SegStereo: Exploiting Semantic Information for Disparity Estimation –Supplementary Material

1 Models

Due to space limitations in the paper, we detail the models involved in our paper, including *SegStereo* and *ResNetCorr*.

1.1 SegStereo

We give structural definition of *SegStereo* in Tab. 1. Our model resembles ResNet-50 [6]. It can be divided into three parts: shallow feature extractor, feature aggregator and disparity encoder-decoder. The shallow part is utilized to extract image features on input stereo images. It subsamples the input images in two stages: "conv_block1_1" and "max_pool1", which results in 1/8 scaling to raw images. The feature aggregator realizes the semantic feature embedding, where semantic features computed from PSPNet-50 [15] are concatenated with correlated features and left transformed features. We further utilize disparity encoder-decoder to regress final disparity map.

In Tab. 1, "conv_block" denotes the convolutional block, where a convolutional layer is followed by batch normalization(BN) and ReLU activation. "Res_block" denotes the residual block designed by [6]. The attributes of "res_block" describe the key convolutional layer in residual block. Several convolutional layers in disparity encoder adopt dilated pattern [3] to integrate larger receptive field. The deconvolutional block in disparity decoder is composed of deconvolutional layer, BN layer and ReLU layer.

1.2 ResNetCorr

We introduce *ResNetCorr* model as a baseline that excludes semantic information. Compared to *SegStereo*, the feature aggregator excludes semantic features as shown in Tab 2. In addition, semantic feature warping does not appear in *ResNetCorr*.

2 Segmentation Branch

We give a more detailed description to segmentation branch in *SegStereo*, including semantic feature warping and segmentation results on KITTI dataset.

Table 1: Layers in our *SegStereo* architecture

Layer	Attributes	Channels I/O	Scaling	Inputs
		1. Shallow Feature Extractor		
conv_block1_1	kernel size = 3, stride = 2	3 / 64	1/2	input stereo images
conv_block1_2	kernel size = 3, stride = 1	64 / 64	1/2	conv_block1_1
conv_block1_3	kernel size = 3, stride = 1	64 / 128	1/2	conv_block1_2
max_pool_block1	kernel size = 3, stride = 2	128 / 128	1/4	conv_block1_3
res_block2_1	kernel size = 3, stride = 1	128 / 256	1/4	max_pool_block1
res_block2_2	kernel size = 3, stride = 1	256 / 256	1/4	res_block2_1
res_block2_3	kernel size = 3, stride = 1	256 / 256	1/4	res_block2_2
res_block3_1	kernel size = 3, stride = 1	512 / 512	1/8	res_block2_3
		2. Feature Aggregator		
conv_block_pre	kernel size = 3, stride = 1	512 / 256	1/8	res_block3_1
PSPNet	layers from conv3_2 to conv5_4 [15]	256 / 128	1/8	res_block2_3
corr_1d	max displacement = 24, single direction [10]	256 / 25	1/8	res_block_pre
conv_trans	kernel size = 3, stride = 1	256 / 256	1/8	res_block_pre
concat	**semantic feature embedding**	(128 + 256 + 25) / 409	1/8	PSPNet, corr_1d, conv_trans
		3.1. Disparity Encoder		
res_block3_2	kernel size = 3, stride = 1	409 / 512	1/8	concat
res_block3_3	kernel size = 3, stride = 1	512 / 512	1/8	res_block3_2
res_block3_4	kernel size = 3, stride = 1	512 / 512	1/8	res_block3_3
res_block4_1	kernel size = 3, stride = 1, dilated pattern	512 / 1024	1/8	res_block3_3
res_block4_2	kernel size = 3, stride = 1, dilated pattern	1024 / 1024	1/8	res_block4_1
res_block4_3	kernel size = 3, stride = 1, dilated pattern	1024 / 1024	1/8	res_block4_2
res_block4_4	kernel size = 3, stride = 1, dilated pattern	1024 / 1024	1/8	res_block4_3
res_block4_5	kernel size = 3, stride = 1, dilated pattern	1024 / 1024	1/8	res_block4_4
res_block4_6	kernel size = 3, stride = 1, dilated pattern	1024 / 1024	1/8	res_block4_5
res_block5_1	kernel size = 3, stride = 1, dilated pattern	1024 / 2048	1/8	res_block4_6
res_block5_2	kernel size = 3, stride = 1, dilated pattern	2048 / 2048	1/8	res_block5_1
res_block5_3	kernel size = 3, stride = 1, dilated pattern	2048 / 2048	1/8	res_block5_2
conv_block5_4	kernel size = 3, stride = 1	2048 / 512	1/8	res_block5_3
		3.2. Disparity Decoder		
deconv_block1	kernel size = 3, stride = 2	512 / 256	1/4	conv_block5_4
deconv_block2	kernel size = 3, stride = 2	256 / 128	1/2	deconv_block1
deconv_block3	kernel size = 3, stride = 2	128 / 64	1	deconv_block2
disp_conv	kernel size = 3, stride = 1	64 / 1	1	deconv_block3

2.1 Semantic Feature Warping

We obtain predicted disparity map $\mathcal{D}$ from disparity branch and right semantic feature map ${\mathcal{F}_s}^r$ from segmentation branch. To preserve more semantic information, we extract the features at "conv5_4" layer in PSPNet-50 [15], which gets 1/8 spatial size to raw image. As shown in Fig. 1, before semantic warping on disparity map $\mathcal{D}$, we upsample semantic feature map ${\mathcal{F}_s}^r$ to original size. The warped feature map is downsampled to 1/8 size to adopt the followed convolutional layer "conv6" in PSPNet-50 [15]. We compute the softmax loss $\mathcal{L}_{seg}$ between predicted semantic map and semantic label.

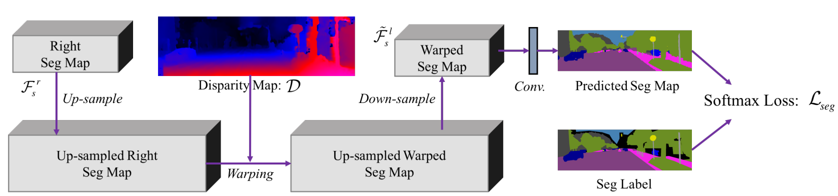

Fig. 1: Diagram of semantic feature warping

Table 2: Layers in baseline *ResNetCorr* architecture

Layer	Attributes	Channels I/O	Scaling	Inputs
1. Shallow Feature Extractor				
conv_block1_1	kernel size = 3, stride = 2	3 / 64	1/2	input stereo images
conv_block1_2	kernel size = 3, stride = 1	64 / 64	1/2	conv_block1_1
conv_block1_3	kernel size = 3, stride = 1	64 / 128	1/2	conv_block1_2
max_pool_block1	kernel size = 3, stride = 2	128 / 128	1/4	conv_block1_3
res_block2_1	kernel size = 3, stride = 1	128 / 256	1/4	max_pool_block1
res_block2_2	kernel size = 3, stride = 1	256 / 256	1/4	res_block2_1
res_block2_3	kernel size = 3, stride = 1	256 / 256	1/4	res_block2_2
res_block3_1	kernel size = 3, stride = 1	512 / 512	1/8	res_block2_3
2. Feature Aggregator				
conv_block_pre	kernel size = 3, stride = 1	512 / 256	1/8	res_block3_1
corr_1d	max displacement = 24, single direction [10]	256 / 25	1/8	res_block_pre
conv_trans	kernel size = 3, stride = 1	256 / 256	1/8	res_block_pre
concat	**excludes semantic features**	(256 + 25) / 281	1/8	corr_1d, conv_trans
3. Disparity Encoder				
res_block3_2	kernel size = 3, stride = 1	409 / 512	1/8	concat
res_block3_3	kernel size = 3, stride = 1	512 / 512	1/8	res_block3_2
res_block3_4	kernel size = 3, stride = 1	512 / 512	1/8	res_block3_3
res_block4_1	kernel size = 3, stride = 1, dilated pattern	512 / 1024	1/8	res_block3_3
res_block4_2	kernel size = 3, stride = 1, dilated pattern	1024 / 1024	1/8	res_block4_1
res_block4_3	kernel size = 3, stride = 1, dilated pattern	1024 / 1024	1/8	res_block4_2
res_block4_4	kernel size = 3, stride = 1, dilated pattern	1024 / 1024	1/8	res_block4_3
res_block4_5	kernel size = 3, stride = 1, dilated pattern	1024 / 1024	1/8	res_block4_4
res_block4_6	kernel size = 3, stride = 1, dilated pattern	1024 / 1024	1/8	res_block4_5
res_block5_1	kernel size = 3, stride = 1, dilated pattern	1024 / 2048	1/8	res_block4_6
res_block5_2	kernel size = 3, stride = 1, dilated pattern	2048 / 2048	1/8	res_block5_1
res_block5_3	kernel size = 3, stride = 1, dilated pattern	2048 / 2048	1/8	res_block5_2
conv_block5_4	kernel size = 3, stride = 1	2048 / 512	1/8	res_block5_3
4. Disparity Decoder				
deconv_block1	kernel size = 3, stride = 2	512 / 256	1/4	conv_block5_4
deconv_block2	kernel size = 3, stride = 2	256 / 128	1/2	deconv_block1
deconv_block3	kernel size = 3, stride = 2	128 / 64	1	deconv_block2
disp_conv	kernel size = 3, stride = 1	64 / 1	1	deconv_block3

2.2 Segmentation Results

Before we train disparity branch of *SegStereo* on KITTI set, we finetune the segmentation sub-network based on the released semantic labels [1]. Fig. 2 shows several examples predicted by our model, which illustrates that our model is able to provide consistent semantic estimates. We further submit the segmentation results of *SegStereo* to KITTI Pixel-level Semantic Evaluation. The final mean IoU over 19 classes is 59.10%. We attempt to use disparity predictions to help semantic estimation, but no obvious improvements are found. We argue that segmentation tends to fail on boundary and small regions which are also hard for disparity estimation.

Table 3: Unsupervised *SegStereo* models trained by softmax loss or pixel-wise distance. The results are evaluated on KITTI stereo 2015 dataset [11]

Loss Combinations	Noc EPE	Noc D1	All EPE	All D1
photometric + smooth + **softmax**	1.61	**8.95**	**1.89**	**10.03**
photometric + smooth + **pixel-wise distance**	**1.57**	**8.95**	1.96	10.36

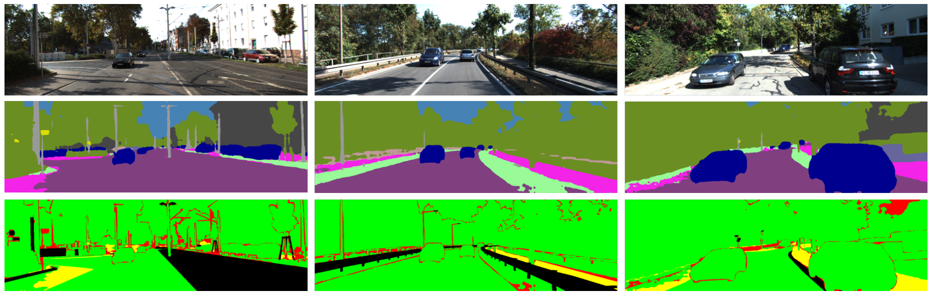

Fig. 2: Segmentation examples of KITTI Semantic Benchmark [1]. From top to bottom: input images, predicted segmentation images and error maps. In error maps, red color indicates wrong label and category. Yellow color denotes the wrong label but the correct category. Green color represents correct label

3 Disparity Results

3.1 Unsupervised SegStereo Trained by Pixel-wise Distance

We also try the pixel-wise euclidean loss between referenced feature maps ${\mathcal{F}_s}^r$ and warped feature maps $\tilde{\mathcal{F}_s}^l$ in segmentation branch. With the constraints measured by pixel-wise distance, our model gets rid of the dependance on semantic labels, which makes a purely unsupervised precedure. In Table 3, compared to original model trained with softmax loss, the model trained with pixel-wise distance achieves similar accuracy on non-occluded regions, which demonstrates usefulness of pixel-wise regularization. For the overall slightly lower accuracy on all regions, it is because pixel-wise distance on occluded areas cannot provide correct constraints owing to lack of corresponding pixels.

3.2 KITTI Stereo 2012 Results

In Tab. 4, we compare *SegStereo* to other methods on KITTI Stereo 2012 benchmark [4]. It can be found that our method outperforms other methods except for PSMNet [2].

3.3 Qualitative results

More Qualitative results are shown in Fig. 3 and Fig. 4. There results are directly grabbed on KITTI benchmark website. Our *SegStereo* model reaches advanced performance with the guidance of semantic information. We provide a video "seg_stereo.mp4" that shows predictions of *SegStereo* on raw KITTI and CityScapes sequences.

Table 4: Compare to other disparity estimation methods on the test set of 2012 dataset [4]. Our method achieves state-of-the-art results on this benchmark

	> 3 pixels		> 4 pixels		> 5 pixels		Mean Error		Runtime
	Noc	All	Noc	All	Noc	All	Noc	All	(s)
L-ResMatch [13]	2.27	3.40	1.76	2.67	1.50	2.26	0.7 px	1.0 px	48
PBCP [12]	2.36	3.45	1.88	2.74	1.62	2.32	0.7 px	0.9 px	68
Displets v2 [5]	2.37	3.09	1.97	2.52	1.72	2.17	0.7 px	0.8 px	265
MC-CNN-arct [14]	2.43	3.63	1.90	2.85	1.64	2.39	0.7 px	0.9 px	67
Content-CNN [9]	3.07	4.29	2.39	3.36	2.03	2.82	0.8 px	1.0 px	0.7
Deep Embed [3]	3.10	4.24	1.73	2.32	1.92	2.68	0.9 px	1.1 px	3
DispNetC [10]	4.11	4.65	2.77	3.30	2.05	2.39	0.9 px	1.0 px	**0.06**
GC-NET [7]	1.77	2.30	1.36	1.77	1.12	1.46	0.6 px	0.7 px	0.9
iResNet [8]	1.71	2.16	1.30	1.63	1.06	1.32	0.5 px	0.6 px	0.12
PSMNet [2]	**1.49**	**1.89**	**1.12**	**1.42**	**0.90**	**1.15**	**0.5 px**	**0.6 px**	0.41
SegStereo(Ours)	1.68	2.03	1.25	1.52	1.00	1.21	**0.5 px**	**0.6 px**	0.6

Fig. 3: **Comparative Qualitative results in the test set of KITTI Stereo 2012 dataset [4].** From top to bottom: left input image, disparity error maps of different methods. The error maps scale linearly between 0 (black) and >=5 (white) pixels error. Red denotes all occluded pixels. By exploiting semantic information, our *SegStereo* model can handle challenging scenarios

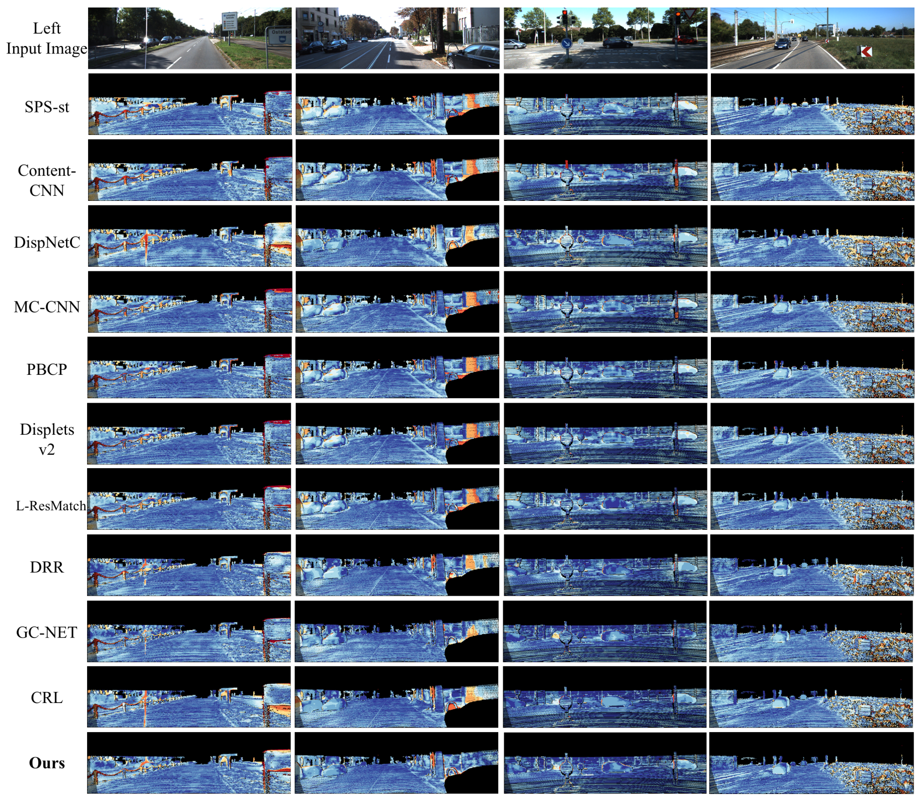

Fig. 4: **Comparative Qualitative results in the test set of KITTI Stereo 2015 dataset** [11]**.** From top to bottom: left input image, disparity error maps of different methods. The error maps use the log-color scale, depicting correct estimates (<3 px or <5% error) in blue and wrong estimates in red color tones. Dark regions in error images denote the occluded pixels. With the guidance of semantic consistency, our model is able to predict reliable disparities on difficult areas, including exposure, shadow, and complex grass. Our *SegStereo* method outperforms other first-class approaches

References

1. Alhaija, H.A., Mustikovela, S.K., Mescheder, L., Geiger, A., Rother, C.: Augmented reality meets deep learning for car instance segmentation in urban scenes. In: BMVC (2017) 3, 4
2. Chang, J., Chen, Y.: Pyramid stereo matching network. In: CVPR (2018) 4, 5
3. Chen, Z., Sun, X., Wang, L., Yu, Y., Huang, C.: A deep visual correspondence embedding model for stereo matching costs. In: ICCV (2015) 1, 5
4. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: CVPR (2012) 4, 5
5. Guney, F., Geiger, A.: Displets: Resolving stereo ambiguities using object knowledge. In: CVPR (2015) 5
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016) 1
7. Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P., Kennedy, R., Bachrach, A., Bry, A.: End-to-end learning of geometry and context for deep stereo regression. In: ICCV (2017) 5
8. Liang, Z., Feng, Y., Guo, Y., Liu, H., Chen, W., Qiao, L., Z., L., Z., J.: Learning for disparity estimation through feature constancy. In: CVPR (2018) 5
9. Luo, W., Schwing, A.G., Urtasun, R.: Efficient deep learning for stereo matching. In: CVPR (2016) 5
10. Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T.: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: CVPR (2016) 2, 3, 5
11. Menze, M., Geiger, A.: Object scene flow for autonomous vehicles. In: CVPR (2015) 3, 6
12. Seki, A., Pollefeys, M.: Patch based confidence prediction for dense disparity map. In: BMVC (2016) 5
13. Shaked, A., Wolf, L.: Improved stereo matching with constant highway networks and reflective confidence learning. In: CVPR (2017) 5
14. Zbontar, J., LeCun, Y.: Stereo matching by training a convolutional neural network to compare image patches. JMLR (2016) 5
15. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: CVPR (2017) 1, 2