Unsupervised Deep Asymmetric Stereo Matching with Spatially-Adaptive Self-Similarity Supplementary Material

Taeyong Song[1] Sunok Kim[2] Kwanghoon Sohn[3,4]
[1]Hyundai Motor Company R&D Division, [2]Korea Aerospace University,
[3]Yonsei University, [4]Korea Institute of Science and Technology (KIST)
taeyongsong@hyundai.com, sunok.kim@kau.ac.kr, khsohn@yonsei.ac.kr

This supplementary material presents the following contents:
Sec. A provides additional experimental results.
Sec. B provides additional implementation details.

A. More Results

A.1. Comparisons with different methods

We use the proposed method with number of sampling patterns $L = 16$ and compare with different methods using different asymmetric factors. As we observe in Tables A1 and A2, our proposed method achieves the best quantitative performance across the different asymmetry factors.

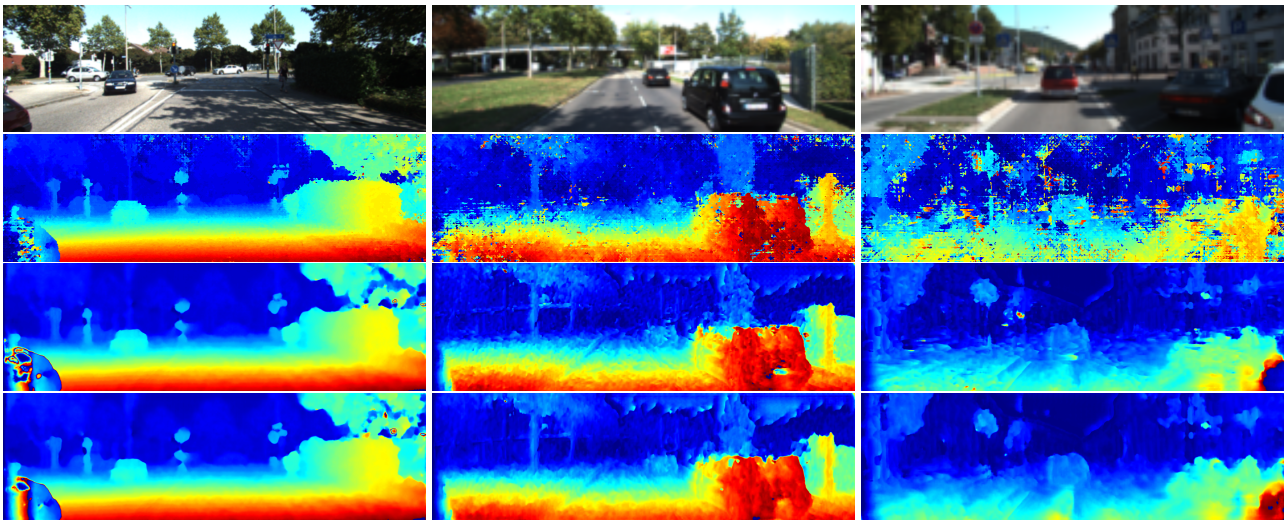
Table A1. Comparisons with different methods with different resolution asymmetry factors.

Method	Resolution asymmetry factor s					
	2		6		8	
	EPE	3PE	EPE	3PE	EPE	3PE
SGM [5]	5.481	36.19	9.617	55.14	14.834	72.46
Restore [7] + SGM [5]	5.326	35.74	8.728	48.39	10.681	56.02
Baseline	2.194	10.81	3.278	23.56	3.856	34.95
Restore [7] + Baseline	2.144	10.53	3.071	20.51	3.710	28.79
DAUS [4]	2.074	9.47	2.657	16.16	2.953	18.44
Proposed Method	**1.981**	**9.20**	**2.544**	**13.63**	**2.836**	**15.43**

Table A2. Comparisons with different methods with different noise asymmetry factors.

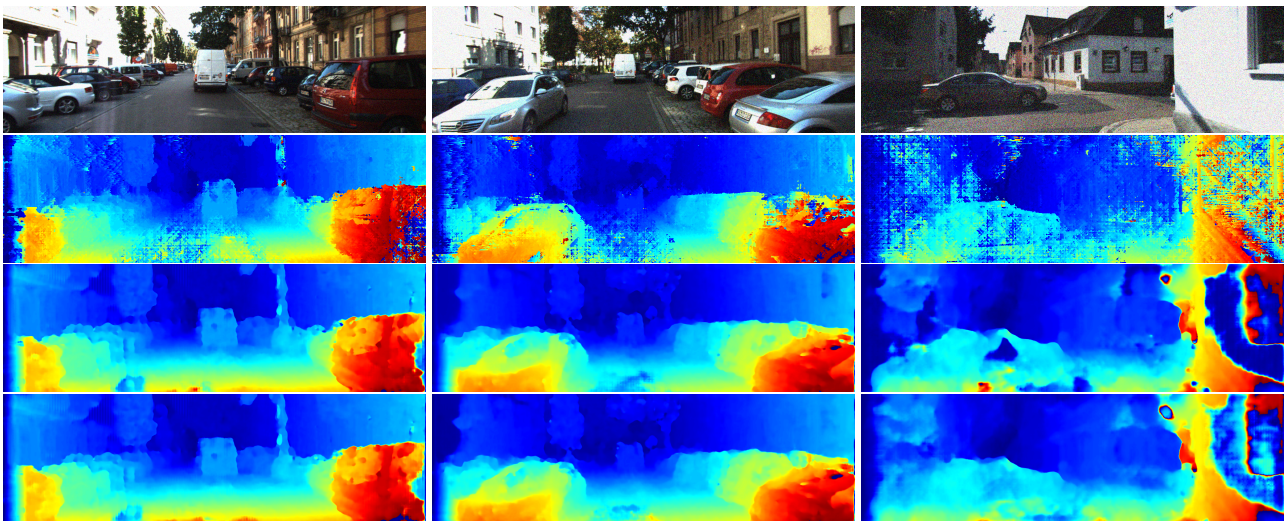
Method	Noise asymmetry factor σ					
	0.05		0.10		0.20	
	EPE	3PE	EPE	3PE	EPE	3PE
SGM [5]	5.204	36.92	9.254	57.64	17.294	75.13
Restore [7] + SGM [5]	5.183	35.13	8.814	53.16	13.164	71.08
Baseline	2.124	11.04	2.571	14.96	6.238	35.82
Restore [7] + Baseline	1.978	10.51	2.284	12.63	4.121	27.14
DAUS [4]	1.984	10.90	2.196	11.89	3.580	21.84
Proposed Method	**1.942**	**10.52**	**2.138**	**11.46**	**3.334**	**20.18**

We present qualitative results in Figs. A1 and A2, for resolution and noise asymmetries, respectively. Compared to the different methods, the proposed method generates better results with less artifacts.



(a) $s = 2$ (b) $s = 6$ (c) $s = 8$

Figure A1. Qualitative results of the different stereo matching methods under re solution asymmetry with factors (a) $s = 2$, (b) $s = 6$, and (c) $s = 8$. (from top to bottom) right image, stereo matching results of: SGM [5], DAUS [4], and the proposed method.



(a) $\sigma = 0.05$ (b) $\sigma = 0.10$ (c) $\sigma = 0.20$

Figure A2. Qualitative results of the different stereo matching methods under noise asymmetry with factors (a) $\sigma = 0.05$, (b) $\sigma = 0.10$, and (c) $\sigma = 0.20$. (from top to bottom) right image, stereo matching results of: SGM [5], DAUS [4], and the proposed method.

A.2. Asymmetries with different type of resolution and noise degradation

In addition to the bilinear blur and Gaussian noise for generating the resolution- and noise-asymmetry images, we conducted experiments using Gaussian blur and Poisson noise. We use Gaussian blur with standard deviation $[1, 2, 3, 4]$, and Poisson noise with Peak parameters $[0.03, 0.05, 0.07, 0.09]$, respectively. The quantitative results are presented in Table. A3. Similar to the results in the main paper, we observe that the proposed method consistently outperforms the different methods in asymmetric stereo matching with different types of resolution and noise degradation.

Table A3. Comparisons of different methods under asymmetries with Gaussian blur and Poisson noise.

Method	Gaussian blur factor							
	1		2		3		4	
	EPE	3PE	EPE	3PE	EPE	3PE	EPE	3PE
SGM [5]	5.231	36.21	5.758	31.44	8.422	52.47	16.173	73.26
Restore [7] + SGM [5]	5.210	35.98	5.677	29.61	7.170	43.26	12.939	66.90
Baseline	2.034	9.44	2.630	16.17	2.838	16.26	3.524	22.04
Restore [7] + Baseline	1.969	9.20	2.590	15.04	2.701	15.21	3.178	18.92
DAUS [4]	1.984	9.24	2.273	12.08	2.561	14.76	2.906	16.83
Proposed Method	**1.905**	**9.11**	**2.169**	**11.51**	**2.382**	**13.29**	**2.766**	**15.42**
Method	Poisson noise Peak parameter							
	0.03		0.05		0.07		0.09	
	EPE	3PE	EPE	3PE	EPE	3PE	EPE	3PE
SGM [5]	11.432	64.63	15.755	72.43	16.640	74.08	18.164	78.98
Restore [7] + SGM [5]	8.262	48.19	11.809	64.21	13.475	68.17	14.014	70.46
Baseline	3.541	21.92	5.744	30.29	5.939	35.29	6.574	38.71
Restore [7] + Baseline	2.920	19.26	3.966	26.83	4.074	25.84	4.731	32.61
DAUS [4]	2.844	16.41	3.610	22.57	3.811	24.16	4.178	29.04
Proposed Method	**2.676**	**14.88**	**3.394**	**18.64**	**3.632**	**23.20**	**3.907**	**25.42**

A.3. Pattern visualization

We visualize the sampling patterns of FCSS [6] and the proposed SASS. The networks are trained under resolution asymmetry with $s = 4$, and we use $L = 8$ for clear visibility of the patterns. The visualization is performed on the image for better understandings, where the self-similarity calculation is applied to the raw feature. We indicate the center pixel $\mathbf{x}_0$ with red circle, and the sampling patterns with squares. Each pattern index $l \in [1, 2, ..., L]$ is indicated with different color.

The visualization is presented in Fig A3. As addressed in the paper, FCSS [6], once trained, generates all the same sampling patterns across different images and regions. In contrast, the proposed SASS adaptively generates the patterns to extract robust features by encoding the structural layouts.

Figure A3. Visualization of the sampling patterns of FCSS [6] (1st row) and our proposed SASS (2nd row). FCSS [6] generates same sampling patterns for all image regions, whereas the proposed SASS generates different patterns for each pixel.

B. Implementation Details

B.1. Architecture details

The original '*stacked hourglass*' architecture of PSMNet [3] contains series of hourglass architecture to estimate the disparities in a cascaded architecture. We reduce the architecture to output a single disparity estimation, and scaled the output with respect to the image width in order to match the input range of bilinear sampling operation. In addition, the overall capacity of the network is reduced by adjusting the number of convolutional layers and channels. We compose the offset generator with three convolutional blocks, where each block consists of convolution and batch normalization layers. ReLU activation is also applied after the batch normalization layer, except for the last block.

Before calculating the SASS feature using (4), the raw feature $\mathbf{F}$ is normalized with L_2 normalization towards the channel axis. The maximum operation $\max_{\hat{\mathbf{x}} \in \mathcal{N}_{\mathbf{x}}}$ is realized with a max pooling layer with 2×2 window. The process also is applied when calculating FCSS [6] feature.

B.2. Determining hyper-parameters

In order to determine the loss weights in (10), we first fixed $\lambda_{pm} = 1.0, \lambda_{ds} = 0.5$, then conducted grid search for λ_{fm} and λ_{cs}. The followed the practice in [6] and [4] to set the exponential bandwidth $\gamma = 0.5$ (4) and $\alpha_{pm} = \alpha_{fm} = 0.15$ (7), (8). We set $\tau = 3$, considering the trade-off between correctness of the estimated disparities and ratio between positive and negative pixels.

In order to determine the margin hyper-parameter M in (6), we observe average L_2 distance between the normalized raw feature values extracted from the *aligned* high- and low-quality images. To this end, we simulate the degradation to the left image, and extract the features from the original and degraded images using the encoder in a trained stereo network. Observing average of approximately 0.3 in L_2 distance, we perform grid search in range [0.1, 0.7], and finally set $M = 0.5$.

B.3. Comparison methods

The image restoration method [7][1] provides the pre-trained models for denoising and super-resolution, trained with real-world datasets without artificial degradation [1, 2]. The pre-trained denoising model is trained with unspecified noise level, and the super-resolution model is trained with $4\times$ setting. For the results in the supplementary material (Tables. A1, A2), we use the pre-trained model for all noise asymmetry settings, and re-trained the model for super-resolution using artificial degradation using the corresponding low-resolution degradation. We used the default settings of the released source code except the batch size, which is reduced to 4. We used a python implementation of SGM [5][2], which uses Census transformation as image feature extraction.

References

[1] Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. A high-quality denoising dataset for smartphone cameras. In *CVPR*, pages 1692–1700, 2018. 5

[2] Jianrui Cai, Hui Zeng, Hongwei Yong, Zisheng Cao, and Lei Zhang. Toward real-world single image super-resolution: A new benchmark and a new model. In *ICCV*, pages 3086–3095, 2019. 5

[3] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *CVPR*, pages 5410–5418, 2018. 4

[4] Xihao Chen, Zhiwei Xiong, Zhen Cheng, Jiayong Peng, Yueyi Zhang, and Zheng-Jun Zha. Degradation-agnostic correspondence from resolution-asymmetric stereo. In *CVPR*, pages 12962–12971, 2022. 1, 2, 3, 4

[5] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE TPAMI*, 30(2):328–341, 2008. 1, 2, 3, 5

[6] Seungryong Kim, Dongbo Min, Bumsub Ham, Sangryul Jeon, Stephen Lin, and Kwanghoon Sohn. FCSS: Fully convolutional self-similarity for dense semantic correspondence. In *CVPR*, pages 6560–6569, 2017. 4

[7] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Learning enriched features for fast image restoration and enhancement. *IEEE TPAMI*, 2022. 1, 3, 5

[1] https://github.com/swz30/MIRNet

[2] https://github.com/beaupreda/semi-global-matching