In the supplementary material, we present more details on the implementation and experimental settings. We also provide detailed performance data, analysis of anchor points and learned attention weights, and more qualitative and quantitative results on geometry matching, visual localization, and other tasks.

I. Implementation Details

a) Feature Extractor: The feature extractor consists of a backbone based on ResNet [1] and an upsampling module. We adopt ResNet-18 as the backbone which contains one convolutional layer, one batch norm layer, and an activation layer, followed by three residual blocks. We remove the max pooling layer from the original ResNet-18 to ensure that the output feature maps are $\frac{1}{2}$ and $\frac{1}{8}$ of the input image size. The upsampling module performs upsampling in two stages. In each stage, we first upsample the input feature map by a scale of 2, then concatenate it with the corresponding output of the backbone model and finally feed the concatenated feature to a refinement block. The detailed model architecture is illustrated in Fig. 1.

b) Training Strategy: We use MegaDepth [2] and ScanNet [3] dataset for training. For MegaDepth, we use the preprocessed data from [4], and for ScanNet, We use the original data with the training image pairs from [5]. Since the preprocessed MegaDepth data from [4] does not contain ground truth depth maps, we get the original dataset from the MegaDepth project page [1]. Since the depth maps from the original dataset are size-free, we crop and resize them to match the preprocessed images. Then we preprocess the images using the method proposed in CAPS [4]. For each image with an arbitrary size, we extract the largest rectangle in the center of the image with an aspect ratio of $4:3$. We then crop the image using the rectangle and scale it to 640×480.

During training, given a pair of input images I_a and I_b, we first generate the visibility map for I_a and then calculate the visibility of each pixel by reprojecting it to I_b and comparing the reprojected point's depth with the corresponding depth from I_b. We consider a pixel in I_a is visible in I_b if and only if the relative depth error ($\frac{d-d_{gt}}{d_{gt}}$) is no more than a threshold (we set 0.1 for MegaDepth and 0.2 for ScanNet). Based on the visibility map, we uniformly sample $10 \times N$ anchor points candidates for training. Then we apply a grid filter to further select N anchor points which are evenly distributed across the image. Specifically, we divide the image to $4^{\lfloor \log_4 N \rfloor}$ grids of the same size. In each pass, we select one random candidate from each grid (if existing). We repeat the same procedure until N points are selected.

Our model is completely trained from scratch. The learning rate is initialized to 0.0001, and reduced by half after each 50K iterations. We first train the model for 120K iterations using the ground truth anchor points. Then we replace the ground truth anchor points with the output of SuperGlue [6] and fine-tune for another 20K iterations. The fine-tuning process narrows down the gap between anchor points from ground truth and SuperGlue matches especially when the matching predicted by SuperGlue is not highly reliable. It improves the accuracy of pose estimation of ScanNet by approximately 4 points, while does not influence the result of MegaDepth, because SuperGlue performs much better on MegaDepth. When testing, we limit the number of input anchor points to 500.

We use PyTorch [7] to train our model on a single NVIDIA Tesla V100 with batch size 5, which requires approximately 28 GB memory. The total training time is about 36 hours.

II. Experimental Details

In the evaluation of geometric matching, our goal is to evaluate the average quality of correspondences of all pixels between image pairs. During the experiment in the main paper, we randomly sample visible points between image pairs and evaluate them only, to increase the evaluation efficiency. When evaluating our model on HPatches [8], we divide the image into grids of 16×16 size, and choose all grid centers which have correspondences on the other image (calculated by their homography). As for MegaDepth [2], we uniformly sample 500 visible points based on their visibility map. We apply the sampling strategy for both images and combine the selected points as our query points. We do not filter out any predicted correspondences in this experiment and evaluate all of them by the $l2$-distance to the ground truth correspondences.

In the ablation study, all of the tested models are trained with ground truth anchor points, and all query points are generated from SIFT [9]. For every image pair, we filter out all the correspondences with the cycle consistency larger than 10 pixels and finally select the top 2000 matches as the output.

III. Performance Data

We detail the time and memory cost of each component in our model. We test on 200 image pairs with size 640×480, where 2000 query points are sampled for each image pair. We divide our inference pipeline into multiple stages and record the time and memory cost for each stage. The memory cost is measured by the following steps: at the end of each stage, we first query the total cached memory in GPU (pre-memory). Then we clear all the GPU cache and query the memory usage again (post-memory). We report the pre-memory of each stage as the total memory and the difference between the pre-memory of this stage and the post-memory of the previous stage as the net memory of this stage. We show the time and memory cost in Tab. I.

We show that the cost of our message-passing is much less in terms of time and memory compared to the total cost while generating local features from the backbone and predicting the final correspondences being the bottleneck of the full pipeline. By further optimizing these procedures, we can make our model even more lightweight and efficient.

We also compare our model with two state-of-the-art works COTR [10] and DualRC-Net [11]. All the models are evaluated on an NVIDIA GeForce GTX 1080 Ti using 20 test image

[1] https://www.cs.cornell.edu/projects/megadepth/

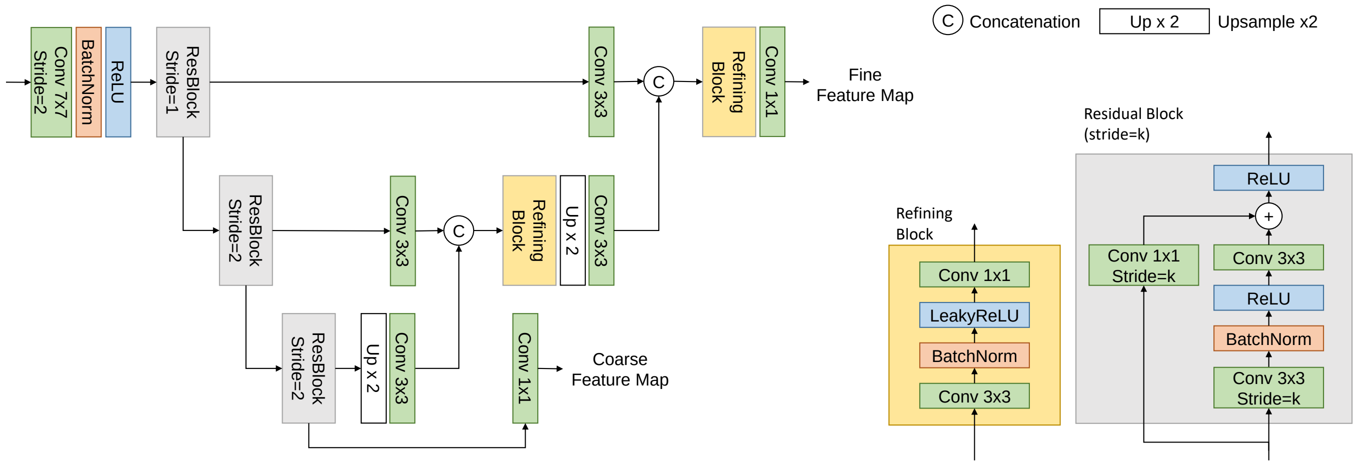

Fig. 1. Overview of the feature extractor. The left shows the overall model architecture, and the right shows details of the refinement block and the residual block.

TABLE I

TIME AND MEMORY COST OF EACH STAGE. EACH COLUMN FROM TOP TO BOTTOM INDICATES CUMULATIVE TIME, NET TIME, TOTAL MEMORY AND NET MEMORY FOR EACH STAGE. EACH ROW FROM LEFT TO RIGHT REPRESENTS THE STAGES INCLUDING EXTRACTING LOCAL FEATURES FROM THE BACKBONE AND THE FEATURES OF ANCHOR POINTS BY INTERPOLATION, PROPAGATING THROUGH INTRA-POINTS, INTER-POINTS, AND POINTS-TO-IMAGE MESSAGE-PASSING LAYERS, REFINING THE FEATURES AND GENERATING THE FINAL CORRESPONDENCE OUTPUT.

	Feature Extraction		**Propagation**		**Refinement**	**Output**
	Local features	Anchor point features	Intra- and Inter-points	Points-to-image	Refined features	Final correspondence
Total time(ms)	26	58	74	82	87	173
Net time(ms)	26	32	16	8	5	86
Total mem(GB)	1.316	0.490	0.493	0.581	1.062	1.485
Net mem(GB)	1.006	0.090	0.092	0.174	0.578	0.634

pairs with 2000 points queried for each pair. The result are shown in Tab. II. By comparing the runtime and the memory for different resolutions, we find that our method significantly outperforms in the high-resolution setting. Note that COTR [10] always resizes and crops the input image to 256×256 no matter how large the input is.

IV. More Experimental Analysis

a) Additional Ablation Study on HPatches: In addition to our ablation study in the main paper, which uses MegaDepth as testing data, we also provide ablative results tested on HPatches [8]. We test our model with three variants: *model w/ DS* downsamples input images to 640*480, *model w/ SIFT* uses anchor points matched by FLANN on SIFT feature, and *model w/o APE* removes the adaptive position embedding during the training. Results are shown in Tab. III. The score of first two variants shows that our model is robust to low resolution of inputs and unreliable anchor points. Although the anchor points by SIFT are highly unreliable to estimate accurate correspondence, our model with such sparse prior is able to generate a dense feature map and achieve comparable matching accuracy with the one using SuperGlue [5]. The last one proves the effectiveness of the proposed adaptive position embedding.

b) Oracle Anchor Points: In this main paper, we have shown that our model is robust to the quantity and quality of anchor points to some degree in Sec 4.4. However, the quality improvement of anchor points can help boost the model performance. To verify this, we conduct a study of oracle anchor points that indicate very confident correspondence prior. At inference, we progressively improve anchor points generated by SuperGlue [5] by replacing a certain number of the anchor points using new anchor points randomly sampled from the ground truth correspondence.

We show the quantitative results on the task of indoor pose estimation in Tab. IV with different numbers of ground truth anchor points (*i.e.* 20, 50, 100). We also demonstrate the qualitative results of extremely difficult cases where SuperGlue [5] almost fails to predict correct correspondences (see Fig. 2). Our model has shown great potential to produce more accurate correspondences given more confident anchor points as input, even in the very challenging application scenarios.

c) Attention Weights: In the intra-points message-passing layer and points-to-image message-passing layer, we use the attention mechanism to propagate information across all intra-

TABLE II

PERFORMANCE COMPARISON. THE LEFT TWO COLUMNS ARE TIME AND MEMORY COSTS TESTED WITH LOW RESOLUTION INPUTS (480×640) AND THE RIGHT TWO COLUMNS ARE RESULTS WITH HIGH RESOLUTION INPUTS (1200×1600). FOR FAIRNESS, WE ALSO REPORT THE SUMMED UP COST OF OUR MODEL AND THE PROCESS OF GENERATING ANCHOR POINT FROM SUPERGLUE [6].

Resolution/Metrics	480×640/Time	480×640/Memory	1200×1600/Time	1200×1600/Memory
DualRC-Net [11]	0.25s	0.77G	4.34s	8.70G
COTR [10]	200s	5.45G	200s	5.45G
Ours	0.18s	1.17G	1.11s	5.69G
Ours(w/ SuperGlue [6])	0.18s+0.11s=0.29s	1.17G (0.21G)	1.11s+0.27s=1.38s	5.69G (1.00G)

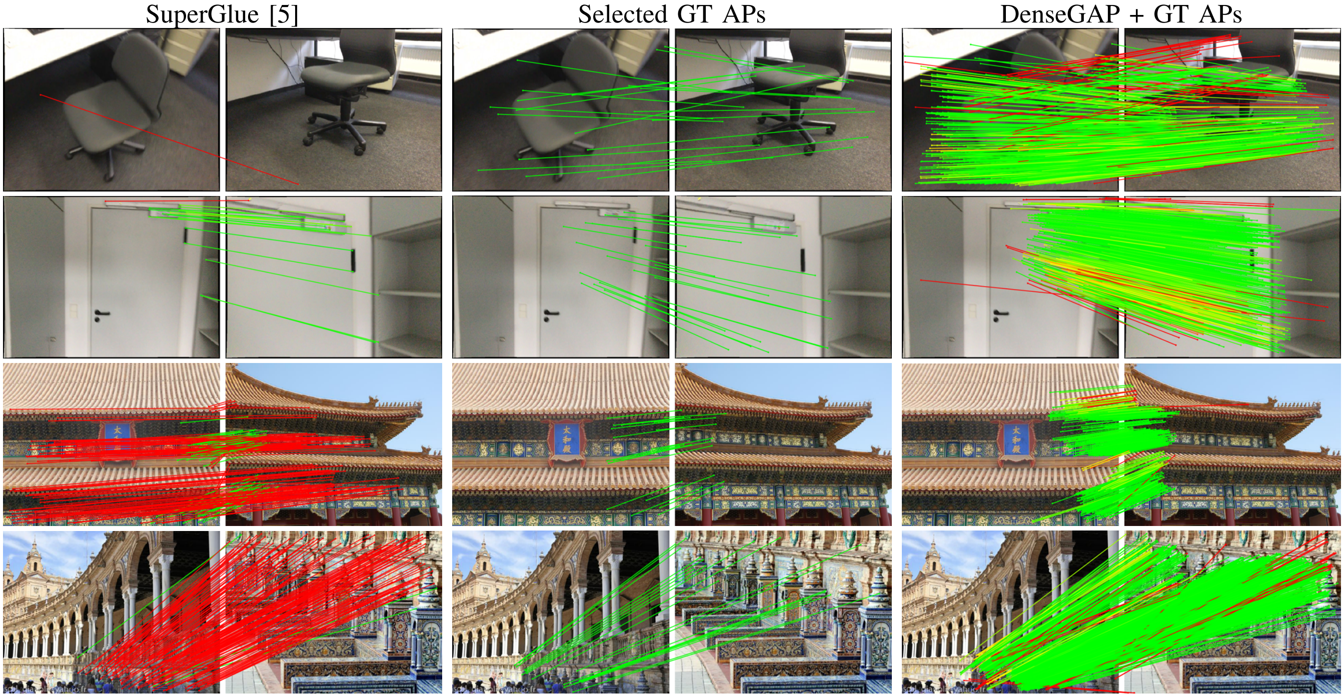

Fig. 2. Qualitative results of oracle anchor points. From left to right: results of SuperGlue [5]; 20 randomly sampled ground truth anchor points; results of DenseGAP with 20 ground truth anchor points (GT APs) as input. We show top 500 correspondences and top 1000 correspondences in indoor scenes and outdoor scenes respectively using our model. The correspondences are colored by their epipolar errors calculated based on ground truth relative poses (green means inliers, and red means outliers). We set the error threshold to 1×10^{-3} for both indoor and outdoor scenes.

TABLE III

ABLATION STUDY ON HPATCHES. WE REPORT THE MMA UNDER THE THRESHOLD OF 1, 3, 5 AND 10 PIXELS. DOWNSAMPLING INPUTS OF OUR MODEL WILL SLIGHTLY CHANGE THE PERFORMANCE IN LOW THRESHOLD BUT WILL NOT CAUSE MAJOR EFFECT IN LARGE THRESHOLD; USING ANCHOR POINTS FROM SIFT FEATURES HAS SOME DRAWBACKS BUT THE RESULTS ARE STILL COMPETITIVE TO THE FULL MODEL; AND REMOVING ADAPTIVE POSITIONAL EMBEDDING CAN GREATLY DEGRADE THE PERFORMANCE OF OUR MODEL.

Method	MMA(1px)	MMA(3px)	MMA(5px)	MMA(10px)
Full Model	**0.565**	**0.898**	**0.958**	**0.979**
Model w/ DS	0.505	0.871	0.945	0.975
Model w/ SIFT	0.533	0.845	0.903	0.927
Model w/o APE	0.384	0.731	0.833	0.909

TABLE IV

INDOOR POSE ESTIMATION EVALUATION ON SCANNET [3] DATASET USING GROUND TRUTH ANCHOR POINTS. THE ASTERISK SUPERSCRIPT (*) INDICATES THE MODEL IS TRAINED USING 50 GROUND TRUTH ANCHOR POINTS AND NOT FINE-TUNED ON SUPERGLUE [5] WHILE DENSEGAP REPRESENTS THE MODEL WITH FINE-TUNING. NOTE THAT THE MODELS WITHOUT FINE-TUNING ARE GENERALLY BETTER THAN THOSE WITH FINE-TUNING DUE TO THE SMALLER GAP BETWEEN THE ANCHOR POINTS USED FOR TRAINING AND TESTING WHEN INTRODUCING GROUND TRUTH ANCHOR POINTS.

Method	AUC(5)	AUC(10)	AUC(20)
DenseGAP+SuperGlue	17.01	36.07	55.66
DenseGAP+20 GT AP	16.11	34.21	54.06
DenseGAP+50 GT AP	19.14	38.78	58.41
DenseGAP+100 GT AP	21.35	41.35	60.68
DenseGAP*+20 GT AP	17.79	37.07	56.84
DenseGAP*+50 GT AP	29.15	49.24	67.44
DenseGAP*+100 GT AP	**34.62**	**55.41**	**71.81**

image edges. We visualize the attention weights in Fig. 3. Since all the intra-points message-passing layers have similar attention patterns, here we show the last layer only. Each layer contains 4-head attention, and we choose the first two heads for visualization. We observe that the attention in the points-to-image layer is sparser and more condensed than intra-points layers, mostly attending to a small neighborhood around it.

TABLE V

Visual localization evaluation on InLoc [2] dataset. We evaluate our model using the pipeline proposed by HLoc [6].

Method	Duc1	Duc2
	(0.25m,10°) / (0.5m,10°) / (1.0m,10°)	
HL [6]+SP [12]+SuperGlue [5]	46.5 / 65.7 / 77.8	**51.9** / 72.5 / **79.4**
HL [6]+DenseGAP	**48.5 / 69.2/ 81.8**	48.9 / **74.0** / **79.4**

TABLE VI

Pose Estimation Evaluation on the ScanNet dataset. The * indicates the model trained on MegaDepth [2].

Method	AUC(5)	AUC(10)	AUC(20)
ORB [13]+GMS [14]	5.21	13.65	25.36
D2-Net [15] +NN	5.25	14.53	27.96
ContextDesc [16]+Ratio [9]	6.64	15.01	25.75
DualRC-Net [11]*	6.94	17.06	29.58
SP [12]+SuperGlue [5]	16.16	33.81	51.84
LofTR [17]	22.06	40.8	57.62
DenseGAP	17.01	36.07	55.66
DenseGAP+100 GT AP	**34.62**	**55.41**	**71.81**

TABLE VII

Results on KITTI flow benchmark. We test our model on the 2012 and 2015 version of the KITTI dataset, and reports the Average End Point Error (AEPE) and the flow outlier ratio ('Fl'). Note that our model is about three magnitudes faster than COTR [10] as shown in Tab. II.

Method	KITTI-2012		KITTI-2015	
	AEPE↓	Fl.[%]↓	AEPE↓	Fl.[%]↓
GLU-Net [20]	3.34	18.93	9.79	37.52
GOCor [21]	2.68	15.43	6.68	27.57
COTR [10]	**1.28**	7.36	**2.62**	**9.92**
Ours	1.69	**6.21**	3.47	10.8

V. More Results

a) Indoor Pose Estimation: In our paper, we compare our model with two state-of-the-art methods: SuperGlue [5] and DualRC-Net [11] on the task of pose estimation. In Tab. VI we further provide results of other methods tested on the ScanNet [3] dataset. We notice that one recent work LofTR [17] achieved better result compared to ours on this task. However, as shown in the last row of the table (and Sec.IV-0b), our model has great potential to improve when combined with better anchor points.

b) Geometric Matching: In addition to the PCK evaluation shown in the main paper (Sec. 4.2) where we use randomly sampled points as queries, we evaluate pixel-wise dense correspondence, by generating correspondences for all pixels in the query image (a.k.a. a dense flow). It takes about 10 seconds on an NVIDIA GTX 1080 Ti to generate a dense flow for a 480×640 image pair. Most of the time is consumed by memory allocation. The time cost can be reduced to about 2 seconds using a graphic card with larger memory (*e.g.* NVIDIA Tesla V100). We show the qualitative results in Fig. 4. By observing the confidence map in the $5th$ column computed using the cycle consistency we define in the main paper (Sec. 3.4), we find that our model can predict highly confident correspondences for most of the visible pixels (*e.g.* pixels in blue) while less confident matching usually occurs in the occluded region (*e.g.* pixels in black). Compared to the baseline method, our approach can generate better quality results with fewer distortion artifacts and preserve the building structures consistent with the query image.

c) Optical Flow: KITTI flow benchmark [18], [19] is one of the major evaluations used in many works of dense correspondence. We test our model on this evaluation and compare it with other state-of-the-art methods, including the concurrent work COTR [10]. As our model also generates confidence for each prediction, we follow the same way by COTR [10] to ensure a fair comparison. Same as COTR [10], we randomly sample multiple points per image pair and evaluate only on points where our method returns confident results from these query points (whose error is under the threshold of 5 pixels). For reference, we also compare with the state-of-the-art dense correspondence methods (GLU-Net [20] and GOCor [21]) in Tab. VII, with the number from COTR [10]. All of the results are shown in Tab. VII. Our model achieves competitive results with COTR [10], while our inference speed is about three magnitudes faster than COTR [10] as shown in Tab. II.

d) Visual Localization: We also evaluate our model on the task of visual localization, which aims to estimate 6 DoF poses of the input images given the 2D to 3D correspondence between an image database and its corresponding 3D models. By matching each input image to the database, we can establish the correspondence between the input image and 3D models, thus predicting the absolute camera pose of the input. We test our model using the Long-Term Visual Localization Benchmark [22] on the popular InLoc dataset which contains challenging indoor scenes with lots of textureless regions and also provides ground truth 3D scene models for evaluating the localization. To implement the full pipeline of the visual localization, we adopt the public repository of Hierarchical Localization [6], by only replacing its matching module with our matching results. We use grid sampling to generate query points from the image in the database, then find its corresponding points in each input image, and finally, use the estimated correspondence to compute the camera pose. We show the quantitative result in Tab. V. Our model outperforms the baseline method based on the matches generated by SuperGlue in the same setting where matches less than 20 are skipped.

e) More Qualitative Comparisons: Finally, we showcase more qualitative results on MegaDepth, ScanNet, and HPatches, in Fig. 5, Fig. 6, and Fig. 7 respectively.

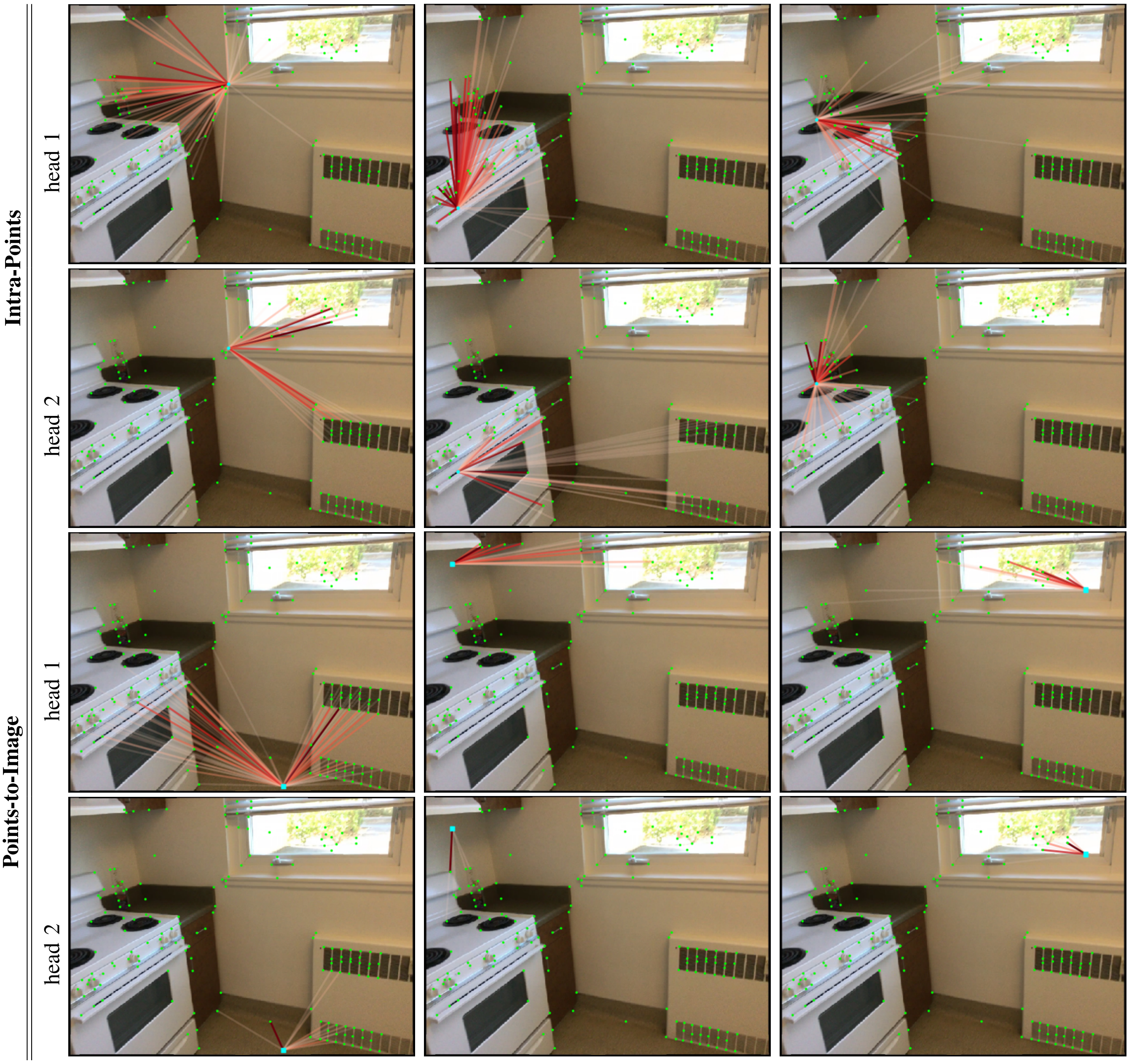

Fig. 3. Visualization of the attention weights in intra-points and points-to-image message-passing layers. We visualize the attention for three target points in each row. The $1st$ and $2nd$ rows share the same target points, and the $3rd$ and $4th$ rows share the same target points. Note that the attention in each head have different patterns where different heads attend to points which have different spatial location distributions.

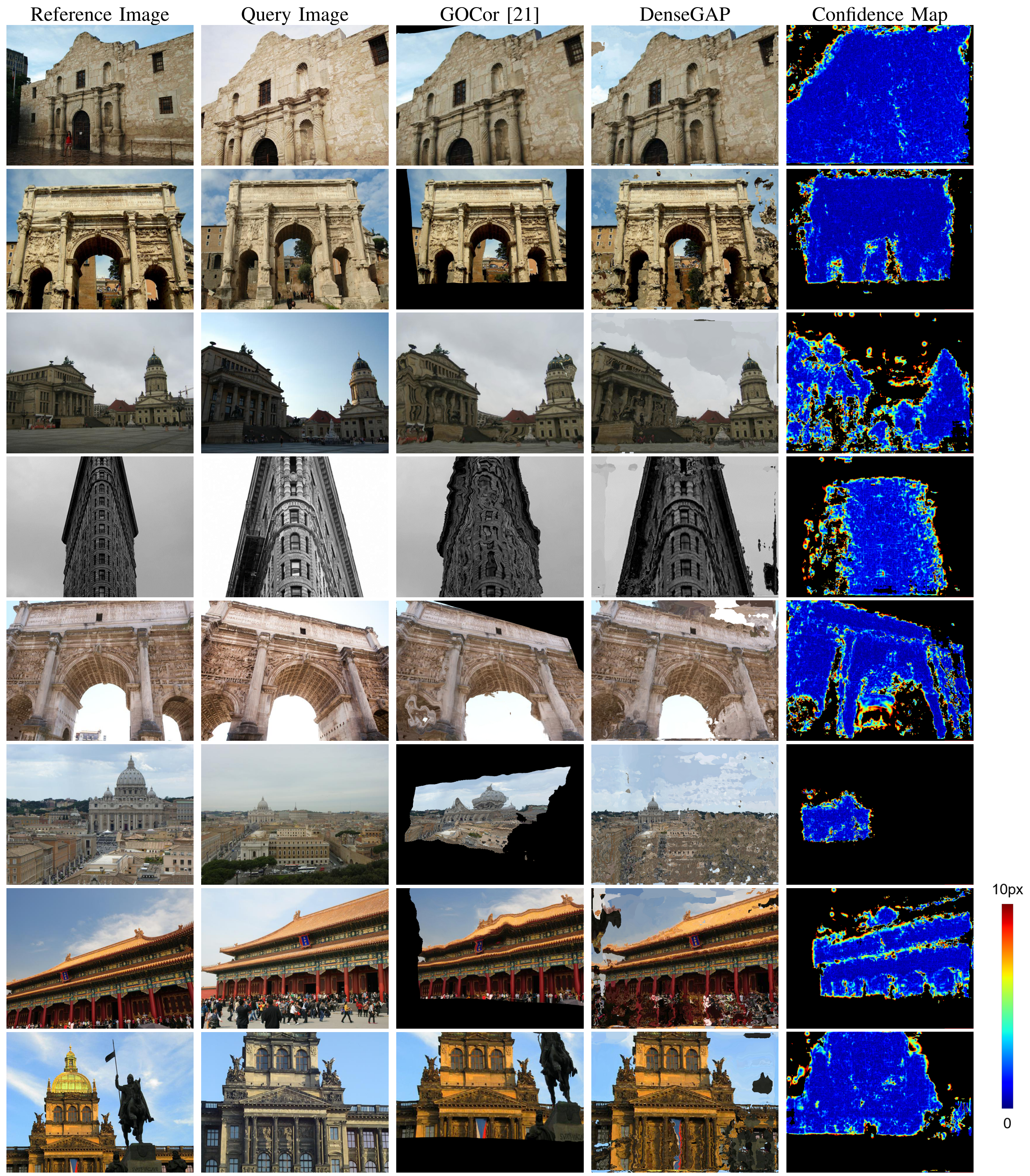

Fig. 4. Qualitative results on geometric matching. The reference image ($1st$ column) is warped to the query image ($2nd$ column) based on the dense correspondences generated by the baseline method ($3rd$) and our model ($4th$). The confidence maps of our predictions (represented by cycle consistency) are also shown in the $5th$ column. The black pixels in the confidence map represent those with the cycle consistency larger than 10 pixels.

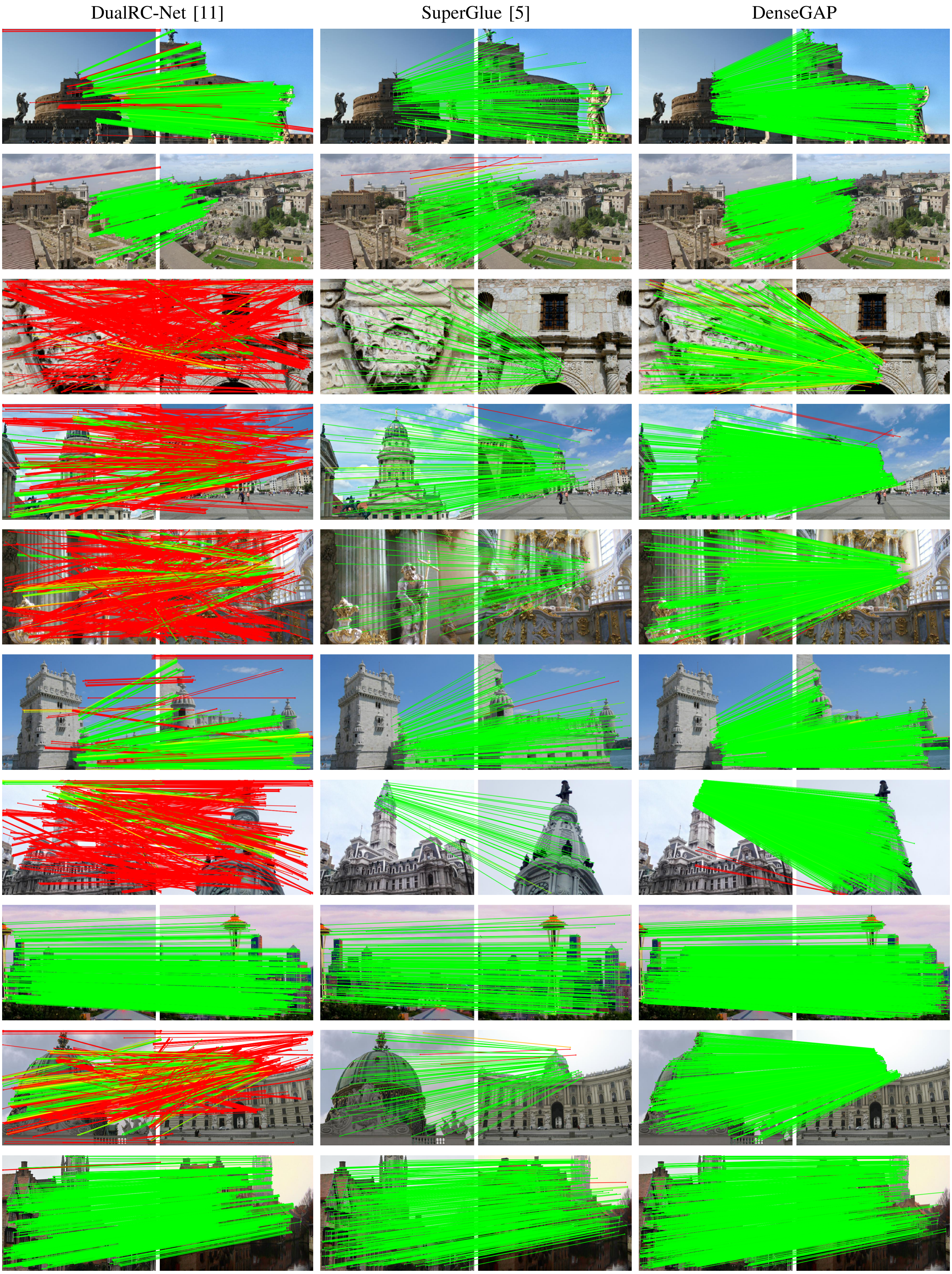

Fig. 5. Matching results on MegaDepth. Correspondences are colored by their epipolar errors calculated based on ground truth relative poses, and the threshold is set to 1×10^{-3}. We select top 1000 matches for both DualRC-Net and our model (green means inliers, and red means outliers).

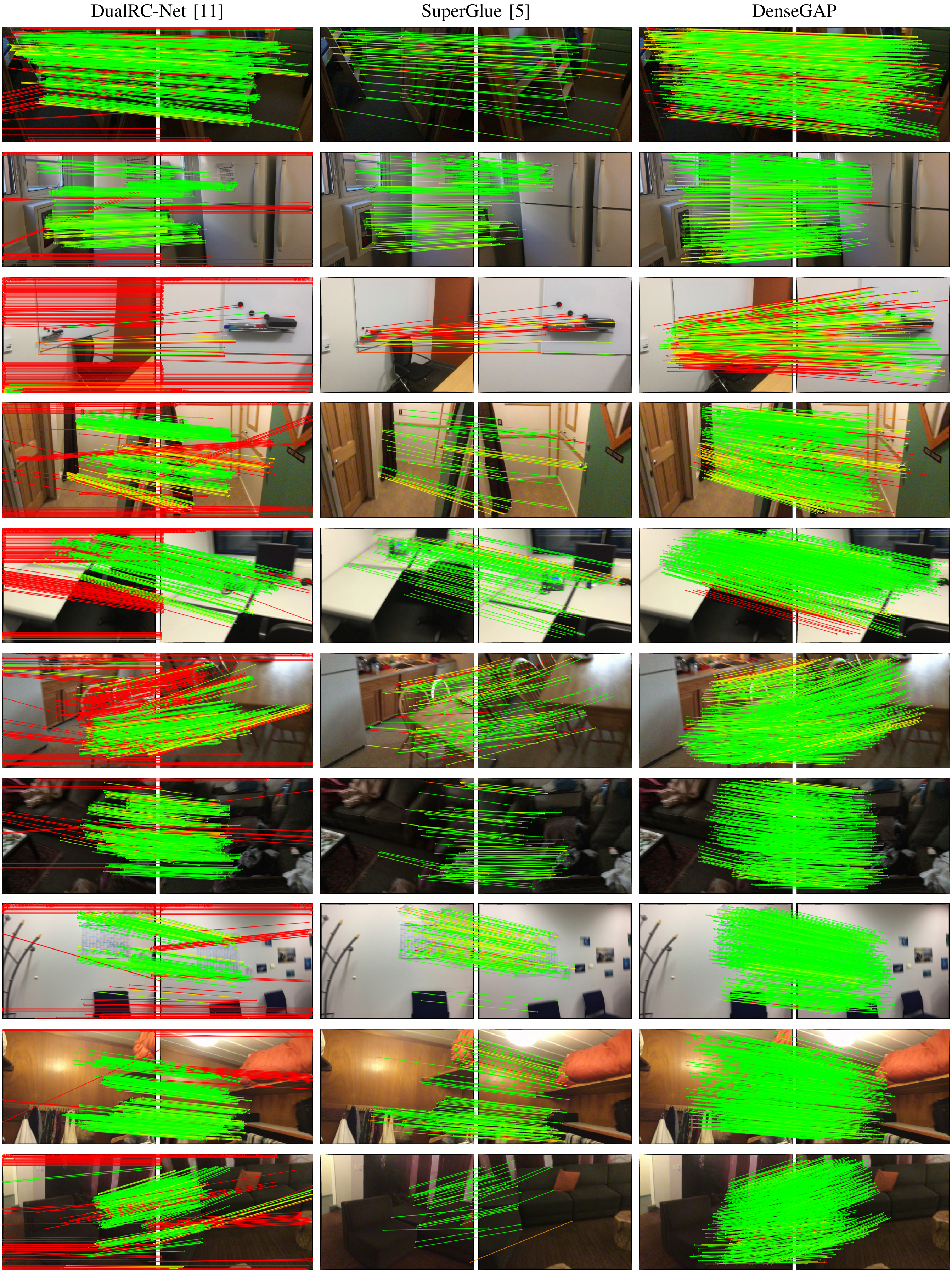

Fig. 6. Matching results on ScanNet. Correspondences are colored by their epipolar errors calculated based on ground truth relative poses, and the threshold is set to 1×10^{-3}. We select top 500 matches for both DualRC-Net and our model (green means inliers, and red means outliers).

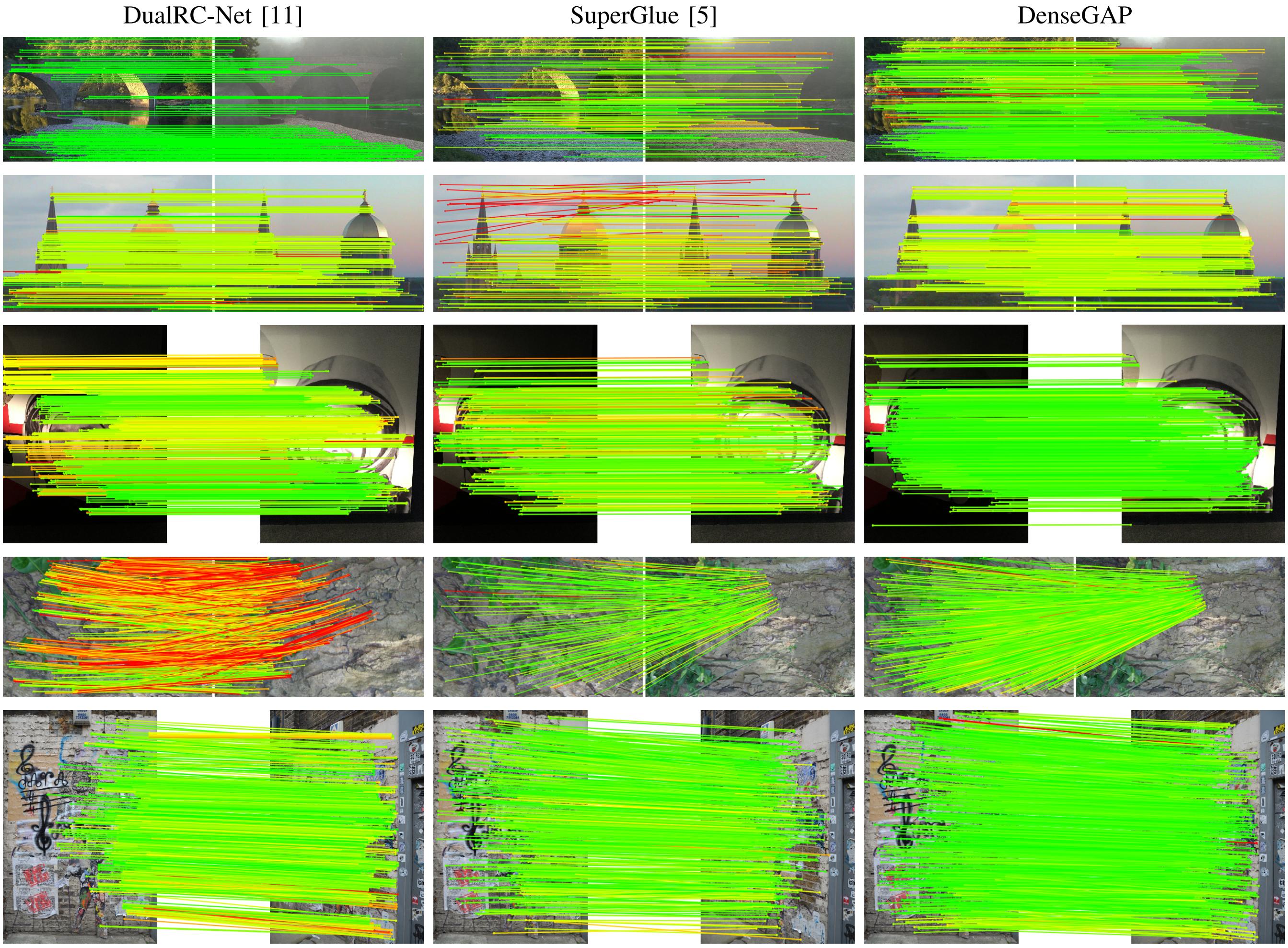

Fig. 7. Matching results on HPatches. Correspondences are colored by their reprojection errors calculated based on ground truth homography, and the threshold is set to 5 pixels. We select top 1000 matches for both DualRC-Net and our model (green means inliers, and red means outliers).

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pages 770–778. IEEE Computer Society, 2016.

[2] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, pages 2041–2050. IEEE Computer Society, 2018.

[3] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pages 2432–2443. IEEE Computer Society, 2017.

[4] Qianqian Wang, Xiaowei Zhou, Bharath Hariharan, and Noah Snavely. Learning feature descriptors using camera pose supervision. In *Computer Vision - ECCV 2020*, volume 12346 of *Lecture Notes in Computer Science*, pages 757–774. Springer, 2020.

[5] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020*, pages 4937–4946. IEEE, 2020.

[6] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 12716–12725. Computer Vision Foundation / IEEE, 2019.

[7] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 8024–8035, 2019.

[8] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pages 3852–3861. IEEE Computer Society, 2017.

[9] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.*, 60(2):91–110, 2004.

[10] Wei Jiang, Eduard Trulls, Jan Hosang, Andrea Tagliasacchi, and Kwang Moo Yi. COTR: correspondence transformer for matching across images. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 6187–6197. IEEE, 2021.

[11] Xinghui Li, Kai Han, Shuda Li, and Victor Prisacariu. Dual-resolution correspondence networks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

[12] 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 (CVPR) Workshops*, June 2018.

[13] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R. Bradski. ORB: an efficient alternative to SIFT or SURF. In *IEEE International Conference on Computer Vision, ICCV 2011*, pages 2564–2571. IEEE Computer Society, 2011.

[14] JiaWang Bian, Wen-Yan Lin, Yasuyuki Matsushita, Sai-Kit Yeung, Tan-Dat Nguyen, and Ming-Ming Cheng. GMS: grid-based motion statistics for fast, ultra-robust feature correspondence. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pages 2828–2837. IEEE Computer Society, 2017.

[15] Mihai Dusmanu, Ignacio Rocco, Tomás Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable CNN for joint description and detection of local features. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019*, pages 8092–8101. Computer Vision Foundation / IEEE, 2019.

[16] Zixin Luo, Tianwei Shen, Lei Zhou, Jiahui Zhang, Yao Yao, Shiwei Li, Tian Fang, and Long Quan. Contextdesc: Local descriptor augmentation with cross-modality context. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 2527–2536. Computer Vision Foundation / IEEE, 2019.

[17] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. *CVPR*, 2021.

[18] Moritz Menze, Christian Heipke, and Andreas Geiger. Object scene flow. *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*, 2018.

[19] Moritz Menze, Christian Heipke, and Andreas Geiger. Joint 3d estimation of vehicles and scene flow. In *ISPRS Workshop on Image Sequence Analysis (ISA)*, 2015.

[20] Prune Truong, Martin Danelljan, and Radu Timofte. Glu-net: Global-local universal network for dense flow and correspondences. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020*, pages 6257–6267. IEEE, 2020.

[21] Prune Truong, Martin Danelljan, Luc Van Gool, and Radu Timofte. Gocor: Bringing globally optimized correspondence volumes into your neural network. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

[22] Carl Toft, Will Maddern, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, et al. Long-term visual localization revisited. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.