Supplementary Material: MCDS-VSS

A Evaluation Metrics

To evaluate MCDS-VSS, we compute its segmentation performance, temporal consistency, throughput and inference speed.

Following the standard practice, we use the mean Intersection over Union (mIoU) to evaluate the segmentation performance.

To evaluate the temporal consistency (TC) of a VSS model, we closely follow the procedure proposed by Liu *et al*. [9], in which we compute the mean flow warping error between every two neighboring frames. More precisely, we use FlowNet2 [7] to compute the optical flow between two adjacent frames, and warp the predicted segmentation maps from time-step $t-1$ into time t. We then calculate the mIoU between the warped and actual target segmentations. Following [9], we evaluate TC using a subset of 100 sequences from the validation set.

We measure the throughput and inference speed of our model in frames per second (FPS) and milliseconds (ms), respectively. For this purpose, we perform inference with MCDS-VSS on 200 different video sequences of 6 frames and average the throughput and time across frames and sequences.

B Implementation Details

B.1 Network Details

In this section, we describe the network architectures and operation of each module in MCDS-VSS. We emphasize that MCDS-VSS is architecture-agnostic and could be implemented with different, e.g. more powerful or efficient, model designs.

Image Encoder & Segmentation Decoder: We implement two distinct MCDS-VSS variants, whose image encoder and segmentation decoder follow the architecture of two popular image semantic segmentation models, namely DeepLabV3+ [1] with a ResNet18 [5] backbone and HRNetV2 [12] with a channel multiplier of 18, respectively. In both cases we initialize the parameters of the encoders with those of the model pretrained on ImageNet, whereas the segmentation decoders are initialized with random weights.

Motion Encoder: The motion encoder concatenates the image features from two consecutive time steps (i.e. $\mathbf{h}_{t-1}$ and $\mathbf{h}_t$) across the channel dimension, and processes this representation with three convolutional layers, followed by batch normalization and ReLU activation functions.

Depth Decoder: The network architecture of our depth decoder, which is reported in Table 1, closely follows [3, 4]. It is composed of five convolutional blocks using reflection

Table 1: Network architecture of the depth decoder module.

Layer	Modules	Output Dim.
Block 1	Conv + ELU	256 × H/8 × W/8
	Conv + ELU	256 × H/8 × W/8
Block 2	Conv + ELU	128 × H/8 × W/8
	Conv + ELU	128 × H/8 × W/8
Block 3	Conv + ELU + Ups.	64 × H/4 × W/4
	Conv + ELU	64 × H/4 × W/4
Block 4	Conv + ELU + Ups.	32 × H/2 × W/2
	Conv + ELU	32 × H/2 × W/2
Block 5	Conv + ELU + Ups.	16 × H × W
	Conv + ELU	16 × H × W
Disp. Pred.	Conv + Sigmoid	1 × H × W

Table 2: Network architecture of the ego-motion decoder module.

Layer	Modules	Output Dim.
Block 1	Conv + BN + ReLU	256 × H/8 × W/8
Block 2	Conv + BN + ReLU	256 × H/8 × W/8
Block 3	Conv + BN + ReLU	128 × H/8 × W/8
Block 4	Conv + Pool + ReLU	128 × H/16 × W/16
Ego Motion	Conv	6 × H/16 × W/16
	Global Avg. Pool	6

padding, followed by ELU nonlinearities. Each of the last three convolutional blocks upsamples the feature maps by a factor of two using nearest-neighbor upsampling. The depth decoder outputs normalized inverse depth maps $\mathbf{d}^{\triangleleft}$, which are then converted into depth maps $\mathbf{d}$ by:

$$\frac{1}{\mathbf{d}} = \frac{1}{D_{\min}} + \left(\frac{1}{D_{\max}} - \frac{1}{D_{\min}}\right) \cdot \mathbf{d}^{\triangleleft}, \tag{1}$$

where $D_{\min}$ and $D_{\max}$ are constant values defining the minimum and maximum depth values in the scene, set to $D_{\min} = 0.1$m and $D_{\max} = 100$m for the Cityscapes dataset.

Ego-Motion Decoder: The ego-motion decoder, which is reported in Table 2, processes the motion features with a series of convolution layers, followed by batch normalization and ReLU nonlinearities. The final layer outputs a 6-dimensional vector representing the translation and rotation (parameterized as Euler angles) of the camera transformation matrix.

Residual Flow Decoder: The residual flow decoder is implemented as a modified lightweight version of RAFT [11]. To seamlessly integrate this module into our MCDS-VSS filter, we modify the implementation of `raft_small` provided by PyTorch[1] by replacing the expensive feature and context encoders with a single convolutional block that directly processes the ego-warped $\mathbf{s}_t^{\text{ego}}$ and image features $\mathbf{h}_t$. We set the number of refinement iterations to 12.

MCDS-VSS Filter: We instantiate the MCDS-VSS filter using the modules described above, after being pretrained for semantic segmentation, SSL of depth and ego-motion, and distillation of object motion. For the first image in a video sequence, MCDS-VSS directly predicts its semantic segmentation, without the use of any temporal filtering. For all other frames, MCDS-VSS employs the structured filtering method described in the paper. The scene feature state is initialized with the image features from the first frame in the video sequence ($\mathbf{s}_1 = \mathbf{h}_1$), whereas the initial camera state $\mathbf{c}_1$ is initialized with zeros. We experimented with learning the initial state representations; however it did not yield any qualitative or quantitative improvements, while increasing the number of learnable parameters.

[1] https://pytorch.org/vision/main/models/raft.html

Table 3: MCDS-VSS training stages and hyper-parameters.

Stage	Training Goal	Loss Function	LR	Batch	Train Time	# Imgs
1	Segmentation & SSL Geometry	$\mathcal{L}_{\text{Segm}} + \lambda_{\text{D}} \cdot \mathcal{L}_{\text{Depth}}$	$2 \cdot 10^{-4}$	12	40h	3
2	Distillation of Object Motion	$\mathcal{L}_{\text{Flow}}$	$1 \cdot 10^{-4}$	4	18h	2
3	Ego-Motion Filter	$\mathcal{L}_{\text{Ego}}$	$8 \cdot 10^{-5}$	8	8h	6
4	Temporal Integration	$\mathcal{L}_{\text{Segm}} + \lambda_{\text{TC}} \cdot \mathcal{L}_{\text{TC}}$	$8 \cdot 10^{-5}$	4	12h	6

Table 4: Throughput, inference speed (in ms) and number of learnable parameters for MCDS-VSS based on DeepLabV3+.

Model	# Params.	FPS	Inf. (ms)
Image Enc $\mathcal{E}_{\text{x}}$	15.3M	76.9	12.9
Motion Enc $\mathcal{E}_{\text{m}}$	4.8M	279.4	3.6
Motion Update	2.8M	476.2	2.1
Ego-Motion Dec $\mathcal{D}_{\text{c}}$	1.6M	492.8	2.0
Depth Dec $\mathcal{D}_{\text{d}}$	1.9M	227.1	4.4
Ego-Motion Comp.	0	63.4	15.8
Residual Flow Dec $\mathcal{R}_{\text{f}}$	2.7M	14.0	71.8
Object Motion Comp.	0	775.6	1.3
Feature Fusion	2.6M	215.3	4.6
Segmentation Dec $\mathcal{D}_{\text{y}}$	1.3M	196.7	5.1
Total MCDS-VSS	32.9M	9.0	111.6

Table 5: Throughput, inference speed (in ms) and number of learnable parameters for MCDS-VSS based on HRNetV2.

Model	# Params.	FPS	Inf. (ms)
Image Enc $\mathcal{E}_{\text{x}}$	9.5M	36.7	27.2
Motion Enc $\mathcal{E}_{\text{m}}$	5.0M	75.8	13.2
Motion Update	2.8M	166.7	6.0
Ego-Motion Dec $\mathcal{D}_{\text{c}}$	1.6M	174.4	5.7
Depth Dec $\mathcal{D}_{\text{d}}$	1.9M	62.4	16.0
Ego-Motion Comp.	0	47.9	20.9
Residual Flow Dec $\mathcal{R}_{\text{f}}$	2.9M	13.4	74.6
Object Motion Comp.	0	886.2	1.1
Feature Fusion	2.6M	76.8	13.0
Segmentation Dec $\mathcal{D}_{\text{y}}$	78.8K	627.6	1.6
Total MCDS-VSS	26.2M	5.6	177.6

B.2 Training and Inference

All our models are implemented in PyTorch [10] and trained with two NVIDIA A100 (80GB) GPUs. For each of the four training stages undergone by MCDS-VSS, we report in Table 3 the most relevant hyper-parameters, including the approximate training time, learning rate, batch size and number of images per sequence. We empirically set the loss weight values to $\lambda_{\text{D}} = 2$, $\lambda_{\text{Reg}} = 10^{-5}$ and $\lambda_{\text{TC}} = 1$.

Tables 4 and 5 report the number of learnable parameters, throughput and inference time for each individual module, as well as for the complete model, for the MCDS-VSS variants based on DeepLabV3+ and HRNetV2, respectively. We emphasize that the ego-motion and object motion compensation modules do not have any learnable parameters, but instead hard-wire our knowledge from the moving camera dynamic scene domain to project the previous scene features into the current time-step using geometry and motion representations. We also observe that MCDS-VSS inference is severely limited by its residual flow decoder. Adapting such module to exploit recent advances in fast optical flow estimation [8]), as well as using more efficient image encoders [6], could allow MCDS-VSS to be used for real time video semantic segmentation.

C Quantitative Results

In Table 6 we compare for individual classes of the Cityscapes dataset the segmentation performance and temporal consistency of MCDS-VSS with an HRNetV2 baseline trained for semantic segmentation and SSL of geometry and motion. MCDS-VSS achieves the

Table 6: Segmentation performance (mIoU) and temporal consistency (TC) for individual Cityscapes classes. We compare MCDS-VSS with HRNetV2 backbone with an HRNetV2 model trained for semantic segmentation and SSL of geometry and motion.

		Road	*Sidewalk*	*Building*	*Wall*	*Fence*	*Pole*	*Traf. Light*	*Traf. Sign*	*Person*	*Rider*	*Car*	*Truck*	*Bus*	*Train*	*Motorbike*	*Bicycle*	*Mean*
mIoU	Baseline	**98.1**	84.3	92.5	50.3	59.2	**66.5**	70.3	**79.3**	**81.8**	63.5	94.6	72.4	83.5	67.7	61.4	76.6	76.5
	Ours	**98.1**	**84.7**	**92.7**	**51.3**	**59.6**	65.9	**70.5**	**79.3**	81.6	**63.7**	**94.9**	**77.6**	**85.0**	**70.6**	**61.7**	**76.7**	**77.1**
TC	Baseline	98.7	86.0	91.1	60.8	60.1	67.1	62.4	77.1	69.0	65.8	89.0	58.1	66.8	45.8	49.4	66.6	71.7
	Ours	**98.9**	**88.2**	**92.1**	**64.1**	**65.9**	**69.1**	**69.2**	**79.5**	**70.5**	**67.8**	**90.5**	**65.2**	**69.6**	**48.3**	**53.6**	**71.0**	**75.3**

best segmentation performance and temporal consistency for most classes in the dataset, especially for those corresponding to moving objects, such as *car*, *truck*, *bus* or *train*.

D Qualitative Results

D.1 Effect of Each MCDS-VSS Stage

In Figure 1 we display the semantic segmentations obtained when decoding the scene features from different stages of our MCDS-VSS filter. We can observe how the segmentations after ego-motion compensation (Figure 1 b) atone for the movement of the ego-vehicle, correctly representing static scene features such as buildings or the bicycle. However, the dynamics of moving objects (e.g. yellow car) are not addressed in this step, thus not compensating for such movement. This limitation is addressed in the object motion compensation step (Figure 1 c). However, the disocclusions resulting from the moving car and inaccuracies in the residual flow estimation can lead to segmentation errors. Finally, fusing the projected scene state features with the current observations (Figure 1 d) leads to a more accurate segmentation of the scene. In (Figure 1 e) we visualize the update gate mask $\mathbf{u}$ employed for feature fusion. For visualization purposes, we take the mean across all channels and assign lighter colors to spatial locations where MCDS-VSS relies on the propagated state $\mathbf{s}_t^{\text{full}}$, whereas darker colors correspond to the current features $\mathbf{h}_t$. We observe that MCDS-VSS relies on the scene state to represent static areas such as buildings or the street, whereas it relies heavier on observations for accurately segmenting disoccluded areas of the image, fast moving objects or thin structures (e.g. poles or street signs).

D.2 MCDS-VSS Qualitative Results

In Figure 2, we show a qualitative result comparing MCDS-VSS with the HRNetV2 baseline. Our method achieves more accurate and temporally consistent segmentations compared to HRNet, correctly segmenting the traffic signs and reducing the amount of flickering between frames.

In Figures 3–6, we show on four validation sequences how MCDS-VSS obtains an accurate and temporally consistent segmentation of the scene, estimates the scene depth, and computes the residual motion flow, which encodes the movement of dynamic objects in the scene, as well as some minor motion corrections for other objects and scene features.

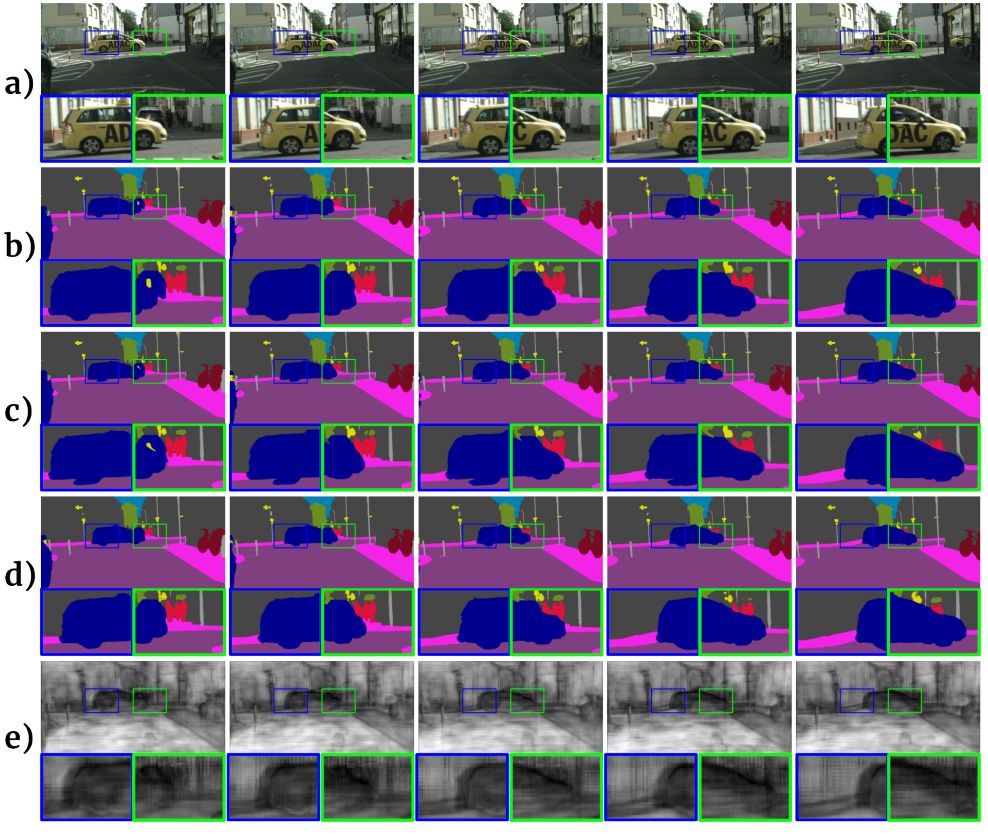

Figure 1: Video segmentation for each stage in MCDS-VSS. **a)** Input images, **b)** segmentation after ego-motion compensation, **c)** segmentation after object motion compensation, **d)** segmentation after feature fusion, **e)** feature fusion update mask, lighter colors mean that filter information is used, whereas darker ones correspond to observations.

D.3 Cross-Dataset Evaluation

We qualitatively evaluate the robustness of MCDS-VSS by performing a cross-dataset validation in which a DeepLabV3+ baseline and our MCDS-VSS model trained on Cityscapes are qualitatively evaluated without retraining on sequences from the KITTI [7] dataset. Figures 7 and 8 illustrate the semantic segmentation predictions of both models, as well as the MCDS-VSS depth estimates on two validation sequences of the KITTI dataset. Due to differences with respect to the training data in the camera model and calibration, as well as different image resolution and aspect ratio, the segmentation performance of both models on the KITTI dataset is severely degraded with respect to Cityscapes. However, MCDS-VSS achieves a more accurate and temporally consistent video segmentation, thus verifying that incorporating geometry and motion inductive biases from the moving camera dynamic scene domain into the VSS model design leads to more robust representations and segmentation results.

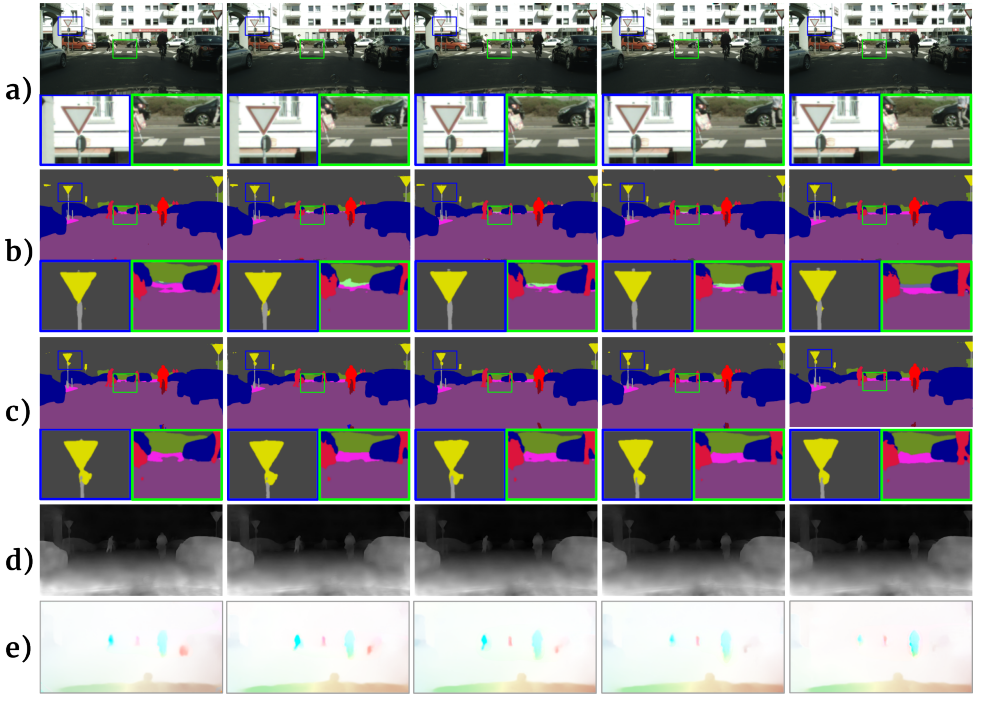

Figure 2: Qualitative evaluation. **a)** Input frames, **b)** HRNetV2, **c)** MCDS-VSS , **d)** Estimated scene depth, **e)** Estimated residual flow. We highlight areas of the segmentation masks where MCDS-VSS obtains visibly more accurate and temporally consistent segmentations.

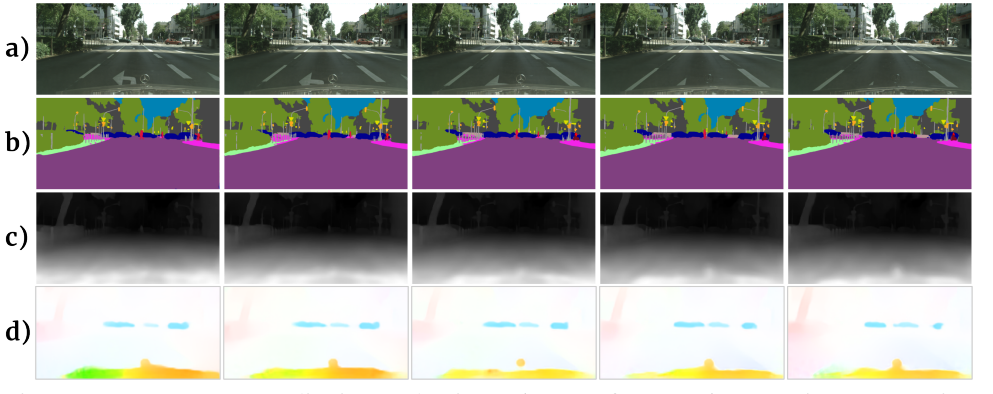

Figure 3: MCDS-VSS qualitative evaluation. **a)** Input frames, **b)** semantic segmentation, **c)** scene depth, **d)** residual flow.

D.4 Point Clouds

Figures 9–12 show examples of RGB and semantic point clouds rendered by backprojecting image values and semantic labels using the depth maps estimated by MCDS-VSS and known camera intrinsics. The high-quality depth maps computed by MCDS-VSS allow for an accurate 3D representation of the scene.

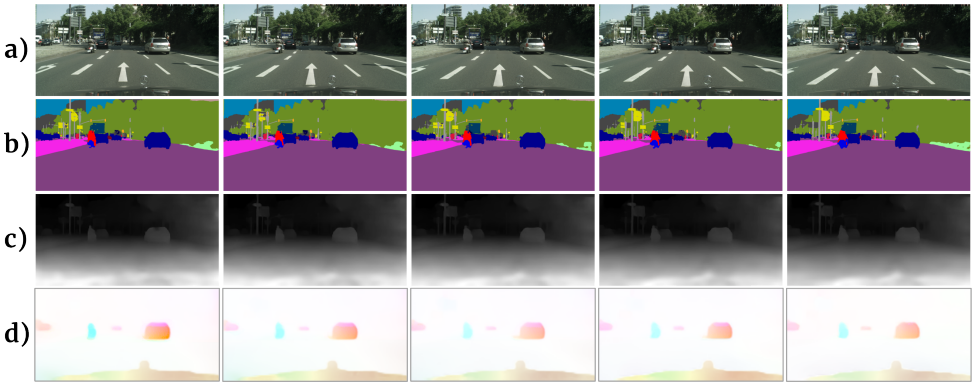

Figure 4: MCDS-VSS qualitative evaluation. **a)** Input frames, **b)** semantic segmentation, **c)** scene depth, **d)** residual flow.

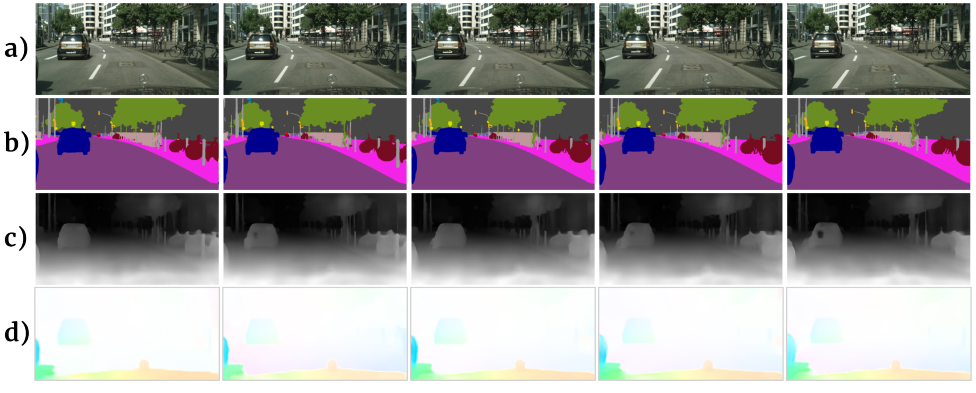

Figure 5: MCDS-VSS qualitative evaluation. **a)** Input frames, **b)** semantic segmentation, **c)** scene depth, **d)** residual flow.

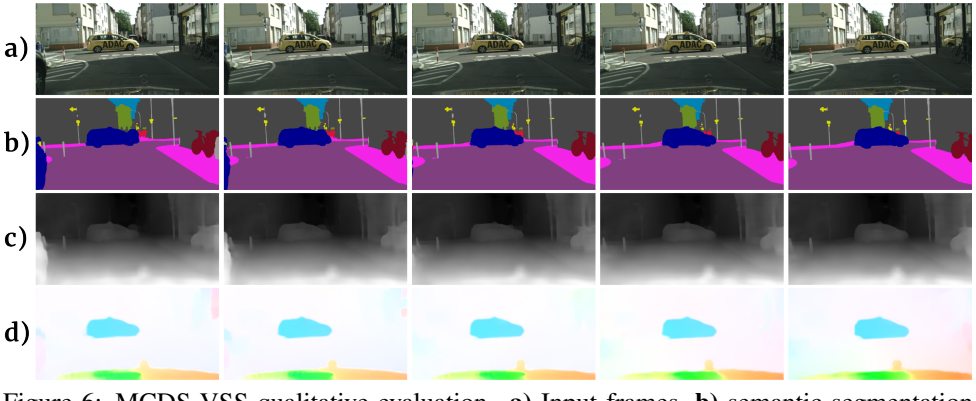

Figure 6: MCDS-VSS qualitative evaluation. **a)** Input frames, **b)** semantic segmentation, **c)** scene depth, **d)** residual flow.

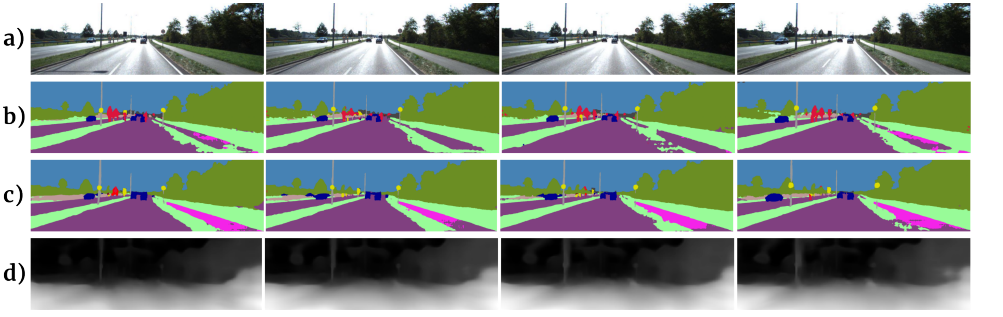

Figure 7: Cross-dataset qualitative evaluation of models trained on Cityscapes and evaluated on KITTI. **a)** Input frames, **b)** DeepLabV3+ baseline, **c)** MCDS-VSS (ours), **d)** estimated scene depth.

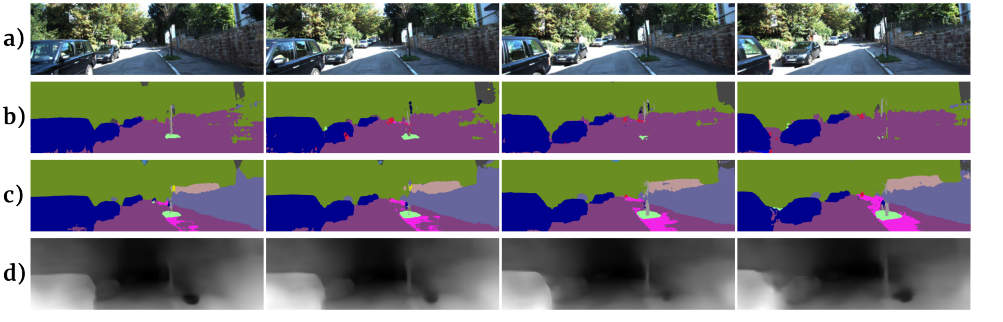

Figure 8: Cross-dataset qualitative evaluation of models trained on Cityscapes and evaluated on KITTI. **a)** Input frames, **b)** DeepLabV3+ baseline, **c)** MCDS-VSS (ours), **d)** estimated scene depth.

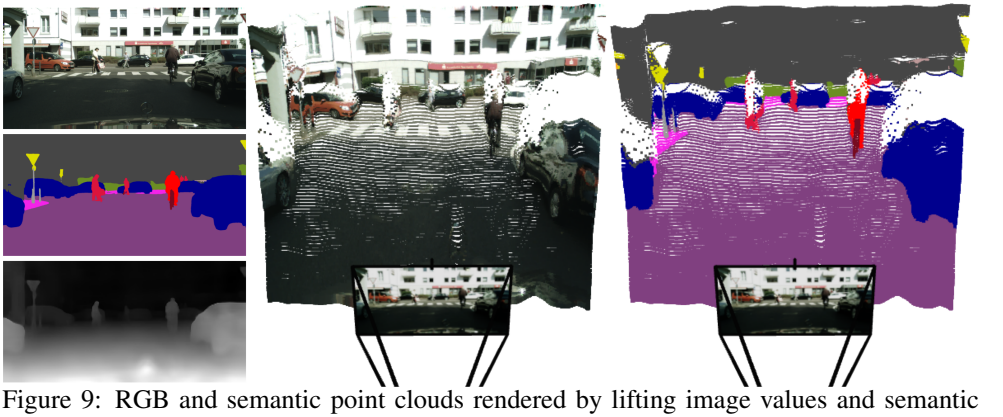

Figure 9: RGB and semantic point clouds rendered by lifting image values and semantic labels to 3D space.

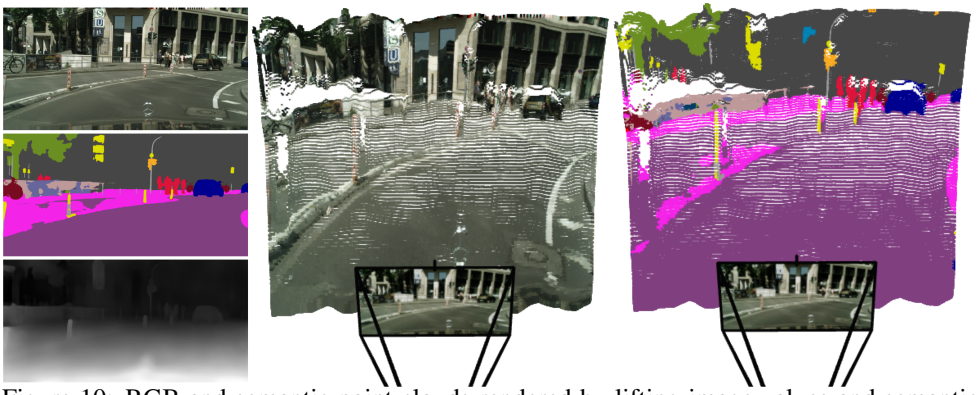

Figure 10: RGB and semantic point clouds rendered by lifting image values and semantic labels to 3D space.

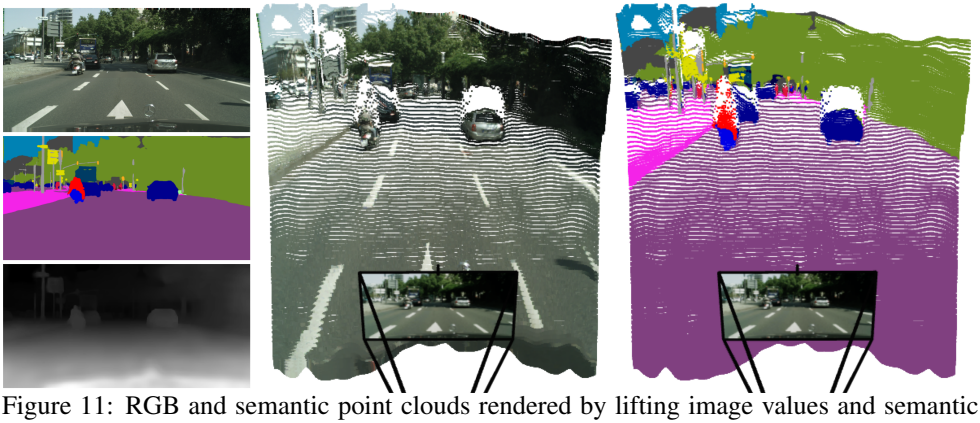

Figure 11: RGB and semantic point clouds rendered by lifting image values and semantic labels to 3D space.

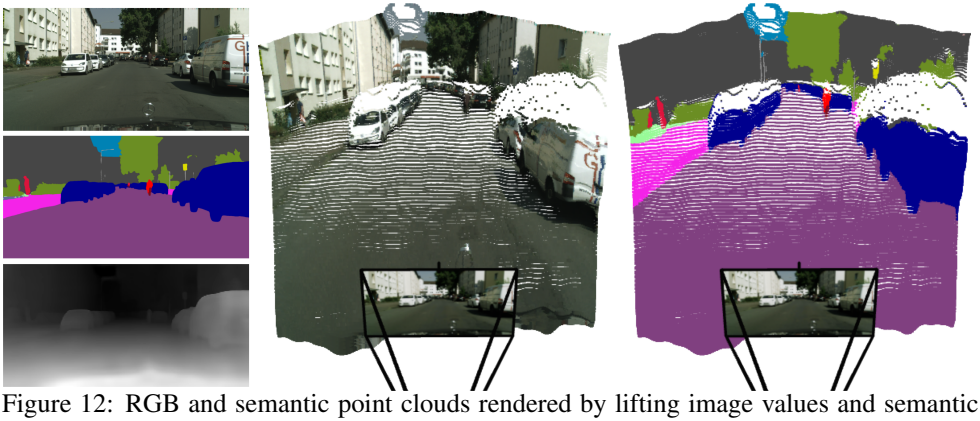

Figure 12: RGB and semantic point clouds rendered by lifting image values and semantic labels to 3D space.

References

[1] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *European Conference on Computer Vision (ECCV)*, pages 801–818, 2018.

[2] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[3] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 270–279, 2017.

[4] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3828–3838, 2019.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[6] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for MobileNetV3. In *IEEE/CVF International Conference on Computer Vision (CVPR)*, pages 1314–1324, 2019.

[7] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2462–2470, 2017.

[8] Lingtong Kong, Chunhua Shen, and Jie Yang. FastFlowNet: A lightweight network for fast optical flow estimation. In *International Conference on Robotics and Automation (ICRA)*, pages 10310–10316. IEEE, 2021.

[9] Yifan Liu, Chunhua Shen, Changqian Yu, and Jingdong Wang. Efficient semantic video segmentation with per-frame inference. In *European Conference on Computer Vision (ECCV)*, pages 352–368, 2020.

[10] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *International Conference on Neural Information Processing Systems Workshops (NeurIPS-W)*, 2017.

[11] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision (ECCV)*, pages 402–419, 2020.

[12] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.