Degradation-agnostic Correspondence from Resolution-asymmetric Stereo Supplementary Material

Xihao Chen Zhiwei Xiong Zhen Cheng
Jiayong Peng Yueyi Zhang Zheng-Jun Zha
University of Science and Technology of China

This supplementary document is organized as follows:
Sec. 1 provides more visual results of the real-world dataset.
Sec. 2 provides the visualization of different feature maps and more analysis.
Sec. 3 provides additional implementation details of different comparison methods and ours.
Sec. 4 provides supplementary validation results of the self-boosting strategy.

1. More Visual Results of The Real-world Dataset

We show the visual results of the other three testing scenes from the self-collected real-world dataset in Fig. 1.

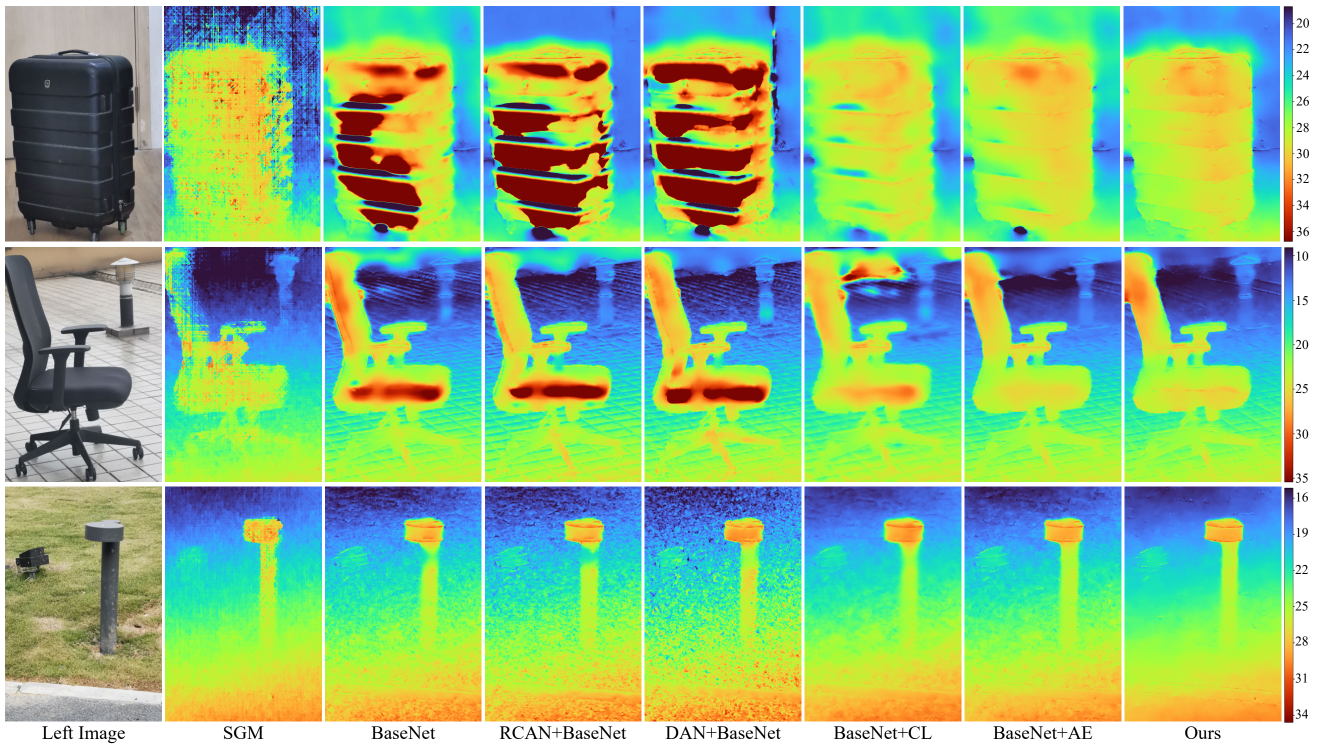

Figure 1. Disparity maps of three testing scenes from the self-collected real-world dataset. The colorbars show the value of disparity.

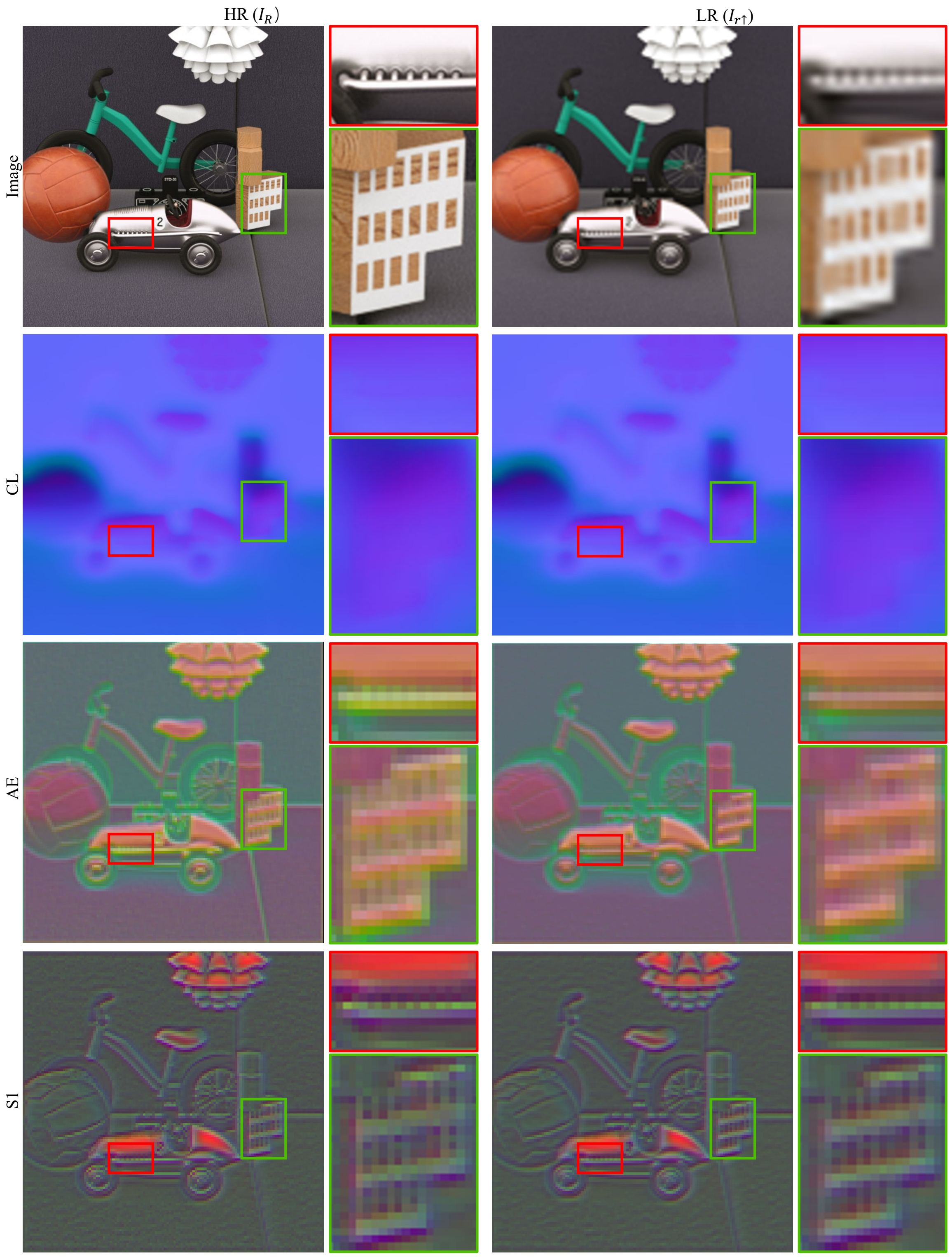

Figure 2. The HR and LR feature maps (2nd to 4th row) generated from the HR and LR right view images (1st row) by different feature-metric learning methods. Feature maps are visualized by projecting the high-dimensional feature spaces onto a three-dimensional space (*i.e.*, RGB) using PCA. Note that, due to dimension reduction, the original feature spaces may contain more information than displayed here. This scene is from the Inria_SLFD dataset and simulated with an asymmetric factor of 4 and under the BIC degradation.

2. A Closer Look at Different Feature Spaces

2.1. Feature Map Visualization

We use PCA to project the high-dimensional feature spaces onto a three-dimensional space (*i.e.*, RGB) and visualize different feature maps in Fig. 2. As can be seen, CL generates blur feature maps for both HR and LR images, since the wrong positive samples of the contrastive loss make the feature network assign similar features to pixels belonging to different scene points. Although these blur feature maps contribute to a high PSNR value, they are not suitable for matching, demonstrated by the unsatisfactory 3PE result. AE learns relatively discriminative features for matching, but it generates notably different features for HR and LR images at the regions with high-frequency components, as shown in the marked rectangles. In contrast, S1 generates more consistent features in these regions with fewer color/structure deviations, thus presenting a higher PSNR value. It is worth noting that, compared with the image space where the HR and LR patches are drastically different, the visual gap of corresponding feature maps in the feature space of S1 is much smaller, which indicates the degradation-agnostic property. Meanwhile, S1 produces the most discriminative features for different regions, including those with tiny textures, such as the surface of the sofa, which indicates the matching-specific property and is justified by the best 3PE result.

2.2. Analysis from The Noisy Label Perspective

Typically, a stereo matching network Φ is comprised of a feature extractor Φ_F and a matching module Φ_M. When trained with the symmetric loss (S3), Φ_F can extract discriminative features to compute a cost volume, while Φ_M can regularize the cost volume and regress an accurate disparity map. Interestingly, Φ_F can maintain its functionality when trained with the asymmetric loss (S1) as mentioned in the paper.

It could be associated with the theory of learning with noisy labels [6, 8]: If a network architecture suits a specific task, training with noisy labels can induce useful features, even when the model generalizes poorly. In other words, the earlier layers of the model are less negatively affected by noisy labels than the later layers. For resolution-asymmetric stereo matching, the asymmetric loss provides noisy supervision due to the photometric inconsistency. Therefore, as the theory indicates, Φ_F (*i.e.*, the earlier layers) would be more robust to the noisy supervision, while Φ_M (*i.e.*, the later layers) would be more negatively affected.

To validate this explanation, a proof-of-concept experiment is conducted on the Inria_SLFD dataset with an asymmetric factor of 4. Specifically, with the symmetric loss, we finetune the feature extractor (or matching module) pre-trained under the setting of S1 while fixing the pre-trained matching module (or feature extractor). The input of the network is still asymmetric, and thus the performance upper bound is S3 (6.39%). The 3PE results of finetuning the feature extractor and finetuning the matching module are 7.67% and 6.60%, respectively. As expected, in an ideal case, finetuning the matching module gets much closer to the upper bound (0.21% gap) than finetuning the feature extractor (1.28% gap). It suggests that the matching module is indeed more negatively affected by the noisy supervision of the asymmetric loss in practice, as explained above.

3. Implementation Details

3.1. Feature-metric Learning Methods

CL. CL is a solution that jointly learns resolution-asymmetric stereo matching and dense features in an unsupervised manner, which is adapted from [18]. As illustrated in Fig. 3(a), CL consists of a stereo matching network Φ_{disp}, which predicts the disparity map d_L between a stereo pair (*i.e.*, I_L and $I_{r\uparrow}$), and a feature network Φ_{feat}, which predicts dense feature maps (*i.e.*, F_L and $F_{r\uparrow}$). CL jointly utilizes the photometric loss $\mathcal{L}_{pm}$, the feature-metric loss $\mathcal{L}_{fm}$, and the weighted smoothness loss $\mathcal{L}_{sm}$ to train Φ_{disp}, and utilizes a contrastive loss $\mathcal{L}_C$ to train Φ_{feat}. For $\mathcal{L}_C$, the set of positive correspondences is provided by d_L, while the set of negative examples is generated by using a spatial negative mining technique [17]. For $\mathcal{L}_{fm}$, it is defined on the features from Φ_{feat}.

AE. As illustrated in Fig. 3(b), AE pre-trains an auto-encoder network, which learns to reconstruct its input image, with a reconstruction loss $\mathcal{L}_{recon}$. After pre-training, the encoder of the auto-encoder produces the feature space of AE, which can be used to define the feature-metric loss $\mathcal{L}_{fm}$ for stereo matching. As in [15], two additional losses are added to improve the loss landscape of $\mathcal{L}_{fm}$, including a discriminative loss $\mathcal{L}_{dis}$ and a convergent loss $\mathcal{L}_{cvt}$. Specifically, $\mathcal{L}_{dis}$ encourages the features to present large gradients, while $\mathcal{L}_{cvt}$ imposes a smoothness constraint on the gradients of the features to ensure large convergence radii for $\mathcal{L}_{fm}$.

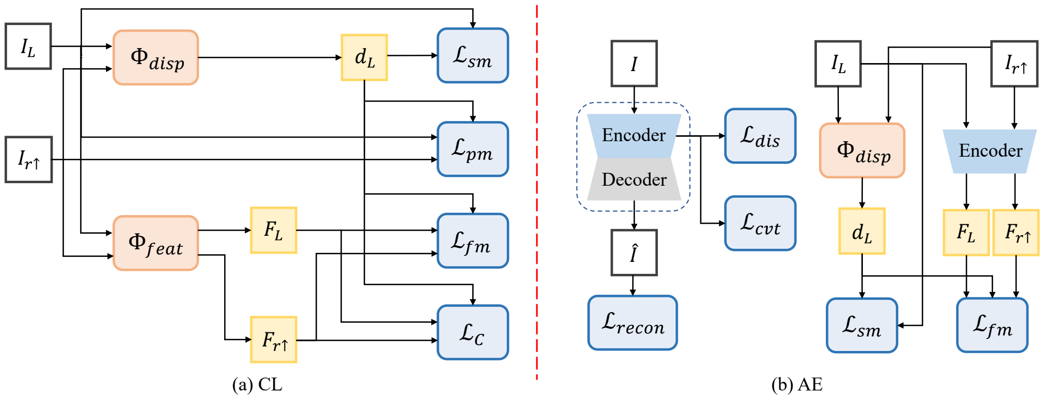

Figure 3. Illustration of two feature-metric learning methods.

3.2. Our Self-boosting Strategy

We provide the detailed algorithm of our self-boosting strategy in Algorithm 1.

Algorithm 1 Self-boosting Strategy

```
Require: Resolution-asymmetric stereo dataset ${I_L, I_{r\uparrow}}_N$, the number of stages $K$, the mini-batch size $n$.
 1: Randomly initialize a stereo matching network $\Phi(\cdot;\theta_F^0,\theta_M^0)$;
 2: while $\Phi(\cdot;\theta_F^0,\theta_M^0)$ does not converge do
 3:     Randomly choose mini-batch ${I_L, I_{r\uparrow}}_n$;
 4:     Feed forward ${I_L, I_{r\uparrow}}_n$;
 5:     Update $\theta_F^0$ and $\theta_M^0$ with $\mathcal{L}_{pm}$.
 6: end while
 7: for $k \leftarrow 1$ to $K$ do
 8:     Initialize a new stereo matching network $\Phi(\cdot;\theta_F^k,\theta_M^k)$ with $\theta_F^{k-1}$ and $\theta_M^{k-1}$;
 9:     Fix the feature extractor $\Phi_F(\cdot;\theta_F^{k-1})$ of the network $\Phi(\cdot;\theta_F^{k-1},\theta_M^{k-1})$;
10:     while $\Phi(\cdot;\theta_F^k,\theta_M^k)$ does not converge do
11:         Randomly choose mini-batch ${I_L, I_{r\uparrow}}_n$;
12:         Feed forward ${I_L, I_{r\uparrow}}_n$;
13:         Compute $\mathcal{L}_{fm}^{k-1}$ with $F_L^{k-1} = \Phi_F(I_L;\theta_F^{k-1})$ and $F_{r\uparrow\to L}^{k-1} = \Phi_F(I_{r\uparrow\to L};\theta_F^{k-1})$;
14:         Update $\theta_F^k$ and $\theta_M^k$ with $\mathcal{L}_{fm}^{k-1}$.
15:     end while
16: end for
17: return $\Phi(\cdot;\theta_F^K,\theta_M^K)$.
```

3.3. Generation of Gaussian Kernels

We generate the isotropic and anisotropic Gaussian kernels following [7, 16]. For the isotropic one, its variance is set as 4.0 and its kernel size is set as 21×21. The anisotropic Gaussian kernel is determined by a covariance matrix

$$\mathbf{\Sigma} = \begin{bmatrix} \cos(\Theta) & -\sin(\Theta) \\ \sin(\Theta) & \cos(\Theta) \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \cos(\Theta) & \sin(\Theta) \\ -\sin(\Theta) & \cos(\Theta) \end{bmatrix}, \tag{1}$$

where Θ, λ_1, and λ_2 are sampled from $U[0, \pi]$, $U[1, 10]$, and $U[1, \lambda_1]$, respectively. The kernel width is set as 15×15.

3.4. Dataset Division

We simulate four resolution-asymmetric stereo datasets, two from the widely used stereo datasets Middlebury [4] and KITTI2015 [9] and two from the light field datasets Inria_SLFD [14] and HCI [5].

Inria_SLFD. Inria_SLFD is a synthetic light field dataset with 53 *sparsely* sampled light field scenes, each of which has 9×9 views. We randomly split 13 scenes as the testing set and the others as the training set. For each scene, we take the view (5,5) as the left view of a stereo pair and the view (5,6) as the right view.

HCI. HCI is the other light field dataset with 22 *densely* sampled light field scenes, each of which has 9×9 views. We randomly split 7 scenes as the testing set and the others as the training set. For each scene, we take the view (5,5) as the left view of a stereo pair and the view (5,9) as the right view. We do not choose two adjacent views since the baseline is too small.

Middlebury. The Middlebury stereo datasets were published in five separate works in [4, 10–13]. We adopt the 2005 dataset (6 scenes) as the testing set and the 2006 dataset (24 scenes) as the training set, since both datasets have a baseline closer to the configuration on smartphones.

KITTI2015. KITTI2015 [9] contains 200 training stereo pairs with ground-truth disparity labels and 200 testing stereo pairs without ground-truth disparity labels. To facilitate the evaluation, we take the original training pairs as the testing set and take the original testing pairs as the training set.

3.5. Training Details

For Inira_SLFD, HCI, and Middlebury, we set the batch size as 6, the resolution of input images as 512×512 (after upsampling/SR for the LR view) during training. For KITTI2015, we set the batch size as 4, the resolution of input images as 640×320. Data augmentations are performed following [2]. In specific, we horizontally flip the input stereo pair and swap its two views to obtain the mirrored version, with a 50% chance. Besides, we add color augmentations by adjusting gamma, brightness, and color in the ranges [0.8,1.2], [0.8,1.2], and [0.95,1.05], respectively. Note that color jitting is performed for each color channel separately.

For methods using the photometric loss (*i.e.*, BaseNet, RCAN+BaseNet, and DAN+BaseNet), λ is set as 0.05. We adopt the hyper-parameter settings suggested in the original papers [15, 18] for BaseNet+CL and BaseNet+AE. For our method, we set $\lambda = 1$ for Inira_SLFD, HCI, and Middlebury, and $\lambda = 0.5$ for KITTI2015. For SGM [3], we use the function $\mathrm{disparitySGM}$ in the Matlab Toolbox.

3.6. Network Architecture

The default backbone network of all learning-based solutions is the popular PSMNet [1]. Considering the limited training samples, we reduce the network capacity. Table 1 shows the detailed architecture of the modified PSMNet.

Table 1. The detailed architecture of the modified PSMNet. D denotes the maximum disparity. $[\cdot]$ indicates that residual connection is adopted. "Dila" indicates the dilated convolution.

Name	Layer Setting	Output dimension
input		$H \times W \times 3$
Feature Extractor		
conv0_x	3×3, 16 3×3, 16 3×3, 16	$\frac{1}{2}H \times \frac{1}{2}W \times 16$
conv1_x	$\begin{bmatrix} 3 \times 3, 16 \\ 3 \times 3, 16 \end{bmatrix} \times 2$	$\frac{1}{2}H \times \frac{1}{2}W \times 16$
conv2_x	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 8$	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
conv3_x	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\frac{1}{4}H \times \frac{1}{4}W \times 64$
conv4_x	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$, dila = 2	$\frac{1}{4}H \times \frac{1}{4}W \times 64$
branch_1	16×16 avg. pool 3×3, 16 bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 16$
branch_2	8×8 avg. pool 3×3, 16 bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 16$
concat[conv2_8, conv4_2,branch_1,branch_2]		$\frac{1}{4}H \times \frac{1}{4}W \times 128$
fusion	3×3, 64 1×1, 16	$\frac{1}{4}H \times \frac{1}{4}W \times 16$
Cost volume		
Concat left and shifted right		$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 32$
Matching Module		
3Dconv0	$3 \times 3 \times 3, 16$ $3 \times 3 \times 3, 16$	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 16$
3Dconv1	$\begin{bmatrix} 3 \times 3 \times 3, 16 \\ 3 \times 3 \times 3, 16 \end{bmatrix}$	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 16$
3Dstack_1	$3 \times 3 \times 3, 32$ $3 \times 3 \times 3, 32$	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 32$
3Dstack_2	$3 \times 3 \times 3, 32$ $3 \times 3 \times 3, 32$	$\frac{1}{16}D \times \frac{1}{16}H \times \frac{1}{16}W \times 32$
3Dstack_3	deconv $3 \times 3 \times 3, 32$ add 3Dstack_1	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 32$
3Dstack_4	deconv $3 \times 3 \times 3, 16$ add 3Dconv1	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 16$
output	$3 \times 3 \times 3, 16$ $3 \times 3 \times 3, 1$	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 1$
upsampling	Bilinear interpolation	$D \times H \times W$
	Disparity regression	$H \times W$

4. More Validation Results of The Self-boosting Strategy

We provide supplementary validation results of the self-boosting strategy. On the one hand, we quantitatively evaluate the performance of the stereo matching networks at different stages. Table 2 shows the results on different datasets simulated with an asymmetric factor of 4 and under the BIC degradation. As can be seen, the network is progressively improved with the increase of stage number k. On the other hand, we also provide the ablation of the self-boosting strategy on the self-collected real-world dataset with visual results. As shown in Fig. 4, our method with the self-boosting strategy produces sharper edges. It indicates that matching ambiguities are better resolved with the help of the strengthened feature-metric consistency.

Table 2. Validation of the self-boosting strategy on different simulated datasets. The 3PE (%) / EPE metrics are evaluated.

Dataset	Stage Number k			
	0	1	2	3
HCI	5.95/0.891	4.65/0.703	4.29/0.679	**4.08/0.637**
Middlebury	8.72/1.363	7.08/1.187	6.00/1.117	**5.78/1.088**
KITTI2015	11.32/2.014	9.33/1.869	9.14/1.827	**8.66/1.801**

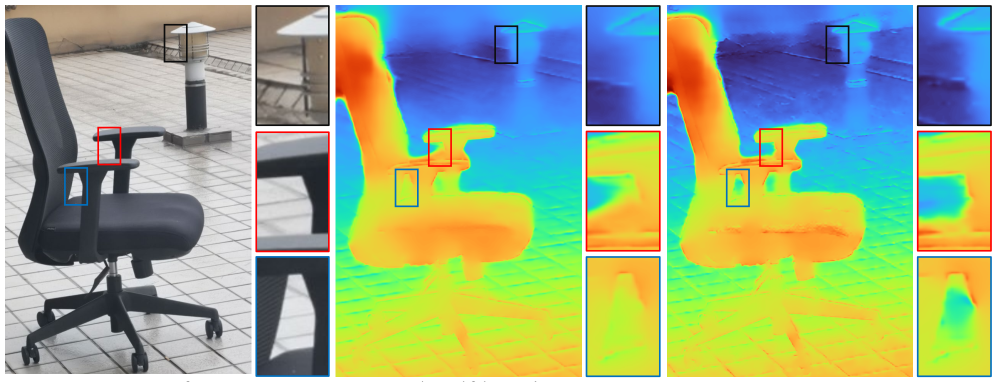

Figure 4. Validation of the self-boosting strategy on the self-collected real-world dataset.

References

[1] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *CVPR*, 2018. 5

[2] Clement Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017. 5

[3] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, 2007. 5

[4] Heiko Hirschmuller and Daniel Scharstein. Evaluation of cost functions for stereo matching. In *CVPR*, 2007. 5

[5] Katrin Honauer, Ole Johannsen, Daniel Kondermann, and Bastian Goldluecke. A dataset and evaluation methodology for depth estimation on 4d light fields. In *ACCV*, 2016. 5

[6] Jingling Li, Mozhi Zhang, Keyulu Xu, John P Dickerson, and Jimmy Ba. Noisy labels can induce good representations. *arXiv preprint arXiv:2012.12896*, 2020. 3

[7] Zhengxiong Luo, Yan Huang, Shang Li, Liang Wang, and Tieniu Tan. Unfolding the alternating optimization for blind super resolution. In *NeurIPS*, 2020. 5

[8] Hartmut Maennel, Ibrahim M Alabdulmohsin, Ilya O Tolstikhin, Robert Baldock, Olivier Bousquet, Sylvain Gelly, and Daniel Keysers. What do neural networks learn when trained with random labels? In *NeurIPS*, 2020. 3

[9] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *CVPR*, 2015. 5

[10] 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. 5

[11] Daniel Scharstein and Chris Pal. Learning conditional random fields for stereo. In *CVPR*, 2007. 5

[12] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1):7–42, 2002. 5

[13] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *CVPR*, 2003. 5

[14] Jinglei Shi, Xiaoran Jiang, and Christine Guillemot. A framework for learning depth from a flexible subset of dense and sparse light field views. *IEEE Transactions on Image Processing*, 28(12):5867–5880, 2019. 5

[15] Chang Shu, Kun Yu, Zhixiang Duan, and Kuiyuan Yang. Feature-metric loss for self-supervised learning of depth and egomotion. In *ECCV*, 2020. 4, 5

[16] Jae Woong Soh, Sunwoo Cho, and Nam Ik Cho. Meta-transfer learning for zero-shot super-resolution. In *CVPR*, 2020. 5

[17] Jaime Spencer, Richard Bowden, and Simon Hadfield. Scale-adaptive neural dense features: Learning via hierarchical context aggregation. In *CVPR*, 2019. 4

[18] Jaime Spencer, Richard Bowden, and Simon Hadfield. Defeat-net: general monocular depth via simultaneous unsupervised representation learning. In *CVPR*, 2020. 4, 5