Supplementary Material
DISeR: Designing Imaging Systems with Reinforcement Learning

Cam Config	Mean (x,z)	Std (x,z)	Mean Yaw	Std Yaw
1	$(-4.6, 79.2)$	$(10.0, 1.9)$	-15.7	39.8
2	$(-8.3, 78.3)$	$(7.8, 2.7)$	-3.6	43.3
	$(4.6, 77.7)$	$(9.1, 3.2)$	8.8	43.7
3	$(-10.4, 77.8)$	$(6.4, 2.9)$	-0.6	43.7
	$(-1.1, 77.6)$	$(8.6, 3.1)$	9.3	43.1
	$(8.5, 77.3)$	$(7.2, 3.3)$	15.4	41.2
4	$(-11.4, 77.7)$	$(5.4, 3.0)$	3.2	45.1
	$(-4.3, 77.6)$	$(7.5, 3.2)$	11.4	43.5
	$(3.5, 77.2)$	$(7.5, 3.2)$	15.1	41.7
	$(10.9, 77.4)$	$(5.5, 3.2)$	17.0	40.7
5	$(-12.1, 77.7)$	$(4.6, 3.1)$	5.4	43.4
	$(-6.5, 77.9)$	$(6.7, 3.0)$	8.0	44.2
	$(-0.17, 77.4)$	$(7.3, 3.3)$	14.0	41.5
	$(6.6, 77.1)$	$(6.8, 3.4)$	17.9	41.7
	$(12.2, 77.2)$	$(4.5, 3.3)$	18.7	40.5

Table 1: Distribution CD Actions: We show the mean and standard deviation of the actions taken by the CD after training. The CD always chooses to place a camera in the back of the allowed region (green box in Fig. 4) while spreading the rest of the cameras across the x-axis (mean x-position cover the entire box). For instance, the largest baseline between 3,4 and 5 cameras are roughly the same as the CD maximizes the spread of cameras along the x-axis while minimizing the z-axis variation. Additionally, the yaw has the largest variance of the parameters, which suggests that the CD has learned a strategy that exploits the yaw to find the object instead of the position.

1. Additional Results

We provide additional experimental results and details below. We refer to the camera designer as CD and perception model as PM.

1.1. Depth Estimation

We show that the CD and PM are able to learn intuitions that hold true in conventional multi-view stereo. We evaluate the individual components, the CD and PM, in isolation in Fig. 5 of the main paper. We now discuss additional results by analyzing the distribution of actions taken by CD, and the results from supervised training of the PM.

Coverage	L1 Loss
0	14.0
1	9.2
2	7.2
3	5.7

Table 2: We show that the L1 loss consistently decreases as more cameras see the sphere.

Distribution of Actions: Table 1 shows the mean and standard deviations (std) over 7,000 trials for actions taken by the CD based on the final camera configuration (number of cameras) at the end of the episode. We notice that regardless of the camera configuration, the CD almost always chooses to place the camera at the back of the allowed region, maximizing distance to the scene, and thus allowing more of it in its field of view. It maximizes z-position (max: 80) and has a very small std. The mean x-position of the camera is always maximized. For example, the largest baseline between the furthest cameras for camera configurations with 3, 4, and 5 cameras is roughly the same. For the 2 camera-configuration, the left camera is placed at -8.3, and the right camera is placed at 4.6 which is not as wide as it could be. The mean yaw of this configuration shows that on average the cameras face opposing directions: $-3.6°$ for left camera, and $+8.8°$ for right. Moreover, the x-position std is high for the right camera so the limited baseline could be due to the narrow FoV (45°).

We also note the distribution of yaw angles in the camera configurations. In the 2-camera configuration, the yaw angles oppose each other and, as the CD adds more cameras, the yaw of the left-most camera reduces to 0° while the right ones have a yaw 15° to the right. Lastly, we note that the yaw angles have the highest std when compared to the x and z positions' std, which suggests that the CD might have learned to fix the cameras around a certain area and rather exploit the yaw, range of [−60,60], to find the object.

Supervised Learning: To verify if the PM can indeed learn to estimate accurate depth with stereo, rather than monocular, cues in our environment, we

conduct a supervised experiment. We first create two datasets, monocular and stereo, by randomly sampling the sphere from the same region described in Section 4.1.1 of the main text. For each sampled sphere, we sample a random position directly in-front of the sphere and place a monocular camera looking at the object. We also randomly sample two cameras a random distance apart (such that the sphere is visible to both the cameras). The two images from two cameras form the stereo pair for the two camera-configuration setup. We sample 7,500 training samples and then train two PMs in a supervised fashion (same architecture): one network on the one camera-configuration dataset and another on the two camera-configuration dataset.

We show plots in Figure 1 of the training loss and baseline experiments. We show that PM model jointly trained with the CD, shown in the main text, achieves similar results as the supervised model. Specifically, Fig. 1.a shows that the training loss for the two camera-configuration PM is substantially lower, verifying that the lack of monocular cues in our environment enable the stereo setup to achieve more accurate depth estimation. Moreover, on the test sets, the stereo setup outperformed monocular with a test L1 loss of 3.78 vs. 5.40 respectively. In Fig. 1.b, we perform the baseline experiment, described in Sec.4.1.2 of the main text. We show that the behavior of the PM is similar to the joint training setup from the main text i.e. the PM model trained with stereo setup estimates more accurate depth and has lower variance than the monocular setup- which is subject to size of sphere and position of the camera w.r.t sphere. However, the primary difference between the experiments is that the only the number of cameras, not the baseline between the cameras, has an effect on the L1 error in the supervised settings.

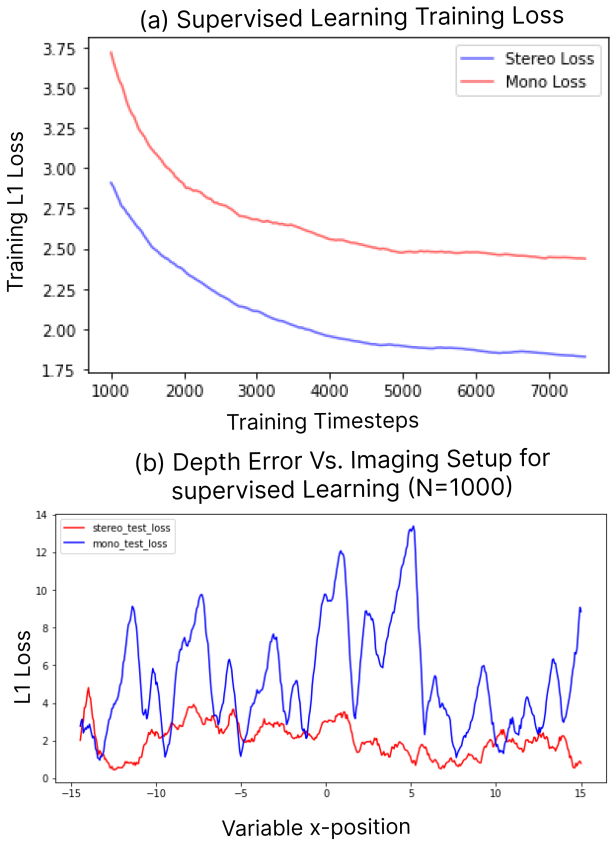

Figure 1: Learning Stereo Cues with Supervised Learning: We train two PMs – one on a one camera configuration and one on a two camera configuration. We show that PM trained with a two camera configuration outperforms the one trained with one camera both during training and when evaluated on the same test set (5.40 vs. 3.78). This result verifies that the lack of monocular cues in our environment enable stereo setup to better estimate depth. In (b) we perform the baseline experiment (described in the main text) on the supervised models and show that the PM model trained in conjunction with the CD shows similar behavior of lower overall depth error and variance with the 2 camera setup.

1.2. AV Camera Rig Design

An overview of our method for autonomous vehicle (AV) camera rig design is shown in Fig. 2, using the same steps shown in Fig. 2 of the main text.

Mean & Std. Training Rewards: We repeat Expt. a and b three times and report the mean and standard deviation (std.) in Fig. 4. The mean and std. for Expt. c is in Fig. 5.

Expt. b: We include additional examples of candidate rig designs generated from the trained CD in Fig. 3. While the FoV and yaw of the cameras vary between candidate camera rigs, we note that the pitch and height are always $-20°$ and 0.5 m above the roof, respectively. We also visualize the BEV segmentation predictions of two Cross View Transformer (CVT) [6] models - one trained with our selected camera rig and one trained with the nuScenes camera rig. Visualizations are shown in Fig. 6; predictions are not thresholded and thus intensity indicates model confidence. These visualizations correspond to the models evaluated in Table 1 of the main text. When looking at the predictions, we observe that the model trained with our selected camera rig has higher confidence than the model trained on the nuScenes rig, suggesting that the CD has learned to place the cameras in such a way that maximizes BEV segmentation prediction confidence.

Expt. c: In expt. c, we train the CD and PM to

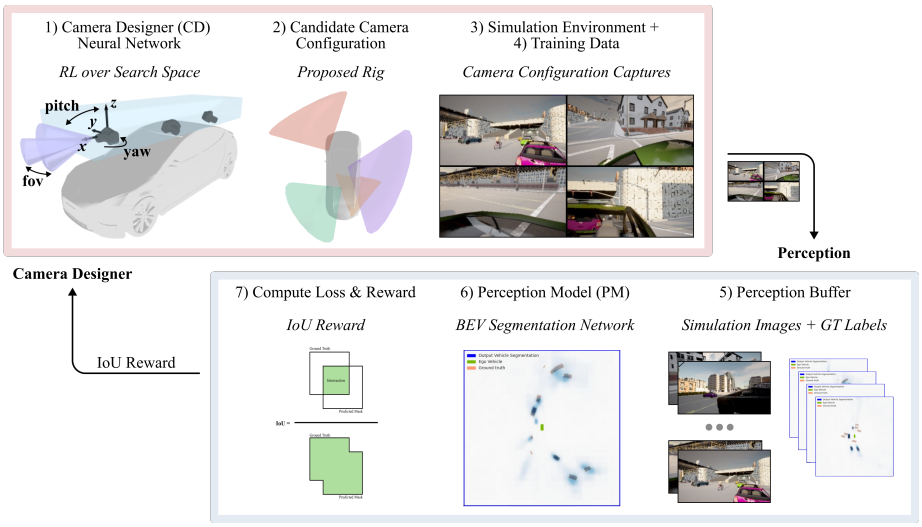

Figure 2: AV Camera Rig Design Method: Detailed visualization of our method for designing camera rigs, using the same steps shown in Fig. 2 of the main text. First, the CD proposes a candidate camera rig and data is collected from that rig in CARLA [1]. Then the data is added the perception buffer, which is used to train the PM. The PM computes the reward, which is used to update the CD.

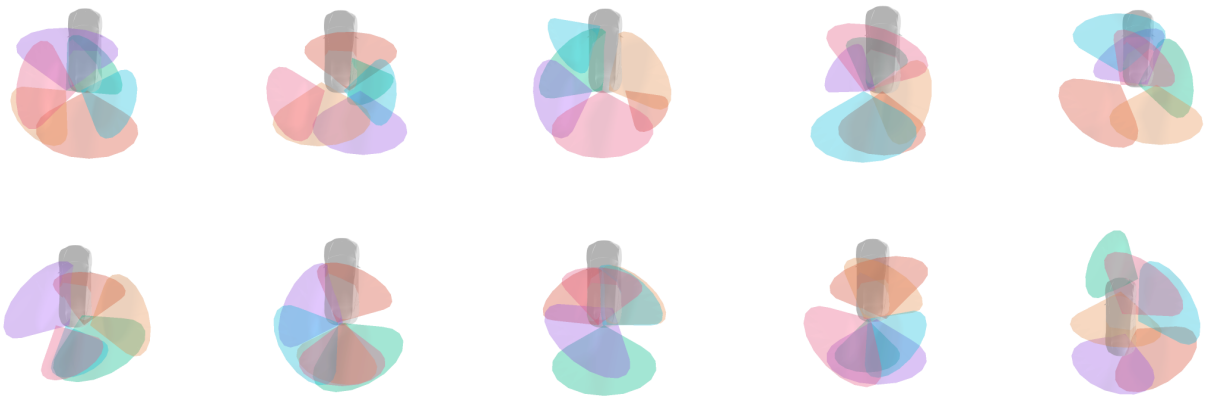

Figure 3: Camera Rig Candidates: Above are ten example candidate camera rigs generated by the learned CD as part of the CD evaluation for expt. b explained in the main text. While yaw and FoV vary between camera rigs, the pitch and height of each camera are consistently $-20°$ and 0.5 m above the roof, respectively.

create a camera rig that minimizes the number of cameras used while maximizing BEV segmentation accuracy. For this experiment, the CARLA environment is modified to only include vehicles in front of the ego-vehicle, between yaw angles -45°and 45°, incentivizing the CD to place cameras facing forward. We additionally place two penalties on camera placement to encourage a resource-constrained camera rig to be created. The first penalty of 0.025 is subtracted from overall reward when any additional cameras are placed. The second penalty subtracted from the overall reward is maximized at a value of 0.3 when an additional camera placed has the same yaw as an existing camera, and minimized at a value of 0 when the yaw difference between the selected camera and the closest existing camera is maximized.

We demonstrate in the main paper that under this resource-constrained, task-specific environment, the selected camera rig that is learned by the CD places only two cameras, both facing forward. Shown in Fig. 5 is the reward curve for this experiment. In this figure, we observe that, as in expts. a and b, the CD is able to learn a policy that improves rewards over time. The maximum reward in this experiment is lower than

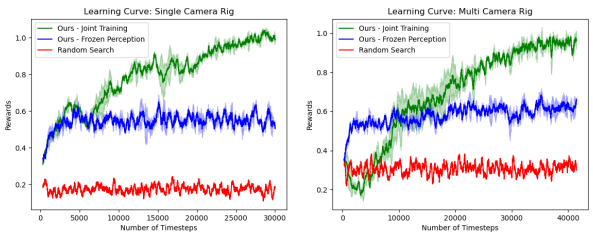

Figure 4: Results for AV Camera Rig Co-Design: Reward curves for experiments a and b, including mean and standard deviation (std) for our method (blue line is frozen PM and green line is jointly trained PM). Mean and std are computed over three trials.

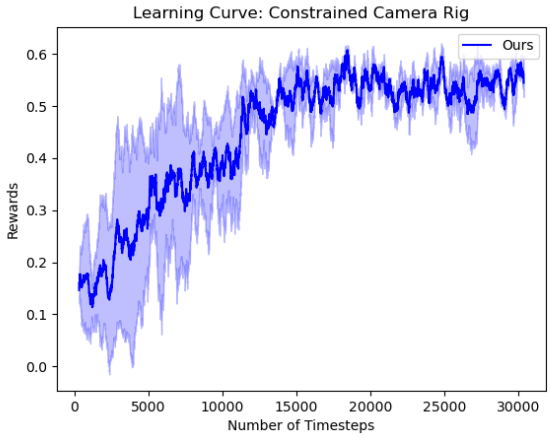

Figure 5: Constrained AV Camera Rig: Shown is the reward curve for Expt. c, where the camera designer (CD) and perception model (PM) are jointly trained to create a *resource-constrained* camera rig. A penalty is added each time a new camera is added to disincentivize the CD from placing unnecessary cameras. The environment is modified to only contain cameras in front of the ego-vehicle.

in expts. a and b because of the resource-constraint penalty subtracted from it. We find that by varying the maximum penalty magnitudes, the CD balances IoU in the BEV segmentation task with sufficiently disincentivizing adding extra cameras.

2. Implementation Details

All code was developed in PyTorch [3] and trained using the Adam optimizer [2]. Code is available from our project page.

2.1. Proximal Policy Optimization

We use proximal policy optimization (PPO) [5] to train the camera designer (CD). In PPO, a policy, π_k, is trained to predict actions that maximize the sum of rewards over time. A value function is trained to predict the advantage of each action, where advantage is the improvement in using the policy's actions compared to random actions. We use the PPO-Clip variant, which disincentivizes the policy from becoming too different from old versions of the policy during training by clipping the probability ratio term to be in $[1-\epsilon, 1+\epsilon]$. where ϵ is a hyper-parameter. The policy, π_k, is then updated via the following rule:

$$\pi_{k+1} =_\theta \underset{s,a\sim\pi_k}{E}[L(s,a,\theta_k,\theta] \quad (1)$$

We use the Stable Baselines3 implementation of PPO [4].

2.2. Depth Estimation

We use a perception buffer size of 35 when jointly training the CD and the PM. At each step of the rollout the loss and the reward is calculated and the PM is trained on the entire dataset for 2 epochs. The architecture is a custom Video ViT network which also inputs a flattened camera matrix per image. All the images and canera matrices, regardless of the number of cameras placed by the CD are, passed into the encoder, which outputs a constant feature vector of size 512 and 64 respectively. The feature vectors are concatenated and passed through a classification head that outputs the depth. We train the PPO in parallel with 5 processes with batch size of 128 and n. steps of 256. The input images are of fixed size (128,128). All actions are predicted in the range of $[-1, 1]$ and then rescaled to their corresponding range described in the main text. The reward is also re-scaled to $[-1, 1]$ where a reward of 1 is given with 0 error and -1 is given when the L1 error between predicted and ground truth depth is equal to, or greater than, the ground truth depth.

2.3. AV Camera Rig Design

We use a perception buffer size of six when training the CD and PM together, meaning that, at every step, the PM is trained with data from the 6 most recent episodes. The PM is trained for two epochs per step. The Cross View Transformers (CVT) model used as the PM is first pre-trained for 40 epochs on 25,000 training samples from random camera rigs. the PM is then jointly trained with the CD. During joint training, we use a learning rate of $4e-4$ for the PM. In all AV rig design experiments, we use the following hyperparameters to train PPO: learning rate=0.0003, n. steps=18, batch size=6, n. epochs=10, gamma=0.99, gae lambda=0.95, clip range=0.2. We parallelize PPO across two processes to reduce training time. All

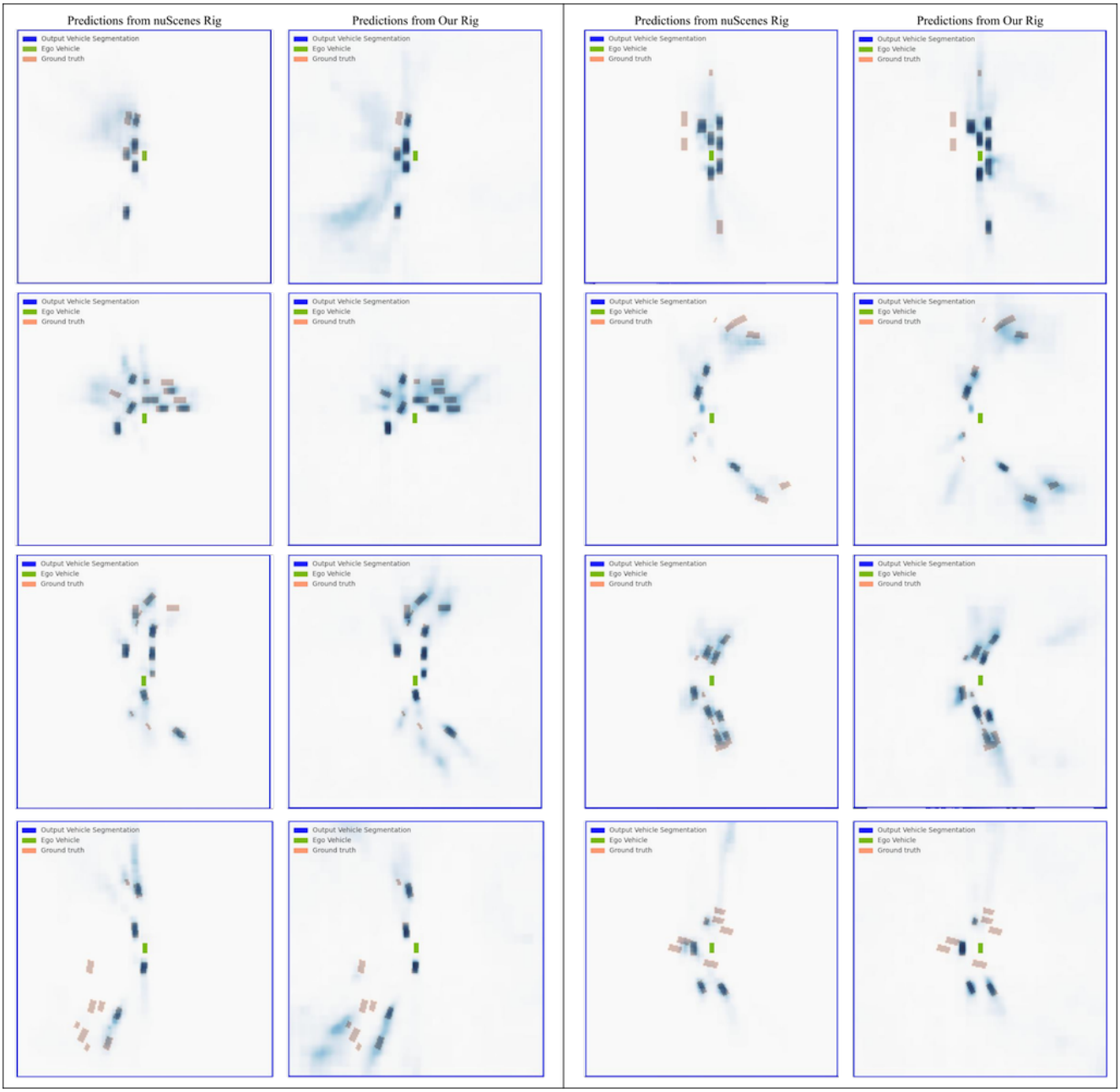

Figure 6: BEV Segmentation Predictions: Visualizations of the predictions made by a BEV segmentation model trained with data from the nuScenes rig (left) compared to our CD-optimized rig (right) on the same test scene. We observe that, on the same scenes, predictions from our rig are, in general, more accurate and have higher confidence.

actions are predicted in the range $[-1, 1]$ and then rescaled to their corresponding range described in the main text. Image resolution is fixed to (400,224) in all experiments.

References

[1] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017. 3

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[3] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 4

[4] Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3, 2019. 4

[5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 4

[6] Brady Zhou and Philipp Krähenbühl. Cross-view transformers for real-time map-view semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13760–13769, 2022. 2