Supplementary Material
Learning Graph Regularisation for Guided Super-Resolution

A. Hyperparameters

We report the hyperparameter settings for the evaluation of our proposed method for RGB-guided depth map super-resolution. Hyperparameters are reported for each dataset and method. The experiments were conducted on the three RGB-D datasets Middlebury [6, 13–16], NYUv2 [12] and DIML [1, 9–11] with the following methods: Guided Filter (GF) [4], the Static/Dynamic filter (SD) [3], the Pixtransform [2], the MSG-Net [7], the Deformable Kernel Network (DKN) and its fast version (FDKN) [8], the PMBANet [17], and the Fast Depth Super-Resolution (FDSR) [5].

We train all learned methods using the Adam optimiser with default parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ and a batch size of 8. For the Middlebury dataset, we train for 2,500 epochs with an initial learning rate of 10^{-4} and reduce it by factor 0.9 every 100 epochs. For the NYUv2 dataset, we train for 250 epochs with the same initial learning rate that is reduced by a factor of 0.9 every 10 epochs. For the DIML dataset, the methods are trained for 150 epochs, again using an initial learning rate of 10^{-4} with a reduction by a factor of 0.9 every 6 epochs. For all learned methods we additionally tested the initial learning rate and learning rate schedule proposed by the respective authors (if available), and used the best configuration. For FDSR, we therefore deviated from our default settings and chose an initial learning rate of $5.0 \cdot 10^{-4}$, with a $0.5\times$ reduction every 80,000 *iterations*. For PMBANet, we found a $0.1\times$ reduction every 100 (1000, 60) epochs for NYUv2 (Middlebury, DIML) to work best. Note that we did not conduct any hyperparameter search for our proposed method.

We also report the hyperparameters used for the non-learned approaches. We used the following values for the SD filter, which we found to be the best performing among the tested configurations: $\lambda = 0.1$, $\sigma_g = 60$ and $\sigma_u = 30$. For Pixtransform we used the hyperparameters suggested in the original manuscript.

B. Qualitative Results

In Figures A1, A2 and A3, we provide additional examples for qualitative comparison between our method and selected methods from our quantitative evaluation; for the Middlebury, NYUv2 and DIML datasets respectively. We provide two additional examples for each upsampling factor and dataset combination. These results further confirm that the predictions obtained with our method compare favourably to existing methods. In particular, we observe the magnitude of errors for our method to be smaller compared to the other approaches, especially along edges. For many examples, we can additionally see that our method facilitates smooth depth predictions in continuous areas, whereas other methods exhibit higher noise. Visually, the differences seem small for an upsampling factor of $\times 4$, however with larger upsampling factors they become more apparent.

C. Learning Graph Weights

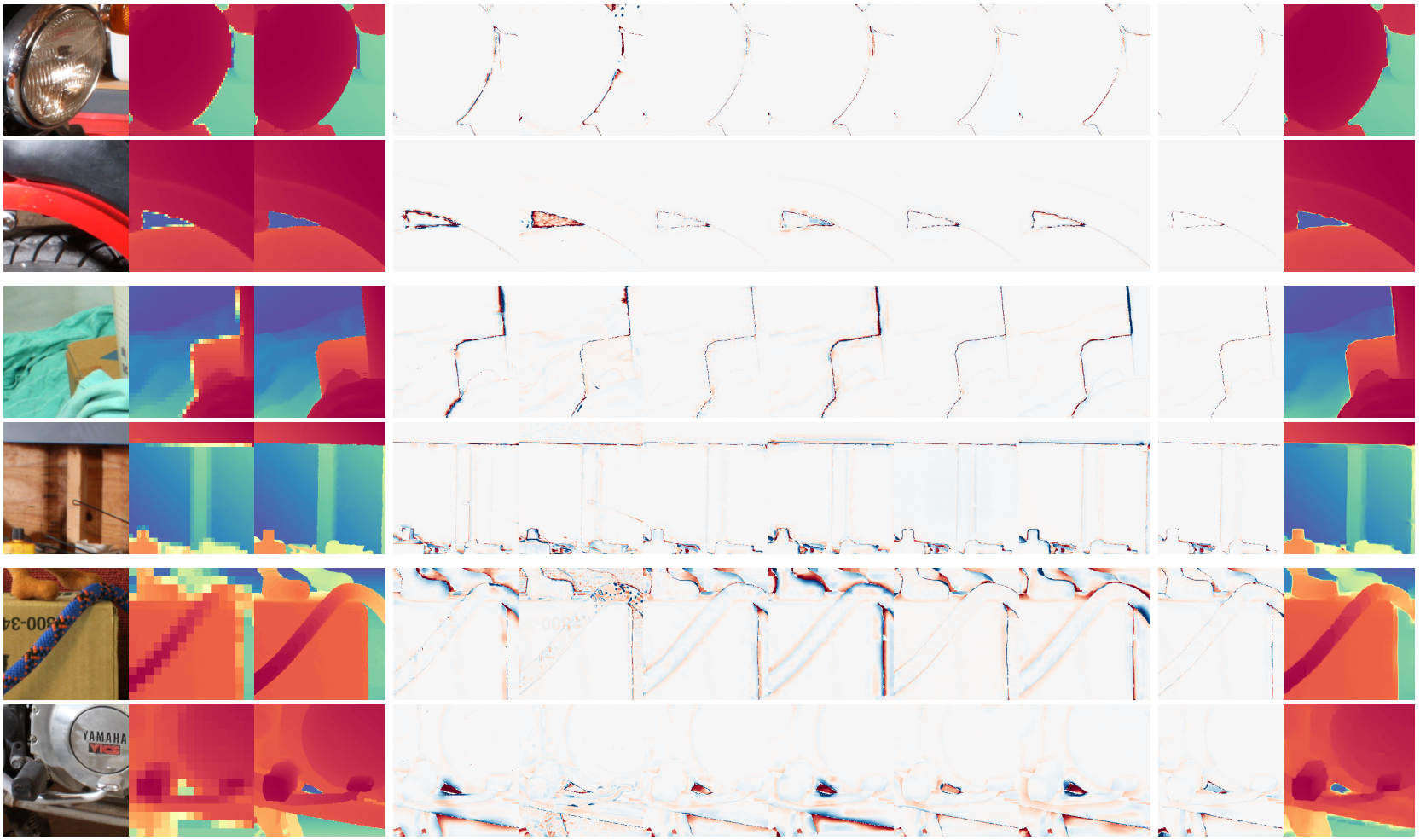
In Figure A4 we show additional examples of the difference between the total affinity of each pixel to its four neighbours when the graph is defined on colour features against the graph defined on learned features. These results further show that our model is able to encourage the optimiser to provide smooth predictions in areas without depth discontinuities. On the other hand, the model also shows low predicted weights in areas that should not be smoothed, thus providing crisper predictions.

D. Forward Pass Timing

We provide some time statistics for the forward pass of the proposed method in Table A1. For reference: FDKN [8] takes about 10 ms for a forward pass independently of the scaling factor. The forward pass for the feature extractor of our proposed method also takes about 15 ms. Pixtransform [2], by far the slowest method among the compared ones, takes more than 120 seconds for a 256^2 patch. Our Python implementation of the SD filter [3] takes few seconds (this also depends on the upsampling factor), although it is not directly comparable to our method as the SD filter implementation does not take advantage of GPU acceleration. The upsampling factor plays a major role for the runtime of our method, because larger upsampling factors lead to downsampling operators $\boldsymbol{D}$ that are less sparse, increasing the time required to solve the linear system. Note that although our method was implemented to leverage GPU parallelisation, there is still potential for further optimisations that could improve runtime performance.

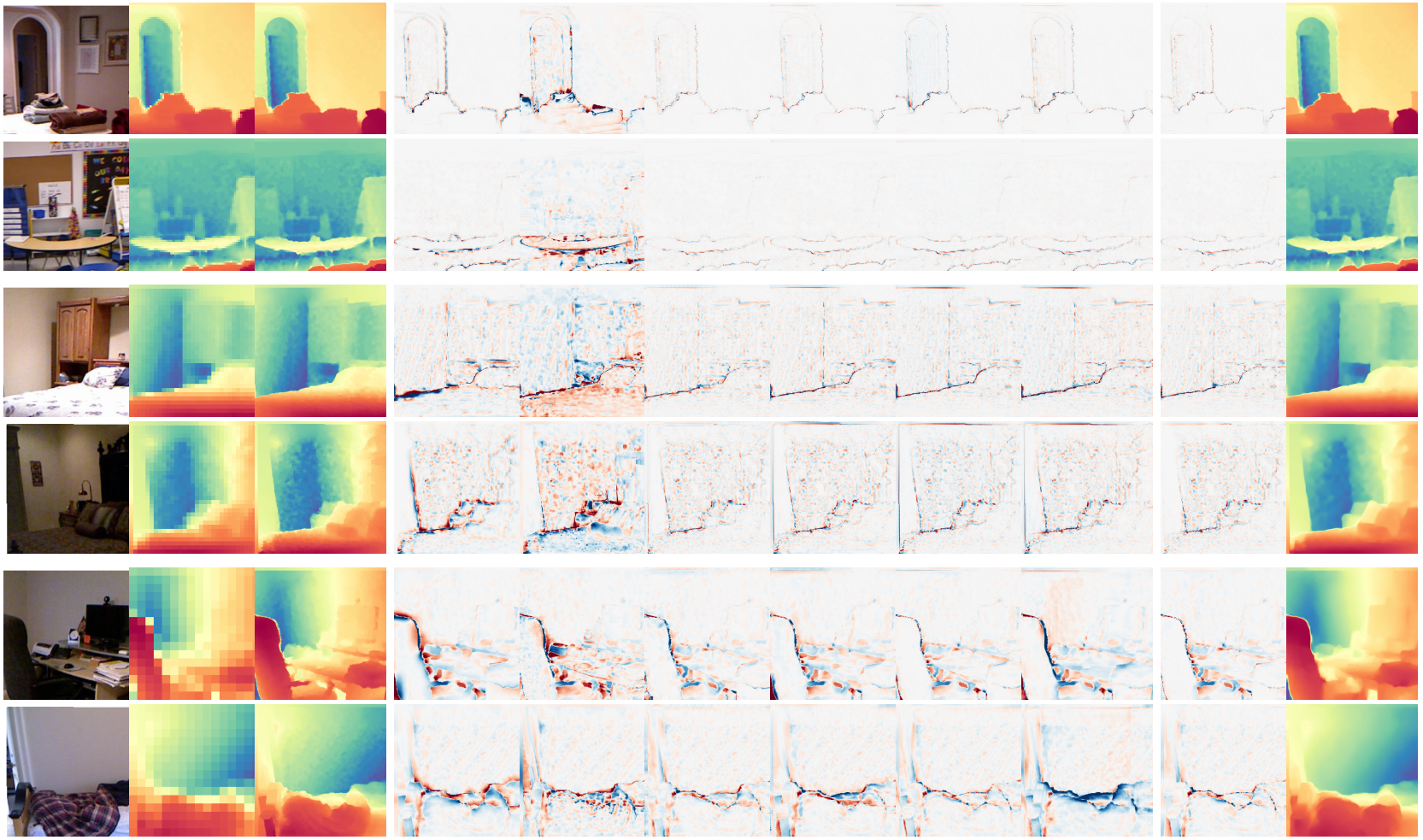
	$\times 4$	$\times 8$	$\times 16$
forward time	79 (21)	111 (35)	305 (92)

Table A1. Forward pass times of our proposed method. Numbers represent the mean time and (standard deviation) measured in milliseconds, computed over the NYUv2 test set on single patches of 256^2 pixels, on an NVIDIA GeForce RTX 2080 Ti.



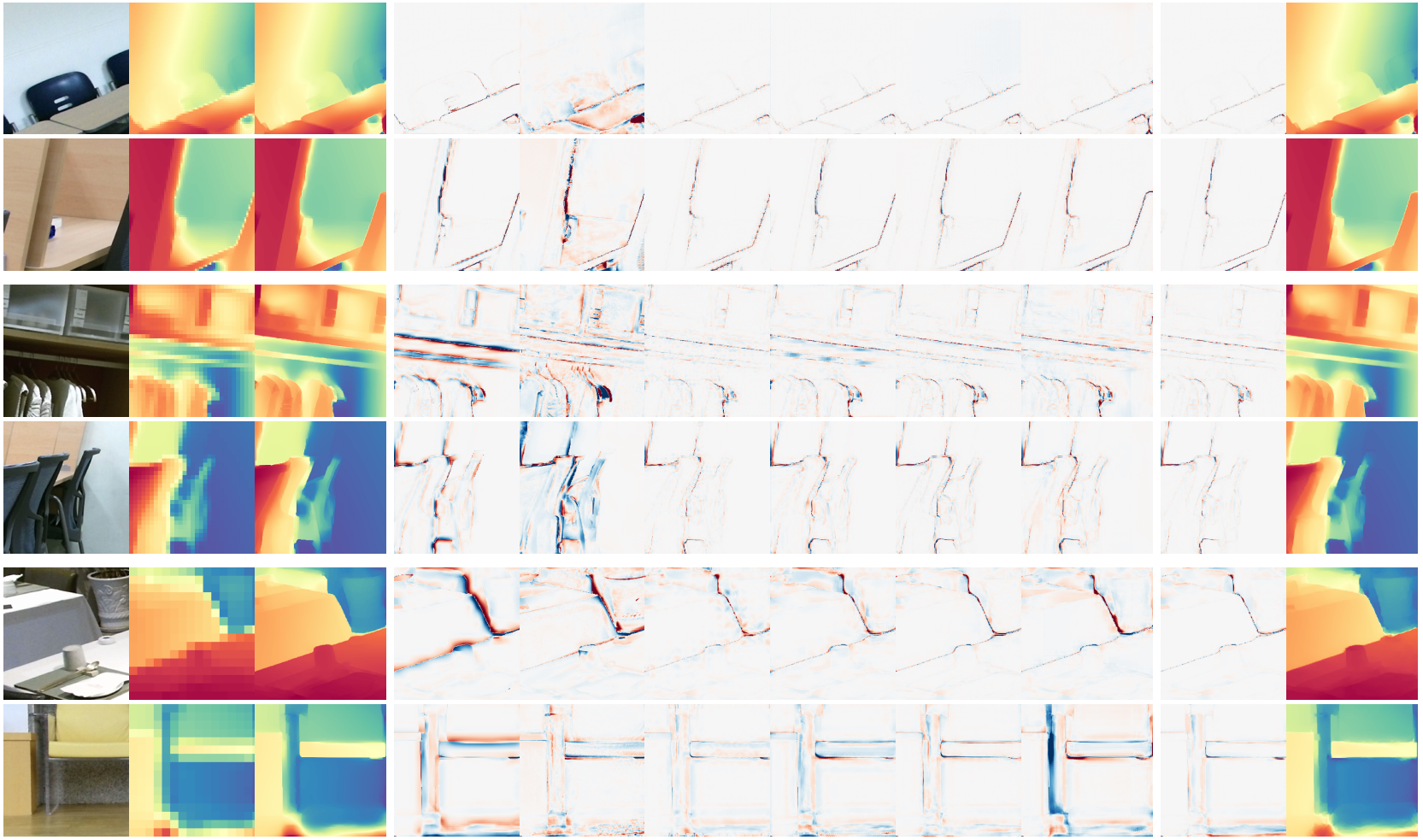
(a) Guide. (b) Source. (c) GT. (d) SD. (e) Pixtrans. (f) MSGNet. (g) FDKN. (h) PMBA. (i) FDSR. (j) Ours. (k) Ours Pred.

Figure A1. Additional qualitative comparison of upsampled depth maps for the Middlebury dataset [6, 13–16]. From top to bottom each group of two rows shows the error of upsampled images, defined as the difference between the prediction and the ground truth, for upsampling factors $\times 4$, $\times 8$ and $\times 16$ respectively. From left to right, the first group of columns are (a) Guide, (b) Source and (c) Ground Truth; the second group includes selected methods from our quantitative evaluation, (d) SDF [3], (e) Pixtransform [2], (f) MSGNet [7], (g) FDKN [8], (h) PMBANet [17] and (i) FDSR [5]; the last two columns represent (j) the error for the prediction of our model and (k) the prediction itself.



(a) Guide. (b) Source. (c) GT. (d) SD. (e) Pixtrans. (f) MSGNet. (g) FDKN. (h) PMBA. (i) FDSR. (j) Ours. (k) Ours Pred.

Figure A2. Additional qualitative comparison of upsampled depth maps for the NYUv2 dataset [12]. From top to bottom each group of two rows shows the error of upsampled images, defined as the difference between the prediction and the ground truth, for upsampling factors $\times 4$, $\times 8$ and $\times 16$ respectively. From left to right, the first group of columns are (a) Guide, (b) Source and (c) Ground Truth; the second group includes selected methods from our quantitative evaluation, (d) SDF [3], (e) Pixtransform [2], (f) MSGNet [7], (g) FDKN [8], (h) PMBANet [17] and (i) FDSR [5]; the last two columns represent (j) the error for the prediction of our model and (k) the prediction itself.



(a) Guide. (b) Source. (c) GT. (d) SD. (e) Pixtrans. (f) MSGNet. (g) FDKN. (h) PMBA. (i) FDSR. (j) Ours. (k) Ours Pred.

Figure A3. Additional qualitative comparison of upsampled depth maps for the DIML dataset [1, 9–11]. From top to bottom each group of two rows shows the error of upsampled images, defined as the difference between the prediction and the ground truth, for upsampling factors $\times 4$, $\times 8$ and $\times 16$ respectively. From left to right, the first group of columns are (a) Guide, (b) Source and (c) Ground Truth; the second group includes selected methods from our quantitative evaluation, (d) SDF [3], (e) Pixtransform [2], (f) MSGNet [7], (g) FDKN [8], (h) PMBANet [17] and (i) FDSR [5]; the last two columns represent (j) the error for the prediction of our model and (k) the prediction itself.

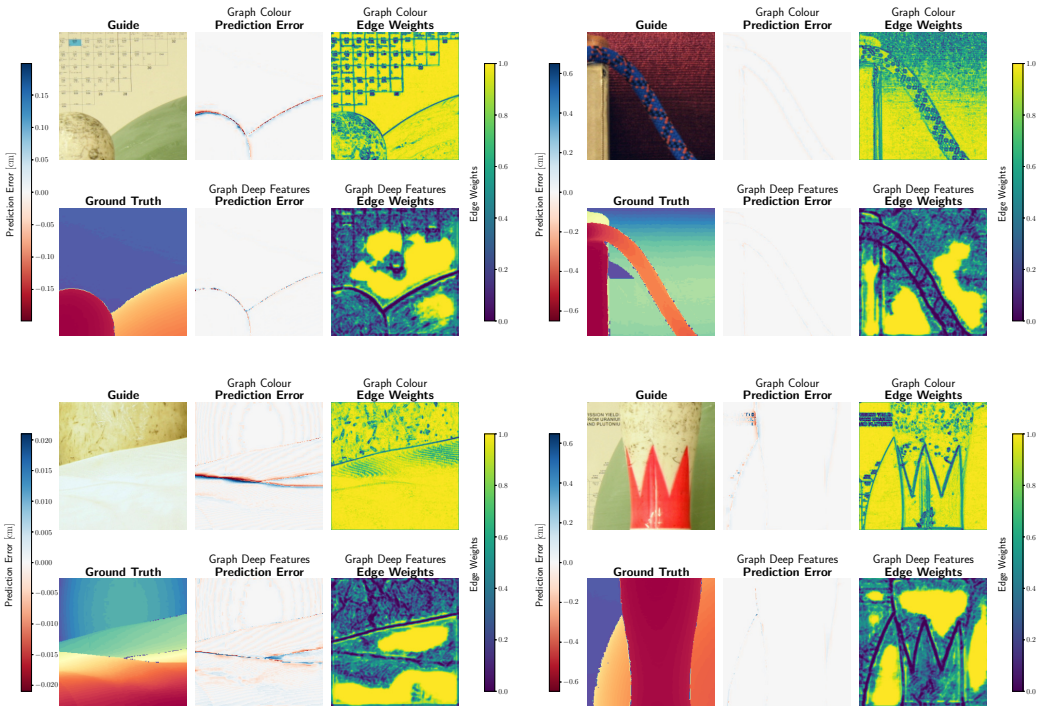


Figure A4. Additional examples of the importance of learned edge potentials. We visualise the total affinity of each pixel to its four neighbours when derived from raw colour (top) or from deep features (bottom). Examples are from the Middlebury test set.

References

[1] Cho, Jaehoon and Min, Dongbo and Kim, Youngjung and Sohn, Kwanghoon. Deep monocular depth estimation leveraging a large-scale outdoor stereo dataset. *Expert Systems with Applications*, 2021. 1, 4

[2] Riccardo de Lutio, Stefano D'Aronco, Jan D. Wegner, and Konrad Schindler. Guided super-resolution as pixel-to-pixel transformation. In *ICCV*, 2019. 1, 2, 3, 4

[3] Bumsub Ham, Minsu Cho, and Jean Ponce. Robust guided image filtering using nonconvex potentials. *TPAMI*, 2018. 1, 2, 3, 4

[4] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. *TPAMI*, 2013. 1

[5] Lingzhi He, Hongguang Zhu, Feng Li, Huihui Bai, Runmin Cong, Chunjie Zhang, Chunyu Lin, Meiqin Liu, and Yao Zhao. Towards fast and accurate real-world depth super-resolution: Benchmark dataset and baseline. In *CVPR*, 2021. 1, 2, 3, 4

[6] Heiko Hirschmüller and Daniel Scharstein. Evaluation of cost functions for stereo matching. In *CVPR*, 2007. 1, 2

[7] Tak-Wai Hui, Chen Change Loy, and Xiaoou Tang. Depth map super-resolution by deep multi-scale guidance. In *ECCV*, 2016. 1, 2, 3, 4

[8] Beomjun Kim, Jean Ponce, and Bumsub Ham. Deformable kernel networks for joint image filtering. *IJCV*, 2021. 1, 2, 3, 4

[9] Sunok Kim, Dongbo Min, Bumsub Ham, Seungryong Kim, and Kwanghoon Sohn. Deep stereo confidence prediction for depth estimation. In *ICIP*, 2017. 1, 4

[10] Youngjung Kim, Bumsub Ham, Changjae Oh, and Kwanghoon Sohn. Structure selective depth superresolution for rgb-d cameras. *TIP*, 2016. 1, 4

[11] Youngjung Kim, Hyungjoo Jung, Dongbo Min, and Kwanghoon Sohn. Deep monocular depth estimation via integration of global and local predictions. *TIP*, 2018. 1, 4

[12] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 1, 3

[13] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nešić, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *GCPR*, 2014. 1, 2

[14] Daniel Scharstein and Chris Pal. Learning conditional random fields for stereo. In *CVPR*, 2007. 1, 2

[15] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *CVPR*, 2001. 1, 2

[16] Daniel Scharstein, Richard Szeliski, and Ramin Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 2002. 1, 2

[17] Xinchen Ye, Baoli Sun, Zhihui Wang, Jingyu Yang, Rui Xu, Haojie Li, and Baopu Li. PMBANet: Progressive multi-branch aggregation network for scene depth super-resolution. *TIP*, 2020. 1, 2, 3, 4