Learned Two-Plane Perspective Prior based Image Resampling for Efficient Object Detection

A. Implementation, Training and Evaluation Details

A.1. Implementation Details

We implemented our approach using Pytorch [13] and mmdetection [3]. The two-plane perspective prior is implemented as a neural network layer with learnable parameters that are global (fixed warps parameterized by vanishing point). We employ differentiable versions of Direct Linear Transform [6] and $warp\_perspective$ from Kornia [15], while we reuse implementation of separable neural warps from [16]. To detect vanishing points, we employ NeurVPS [19] for fixed cameras. We use VPNet [12] with ResNet18 backbone for autonomous navigation.

Parameter Initialization We selected one representative image, and initialized the learnable parameters (i.e. θ's and α's) via visual inspection. The guiding principal was to enlarge far objects while trying to distort the close-by objects as less as possible. The same initial parameters are used for all the datasets. Please look at our code for the initial parameters.

A.2. Evaluation Details

Detection Model Choice We experiment with Faster R-CNN as our base detection model as prior work has shown it occupies the optimal sweet-spot [8] w.r.t latency and accuracy on modern GPUs. However, our approach is agnostic to the choice of detector and our results generalize to other detectors.

Latency: We capture end-to-end latency in milliseconds, that includes image pre-processing, network inference and post-processing, following protocol from prior work [16].

Scale The image down-sampling factor is equal in both spatial dimensions. So an image originally 1920×1200 (1x scale) when down-sampled to 0.25x is a 480×300 image.

A.3. Training Details

For training our proposed approaches, to train the Faster R-CNN model we use the Adam optimizer with a learning rate of 3×10^{-4}. For training any methods by Fovea [16] we follow their protocol. We follow the protocol mentioned by [4, 5] for training their approach.

Argoverse-HD We considered the same base architecture (Faster R-CNN) for all the methods. We compare with SOTA [16] using models provided by their public code release, and follow the training protocol prescribed in their work, training our models for 3 epochs.

WALT We considered the same base architecture (Faster R-CNN) for all the methods. We trained the models (and learnt the warping function parameters, if applicable) using the Adam optimizer with the same learning rate and other parameters for 6 epochs.

Vanishing Point Estimation For NeurVPS, we directly employ the pre-trained model trained on Natural Scenes (TMM17) dataset [20] part of their public code release. While for VPNet [12], as there is no public code release, we implement this architecture employing a ResNet18 backbone attached to a modified YOLO head. We omit the upsampling refinement procedure described in [12], as model's median error in vanishing point prediction is around 10 pixels with an average latency of 28 ms, which is sufficient for our method to work. The off-the-shelf model is executed at $n_v = 30$ to amortize the cost of executing this model. We also tried using LaneAF [1] to obtain lane lines (similar latency), however, we observed the method was prone to errors while clustering lines and obtaining the vanishing point.

B. Multiple Vanishing Points

Our method can consider additional planes that correspond to lines meeting at a different vanishing point. For example, a traffic camera with a wide field of view that is placed at an intersection observing two roads simultaneously would benefit from this. Assuming N vanishing points, considering Saliency S_{v_i} corresponding to vanishing point v_i,

$$S = \sum_{i=0}^{N} \lambda_i S_{v_i} \tag{1}$$

where λ_i's are learnable, initialized as $\frac{1}{N}$. Please observe the case of $N = 2$ in an image from the commuter bus dataset in Figure 1, wherein combining saliencies from two vanishing points (obtained from [10]) ensures far away objects of interest are sampled more.

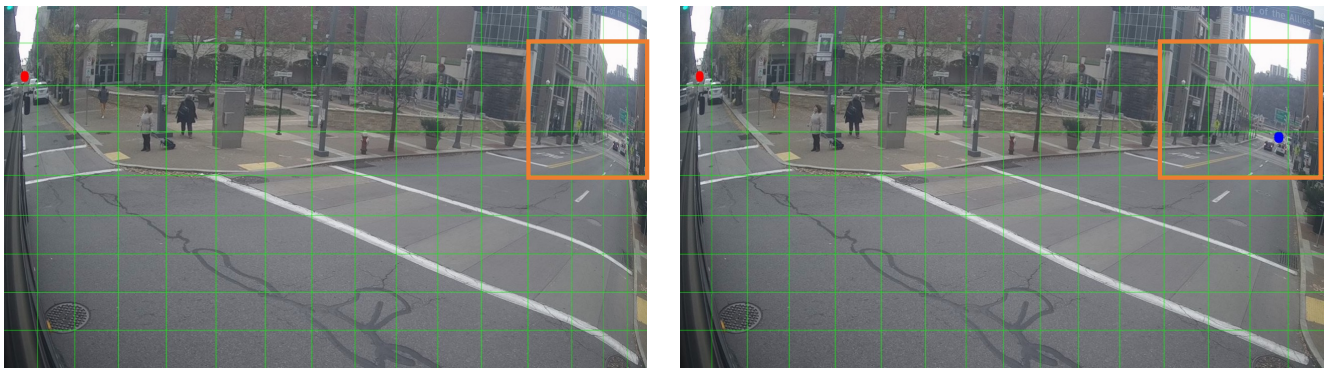

Figure 1. **Multiple Vanishing Points:** Saliency from First Vanishing Point parallel to sidewalk "compresses" far away cars on perpendicular road. Second Vanishing Point ensures those cars are not compressed. Please zoom in to observe the vanishing points and deformation.

Method	Scale	AP	AP_S	AP_M	AP_L
RetinaNet	0.5x	22.6	4.0	22.0	53.1
Fovea (S_I) [16]	0.5x	24.9	7.1	27.7	50.6
Two-Plane Prior	0.5x	**26.3**	**10.1**	**29.2**	50.5
Baseline at higher scales					
RetinaNet	0.75x	29.9	9.7	32.5	54.2

Table 1. **Alternate Detector:** We replace Faster R-CNN with RetinaNet (archetypal one-stage detector), and observe considerable improvements over Baseline (RetinaNet with uniform downsampling) and SOTA trained on Argoverse-HD dataset.

We observed in the datasets we considered, multiple vanishing points were rare as it generally requires a camera with a large field of view. Thus, we employed models [12, 19] trained on Natural Scenes dataset, which predict only one vanishing point. However, vanishing points can be estimated from other methods [10, 11, 19] which do predict all the vanishing points, but incur higher overheads.

C. Results on Another Detector

Adaptive spatial sampling mechanisms leverage and exploit priors corresponding to the input images in a way that is agnostic to the detection method. We expect our approach to generalize across detectors, similar to observations by such warping mechanisms and saliency priors proposed earlier [16]. We choose RetinaNet [9], a popular single-stage object detector as our archetypal example (Faster R-CNN [14] is the two-stage archetype). Results can be viewed in Table 1. Our approach improves upon both the baseline Faster R-CNN and SOTA, specially for small and medium sized objects, following the trends observed in the main manuscript.

Method	Scale	Model	AP	AP_S	AP_M	AP_L
Faster R-CNN	0.5x	COCO	15.3	1.1	12.5	40.5
Faster R-CNN	0.5x	AVHD	15.1	1.0	10.6	39.0
Fovea (S_D) [16]	0.5x	AVHD	13.7	1.3	10.0	34.7
Fovea (S_I) [16]	0.5x	AVHD	16.4	2.1	12.8	38.6
Two-Plane Prior (Psuedo.)	0.5x	AVHD	16.2	**4.7**	**15.9**	33.3
Two-Plane Prior (Psuedo.)	0.5x	COCO	**20.9**	**5.8**	**19.4**	**44.2**
Baseline at higher scales						
Faster R-CNN	0.75x	AVHD	19.7	3.0	16.1	44.2
Faster R-CNN	0.75x	COCO	20.3	3.7	18.2	45.3
Faster R-CNN	1x	AVHD	22.6	5.7	20.1	45.7
Faster R-CNN	1x	COCO	23.1	6.5	21.7	46.1

Table 2. **Generalization to BDD100K:** Scale in this case is fixed to 0.5x, AVHD refers to Argoverse-HD and COCO datasets respectively. AVHD models are finetuned from the pre-trained COCO model. We compare generalization on the BDD100K dataset. Our method assumes availability of training set images of BDD100K **and not labels**, we generate pseudo-labels from the available model (Section 3.6) to learn the Two Plane prior.

Method	sAP	sAP_S	sAP_M	sAP_L
StreamYOLO-L [17]	25.9	8.6	24.2	40.9
StreamYOLO-M [17]	25.9	9.2	24.8	41.0
StreamYOLO-S [17]	29.6	11.0	30.9	51.6
Ours	**30.0**	**13.7**	**31.5**	**52.2**

Table 3. **"Real-Time" Detectors**: Streaming Comparison on Argoverse-HD on Titan X. StreamYOLO-M and StreamYOLO-L single-frame latency is 45.8 ms and 62.9 ms respectively, is greater than 33ms, violating [17]'s "real-time" restriction. StreamYOLO-S satisfies (**20.8 ms**), hence has better performance.

D. Additional Results on Autonomous Driving

We shall consider the comparisons made in Section 5 on Argoverse-HD in the main manuscript and present some additional results (specially across different categories present

	Method	Scale	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	person	mbike	tffclight	bike	bus	stop	car	truck
	Faster R-CNN (Pre.)	0.5x	21.5	35.8	22.3	2.8	22.4	50.6	20.8	9.1	13.9	7.1	48.0	16.1	37.2	20.2
	Faster R-CNN	0.5x	24.2	38.9	26.1	4.9	29.0	50.9	22.8	7.5	23.3	5.9	44.6	19.3	43.7	26.6
	Fovea (Learned Nonsep.) [16]	0.5x	25.9	42.9	26.5	10.0	28.4	48.5	25.2	11.9	20.9	7.1	39.5	25.1	49.4	28.1
	Fovea (Learned Sep.) [16]	0.5x	27.2	44.8	28.3	12.2	29.1	46.6	24.2	14.0	22.6	7.7	39.5	31.8	50.0	27.8
SOTA	Fovea (S_D) [16]	0.5x	26.7	43.3	27.8	8.2	29.7	54.1	25.4	13.5	22.0	8.0	45.9	21.3	48.1	29.3
SOTA	Fovea (S_I) [16]	0.5x	28.0	45.5	29.2	10.4	31.0	**54.5**	27.3	16.9	**24.3**	9.0	44.5	23.2	50.5	28.4
SOTA	Fovea (L:S_I) [16]	0.5x	28.1	45.9	28.9	10.3	30.9	54.1	27.5	**17.9**	23.6	8.1	45.4	23.1	50.2	28.7
Ours	Ground Plane Prior	0.5x	**29.1**	**46.2**	**30.5**	**15.5**	27.5	48.3	**28.6**	15.0	10.4	**10.5**	33.7	**46.2**	**55.4**	**32.9**
Ours	Two-Plane Pr. (Psuedo.)	0.5x	27.1	43.4	28.4	9.8	28.9	50.2	**28.2**	16.2	20.6	8.1	**50.8**	26.1	45.2	21.7
Ours	Two-Plane Pr. (Avg VP)	0.5x	**29.6**	**45.9**	**31.6**	**12.7**	**30.7**	52.7	**28.4**	14.3	**24.3**	**11.9**	38.5	**31.6**	**53.9**	**34.2**
Ours	Two-Plane Prior	0.5x	**30.8**	**47.2**	**33.2**	**14.5**	**31.6**	52.9	**30.0**	16.7	24.1	**13.7**	35.9	**35.9**	**55.3**	**35.3**
	Baseline at higher scales															
	Faster R-CNN	0.75x	29.2	47.6	31.1	11.6	32.1	53.3	29.6	12.7	30.8	7.9	44.1	29.8	48.8	30.1
	Faster R-CNN	1.0x	33.3	53.9	35.0	16.8	34.8	53.6	33.1	20.9	38.7	6.7	44.7	36.7	52.7	32.7

Table 4. **Evaluation on the Argoverse-HD dataset:** This is an expanded version of the Table present in the manuscript. We see improvements for most of the objects that are on the ground, with much better overall performance for both small and medium sized objects along with improvements in AP_{50} and AP_{75}. Notice that Two-Plane prior performs at par with SOTA on "traffic-light" category. Pre. denotes Pretrained model, while Psuedo. denotes model trained with Psuedo-Labels from pretrained model (no access to Argoverse-HD labels) as described in Section 3.6. *We have bolded all of our method variations that perform better than SOTA.*

in the dataset).

Comparison with Learned Fovea [16]: Fovea [16] also proposed end-to-end global, dataset-wide saliency map S learned via backpropagation (Learned Seperable and Learned Nonseperable). However, they observed worse performance compared to their bounding box priors (S_D and S_I), see Table 4. We show that end-to-end learned saliency is better, with careful geometric parameterization.

Improved Performance on ground plane: We observe improved performance over state-of-the-art on every object category for objects on the ground plane (person, traffic light, bike, stop-sign, car, truck) apart from motorbike (See Table 4). On further observation, this might be an artifact of the label skew of the Argoverse-HD dataset ($mbike$ has the least number of instances). For objects not on the ground plane, like traffic-light, we observe performance as good as SOTA.

Generalization to BDD100K: We compare generalization from approaches trained on Argoverse-HD and COCO datasets to BDD100K MOT dataset (See Table 2). Our approach assumes access to training set images from BDD100K dataset, but not it's ground truth labels. As our priors can be learnt without access to ground truth data, we employ the method detailed in Section 3.6 to generate pseudo labels to learn and adapt the geometric parameters on this dataset by training on these pseudo-labels for 1 epoch only.

We observe that our method nearly matches SOTA when adapted starting from a model finetuned on Argoverse-HD, however, dramatically exceeds it's performance when adapted from a model solely trained on COCO. We believe the reason for this mismatch is due to catastrophic forgetting [7] observed in finetuned models when evaluated on out-of-distribution data. Lastly, the results indicate the benefits of learnability of our perspective prior, we observe increase in performance for "free" even when images are available without access to ground truth.

Comparison with "Real-Time" Detectors: Real-time detectors like [17] have been recently proposed which predict boxes G_{t+1} at time t (of frame F_{t+1}; available solely during training and not testing) given F_t to satisfy sAP. Approaches like these constraint the detector to perform the computation within a latency budget ($<$ 33ms or 30 FPS). Our methods are complementary to such detectors, as long as *their* constraint is satisfied.

However, real-time detectors (termed as "fast" strategy) might be suboptimal [8]. Satisfying the real-time detector constraint may not be optimal for every hardware platform, specially on slower edge devices. Such methods [17] are **not** hardware-agnostic, and model architecture choices are optimized for specific hardware (in their case, for a V100 GPU). On Titan X (See Table 3), their streaming performance (which is hardware dependent) is worse.

E. Tracking Smaller Objects for Longer

We provide an analysis of our approach observing how it improves object tracking. We wish to observe if the gains from our method translates to detecting far-away objects for longer period of time. We employ Argoverse-HD dataset for our experiments which have ground truth object IDs.

Setup: We employ a Faster R-CNN as our baseline and the tracker is fixed to IOU Tracker [2]. We additionally pair

the priors proposed by Fovea [16] for comparison. All the detectors are executed at 0.5x scale for fair comparison.

Tracking Visualizations: We present some tracking visualization in Figures 2 and 3. These visualizations motivate us to define the following metrics.

Detecting and Tracking for Longer: We wish to understand if Two-Plane Prior is able to detect an object for a longer lifespan. This is important in autonomous driving situations, wherein we want to detect far-away objects as quickly as possible or any object moving away from us.

Prior tracking quality metrics such as MT% and ML% check the ratio of tracks that are mostly tracked or mostly lost. However, this does not capture the track length improvements. We propose to compare the average extension of a track (**ATE**) compared to the baseline detection method. Given a track τ, E_τ can be positive or negative, and is given by,

$$E_\tau(m, b, gt) = (L_m - L_b)/L_{gt} \tag{2}$$

where m is the method, b is baseline and gt is the ground truth track, while L denotes track length. **ATE** is the average over tracks across all sequences. However, as the metric weighs all tracks equally, which is unfair for extremely small track lengths, thus, we only consider ground truth tracks which are atleast 5 seconds or 150 frames long.

Detecting and Tracking Smaller Objects: Given a track, we wish to observe if Two-Plane Prior is able to detect an object when it's "smaller" compared to other methods. This is important in autonomous driving situations, wherein we would like to detect further away objects, which would appear smaller. We wish to compute the minimum object size tracked (**MOS**). We employ a proxy for object size, $size(x) = log(area(x))$ where x denotes an object bounding box, as the area quadratically increases. For a given ground truth track τ, let o_τ denote minimum object size of an object, while O_τ denotes maximum object size. Let c_τ denote the minimum object size in the predicted track currently considered. We can write,

$$M_\tau = \frac{c_\tau - o_\tau}{O_\tau - o_\tau} \tag{3}$$

M_τ is averaged over all tracks across all sequences to obtain **MOS**.

F. Detection on the Commuter Bus

The bus is equipped with a Jetson AGX edge device. The edge device communicates with a modified onboard-NVR recording bus data from 7 cameras, two inside the bus and five on the outside of the bus. The cameras record data at $5FPS$ at 720P resolution for 8 working hours of the bus, totalling 1.08 million frames everyday. It is not feasible to transmit and process this data on the cloud due to bandwidth and compute limitations, and privacy concerns. Thus, the edge device and the NVR are part of a distributed edge-cloud infrastructure wherein the edge device is employed to process these simultaneous streams, only relevant frames are transmitted to cloud machines where we do further offline analysis.

Method	Scale	AP_{50}	AR	AR_S	AR_M	AR_L	Latency (ms)
Faster R-CNN	0.5x	45.0	31.7	0.5	37.3	46.9	154 ± 8.5
Two-Plane Prior	0.5x	**77.2**	**61.7**	**16.4**	**69.9**	**68.7**	158 ± 7.5
Faster R-CNN	0.75x	58.6	41.1	10.5	45.3	54.7	240 ± 8.5
Two-Plane Prior	0.75x	**84.5**	**68.3**	**38.8**	**72.9**	**73.3**	245 ± 10
Baseline at higher scales							
Faster R-CNN	1x	68.2	41.5	16.9	47.5	57.1	350 ± 15

Table 5. **Rare Object Detection on the Commuter Bus:** We compare our approach with a baseline Faster R-CNN. We observe improved precision and recall over the baseline, specially for small and medium sized objects. Do note, for ≈1FPS throughput over five simultaneous streams, average latency of 200ms should be achieved (however, this is not an enforced latency budget for streaming perception [8]).

We analyze bus streams to build an actionable map of public infrastructure, for instance, which areas need a trash pickup or where does snow needs to be shovelled. We also provide real-time feedback to the bus driver, informing them of people who may need assistance (say, on wheelchairs, or with a stroller or service animal) getting on the bus. Thus we employ an object detector to detect trash cans, garbage bags and people with an assistive device. Our system has to operate at near real-time on all streams simultaneously, rendering cloud-transmission-turn-around infeasible.

As we employ the edge device to filter out relevant frames, detecting all the objects in the scene is more important than the precision and localization accuracy (a frame once, marked "relevant", is sent to cloud where we employ larger models at higher resolutions without constraints).

Dataset Acquisition: For research purposes, we do record all the data[1], which is humongous (≈30 Terabytes till now) and the instances are rare, we were able to identify 3.5K such frames (temporally subsampled to 750) through a semi-automatic method. Firstly, we only sampled frames from the camera that is facing the sidewalk (people entering the bus are visible). We then geo-fenced images from bus-stop locations and major intersections on the bus route reducing the set to 780K images. Then, we employ off-the-shelf Detic Swin-B Large Faster R-CNN with CLIP (for custom vocabulary) [18] and find images with "wheelchair", "stroller", "walker", "crutches", "cane", "dog", "animal", "trolley", "cart", "trash can", "garbage bin", "garbage", "garbage bag" categories with a confidence threshold of 0.25. This model has a high false positive rate

[1] Transmission is infeasible, HDD's swapped physically. (Sneakernet)

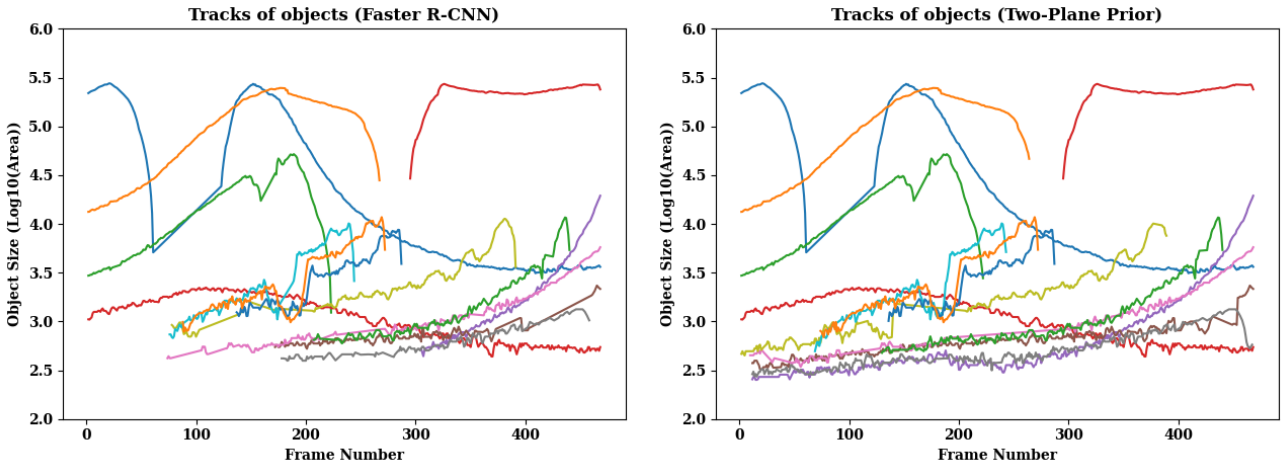

Figure 2. **Tracking Visualization:** To visualize the impact of our two-plane prior, we visualize tracks of length greater than 150 frames tracked by both the methods for a given sequence. We plot object size w.r.t frame numbers (which denotes length). We can observe that some objects are detected earlier and are tracked for a longer time.

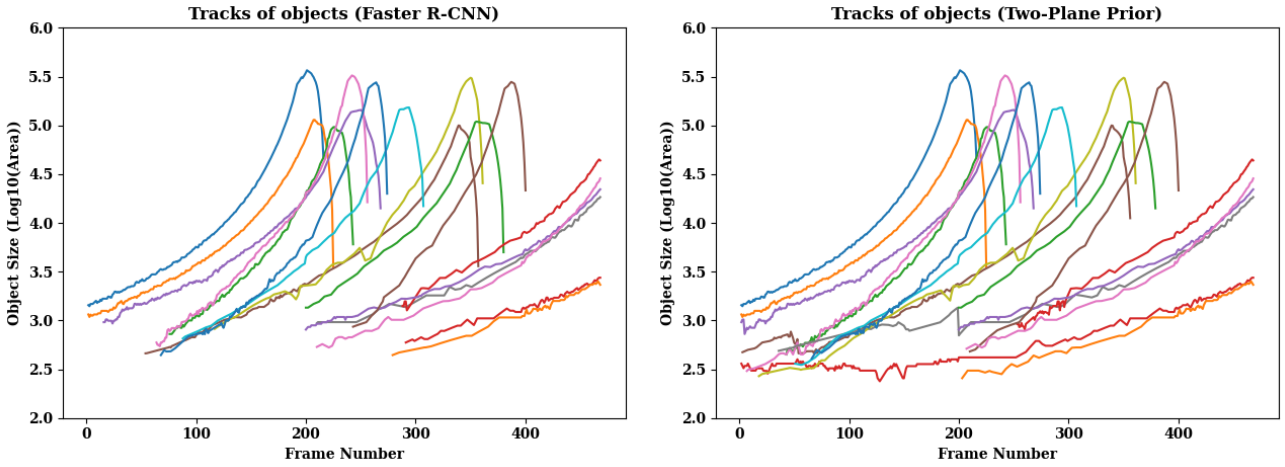

Figure 3. **Tracking Visualization:** To visualize the impact of our two-plane prior, we visualize tracks of length greater than 150 frames tracked by both the methods for a given sequence. We plot object size w.r.t frame numbers (which denotes length). The severe drops of the object sizes for some tracks correspond to nearby object overtaken by our vehicle. We can observe that some objects are detected earlier and are tracked for a longer time.

for these rare classes, and we were able to automatically filter a set of 21K images, and manually filtered these to yield 3.5K images. As many of these images were part of dense temporal sequences, we further sub-sampled temporally within each sequence yielding 750 samples. We manually annotated these images with object bounding boxes and categories (”trash-can”, ”garbage-bag” and ”person-requiring-assistance”; labels from Detic [18] were not accurate). As the data is recorded over the course of a year, we split the train and test test (70% - 30%) using the date stamp (images taken on the same day are in the same split) so that the model doesn't overfit.

Hardware Platform: We set the Jetson AGX to consume 30+ Watts (MAXN configuration; no power budget). Memory is measured using the `tegrastats` utility, while we use Jetpack 4.6.1 and pytorch 1.6, mmdetection 2.7 (+ mmcv 1.15) compiled for Jetson AGX to measure latency consistently across methods (models can be compiled with TensorRT and trained with mixed precision for additional orthogonal improvements).

Results: In this case, just like autonomous driving, we observe that the vanishing point is highly local. Due to overheads of vanishing point estimate on our edge device, we instead employ the average vanishing point, and cache

saliency S, considerably reducing our approach's latency and memory while maximizing accuracy. From Table 5, we observe AR and mAP_{50} for the baseline (Faster R-CNN) and our approach at 0.5x and 0.75x scales. Our method consistently outperforms the baseline method at the same scale, showing both better precision and recall while incurring only **4ms** additional latency and **22 MB** memory overheads.

G. Qualitative Results

We present the variations of our proposed Two-Plane Perspective Prior across different datasets and scenarios in Figure 4. We also show case of the major failure mode of just employing Ground Plane Prior in Fig 5. We also show a qualitative comparison with prior work in Figures 6, 7 and 8. Lastly, we take a closer look at some of the far away objects that were detected in Figures 9 and 10. The accompanying website further illustrates some of the aspects of our method.

References

[1] Hala Abualsaud, Sean Liu, David B Lu, Kenny Situ, Akshay Rangesh, and Mohan M Trivedi. Laneaf: Robust multi-lane detection with affinity fields. *Robotics and Automation Letters*, 2021. 1

[2] Erik Bochinski, Volker Eiselein, and Thomas Sikora. High-speed tracking-by-detection without using image information. In *AVSS*, 2017. 3

[3] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 1

[4] Ting-Wu Chin, Ruizhou Ding, and Diana Marculescu. Adascale: Towards real-time video object detection using adaptive scaling. *Machine Learning and Systems*, 1:431–441, 2019. 1

[5] Anurag Ghosh, Akshay Nambi, Aditya Singh, Harish YVS, and Tanuja Ganu. Adaptive streaming perception using deep reinforcement learning. *arXiv preprint arXiv:2106.05665*, 2021. 1

[6] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 1

[7] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *AAAI*, 2018. 3

[8] Mengtian Li, Yu-Xiong Wang, and Deva Ramanan. Towards streaming perception. In *ECCV*, 2020. 1, 3, 4

[9] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 2

[10] Yancong Lin, Ruben Wiersma, Silvia L Pintea, Klaus Hildebrandt, Elmar Eisemann, and Jan C van Gemert. Deep vanishing point detection: Geometric priors make dataset variations vanish. In *CVPR*, 2022. 1, 2

[11] Shichen Liu, Yichao Zhou, and Yajie Zhao. Vapid: A rapid vanishing point detector via learned optimizers. In *ICCV*, 2021. 2

[12] Yin-Bo Liu, Ming Zeng, and Qing-Hao Meng. D-vpnet: A network for real-time dominant vanishing point detection in natural scenes. *Neurocomputing*, 2020. 1, 2

[13] 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. *NeurIPS*, 2019. 1

[14] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 2015. 2

[15] Edgar Riba, Dmytro Mishkin, Daniel Ponsa, Ethan Rublee, and Gary Bradski. Kornia: an open source differentiable computer vision library for pytorch. In *WACV*, 2020. 1

[16] Chittesh Thavamani, Mengtian Li, Nicolas Cebron, and Deva Ramanan. Fovea: Foveated image magnification for autonomous navigation. In *ICCV*, 2021. 1, 2, 3, 4

[17] Jinrong Yang, Songtao Liu, Zeming Li, Xiaoping Li, and Jian Sun. Real-time object detection for streaming perception. In *CVPR*, 2022. 2, 3

[18] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra. Detecting twenty-thousand classes using image-level supervision. In *ECCV*, 2022. 4, 5

[19] Yichao Zhou, Haozhi Qi, Jingwei Huang, and Yi Ma. Neurvps: Neural vanishing point scanning via conic convolution. *NeurIPS*, 2019. 1, 2

[20] Zihan Zhou, Farshid Farhat, and James Z Wang. Detecting dominant vanishing points in natural scenes with application to composition-sensitive image retrieval. *Transactions on Multimedia*, 2017. 1

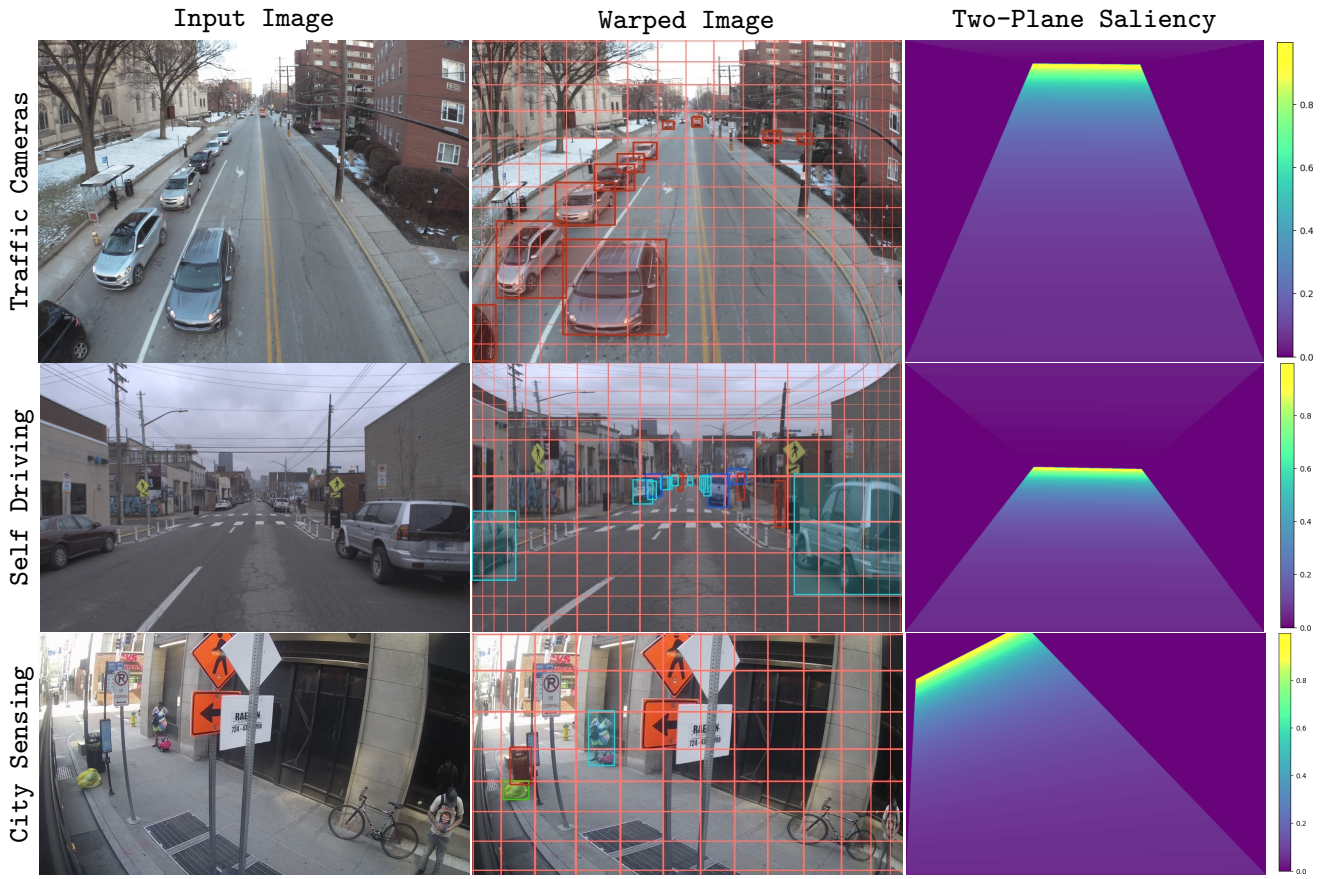

Figure 4. **Two-Plane Prior Based Warping:** Two-Plane Prior is defined by a few parameters that describe two planar regions in the direction of the vanishing point in the 3D scene (See Section 3.1 and 3.2 in manuscript). Firstly, we can observe the Two-Plane Prior's explicit dependence on the vanishing point v in the saliency maps. Next, as we can observe from grid lines (equidistant in the original image) overlaid on top of the warped images, the extent of spatial warping varies across datasets (WALT, Argoverse-HD and Commuter Bus), showing us the need for learnable parameter ν over prior work which do not directly model this relationship. Lastly, notice the second plane's effect in sampling. The second plane acts as a "counter-balance" to reduce distortion, and the plane is **faintly observable** (contrast adjusted for better visibility).

Figure 5. **Ground Plane Prior vs Two-Plane Prior:** This figure demonstrates how crucial it is to model the second plane. Learning is difficult with Ground Plane Prior (Section 5.1 in the manuscript) and causes heavy distortion of non-ground-plane regions.
Scene 1: Detector with Ground Plane Prior misses nearby tall objects because of heavy distortion. Turquoise colored bus on the right (blue box) is detected when Two-Plane prior is used and missed with Ground Plane prior.
Scene 2: Objects not on the ground plane are missed as they are squished by the Ground Plane Prior. Yellow boxes denote the traffic lights. All 6 traffic lights in the scene were detected when Two-Plane prior is used while Ground Plane prior missed 4 traffic lights.

Figure 6. **Qualitative Comparison with Fovea Warps on Argoverse-HD:** We observe that the reliance on the vanishing point v allows the warp to sample in the direction of the road even while making turns. Far ahead on the road, a $truck$ (dark-blue) is not detected by Fovea (S_D or S_I), but correctly detected by our approach.

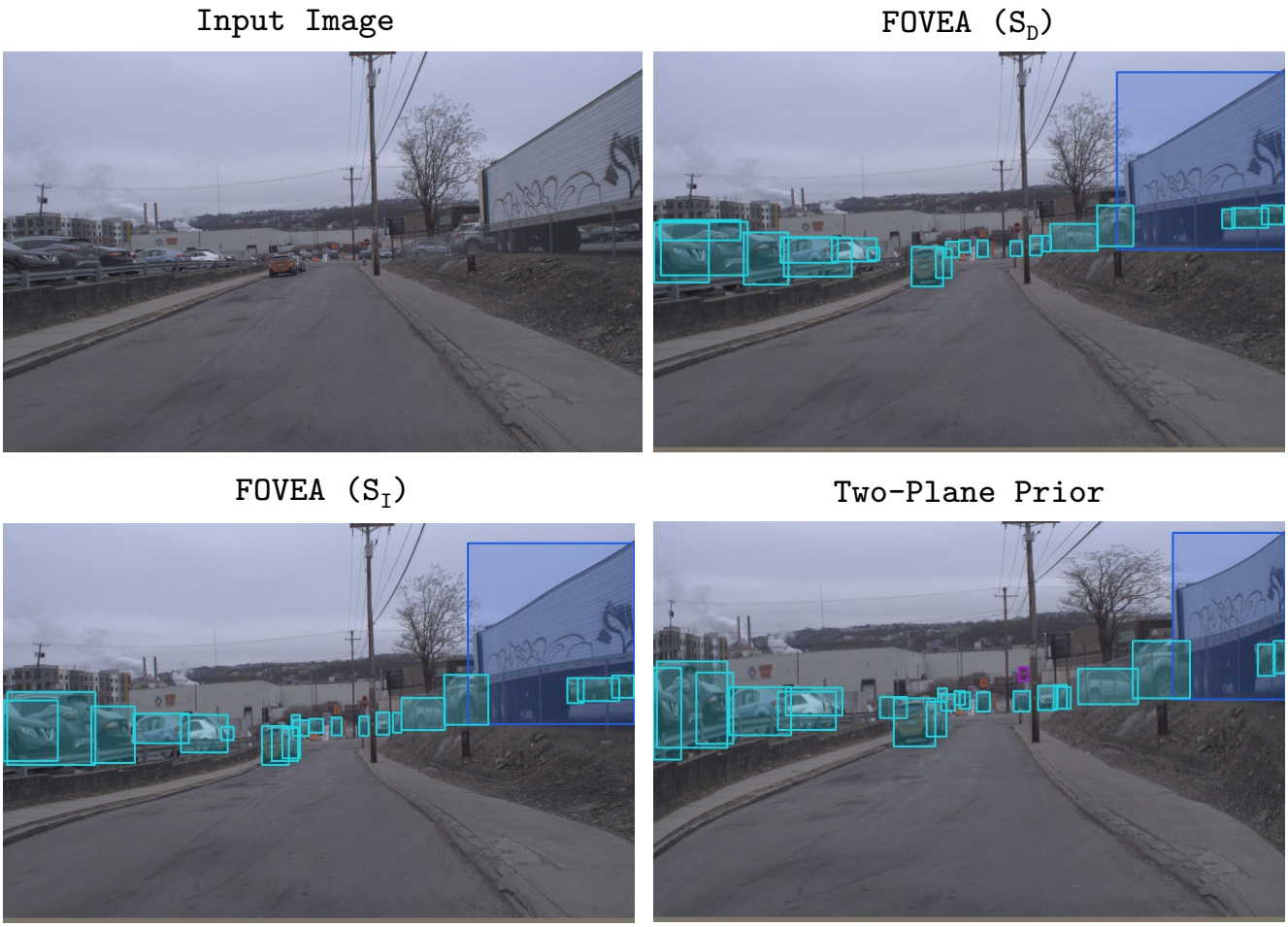

Figure 7. **Qualitative Comparison with Fovea Warps on Argoverse-HD:** We observe that scale factor ν models the extent of sampling better. Fovea (S_D or S_I) misses the $stop - sign$ (a magenta box in the middle of the image) which our method is able to detect (as it's larger in the warped image). Fovea (S_I) notes that in their method, regions immediately adjacent to magnified regions are often contracted which is noticed in this case.

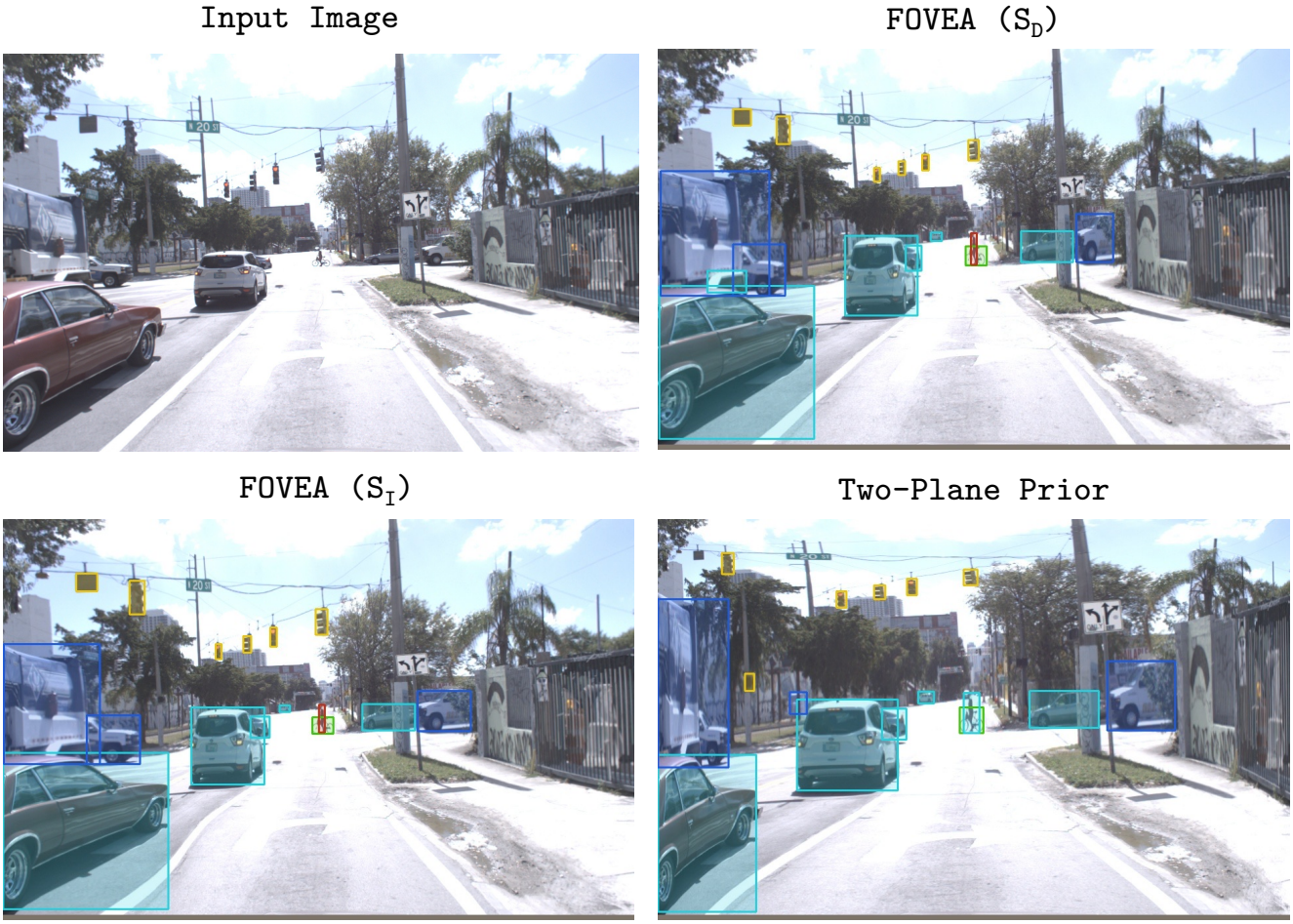

Figure 8. **Qualitative Comparison with Fovea Warps on Argoverse-HD:** *Failure Case:* The model has misclassified a pedestrian as car in the image warped by the Two-Plane Prior while correctly classified by Fovea (S_D or S_I) (red), likely due to the presence of bicycle and heavier distortion.

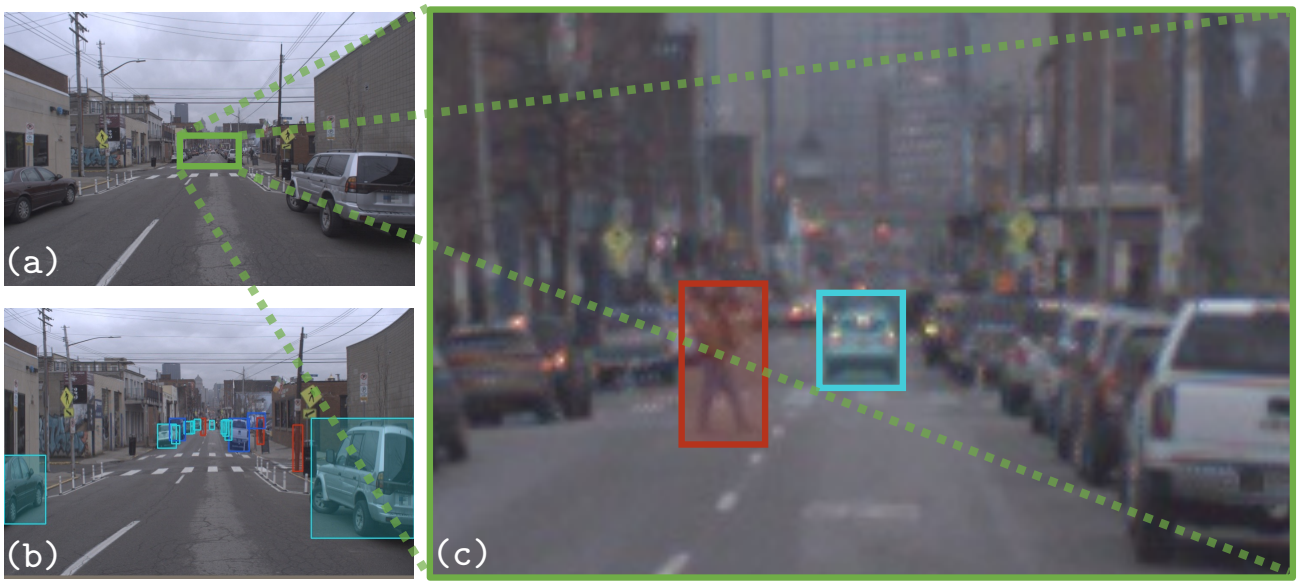

Figure 9. **Detection of Far Away Objects:** Our Two-Plane Prior boosts the detection of small far-away objects at lower resolutions (depicted image from Argoverse-HD dataset). The cropped green region in the (a) original image is (c) zoomed in while (b) shows all the detections in the warped image. Our method detects far-away pedestrian and car.

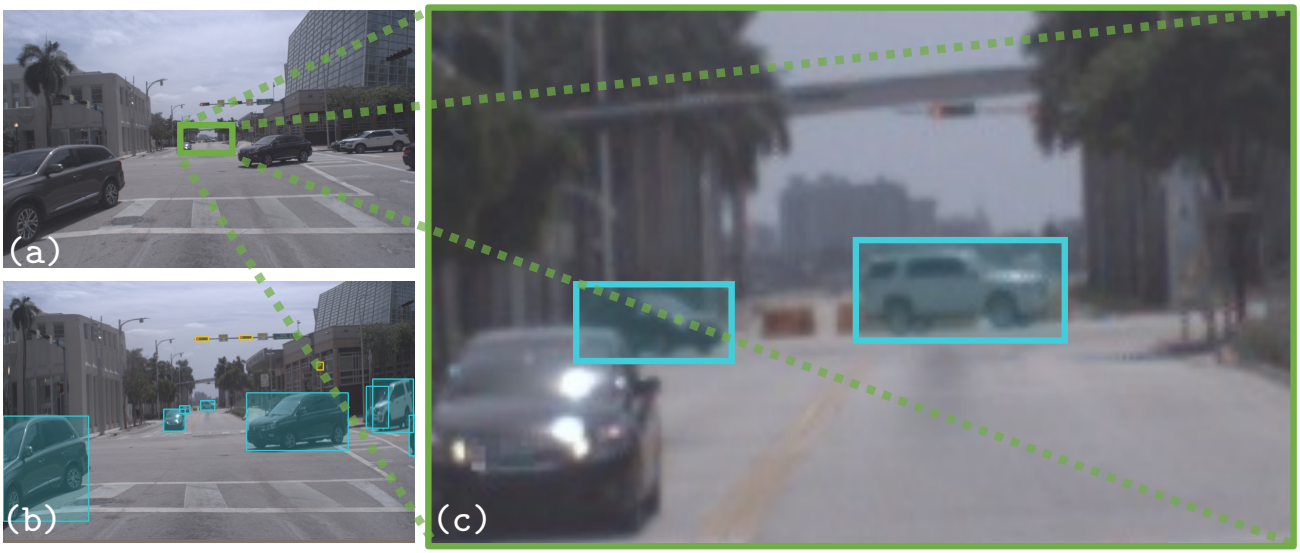

Figure 10. **Detection of Far Away Objects:** Our Two-Plane Prior boosts the detection of small far-away objects at lower resolutions (depicted image from Argoverse-HD dataset). The cropped green region in the (a) original image is (c) zoomed in while (b) shows all the detections in the warped image. Our method is able to detect the occluded car.