Supplementary Material for Generative View Synthesis: From Single-view Semantics to Novel-view Images

Tewodros Habtegebrial^{1,4} **Varun Jampani**^{2} **Orazio Gallo**^{3} **Didier Stricker**^{1,4}
^{1}`TU Kaiserslautern` ^{2}`Google Research` ^{3}`NVIDIA` ^{4}`DFKI`

1 Additional Experimental Results

In this supplementary, we provide additional details on depth estimation and evaluation(in section 1.3), evaluation with different style images(in section 1.4), network architecture(2), comparison with more advanced baselines (1.2), and additional training details(in section 3).

1.1 Visual Results

We have included additional visual results of our approach along with different baseline techniques in the **supplementary video**. We highly encourage the readers to watch the supplementary video. More visual results on depth estimation are included in Figure 1.

1.2 Comparison RGB+Seg View Synthesis Methods

In our the paper we compared against baseline constructed as a sequential application of Image-to-Image Translation and Monocular Novel-view Synthesis (MNVS). The results indicate that our method preserves the semantics/geometry of the input scene. In-order to further improve the baselines, we experimented with feeding the MNVS part with RGB and Semantics. The RGB and Semantics are concatenated channel wise and fed as input to MNVS methods. As expected, feeding RGB and Semantics improves the performance of all baselines. Table 1 shows that the SPADE [9]+SM [11], baseline that takes an additional semantic map (indicated as SPADE [9]+SM [11]$^{+}$) performs better than its RGB counterparts.

Method	Cls. Acc ↑	IoU ↑	PD↓	FID↓
GVSNet	**74.34**	**66.43**	**1.74**	**62.06**
SPADE+SM	69.93	60.82	1.95	75.81
SUN+SPADE	72.92	65.52	1.75	68.96
SPADE+SM^{+}	72.29	63.55	1.75	73.74

Table 1: Comparing against an SPADE [9]+SM [11] baseline, where the SM network takes a translated image together with the input semantics. As show in rows 3 and 4 the feeding the SM network with semantics leads to better performance. However, as can be seen in the first column our method consistently outperforms the improved and the original baselines.

1.3 Depth Estimation and Evaluation

Extracting Depth Maps from Multi-Plane Transparencies. Our Semantic Uplifting Network (SUN) estimates multi-plane transparencies α, in the reference camera. Here we show a simple way to convert $\alpha \in R^{n \times m}$ in to a depth map $\hat{Z} \in R^{n}$, where n is the number of pixels and m is the number of planes in the MPI representation. Suppose the MPI planes are located at distances $\{d_1, d_2, \ldots, d_m\}$ and are fronto-parallel to the reference camera. The depth value at a pixel p, $\hat{Z}(p)$,

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

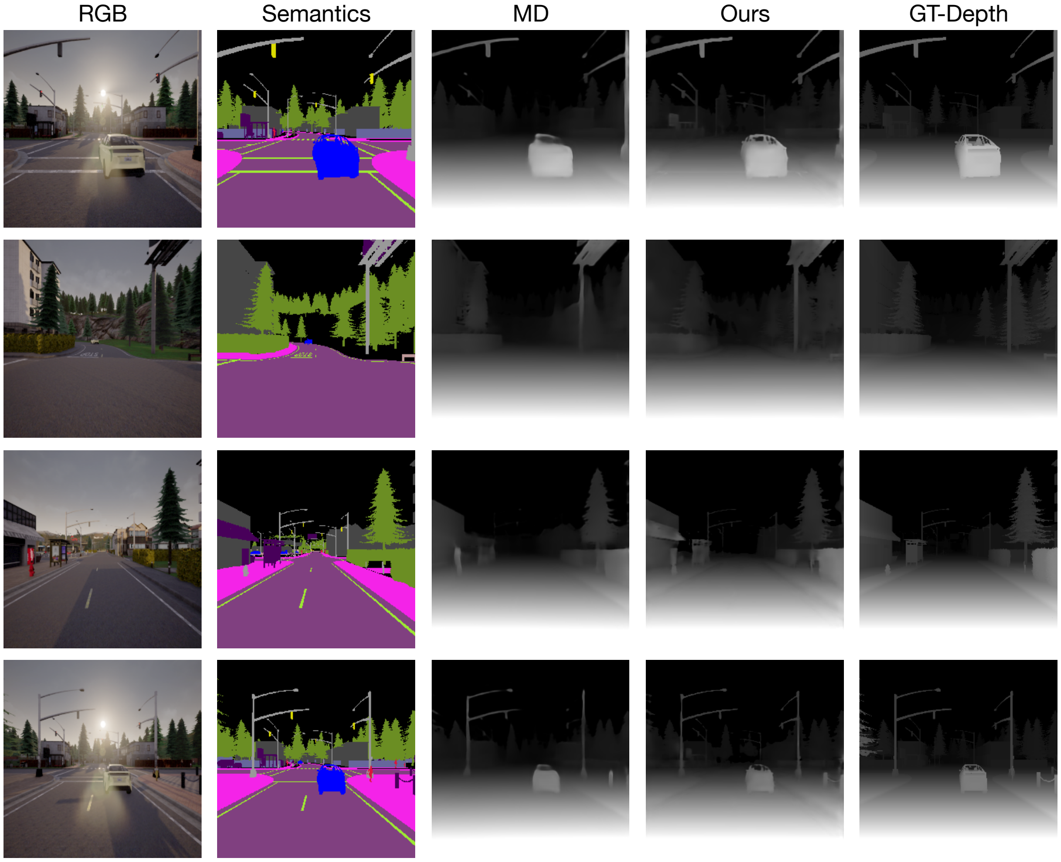

Figure 1: **Depth Estimation Results.** Here we present more visual depth estimation results from our SUN model and MonoDepth (MD) model trained with ground-truth depths.

is computed by alpha compositing the depth of the MPI planes with the alpha values at pixel p, as follows.

$$\hat{Z}(p) = \sum_{i=1}^{m} \left[d_i \, \alpha(p,i) \left[\prod_{j=1}^{i-1} (1 - \alpha(p,j)) \right] \right] \tag{1}$$

Depth Evaluations. In Table 2, we show accuracy of the depths estimated by our SUN method and compare it against a baseline network trained with ground truth depths. The baseline network uses an RGB input while our method uses semantics as input. The baseline network is taken from the MonoDepth [5]; it is an encoder decoder network with ResNet-18 backbone. The network can be trained in a self-supervised manner, however for a fairer comparison, we train the network using ground-truth depth.

We evaluated these networks following traditional depth accuracy metrics [10]: *scale invariant depth* (SC_Inv), *relative depth* (L1_Rel) and *inverse depth* (L1_Inv) metrics. Using only semantics as input, which is devoid of photo-metric details, our network produces favourable depth estimations when compared to the baseline. Our method outperforms the MonoDepth network on SC_Inv and L1_Inv metrics. However, on L1_Rel, our method under-performs compared to MonoDepth. We believe this is due to the fact that MPI planes are distributed by sampling the inverse depth linearly. This results in having few planes far from the camera. Nonetheless, for the view synthesis tasks it is desirable to have higher accuracy at closer ranges.

Method	Input	Supervision	SC_Inv	L1_Rel	L1_Inv	Depth range (in meters)
Ours	Sem	Sem + Depth	**0.200**	0.129	**0.006**	1 - 100
Mono-Depth	RGB	Depth	0.225	**0.125**	0.008	1 - 100
Ours	Sem	Sem + Depth	**0.252**	0.184	**0.006**	1 - 200
Mono-Depth	RGB	Depth	0.269	**0.155**	0.008	1 - 200
Ours	Sem	Sem + Depth	0.**331**	0.395	**0.006**	1 - 1000
Mono-Depth	RGB	Depth	0.358	**0.268**	0.007	1 - 1000

Table 2: **Depth accuracy evaluation on CARLA [4] dataset.** Depth maps generated by our SUN model and monocular depth estimation network from [5]. SC_inv, L1_Rel and L1_Inv stand for scale invariant, L1 relative and L1 inverse depth metrics, respectively.

1.4 Evaluations with Different Style Images.

As can be seen in the Figure-1 of the main paper and in the supplementary video, our method can produce novel-views with different styles, as dictated by the given style images. In image-to-image literature [9], it is customary to use the target color image as a style during evaluation. Following this custom, we use the color image from the source camera as a style guidance for the novel-view image generation. In order to verify the efficacy of method under arbitrary styles, we perform an additional test where our method is not fed the input image as style, rather a random frame the same sequence is used as style input. In Table 3, we show semantic evaluation results using this procedure. The results show that our method still outperforms the baselines under this setting of using different style image.

Method	Cls. Acc. ↑	Mean IoU ↑
GVSNet Variations		
GVSNet (Full Model)	**73.16**	**65.40**
SUN+SPADE [9]	70.95	63.73
SPADE [9] + SM [11]	67.26	58.71
SPADE [9] + CVS [2]	64.70	55.67
SPADE [9] + AF [12]	63.28	54.05
Target GT Images	77.47	69.67

Table 3: **Semantic evaluations on CARLA [4] dataset.** Metrics are computed using a single randomly chosen style frame per sequence. This test shows that our method handles arbitrary style images. Our Class Accuracy and Mean IoU results here are also close the results achieved using the source-view color image as style input. As reported in the main paper paper, using source-view color image as style we obtain Cls. Acc. of 77.34 and mean IoU of 66.43.

2 Network Architectures

Semantic Uplifting Network. Our Semantic Uplifting Network (SUN), is a 2D encoder-decoder network with 3 outputs: lifted semantics, MPI transparency α and association function Φ. These 3 outputs are predicted jointly, with the network architecture shown in Table 4.

Appearance Decoder Network. Table 5 shows the detailed architecture of the Appearance Decoder Network (ADN) network.

Layered Translation Network. Layered Translation Network (LTN) consists of layered appearance generator and style encoder networks. The architecture of both the appearance generator and encoder networks are similar to those used in the SPADE [9] paper. However, in this work the generator network performs layered translation of k semantic maps. The output of the network is also different since we predict higher dimensional features, not color images. Our generator, takes input semantics of shape $[H \times W \times k \times l]$ and produces appearance features with dimensions $[H \times W \times l \times f]$. In our experiments, we found that $k = 3$ and $f = 20$ suffice to achieve good results. The GAN part of our work is based on SPADE, therefore, we use the same discriminator network and training losses as in the SPADE work.

3 Additional Experimental Details

3.1 Dataset Details

We use three publicly available datasets for our GVS experiments. In all 3 datasets we used images down scaled to a resolution of 256×256 pixels.

CARLA. Using the CARLA [4] open source simulation environment, we captured 20 independent sequences in 5 towns. We use 16 sequences for training and 4 sequences for testing, taking one test sequence per city. We capture data using camera arrays mounted on top of a car. Each camera captures color, semantic and depth images. We use 3 groups of camera arrays: *horizontal*, *forward* and *side*-camera groups. The horizontal camera array has 5 cameras at uniform spacing along the $x - axis$. The forward camera array has 5 cameras uniformly distributed along $z - axis$. The side camera array contains 3 horizontally shifted cameras facing the side-view of the car. The spacing between cameras within each array is $54cms$. During training and testing we take a random pair of cameras from one of the camera groups and use one as source and the other as target.

Cityscapes is a publicly available dataset of urban scenes captured across several German cities [3]. We use 2975 scenes for training and 500 scenes for test. Each scene is captured with a stereo pair. During training and testing phases, we use one of the two stereo cameras as source and the other as target. Ground truth semantics is available for the left camera images only. We label the right camera images using a pre-trained semantic segmentation network [13]. Citycapes dataset has no ground-truth depth. We generate depth maps by training the DPS [6] network in a self-supervised manner. In order to achieve results which are comparable with SPADE [9], we use instance masks in our experiments on this dataset. We compute a one channel instance mask image, as a gradient of the original instance segmentation. We use instance masks by concatenating the input view instance mask with lifted semantics. Since right camera instance masks are not available, we generate a pseudo ground-truth instance masks by warping the left image masks using the depths estimated by DPS network [6]. While warping, we perform forward-backward depth tests to detect occlusions and in areas where occlusion is detected we leave the instance masks empty. Note, that in other datasets we do not use instance masks. In all of our experiments, we use 19 class labels provided by the dataset.

Virtual-KITTI-2 is a synthetic dataset of urban scenes captured with a stereo camera. Each stereo pair has color images together with ground-truth depths and semantic segmentations. The dataset has 6 sequences captured under 5 weather conditions. Each sequence is randomly divided in to train and test sub-sequences. The training sub-sequence covers 80 % of the frames and the test sub-sequence covers has 20% of the frames.

3.2 Training Details

Depth Loss. We compute the depth reconstruction loss $\mathcal{L}_{dep}$ on a scaled version of the predicted and target inverse depth maps. The scaled inverse depth is computed as $f_x \times 0.54/depth$, where f_x is the focal length. This is equivalent to converting *depth* into *disparity* by assuming imaginary stereo camera with a baseline of $54cms$. Since, all of the datasets used in this paper are large scale outdoor scenes, the same scaling works well for all the datasets.

Loss Weighting. We set the depth loss weighting factor $\lambda_1 = 0.1$, while the other weighting factors (λ_0, λ_2 and λ_3) are all set to 1.

Training Protocol GVSNet is trained in two phases. In the first phase, the Semantic Uplifting Network (SUN) is trained using depth reconstruction and semantic alignment losses. For the CARLA [4] dataset, we train the SUN network for 30 epochs. In Cityscapes [3] dataset, we train for 200 epochs and in Virtual KITTI-2 [1], we train for 60 epochs. In all datasets, we use mini-batch size 12 and Adam [7] optimizer with a $lr = 0.0004$, $\beta_1 = 0.900$, $\beta_2 = 0.999$), and $eps = e{-}08$. This SUN training is performed using 3 NVIDIA GTX-2080-Ti GPUs.

In the second phase, the Layered Translation Network (LTN) and Appearance Decoder Network (ADN), are trained by minimising the color and GAN losses. We train these networks while keeping the SUN network fixed. For CARLA dataset, we train for 20 epochs. In Cityscapes and Virtual KITTI-2 datasets, we train for 250 and 35 epochs respectively. In this phase, we use a batch size of 16. The training is done using 8 NVIDIA GTX-2080-Ti GPUs. We use Adam [7] optimizer with the initial learning rate of $lr = 0.0004$. The learning rate is kept fixed for the first half of the training. In the second half, we start decreasing the learning rate linearly so that it reaches 0 at the last iteration.

Layer	Input	Type	Stride	in_chans	out_chans
conv1a	input sem	Conv2d + ReLU	2	1	32
conv1b	conv1a	Conv2d + ReLU	1	32	32
conv2a	conv1b	Conv2d + ReLU	2	32	64
conv2b	conv2a	Conv2d + ReLU	1	64	64
conv3a	con2b	Conv2d + ReLU	2	64	128
conv3b	conv3a	Conv2d + ReLU	1	128	128
conv4a	conv3b	Conv2d + ReLU	2	128	256
conv4b	conv4a	Conv2d + ReLU	1	256	256
conv5a	conv4b	Conv2d + ReLU	2	256	512
conv5b	conv5a	Conv2d + ReLU	1	512	512
conv6a	conv5b	Conv2d + ReLU	2	512	512
conv6b	conv6a	Conv2d + ReLU	1	512	512
conv7a	conv6b	Conv2d + ReLU	2	512	512
conv7b	conv7a	Conv2d + ReLU	1	512	512
dconv7	conv7a	Conv2d + ReLU	1	512	512
dconv6	dconv7 $\bigoplus$ conv6b	Conv2d + ReLU	1	1024	512
dconv5	dconv6 $\bigoplus$ conv5b	Conv2d + ReLU	1	1024	512
dconv4	dconv5 $\bigoplus$ conv4b	Conv2d + ReLU	1	768	384
dconv3	dconv4 $\bigoplus$ conv3b	Conv2d + ReLU	1	512	256
dconv2	dconv3 $\bigoplus$ conv24	Conv2d + ReLU	1	320	96
dconv1	dconv2 $\bigoplus$ conv1b	Conv2d + ReLU	1	128	96
base_1	dconv1	Conv2d + ReLU	1	96	96
base_2	base_1	ResBlock	1	96	96
base_3	base_2	ResBlock	1	96	96
base_4	base_3	ResBlock	1	96	96
base_5	base_4	ResBlock with BN	1	96	96
base_6	base_5	ResBlock with BN	1	96	96
out_conv_1a	base_6	Conv2d+ReLU	1	96	(l x (k-1) + m*(k+1))/2
out_conv_1b	out_conv_1a	Conv2d+ReLU + BN	1	(l x (k-1) + m*(k+1))/2	(l x (k-1) + m*(k+1))/2
out_conv_a	out_conv_1b	Conv2d+ReLU	1	(l x (k-1) + m*(k+1))/2	(l x (k-1) + m*(k+1))/2
out_conv_1b	out_conv_a	Conv2d	1	(l x (k-1) + m*(k+1))/2	l x (k-1) + m*(k+1)

Table 4: Semantic Uplifting Network Architecture. In this table *in_chans* and *out_chans* refer to input and output number of channels, respectively. The size of the SUN network depends on the following hyper-parameters: number of layered semantic maps (k), number of MPI planes (m) and number semantic classes (l). The output from the *out_conv_1b* layer is split into α, Φ and layered semantics channels. α and Φ take up m and $k \times m$ channels, respectively. Since we create the layered semantic representation includes the input semantics, the network predicts layered semantics only for $(k-1)$ layers, with a total of $(k-1) \times l$ channels. Sigmoid activation is applied on the α and Φ outputs. All of the layers in this network are 2D convolutional layers. Encoder layers $conv1a$ up to $conv7b$ decrease the spatial resolution of their output by a factor of 2 using stride 2 convolutions. Decoder layers $dconv7$ to $deconv1$ apply nearest neighbour up-sampling with a factor of 2 before applying convolution and ReLU. The ResBlock have 2 convolutional layers (the first one has ReLU activation), and the output of the second layer is added to the input. The $\bigoplus$ sign refers to channel-wise concatenation.

References

[1] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual KITTI 2. In *arXiv Preprint*, 2020. 4

Layer	input	in_chans	out_chans
conv_0	appearance features	f	16
conv_1	conv_0	16	32
conv_2	conv_1	32	64
conv_3	conv_2	64	64
conv_4	conv_3	64	64
d_conv_4	conv_4	64	64
d_conv_3	d_conv_4 $\bigoplus$ conv_3	128	32
d_conv_2	d_conv_3 $\bigoplus$ conv_2	64	32
d_conv_1	d_conv_2 $\bigoplus$ conv_1	64	16
output_conv	d_conv_1 $\bigoplus$ conv_0	32	3

Table 5: Architecture of the Appearance Decoder Network. In this table in_chans and out_chans stand for the number of input and output channels. The $\bigoplus$ sign indicates channel-wise concatenation. The ADN network gets $f-channel$ appearance feature as input and returns a $3-channel$ color image as output. Every layer in this network is a convolutional layer with 3×3 kernel and *stride* of 1. All layers except output_conv have spectral normalisation and LeakyReLU non-linearity (with negative slope value of -0.2) after convolution. output_conv layer applies Tanh non-linearity after convolution and it has no spectral normalisation [8]. Layers conv_0 up to conv_4 down scale the spatial dimensions of their output using a bilinear down-sampling kernel. Equivalently, layers d_conv_4 up to d_conv_1 perform bilinear up-sampling with a factor of 2, before convolution is applied.

[2] Xu Chen, Jie Song, and Otmar Hilliges. Monocular neural image based rendering with continuous view control. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 3

[3] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4

[4] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *arXiv Preprint*, 2017. 3, 4

[5] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 3

[6] Sunghoon Im, Hae-Gon Jeon, Stephen Lin, and In So Kweon. DpsNet: End-to-end deep plane sweep stereo. In *arXiv Preprint*, 2019. 4

[7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *arXiv Preprint*, 2014. 4

[8] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv Preprint*, 2018. 6

[9] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 3, 4

[10] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[11] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo Magnification: Learning view synthesis using multiplane images. In *ACM Transactions on Graphics (SIGGRAPH)*, 2018. 1, 3

[12] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *European Conference on Computer Vision (ECCV)*, 2016. 3

[13] Yi Zhu, Karan Sapra, Fitsum A Reda, Kevin J Shih, Shawn Newsam, Andrew Tao, and Bryan Catanzaro. Improving semantic segmentation via video propagation and label relaxation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 4