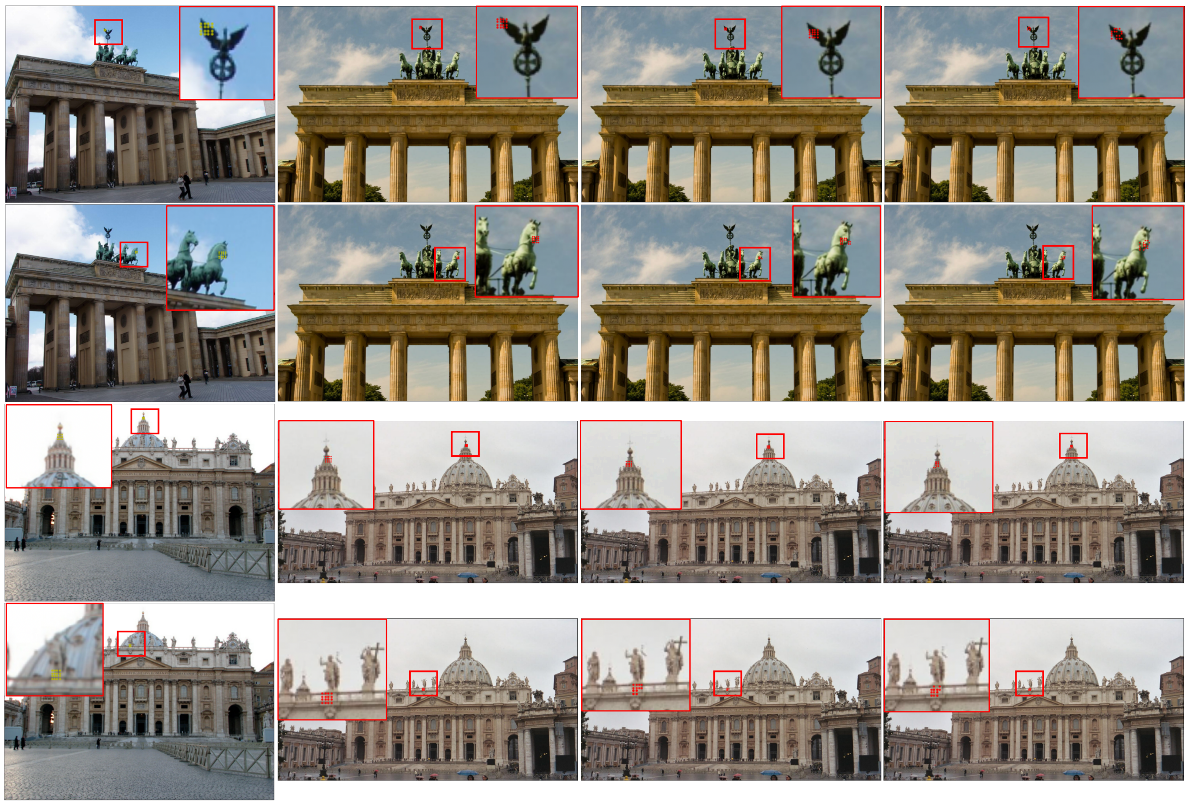
with the same size in target view (red points in Fig. 2(b)). Rather than trivially searching for the nearest neighbor, our CasMTR can gradually refine all matched points of the target view to more exact locations. Thus CasMTR further improves the pose estimation with lower detailed error.

4.8. More Qualitative Comparisons

More qualitative comparisons for MegaDepth [8] and ScanNet [4] are shown in Fig. 3 and Fig. 4 respectively. From Fig. 3, CasMTR-2c outperforms LoFTR [11] and QuadTree [13], while NMS can further improve the results. From Fig. 4, we find that denser matches without NMS are more suitable for ScanNet images with textureless regions and limited resolutions. Besides, note that some sparse matches are filtered by the NMS (row2 of Fig. 4). Because our NMS is only based on the densest confidence map (1/4 of ScanNet). Thus keypoints detected by the NMS may have low-confident scores in the frozen coarse stage, so these points would be eliminated by the confidence threshold (all stages' confidence thresholds in our work are fixed in 0.2). Moreover, qualitative results on HPatches [1] are shown in Fig. 5. And we also provided some results from InLoc [12] of the visual localization task in Fig. 6. Note that the ground truth of InLoc is not provided. So we colorize the matches with model confidence. It seems that our results of Fig. 6(c,e) have lower confidence. Because the results of CasMTR are much denser than LoFTR and baseline. Thus many low-confident matches are remained to cover the high-confident ones. Moreover, our NMS can successfully detect keypoints with locally high confidence, which improves the performance as in Tab. 7.

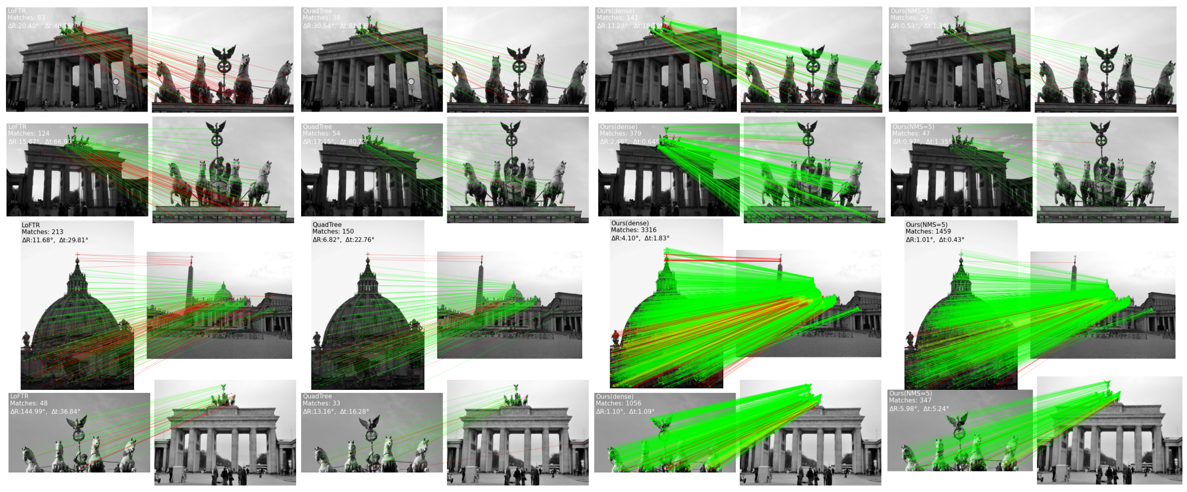
5. Limitation

The proposed CasMTR can achieve good performance in various matching downstream tasks. Although CasMTR-4c with 1/4 feature maps enjoys impressive enough results, CasMTR-2c with 1/2 high-resolution features can further improve the results in most situations. So we think that learning high-resolution attention modules is still necessary for image matching. Though we have made lots of efforts to improve the efficiency, learning in 1/2 features is still



(a) Source view (b) Target view (1/8 match) (c) Target view (1/4 match) (d) Target view (1/2 match)

Figure 2: The effectiveness of each cascade stage from CasMTR. Cascade modules can correctly refine the dense matching results rather than trivially searching for the nearest neighbor. Please zoom-in for details.



(a) LoFTR (b) QuadTree (c) CasMTR-2c (dense) (d) CasMTR-2c (NMS=5)

Figure 3: Qualitative results compared on MegaDepth [8]. Please zoom-in for details.

very challenging as shown in Tab. 2. Therefore, improving the efficiency of the high-resolution feature correlation learning for attention modules should be an interesting future work. On the other hand, NMS fails to be generalized well on texture-less indoor scenes with a frozen coarse stage. Therefore, we consider it as future work to integrate both

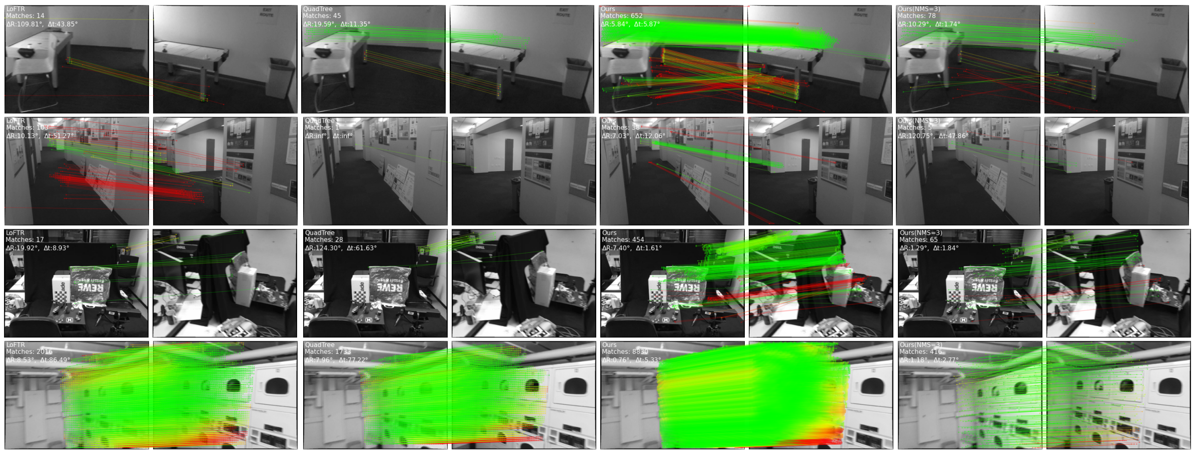

(a) LoFTR (b) QuadTree (c) CasMTR-4c (dense) (d) CasMTR-4c (NMS=3)

Figure 4: Qualitative results compared on ScanNet [4]. Please zoom-in for details.

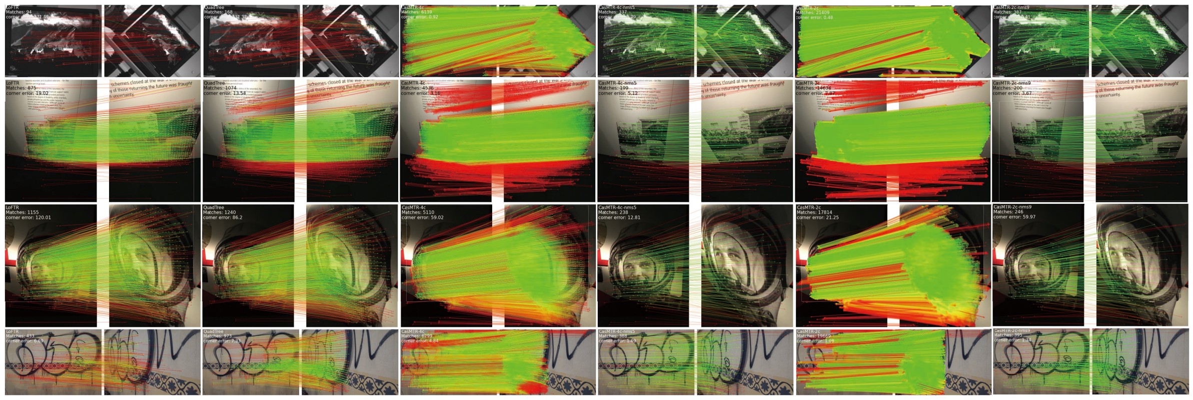

(a) LoFTR (b) QuadTree (c) CasMTR-4c (d) CasMTR-4c (nms5) (e) CasMTR-2c (f) CasMTR-2c (nms9)

Figure 5: Qualitative results compared on HPatches [1]. All image pairs are resized to meet that the short side is 480. We also show the corner error of each instance. The matching color threshold is 2-pixel. Please zoom-in for details.

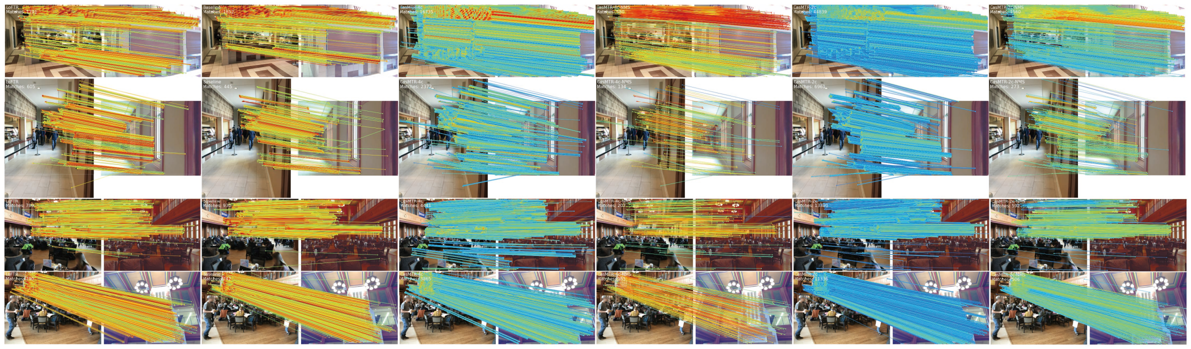

(a) LoFTR (b) Baseline (c) CasMTR-4c (d) CasMTR-4c-NMS5 (e) CasMTR-2c (f) CasMTR-2c-NMS5

Figure 6: Qualitative results compared on InLoc [12]. All image pairs are resized to meet that the short side is 1024. We colorize the matches with model confidence. Red means certain while blue means uncertain. Please zoom-in for details.

trainable and un-trainable confidence for NMS detection in challenging scenes.

References

[1] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of hand-crafted and learned local descriptors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5173–5182, 2017. 3, 4, 6

[2] Hongkai Chen, Zixin Luo, Lei Zhou, Yurun Tian, Mingmin Zhen, Tian Fang, David Mckinnon, Yanghai Tsin, and Long Quan. Aspanformer: Detector-free image matching with adaptive span transformer. *arXiv preprint arXiv:2208.14201*, 2022. 1

[3] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing

Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *Advances in Neural Information Processing Systems*, 34:9355–9366, 2021. 1

[4] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 1, 4, 6

[5] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018. 3

[6] Meng-Hao Guo, Cheng-Ze Lu, Zheng-Ning Liu, Ming-Ming Cheng, and Shi-Min Hu. Visual attention network. *arXiv preprint arXiv:2202.09741*, 2022. 2

[7] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020. 1

[8] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2041–2050, 2018. 1, 4, 5

[9] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12716–12725, 2019. 3

[10] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020. 3

[11] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8922–8931, 2021. 1, 3, 4

[12] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7199–7209, 2018. 3, 4, 6

[13] Shitao Tang, Jiahui Zhang, Siyu Zhu, and Ping Tan. Quadtree attention for vision transformers. *arXiv preprint arXiv:2201.02767*, 2022. 1, 2, 4

[14] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 568–578, 2021. 1

[15] Shiyu Zhao, Long Zhao, Zhixing Zhang, Enyu Zhou, and Dimitris Metaxas. Global matching with overlapping attention for optical flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17592–17601, 2022. 2