ScaleNet: A Shallow Architecture for Scale Estimation

Axel Barroso-Laguna Yurun Tian Krystian Mikolajczyk
Imperial College London
{axel.barroso17, yurun.tian, k.mikolajczyk}@imperial.ac.uk

Appendices

A. Dataset details

Real pair images are obtained directly from the Megadepth dataset [7] as detailed in the main paper. As a reference, we display in figure 1 examples of our dataset with their labeled scale changes. Since our labeling strategy relies on a 3D model, we can capture extreme scale changes between images if the collection of images that was used for the 3D reconstruction was big. While keypoint/descriptors may not be able to match correctly two images when there are strong scale changes, we rely on the fact that 3D models can relate two images if there were images in-between, *i.e.*, we can compute the relationship between image A and B if we have an easier-to-match image C in between the two views. This idea is somehow similar to the strategy proposed in MODS [12], where authors create synthetic images between two different views to be able to relate them. In addition, we also use synthetic pairs to train ScaleNet. Synthetic pairs are computed on the fly, and therefore an unlimited number of examples can be created. We define a set of synthetic transformations with affine parameters: scale $\in [0.16, 6]$, rotation $\in [-30°, 30°]$ and skew $\in [-0.2, 0.2]$. We use the random transformation to wrap the input image and generate the training pair. The ground-truth scale is directly the scale factor used to transform the image.

Figure 1. Image pairs examples of our dataset with their scale factor annotations computed as the average on logarithmic space of local scale changes.

B. Local-ScaleNet

Local-ScaleNet predicts a dense scale factor map between a pair of input images. In contrast to ScaleNet, in which scale estimation is global, pixel-wise Local-ScaleNet estimations offer a more suitable option when scale transformation significantly varies across the images, *e.g.*, images with strong perspective distortion.

Architecture. Local-ScaleNet architecture follows the scheme proposed in ScaleNet (cf. section 3 main paper) with small variations. Analogous to ScaleNet, Local-ScaleNet takes two input images, A and B, and computes features, f_A and f_B, with the same multi-scale feature extractor block. Features, f_A and f_B, go into the correlation layers to calculate the self- and cross-similarities, c_A and c_{A-B}. In contrast to ScaleNet, Local-ScaleNet does not compute the self-similarities in image B. Since Local-ScaleNet estimates the scale factor locally, self-similarities in image B, c_B, are not spatially correlated with image A, and therefore, c_B is not used in the per-pixel scale estimation. Similar to ScaleNet, c_{A-B} and c_A go into our local feature reduction block to process and reduce the channel dimension of the correlation maps. c'_A and c'_{A-B} are then concatenated and fed into the final dense prediction network which outputs the local distribution of scales between images A and B. Furthermore, during test time, we transform our scale distribution into a scale factor map by following the pipeline presented in the main paper (cf. equation 3) in a pixel-wise manner. Due to image A and B not correlating spatially, we do not apply the consistency check from the original ScaleNet. Finally, the scale factor map is resized into the original image A resolution. Full Local-ScaleNet architecture and pipeline are shown in figure 2.

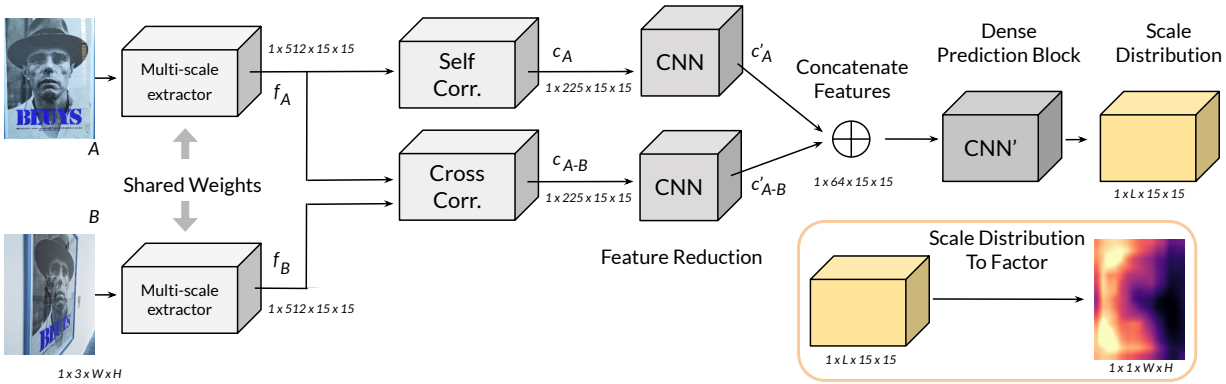

Figure 2. Local-ScaleNet uses a pre-trained VGG-16 and an ASPP block as its multi-scale feature extractor. After features are computed, a combination of self- and cross-correlation layers are used to calculate the relationship within image A and between images A and B. Correlation volumes' dimensionality is reduced through a CNN, and its results are concatenated into a single map. Finally, a dense scale distribution map is calculated by the final dense prediction block. During inference time, the scale distribution map is converted into a scale factor map and resized to the original resolution of image A.

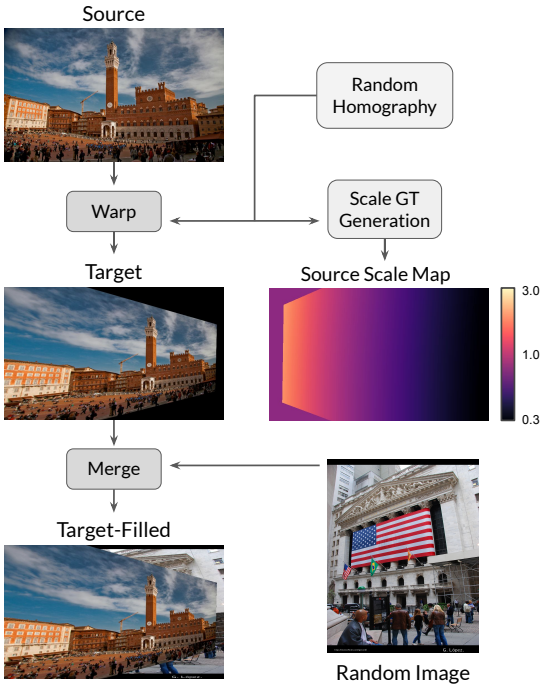

Figure 3. The local dataset generation pipeline, firstly, computes a random homography transformation. The homography is applied to an input image to generate a tuple of images. We refer to them as *Source* and *Target* images. The same homography is used to calculate the source scale map. Moreover, we take an extra random image from the dataset to fill those regions with zero values in the *Target* image due to the homography transformation.

Dataset. As discussed, Local-ScaleNet generates a dense map of scale distributions. To train it, we need pairs of images with their corresponding pixel-wise scale ground-truths. We based our dataset on synthetic pairs of images as displayed in figure 3. Even though real pairs of images could be used to train Local-ScaleNet, real image scenes do not present strong local distortions, *i.e.*, global scale often reflects the scale changes in local regions. Thus, we use synthetic pairs to ensure that the global scale differs from the scales between local regions.

A random homography transformation is generated and applied to an input image, such that the input image, *Source*, and the warped one, *Target*, are geometrically related by our homography. We use the same parameters to build the homography matrix as those presented in appendix A and add a perspective distortion to ensure different local scale factors throughout the *Target* image. Moreover, due to strong homography transformations, *Target* image may contain zero values in the non-overlapping regions as seen in figure 3. To avoid the network being driven by those zero values, we filled the non-overlapping regions between *Source* and *Target* images with a randomly sampled image from the dataset. Therefore, our final training pair is composed by the *Source* and our new *Target-Filled* image. Lastly, we use the same homography to generate the ground-truth scale map between *Source* and *Target-Filled*. As we have the synthetic transformation, we generate the source scale map by sampling local points throughout the whole *Source* image and computing the scale factor within their neighborhood region as the ratio of their distances in the *Source* and *Target-Filled* images.

C. Extended experiments

This section extends the experiments and results presented in the main paper.

C.1. Design choices

ScaleNet embraces some key ideas on its learning scheme and parameterization that help towards delivering good scale estimations. We discuss them here and highlight the importance of our decision during ScaleNet design. Table 1 shows the performance of our baseline, which uses SuperPoint network without any scale rectification. Moreover, we report the results of ScaleNet without the consistency check for an easier comparison against other designs.

Natural vs logarithmic space. The scale factor is a relative ratio operator, hence, it is non-linear. We compute the scale factor as a soft-computation based on quantized scale classes, s_i, and a probability distribution (cf. equation 2 main paper). To avoid being biased by high scale values when computing the soft-scale, we transform the quantized scale classes, s_i , to logarithmic space. Hence, to demonstrate the superiority of this parameterization, we display in table 1 the results of doing the soft-computation directly in natural space (*Natural rep.*). We see that even though the performance of the natural representation model is over the baseline, we can improve upon it by the simple and effective approach of dealing with the scale factors in the logarithmic space.

Regression vs classification model. Even though predicting directly the scale factor between images is theoretically possible, in the practical scenario, it appears to be a much harder task. To prove it, we have trained a regression model with the L2 loss, where our network had to predict a normalized scale factor between the two images in logarithmic space. We use logarithmic space since the previous experiment shows the benefits of this representation. As shown in table 1, the proposed ScaleNet regressor model does not bring any benefit over the baseline accuracy (which does not use any scale rectification). We claim that even though a regressor model could be trained, learning and interpreting the relationships between the quantized scale ranges is an easier task, and hence, we embrace the decision of a classification model rather than a regressor.

Hard-assignment vs soft-assignment. Hard-assignment has been already introduced in the main paper as discrete ScaleNet (D-ScaleNet). D-ScaleNet uses the maximum predicted value of the scale distribution as the scale factor between images, instead of computing the soft-scale (cf. equation 2 main paper). Even though we provided the advantages and drawbacks in terms of computation time and 3D metrics (cf. section 5.4 main paper), we further display their differences in table 1. Results show that the soft-assignment, and hence, being able to interpolate between the quantized scale values, provides better results than D-ScaleNet. However, D-ScaleNet can offer a faster alternative to ScaleNet and still bring notable improvements over the baseline.

	Pose estimation (AUC)		
	at 5°	at 10°	at 20°
Baseline	4.8	7.4	10.4
ScaleNet	**8.4**	**12.3**	**17.6**
Natural rep.	7.2	11.2	15.9
Regression	4.9	7.3	10.1
D-ScaleNet (hard-assig.)	7.9	11.5	16.7

Table 1. Ablation study of the different ScaleNet's design choices. Baseline refers to SuperPoint [5] without scale correction, and ScaleNet refers to the method introduced in the paper without consistency check (cf. equation 3 main paper).

	HPatches MMA-5px (%)		
	Easy	Medium	Hard
Key.Net-SS / HardNet	66.65	47.40	25.52
w/ ScaleNet	**71.98**	**48.40**	21.42
w/ Local-ScaleNet	64.63	45.01	**28.43**

Table 2. Mean matching accuracy (MMA) results on perspective sequences from HPatches [1].

C.2. Image matching

Local scale estimation allows to locally correct the scale of a given keypoint. Local estimations are effective when the scale transformation significantly varies across the image. Therefore, we test the effect of using the dense variant of ScaleNet (Local-ScaleNet) on sequences with a strong perspective change, *i.e.*, with different local scale factors. Local-ScaleNet substitutes the fully connected layers of ScaleNet with the CNN dense prediction block detailed in section B. As the scale correction is local, we combine ScaleNet with Key.Net/HardNet [3, 11], which allows correcting locally its feature extraction. Specifically, we run Key.Net in the original image and use the scale map estimated by Local-ScaleNet to correct the size of the extracted patch before computing a descriptor with HardNet. We split image pairs from HPatches [1] into easy, medium, and hard subsets according to their perspective distortions. Table 2 shows that using ScaleNet for local correction is effective for significant perspective changes but deteriorates the performance for less challenging transformations. Such behavior was also observed for scale and affine invariant handcrafted feature extractors [10]. Moreover, as Local-ScaleNet can only be applied to local descriptors, global ScaleNet is more suitable as a general approach.

C.3. IMC benchmark

Protocol. We follow the protocol proposed in Image Matching Challenge [18] for computing their two main

	mAA (%) at 10°						
	Stereo	**Multiview**	**Average**		**Stereo**	**Multiview**	**Average**
SIFT	43.1	43.2	43.2	SuperPoint	32.9	54.5	43.7
w/ ScaleNet	**43.4**	**47.3**	**45.4**	w/ ScaleNet	**41.3**	**61.9**	**52.0**
R2D2-SS	13.4	12.3	12.9	R2D2-MS	34.9	**48.0**	**41.5**
w/ ScaleNet	**26.3**	**31.7**	**29.0**	w/ ScaleNet	**35.5**	46.7	41.1
Key.Net-SS	37.3	57.4	47.4	Key.Net-MS	60.2	73.6	66.9
w/ ScaleNet	**56.1**	**71.9**	**64.0**	w/ ScaleNet	**62.6**	73.6	**68.1**

Table 3. Mean Average Accuracy (mAA) at 10°on IMC dataset [18].

tasks, wide-baseline stereo and multi-view reconstruction. The benchmark looks to the pose errors under different thresholds and reports the mean Average Accuracy (mAA) of the reconstructions. We follow the evaluation protocol with 2,048 keypoints. Moreover, we select DEGENSAC [4] for geometric verification and first-to-second nearest-neighbour ratio for filtering false-positive matches. For more details on their evaluation refer to [18]. We search for the best configuration in stereo and multi-view independently for each method. We use the validation scenes provided in Image Matching Competition[1] as authors only made public the ground-truth for such scenes. Computing ScaleNet in the test set is not straightforward, since the benchmark had to be modified to accept a different set of features for each pair of images. Thus, as a reference to our method's effect in their benchmark, we report the best results obtained in the three validation scenes, *Reichstag* (74 images), *Sacre Coeur* (100 images), and *St Peters Square* (100 images).

Results in table 3 display the benefits of camera pose error when using ScaleNet. The boost of ScaleNet when working together with a single scale method shows the importance of rectifying the scale difference between images. Even though multi-scale methods look for features on multiple scaled images, the best results are obtained when ScaleNet corrects one of the images before feature extraction. Moreover, ScaleNet brings benefits to current pipelines when the scale change between image pairs is extreme, and therefore, if the scale factor is not significant as in the evaluated scenes, multi-scale feature extraction is sufficient to perform the correspondence search.

C.4. Camera pose

In addition to the results of the main paper, we extend in table 4 the results of the camera pose task. We test our method together with state-of-the-art combination SuperPoint [5] and SuperGlue [14] and report results in the standard Megadepth [7] test split proposed in [16]. This split contains 1,500 image pairs from two different scenes,

[1]https://www.cs.ubc.ca/research/image-matching-challenge

	Pose estimation (AUC)		
	at 5°	at 10°	at 20°
SP + SuperGlue	35.2	54.7	71.6
w/ ScaleNet	**36.3** (+3%)	**55.7** (+2%)	**72.4** (+1%)

Table 4. Camera pose results on Megadepth test split from [16]. When scale changes are small on the image pairs, ScaleNet brings more robustness without hurting the performance.

where image pairs suffer from general viewpoint and illumination challenges but not extreme scale changes. Hence, we see in table 4 that even though ScaleNet does not bring great improvements, it does not either hurt the performance when there are no strong scale changes between the image pairs.

C.5. Dense matching

We also extend the evaluation on dense geometric matching methods, DGC-Net [9] and GLU-Net [17], by reporting the percentage of correct keypoints (PCK) under multiple pixel acceptance thresholds, *i.e.*, 1, 5, and 10 pixels. We use the same sparse correspondence split proposed by [15] in the Megadepth dataset [7]. As in the main paper, we report the results for the full dataset (*All*) and create the *Easy* ($s > 1.2$) and *Hard* ($s > 1.8$) splits, where s indicates scale distortions factor between images. Results in table 5 show that the improvements that ScaleNet brings into the dense methods are similar across all acceptance thresholds, offering extra robustness across all splits, and a large boost when the scale changes between images is strong (*Hard* split).

C.6. Evaluating scale predictions

Despite the evaluation of ScaleNet in different 3D tasks, in this section, we introduce the results in scale prediction accuracy. Although the real impact of ScaleNet is measured in terms of 3D metrics, knowing its scale accuracy gives a better understanding of the capability of our scale predictor.

Protocol. To test the accuracy of our scale predictions, we compute the mean scale ratio, r, between the ground-truth, S_{GT}, and the estimated scale, s. Specifically, we

	All			Easy			Hard		
	PCK-1 (%)	PCK-5 (%)	PCK-10 (%)	PCK-1 (%)	PCK-5 (%)	PCK-10 (%)	PCK-1 (%)	PCK-5 (%)	PCK-10 (%)
DGC-Net	7.4	40.2	51.5	5.3	34.4	48.0	3.2	4.5	9.6
w/ ScaleNet	**7.9 (+7%)**	**41.5 (+3%)**	**53.3 (+4%)**	**5.9 (+11%)**	**36.8 (+7%)**	**51.4 (+7%)**	**3.3 (+3%)**	**20.1 (+347%)**	**28.7 (+199%)**
GLU-Net	21.3	55.5	62.2	19.5	55.4	62.5	1.7	10.8	15.5
w/ ScaleNet	**23.3 (+9%)**	**57.8 (+4%)**	**64.5 (+4%)**	**20.4 (+5%)**	**56.3 (+2%)**	**64.5 (+3%)**	**7.3 (+329%)**	**26.8 (+148%)**	**33.8 (+118%)**

Table 5. Percentage of correct keypoints (PCK) for different pixel thresholds on MegaDepth [7] sparse correspondences. The results are reported for DGC-Net [9] and GLU-Net [17] without and with our ScaleNet, which consistently improves the performance.

Scale Accuracy (r)			
Perfect	Random	Constant	ScaleNet
1.0	4.3	3.2	**1.6**

Table 6. Scale prediction accuracy computed as the mean scale ratios, r, between the ground-truth, S_{GT}, and estimated scales, s.

Time (ms)				
DGC-Net	GLU-Net	R2D2	Key.Net	ScaleNet
35.4	25.1	384.3	640.5	19.8

Table 7. Comparison of computational times in ms for several state-of-the-art methods. We report the multi-scale extraction times for R2D2 and Key.Net/HardNet.

compute the ratio of a scale prediction as $r = s_a/s_b$, $s_a = max(S_{GT}, s)$, and $s_b = min(S_{GT}, s)$. In addition to ScaleNet predictions, we also propose two baselines as a reference: *Random* makes a random prediction with $s \in [0.16, 6]$, and *Constant* uses a fixed scale value such as $s = 1.0$. In addition, we also indicate the results of perfect scale estimation in *Perfect* ($r = 1.0$). We use the test set proposed in the relative camera pose experiment of the main paper (section 5.2).

Results. We report in table 6 the mean scale ratio between ground-truth and predicted scales. ScaleNet predictions ($r = 1.6$) are accurate enough to bring images to scale discrepancies where current local feature extractors can operate. However, even though scale prediction accuracy is relevant, it is hard to quantify. Therefore, we believe that results on downstream applications are needed to understand the impact of ScaleNet image rectification.

C.7. Complexity time

Table 7 introduces the computational time of ScaleNet and other state-of-the-art methods. Although ScaleNet has the fastest inference time, it only predicts a single transformation parameter rather than the flow between images or local keypoints/descriptors. However, we showed that ScaleNet largely benefits other methods if scale changes are presented in the images and increases only 19.8ms the computational time of matching two images. Besides dense geometric matching models, DGC-Net (35.4ms) and GLU-Net (25.1ms), we also provided the times of multi-scale feature extraction of R2D2-MS (384.3ms) and Key.Net-MS/HardNet (640.5ms). As multi-scale methods require a minimum image size to work properly, we report the times when extracting features from an image of size 600×800, and see that the time for ScaleNet is significantly smaller than the feature extractors. Thus, we believe that if scale robustness is needed, the improvements outweigh the computational overhead introduced by our scale correction.

Besides, some strategies could reduce the impact of using scale correction. As discussed in the main paper, ScaleNet could use faster and more efficient feature extractors, *i.e.*, MobileNet [6], or we could precompute ScaleNet backbone features and store them. Catching ScaleNet backbone features would allow us only to run correlation and fully connected layers for every new pairs of images, reducing the complexity of some 3D system, *e.g.*, 3D reconstruction. As a reference, the time on inference when caching backbone features is 1.8ms, meanwhile, the inference time of regular ScaleNet is 19.8ms.

D. Qualitative examples

In this section, we provide some qualitative results from applying ScaleNet in image pairs from the Megadepth dataset [7] that contain strong scale changes. In figure 4, we show the matches that were obtained for each method. We only plot the matches that survive the Lowe's ratio test [8] and MAGSAC [2] geometric fitting method. As in the paper, to avoid the effect of a higher number of keypoints due to one image being upsampled, we only use the top 2,048 keypoint candidates. We observe that all methods benefit from the scale correction when images present severe scale variations. Furthermore, examples show difficult image pairs where the multi-scale methods, SIFT [8], R2D2 [13], and Key.Net/HardNet [3, 11], could not handle the extreme-scale change and found few or non-matches between pairs. We see that on those examples, methods benefit largely from scale rectification. Similarly, SuperPoint [5] could only find correct matches if the scale factor between images is corrected.

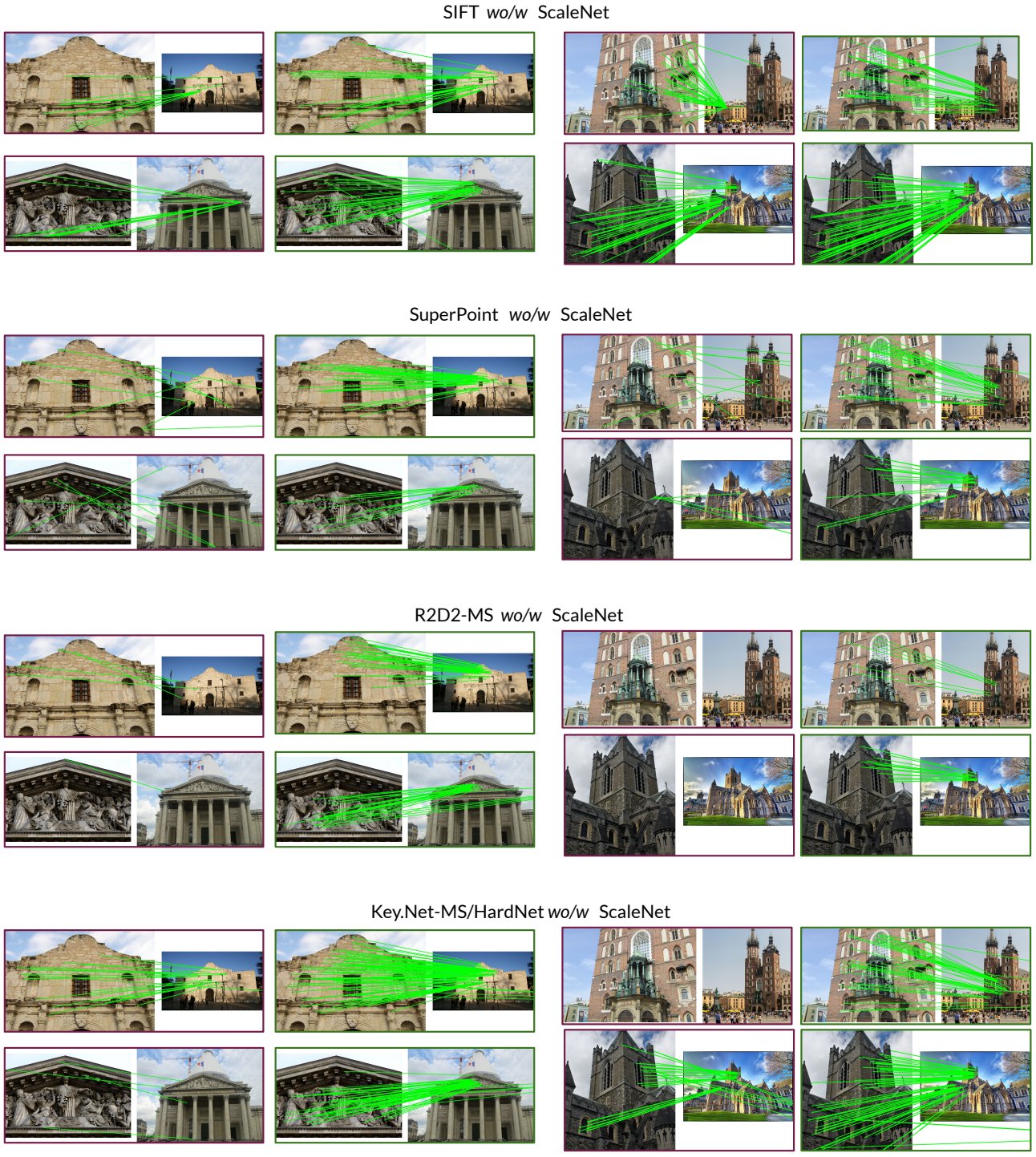

Figure 4. Comparison of matches **without** (red boxes) and **with** (green boxes) the scale rectification proposed by ScaleNet. We only plot the matches that agree with the camera pose computed by the methods, either with or without ScaleNet. The examples display extreme scale conditions, and hence, we observe that the number of matches notably increases when applying ScaleNet's image rectification.

References

[1] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5173–5182, 2017. 3

[2] Daniel Barath, Jiri Matas, and Jana Noskova. Magsac: marginalizing sample consensus. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10197–10205, 2019. 5

[3] Axel Barroso-Laguna, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Key.net: Keypoint detection by handcrafted and learned cnn filters. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 3, 5

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

[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, 4, 5

[6] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. 5

[7] 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 3302–3312, 2018. 1, 4, 5

[8] David G. Lowe. Distinctive image features from scale-invariant keypoints. In *International booktitle of computer vision*, 2004. 5

[9] Iaroslav Melekhov, Aleksei Tiulpin, Torsten Sattler, Marc Pollefeys, Esa Rahtu, and Juho Kannala. DGC-Net: Dense geometric correspondence network. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019. 4, 5

[10] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, number 10, pages 1615–1630. IEEE, 2005. 3

[11] Anastasiia Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. In *Advances in Neural Information Processing Systems*, pages 4826–4837, 2017. 3, 5

[12] Dmytro Mishkin, Jiri Matas, and Michal Perdoch. Mods: Fast and robust method for two-view matching. In *Computer Vision and Image Understanding*, volume 141, pages 81–93. Elsevier, 2015. 1

[13] Jerome Revaud, Philippe Weinzaepfel, César De Souza, and Martin Humenberger. R2d2: Repeatable and reliable detector and descriptor. In *Advances in Neural Information Processing Systems*, 2019. 5

[14] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 4

[15] Xi Shen, François Darmon, Alexei A Efros, and Mathieu Aubry. Ransac-flow: generic two-stage image alignment. *arXiv preprint arXiv:2004.01526*, 2020. 4

[16] 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. 4

[17] Prune Truong, Martin Danelljan, and Radu Timofte. GLU-Net: Global-local universal network for dense flow and correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 4, 5

[18] Jin Yuhe, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. Image matching across wide baselines: From paper to practice. In *International Journal of Computer Vision*, 2020. 3, 4