Patch2Pix: Epipolar-Guided Pixel-Level Correspondences Supplementary Material

Qunjie Zhou[1] Torsten Sattler[2] Laura Leal-Taixé[1]
[1]Technical University of Munich [2]CIIRC, Czech Technical University in Prague

In this supplementary material, we provide additional information to further understand our proposed refinement network *Patch2Pix*. In Sec. 1, we provide the architecture details of our backbone, regressors and our baseline, *i.e.*, the adapted NCNet, followed by the details of our training data and other implementation details. We presents ablation studies on our architecture and training hyper-parameters in Sec. 2. We further detail the experimental setups for the homography estimation and outdoor/indoor localization in Sec. 3. Finally, Sec. 4 shows qualitative results of matches estimated by *Patch2Pix* on various benchmarks. We will release our code upon the paper's acceptance.

1. Implementation Details.

Backbone. We use a truncated ResNet34 as our backbone to extract features from the input images. It predicts 5 feature maps, from input feature f_0 to the last feature map f_4. The corresponding feature channel dimensions are [3, 64, 64, 128, 256]. To have enough resolution in the last feature map f_4, we change the stride of the convolutional layer to prevent further downscaling, which means the spatial resolution of f_4 is the same as f_3, *i.e.*, $1/8$ of the original image resolution. The ResNet34 backbone is pretrained on ImageNet [5] and frozen during training.

Regressor. Our mid-level and fine-level regressors have the same architecture, as shown in Fig. 1. On the left side of the figure, the collected features from the backbone of a patch pair are fed into a set of layers to aggregate the feature tensors into a single vector. On the right side, the aggregated feature vector is fed into a set of fully connected layers (FC) to output the confidence score c and the coordinates of the detected local match δ_i.

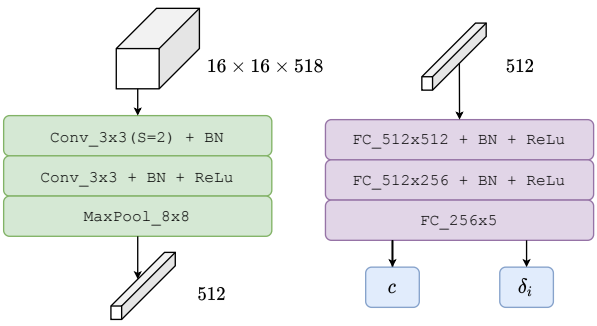

Figure 1. **Regressor Architecture.**

Our Adapted NCNet [15]. To detect match proposals, we use the pretrained NC matching layer from NCNet [15] to match our extracted features. Given a pair of images, features are first extracted from the image and the two last feature maps, *i.e.* f_4^A, f_4^B, which are 8-times downscaled w.r.t. image resolution, are exhaustively matched to produce a correlation map. The size of the correlation map is further reduced using a MaxPool4D operation with window size $k = 2$ for computational efficiency following the original NCNet [15]. The final matching score map is obtained by applying a 4D convolution over the reduced correlation map to enforce neighbourhood consensus. The raw matches are the indices of row-wise and column-wise maximum values of the matching score map. To go back to the matching resolution, the raw matches are shifted to the corresponding pooling location using the index information from the MaxPool4D operation. This results in downscaled matches, with each match corresponding to a pair of local 8×8 patches in the original image. Multiplying a match by 8 gives the two upper-left corners of the two local patches.

We keep only the mutually matched patches and use all of them during training. During inference, we further filter the mutually matched ones with a match score threshold $c = 0.9$ for outlier rejection. We found it produces the best performance across tasks in our experiments.

Training Data Processing. Our refinement network is trained on the large-scale outdoor dataset MegaDepth [10], where images from 196 scenes are obtained from the Internet and then reconstructed using Structure-from-Motion [20]. We first follow the preprocessing steps from [7] to regenerate camera pose labels. We keep images with aspect ratio (width/height) between $[1.3, 1.7]$ from which we randomly select at most 500 pairs per scene which have more than 35% visual overlap. Finally, we obtain in total 60661 pairs across 160 scenes. During training, we

ID	Exp.	Feat.	$\widehat{\theta}_{cls}$ / $\widehat{\theta}_{geo}$	$\widetilde{\theta}_{cls}$/$\widetilde{\theta}_{geo}$	Overall	Illumination	Viewpoint
					Accuracy (%, $\epsilon < 1/3/5$ px)		
I	No	01	50/50	5/5	0.37 / 0.75 / 0.82	0.54 / 0.91 / 0.95	0.20 / 0.60 / 0.70
		012			0.34 / 0.69 / 0.8	0.62 / 0.92 / 0.97	0.08 / 0.47 / 0.64
		123			0.37 / 0.69 / 0.81	0.68 / 0.93 / 0.98	0.09 / 0.48 / 0.66
		01234			0.42 / 0.76 / 0.83	0.62 / 0.92 / 0.97	0.24 / 0.60 / 0.70
		0123			0.45 / 0.77 / 0.85	0.65 /0.93 / 0.98	0.27 / 0.62 / 0.73
II	No	0123	400/400	5/5	0.43 / 0.76 / 0.84	0.56 / 0.93 / 0.97	0.27 / 0.62 / 0.73
			100/100	5/5	0.45 / 0.76 / 0.84	0.67 / 0.93 / 0.98	0.25 / 0.60 / 0.72
			50/50	15/15	0.44 / 0.77 / 0.85	0.68 / 0.93 / 0.97	0.21 / 0.62 / 0.74
			50/50	5/5	0.45 / 0.77 / 0.85	0.65 / 0.93 / 0.98	0.27 / 0.62 / 0.73
			50/50	1/1	0.43 / 0.76 / 0.84	0.62 / 0.91 / 0.97	0.26 / 0.61 / 0.71
			25/25	5/5	0.41 / 0.77 / 0.85	0.61 / 0.94 / 0.98	0.23 /0.62 / 0.73
III	Yes	0123	50/50	15/1	0.42 / 0.78 / 0.85	0.59 / 0.95 / 0.98	0.27 / 0.61 / 0.73
			50/50	5/5	0.47 / 0.78 / 0.85	0.65 / 0.93 / 0.97	0.30 / 0.64 / 0.73
			50/50	5/1	0.45 / 0.76 / 0.85	0.64 / 0.93 / 0.98	0.28 / 0.59 / 0.72
			50/50	1/1	0.44 / 0.76 / 0.84	0.63 / 0.94 / 0.98	0.26 / 0.60 / 0.71

Table 1. *Patch2Pix* **Training Ablation.** All trained variants are evaluated on HPatches [1] for homography estimation. We compare the models within each group and mark the best results in different colors.

crop the image from the right and bottom sides so that its aspect ratio is 1.5 and then resize every image to resolution 480×320.

Training Details. For each training image pair, we randomly select 400 matches from the NCNet match proposals and then apply our expansion mechanism, which gives us 3200 matches to be processed by the two regressors. The regressors are optimized using Adam [9] with an initial learning rate of $5e^{-4}$ for 5 epochs and then $1e^{-4}$ until it converges. Our method is implemented in Pytorch [12] v1.4. Each of our training is performed on a RTX 8000 48GB GPU.

2. Training Ablation Study.

We show *Patch2Pix* variants trained under different training settings including: with or without patch expansion (Exp.), different feature collection for patch pairs (Feat.), the two thresholds $\widehat{\theta}_{cls}, \widehat{\theta}_{geo}$ used to calculate the losses of the mid-level regressor, and the two $\widetilde{\theta}_{cls}, \widetilde{\theta}_{geo}$ for the fine-level regressor (*c.f.* Sec.3.1 & 3.2 in our main paper). We compare all variants when they are trained with a learning rate of $5e^{-4}$ for 5 epochs, since training longer does not change the comparison in our case. Those variants are evaluated using HPatches [1] for homography estimation at a confidence threshold 0.5 for ablation. We report the percentage of correctly estimated homographies whose average corner error distance is below 1/3/5 pixels. We give an ID to different groups of experiments for convenience and present the results in Tab. 1.

Training Hyper-parameters. As shown in Tab. 1, in the experiments of group I, we keep other settings identical and only modify the feature collection. We show that taking features from all layers before the last layer f_4 leads to the best results. We then fix the feature collection and vary the thresholds which are used to identify the labels for classification and the subset of matches to be optimized for regression. Comparing models within group II, we find that models using a threshold of 400 and 50 for the mid-level regressor perform best for viewpoint changes, while the model trained with threhsold 50 is better under illumination changes and thus overall more promising. In group III, we again fix the feature collection and apply patch expansion mechanism to our trainings and further search for the suitable thresholds. We find that the model trained with $\widehat{\theta}_{cls} = \widehat{\theta}_{geo} = 50$ and $\widetilde{\theta}_{cls} = \widetilde{\theta}_{geo} = 5$ overall outperform others, especially under viewpoint changes, which gives the best threshold setting.

Effect of Local Patch Expansion. Comparing the best models from groups II and III, we show that with the same training epochs, our best model learns faster when using patch expansion. We further noticed that with the same thresholds, the model trained without patch expansion converges faster at a similar accuracy, while the model with patch expansion converged slower at a better accuracy.

3. Experiment Details.

Homography Estimation Details To compute the corner correctness metric used in [6,17,22], the four corners of one image are transformed into the other image using the estimated homography to compute the distance to the four GT corners. We report the percentage of correctly estimated homographies whose average corner error distance is below 1/3/5 pixels. To estimate the homography from the predicted matches, we use the *findHomography* function pro-

vided in pydegensac [3,4,11][1], which shows marginally better accuracy compared to the OpenCV [2] implementation. We fix the RANSAC threshold as 2 pixels since it in general works better than other thresholds for all methods. We run all methods on a GTX TITAN X 12GB GPU under our environment using their public implementations.

Quantization Details. As we mentioned in the main paper, we apply quantization to our matches to evaluate on Aachen Day-Night Benchmark (v1.0) [18, 19]. The introduced localization pipelines, *e.g.* HLOC [16], first reconstruct a 3D model using the local features and matches and then register the queries to the built 3D model. Therefore, such pipelines require methods to produce keypoints that are co-occurring in several retrieval pairs to work properly in the triangulation step for reconstruction. However, our method directly regresses matches from a pair of images. As such, the pixel positions found in image A for a pair (A, B) might differ slightly to those found for a pair (A, C). In contrast, all methods that perform separate feature extraction per image will automatically have the same detections in image A for both pairs. Thus, they can easily be used for triangulation. To make our matches work in this setting, we quantize our matches by representing keypoints that are closer than 4 pixels to each other with their mean location, meaning we sacrifice pixel-level accuracy here. After quantization, we remove the duplicated matches by keeping only the one with the highest confidence score. While it is not a perfect solution, we leave it as our future work to either add a loss that enforces detecting the same positions in A for pairs (A, B) and (A, C) or to design a localization pipeline tailored to our matches.

InLoc Evaluation Details. We follow SuperPoint [6]+SuperGlue [17] to evaluate on the same top-40 retrieval pairs, where they perform a temporal consistency check to restrict the retrieval, and adopt their RANSAC threshold of 48 pixels for pose estimation. For all methods, we test their performance two different image sizes, *i.e.*, we resize every image to have a larger side of either 1024 or 1600 pixels. Only for SparseNCNet [14] we also consider its default image size, *i.e.*, 3200 pixels. By default, the local feature detection and description methods use the nearest neighbor mutual matcher to detect matches. We further test those methods when using a threshold of 0.75 on their matching scores, computed by their normalized descriptors, for outlier rejection. We present the complete results of the methods under different settings in Tab. 2. For clarity, we mark their best entries blue which have been presented in Tab. 2 of our main paper.

With the results of the complete table, we show that SuperPoint [6], SuperPoint + CAPS [22] and SIFT + CAPS [22] benefit from using a threshold of 0.75 for the outlier filtering while D2Net [7] and R2D2 [13] perform better without such thresholding. In addition, we observe that SuperPoint, D2Net , SuperPoint + CAPS and Patch2Pix benefit from using a smaller image size of 1024, while SparseNCNet performs best at size of 1600 pixels.

[1] https://github.com/ducha-aiki/pydegensac

Method	Imsize	Supervision	Localized Queries (%, $0.25m$/$0.5m$/$1.0m$, $10°$) DUC1	DUC2
SuperPoint [6] + NN 0.75	1024	Full	40.4 / 58.1 / 69.7	42.0 / 58.8 / 69.5
SuperPoint + NN 0.0	1024	Full	29.8 / 48.5 / 61.6	32.1 / 46.6 / 56.5
SuperPoint + NN 0.75	1600	Full	43.9 / 67.7 / 76.3	39.7 / 58.0 / 71.0
D2Net [7] + NN 0.0	1024	Full	38.4 / 56.1 / 71.2	37.4 / 55.0 / 64.9
D2Net + NN 0.75	1024	Full	31.8 / 49.0 / 55.1	20.6 / 34.4 / 44.3
D2Net + NN 0.0	1600	Full	34.8 / 54.5 / 68.7	34.4 / 50.4 / 62.6
R2D2 [13] + NN 0.0	1600	Full	36.4 / 57.6 / 74.2	45.0 / 60.3 / 67.9
R2D2 + NN 0.75	1600	Full	35.4 / 60.6 / 75.8	42.7 / 57.3 / 65.6
SuperPoint + SuperGlue [17]	1600	Full	49.0 / **68.7** / 80.8	53.4 / 77.1 / **82.4**
SuperPoint + CAPS [22] + NN 0.75	1024	Mix	40.9 / 60.6 / 72.7	43.5 / 58.8 / 68.7
SuperPoint + CAPS + NN 0.0	1024	Mix	39.4 / 61.6 / 72.7	35.1 / 50.4 / 64.1
SuperPoint + CAPS + NN 0.75	1600	Mix	43.9 / 67.7 / 76.3	39.7 / 58.0 / 71.0
SIFT + CAPS [22] + NN 0.75	1600	Weak	38.4 / 56.6 / 70.7	35.1 / 48.9 / 58.8
SIFT + CAPS + NN 0.0	1600	Weak	37.9 / 56.1 / 66.7	30.5 / 43.5 / 53.4
SIFT + CAPS + NN 0.75	1024	Weak	38.4 / 53.5 / 69.7	33.6 / 45.0 / 55.0
SparseNCNet [14] (top2k)	1600	Weak	41.9 / 62.1 / 72.7	35.1 / 48.1 / 55.0
SparseNCNet (top2k)	1024	Weak	37.9 / 54.0 / 70.2	32.8 / 45.8 / 53.4
SparseNCNet (top2k)	3200	Weak	35.4 / 50.5 / 62.1	24.4 / 31.3 / 35.9
Patch2Pix (c=0.25)	1024	Weak	44.4 / 66.7 / 78.3	49.6 / 64.9 / 72.5
Patch2Pix (c=0.25)	1600	Weak	44.9 / 67.2 / 75.8	43.5 / 59.5 / 69.5
Patch2Pix (w.SuperPoint+CAPS)	1024	Mix	42.4 / 62.6 / 76.3	43.5 / 61.1 / 71.0
Patch2Pix (w.SuperGlue)	1600	Mix	**50.0** / 68.2 / **81.8**	**57.3** / **77.9** / 80.2

Table 2. **Complete InLoc [21] Benchmark Results.** We report the percentage of correctly localized queries under specific error thresholds. Methods are evaluated inside HLOC [16] pipeline to share the same retrieval pairs, RANSAC threshold. We mark the best results in **bold**. For each method, we mark its best entry among all settings in blue which corresponds to its result presented in Tab. 2 of our main paper.

4. Qualitative Results.

In Fig. 2, we plot the matches estimated by *Patch2Pix* on the image pairs obtained from the internet, HPatches [1] and PhotoTourism [8]. We use the default setting of our model, *i.e.*, NCNet proposals and confidence score 0.25, to predict matches from the image pairs. We identify the inlier matches using the *findHomography* or *findFundamentalMatrix* function provided in pydegensac [3, 4, 11]. For the HPatches image pairs, we use *findHomography* with a ransac threshold of 2. For other image pairs, we use *findFundamentalMatrix* with a RANSAC threshold of 1. Finally, we plot at most 300 matches for each pair for clear visualization.

Furthermore, we visualize the matches on Aachen Day-Night (v1.0) [18, 19] in Fig. 3 and on InLoc [21] in Fig. 4 and Fig. 5. We show the matches refined by *Patch2Pix* when we use our NCNet baseline, and when we use SuperPoint [6] + SuperGlue [17] for the match proposals. For a randomly selected query, we pick the database images with the most inlier matches identified by the camera pose solver during localization. We plot the inliers in green and other matches in red and count the inlier numbers.

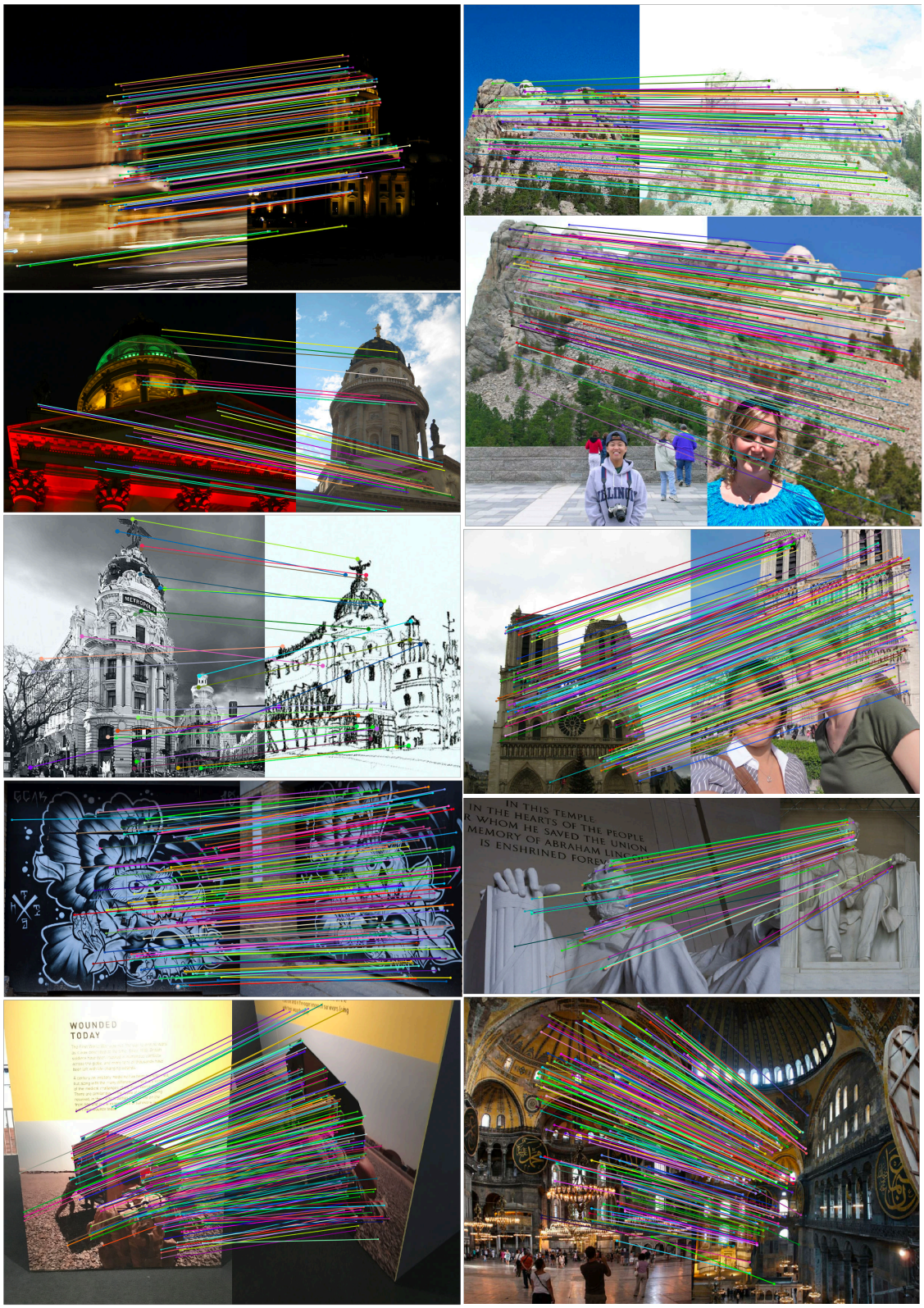

Figure 2. Example matches of *Patch2Pix* on the image pairs obtained from the internet, HPatches [1] and PhotoTourism [8]. *Patch2Pix* can robustly handle strong illumination changes, large viewpoint variations, and repetitive structures.

Figure 3. Example matches of *Patch2Pix* using NCNet proposals (left) and SuperPoint [6] + SuperGlue [17] (right) proposals on night queries of Aachen Day-Night(v1.0) [18, 19].

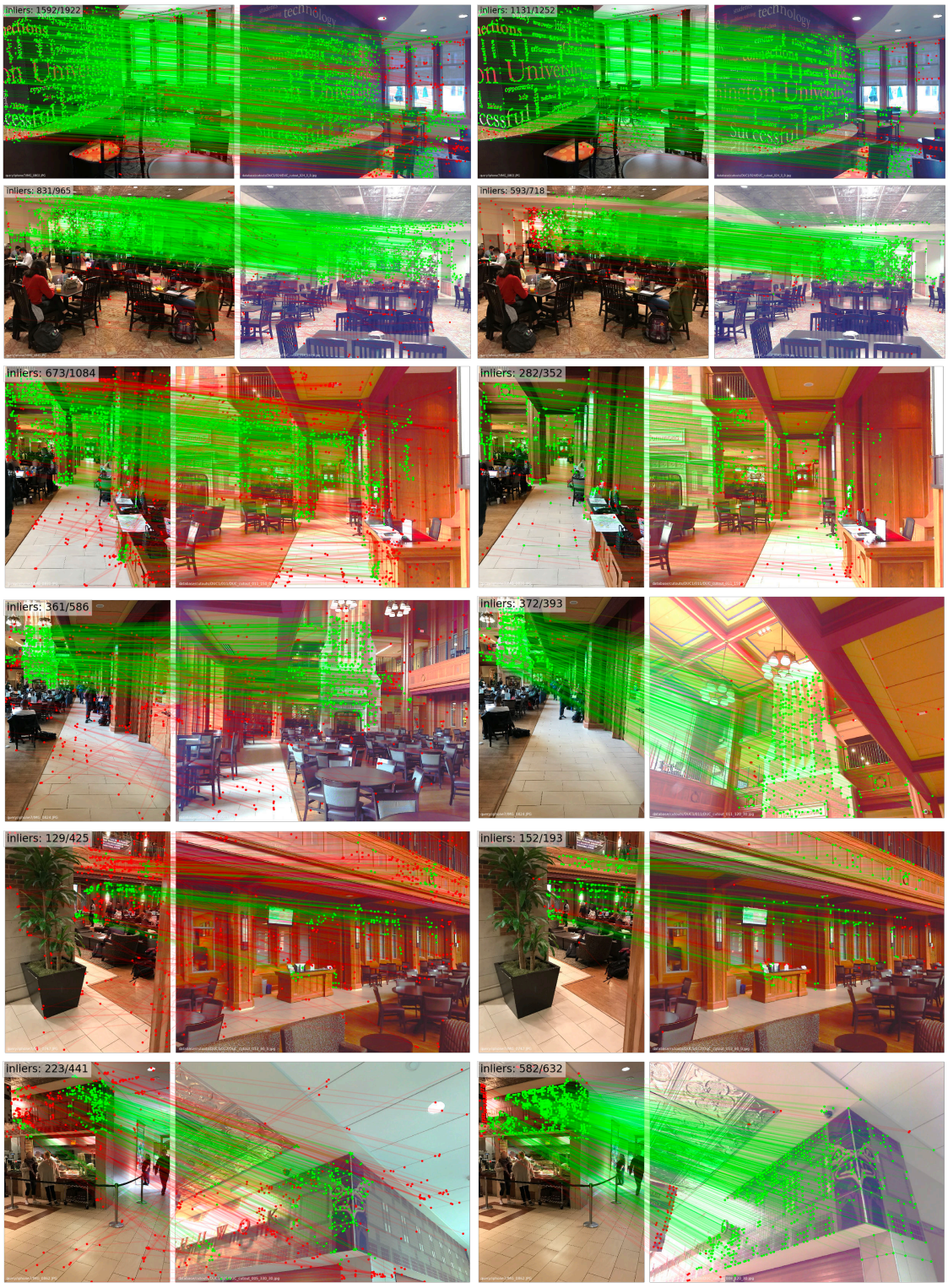

Figure 4. Example matches of *Patch2Pix* using NCNet proposals (left) and SuperPoint [6] + SuperGlue [17] (right) proposals on InLoc [21] DUC1.

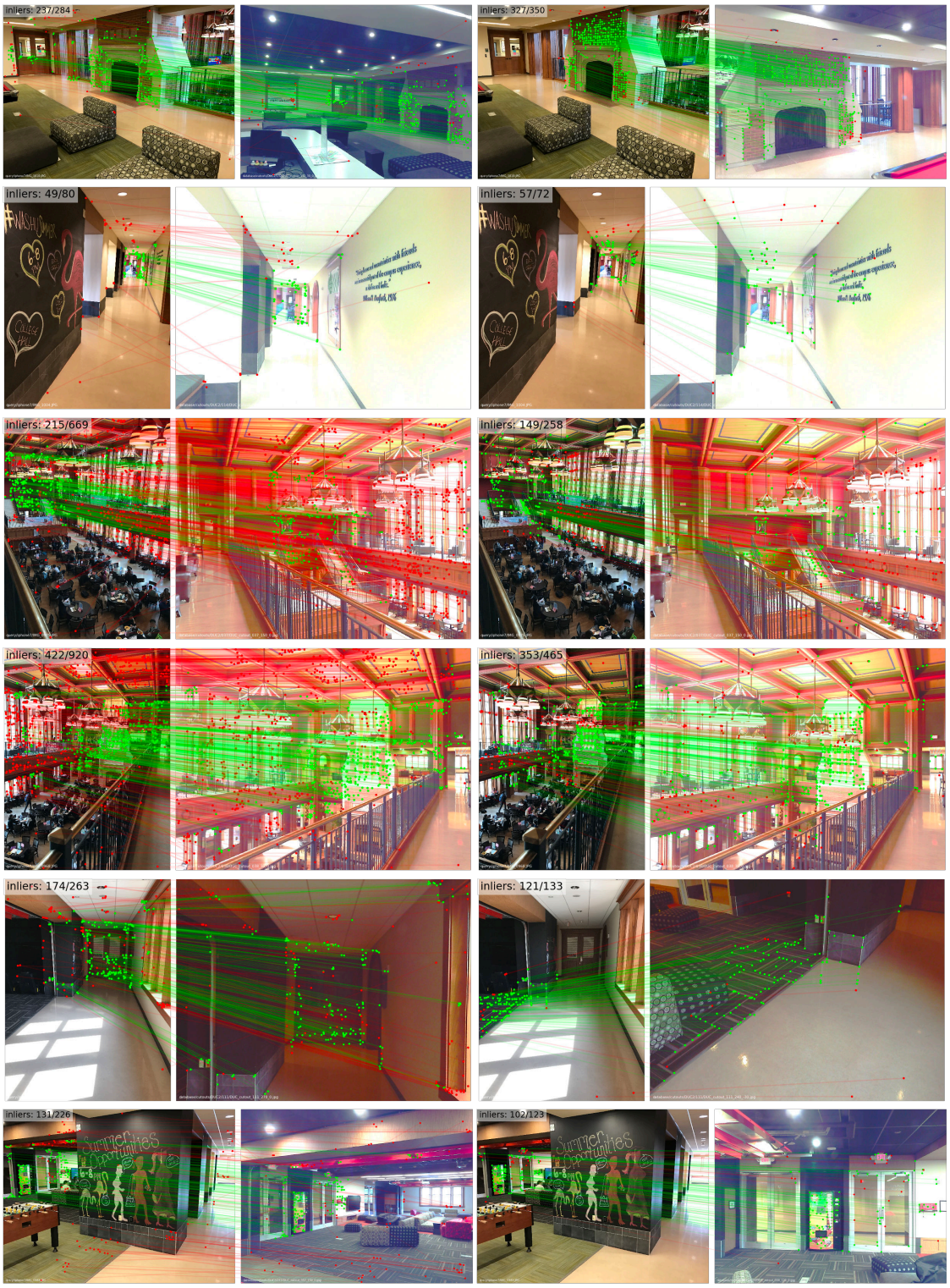

Figure 5. Example matches of *Patch2Pix* using NCNet proposals (left) and SuperPoint [6] + SuperGlue [17] (right) proposals on InLoc [21] DUC2.

References

[1] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *CVPR*, pages 5173–5182, 2017. 2, 3, 4

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 3

[3] Ondřej Chum, Jiří Matas, and Josef Kittler. Locally optimized ransac. In *Joint Pattern Recognition Symposium*, pages 236–243. Springer, 2003. 3

[4] Ondrej Chum, Tomas Werner, and Jiri Matas. Two-view geometry estimation unaffected by a dominant plane. In *CVPR*, volume 1, pages 772–779. IEEE, 2005. 3

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. Ieee, 2009. 1

[6] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPR Workshops*, pages 224–236, 2018. 2, 3, 5, 6, 7

[7] 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. In *CVPR*, 2019. 1, 3

[8] Yuhe Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. Image Matching across Wide Baselines: From Paper to Practice. *IJCV*, 2020. 3, 4

[9] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 2

[10] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *CVPR*, 2018. 1

[11] Dmytro Mishkin, Jiri Matas, and Michal Perdoch. Mods: Fast and robust method for two-view matching. *CVIU*, 141:81–93, 2015. 3

[12] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NeurIPS Workshops*, 2017. 2

[13] Jerome Revaud, Cesar De Souza, Martin Humenberger, and Philippe Weinzaepfel. R2d2: Reliable and repeatable detector and descriptor. In *NeurIPS*, pages 12405–12415, 2019. 3

[14] Ignacio Rocco, Relja Arandjelović, and Josef Sivic. Efficient neighbourhood consensus networks via submanifold sparse convolutions. *ECCV*, 2020. 3

[15] Ignacio Rocco, Mircea Cimpoi, Relja Arandjelović, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Neighbourhood consensus networks. In *NeurIPS*, 2018. 1

[16] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, 2019. 3

[17] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *CVPR*, pages 4938–4947, 2020. 2, 3, 5, 6, 7

[18] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, et al. Benchmarking 6dof outdoor visual localization in changing conditions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8601–8610, 2018. 3, 5

[19] Torsten Sattler, Tobias Weyand, Bastian Leibe, and Leif Kobbelt. Image Retrieval for Image-Based Localization Revisited. In *British Machine Vision Conference (BMCV)*, 2012. 3, 5

[20] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 1

[21] 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. 3, 6, 7

[22] Qianqian Wang, Xiaowei Zhou, Bharath Hariharan, and Noah Snavely. Learning feature descriptors using camera pose supervision. In *ECCV*, 2020. 2, 3