Actually Making BEVs

D.A. Forsyth

BEVs from images (with all tech)

- Idea:
  - Predict and complete labels; project; cleanup

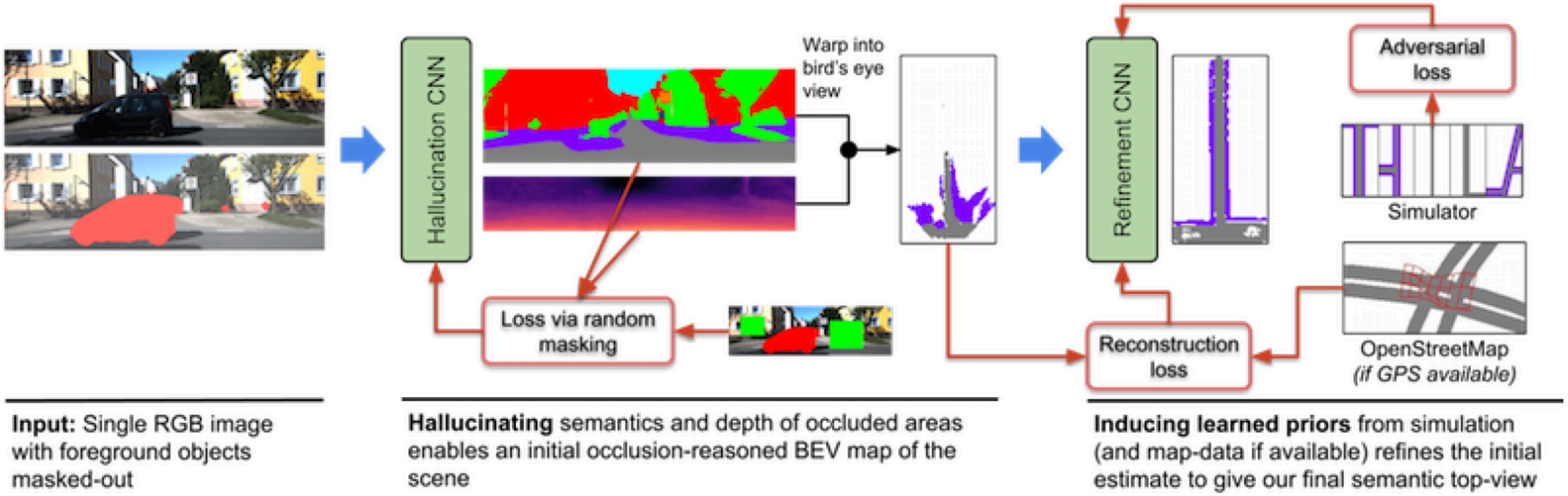
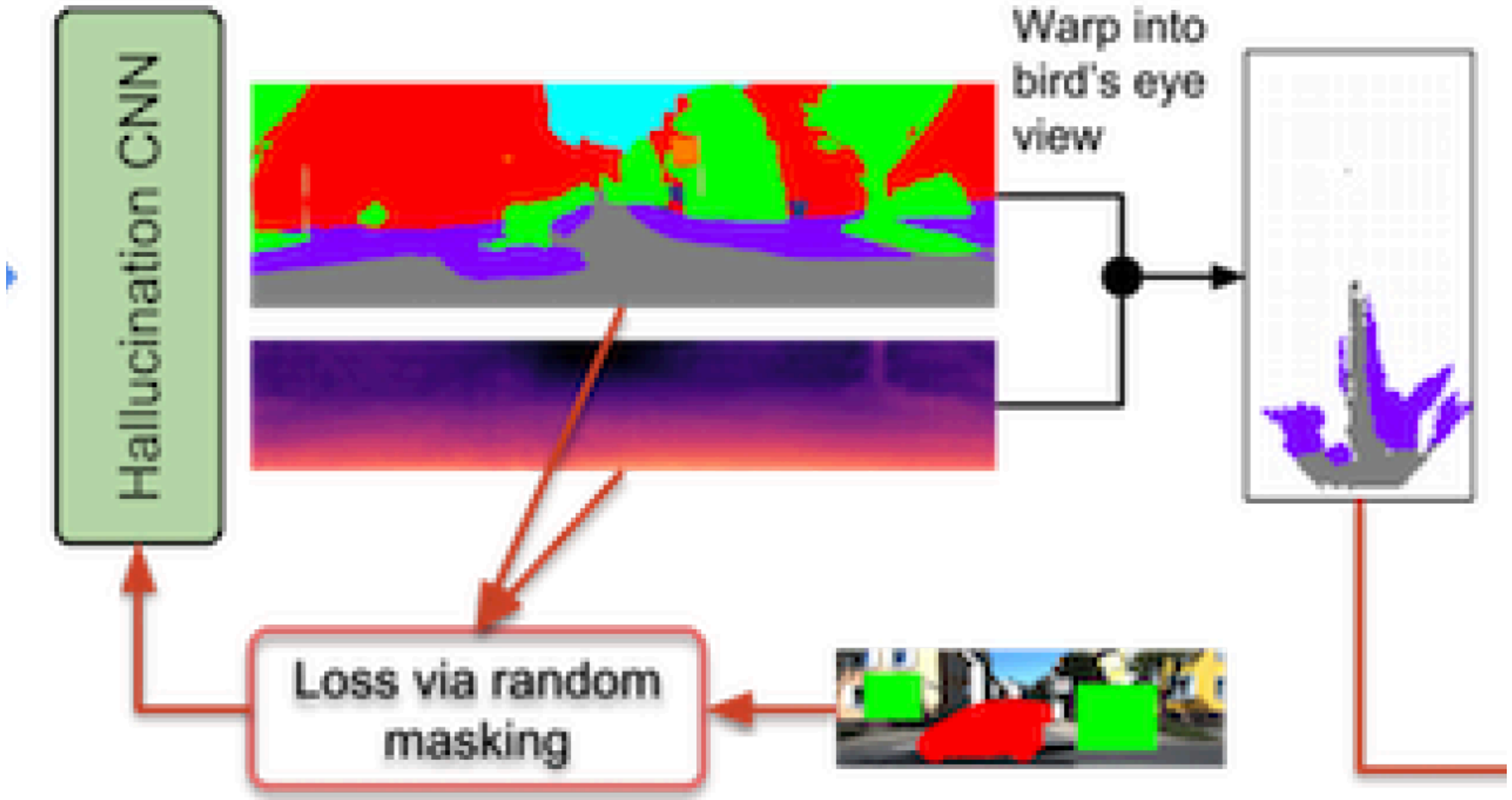

Hallucination CNN
Warp into bird's eye view
Loss via random masking
Refinement CNN
Adversarial loss
Simulator
Reconstruction loss
OpenStreetMap (if GPS available)
Input: Single RGB image with foreground objects masked-out
Hallucinating semantics and depth of occluded areas enables an initial occlusion-reasoned BEV map of the scene
Inducing learned priors from simulation (and map-data if available) refines the initial estimate to give our final semantic top-view

Fig. 1: Given a single RGB image of a typical street scene (left), our approach creates an **occlusion-reasoned semantic map of the scene layout in the bird's eye view**. We present a CNN that can hallucinate depth and semantics in areas occluded by foreground objects (marked in red and obtained via standard semantic segmentation), which gives an initial but noisy and incomplete estimate of the scene layout (middle). To fill in unobserved areas in the top-view, we further propose a refinement-CNN that induces learning strong priors from simulated and OpenStreetMap data (right), which comes at no additional annotation costs.

Schulter et al, 2018



Hallucinating semantics and depth of occluded areas enables an initial occlusion-reasoned BEV map of the scene

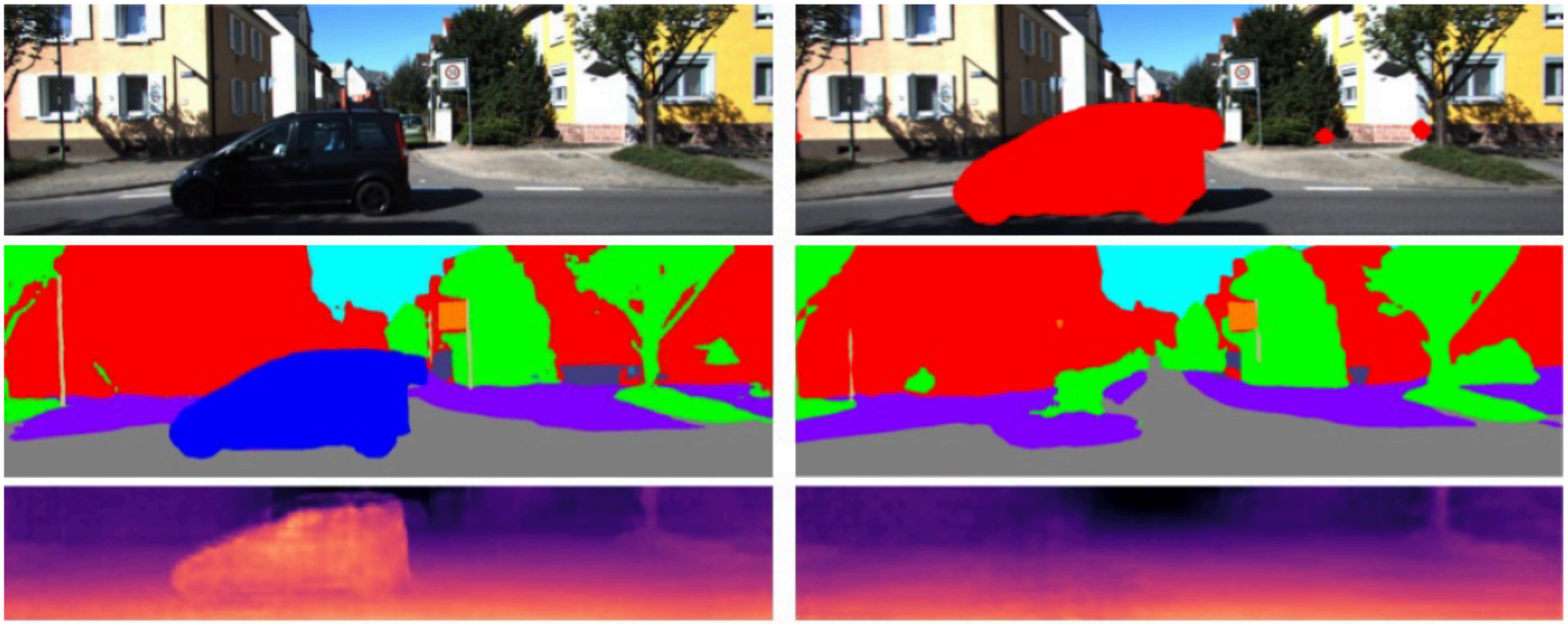
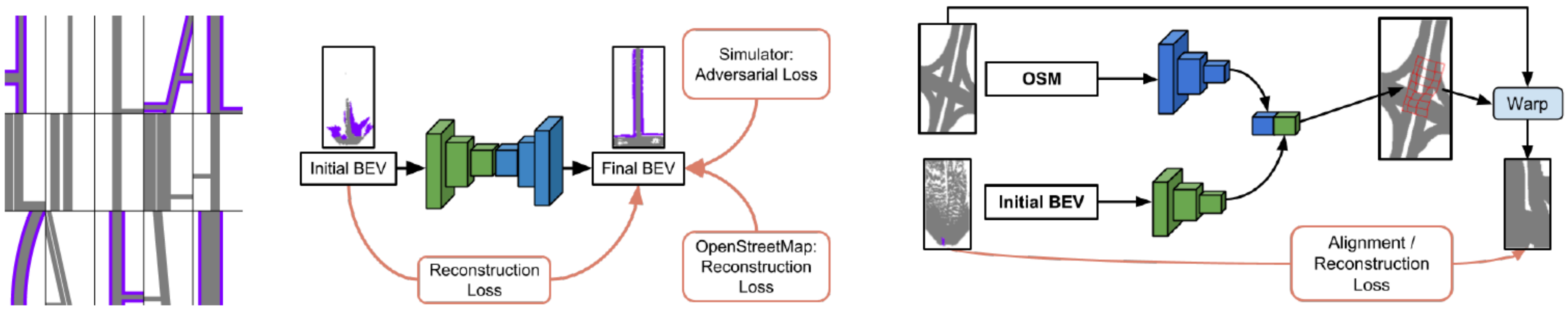


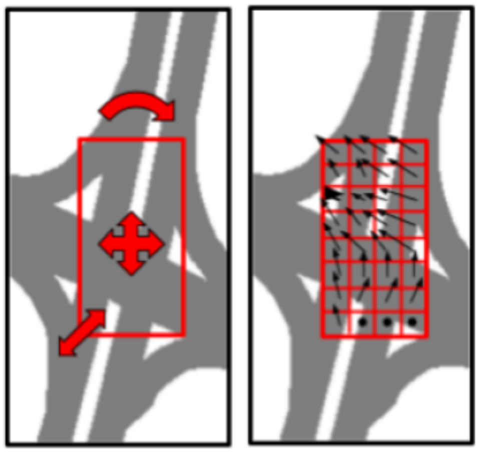
Fig. 6: Qualitative example of our hallucination CNN: Semantics and depth without (left) and with (right) hallucination.


Initial BEV
Final BEV
Simulator: Adversarial Loss
Reconstruction Loss
OpenStreetMap: Reconstruction Loss
OSM
Initial BEV
Warp
Alignment / Reconstruction Loss

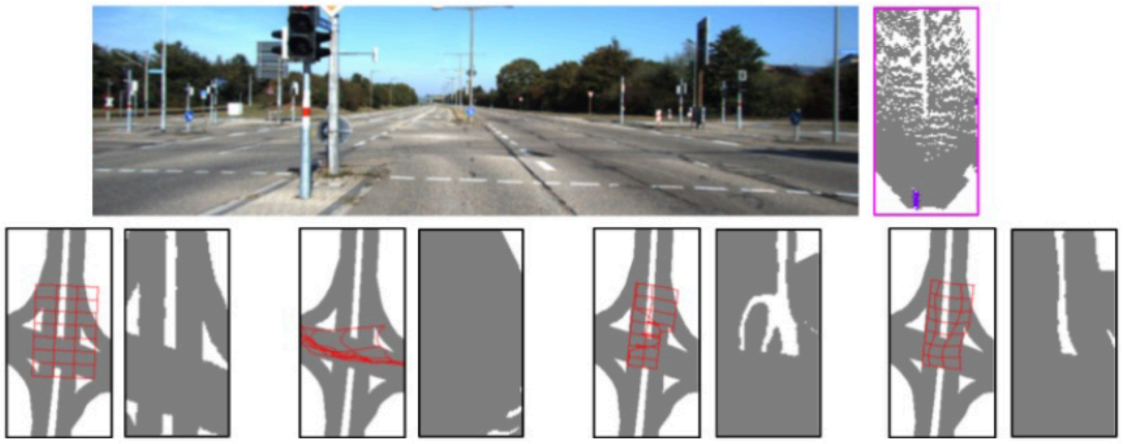
(a) (b) (c)

Fig. 4: **(a) Simulated road shapes** in the top-view. **(b) The refinement-CNN** is an encoder-decoder network receiving three supervisory signals: self-reconstruction with the input, adversarial loss from simulated data, and reconstruction loss with aligned OpenStreetMap (OSM) data. **(c) The alignment CNN** takes as input the initial BEV map and a crop of OSM data (via noisy GPS and yaw estimate given). The CNN predicts a warp for the OSM map and is trained to minimize the reconstruction loss with the initial BEV map.

Schulter et al, 2018



(a)



(b)

Fig. 5: **(a)** We use a composition of similarity transform (left, "box") and a non-parametric warp (right, "flow") to align noisy OSM with image evidence. **(b, top)** Input image and the corresponding B^{init}. **(b, bottom)** Resulting warping grid overlaid on the OSM map and the warping result for 4 different warping functions, respectively: "box", "flow", "box+flow", "box+flow (with regularization)". Note the importance of composing the transformations and the induced regularization.

Schulter et al, 2018

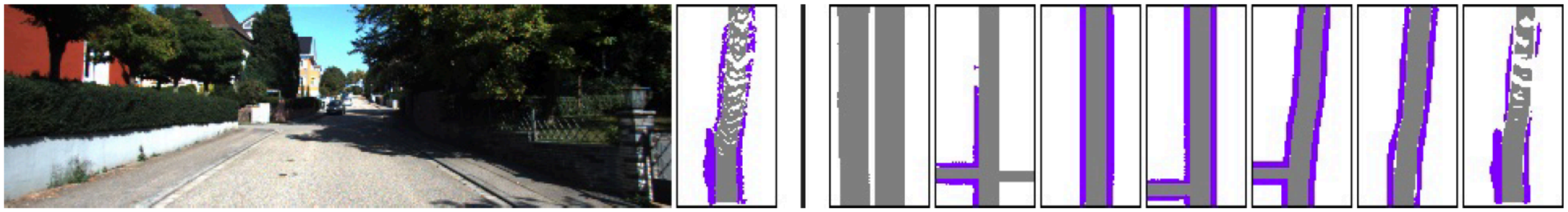


Fig. 7: One example of the influence of λ (trade-off between adversarial and reconstruction loss, i.e., OSM and simulation data). (left) input image and B^{init}; (right) 6 final BEV maps for $\lambda = \{0, 1, 5, 100, 500, 1000, 10^6\}$. One can see that higher λ leads to higher alignment with B^{init}.

Schulter et al, 2018

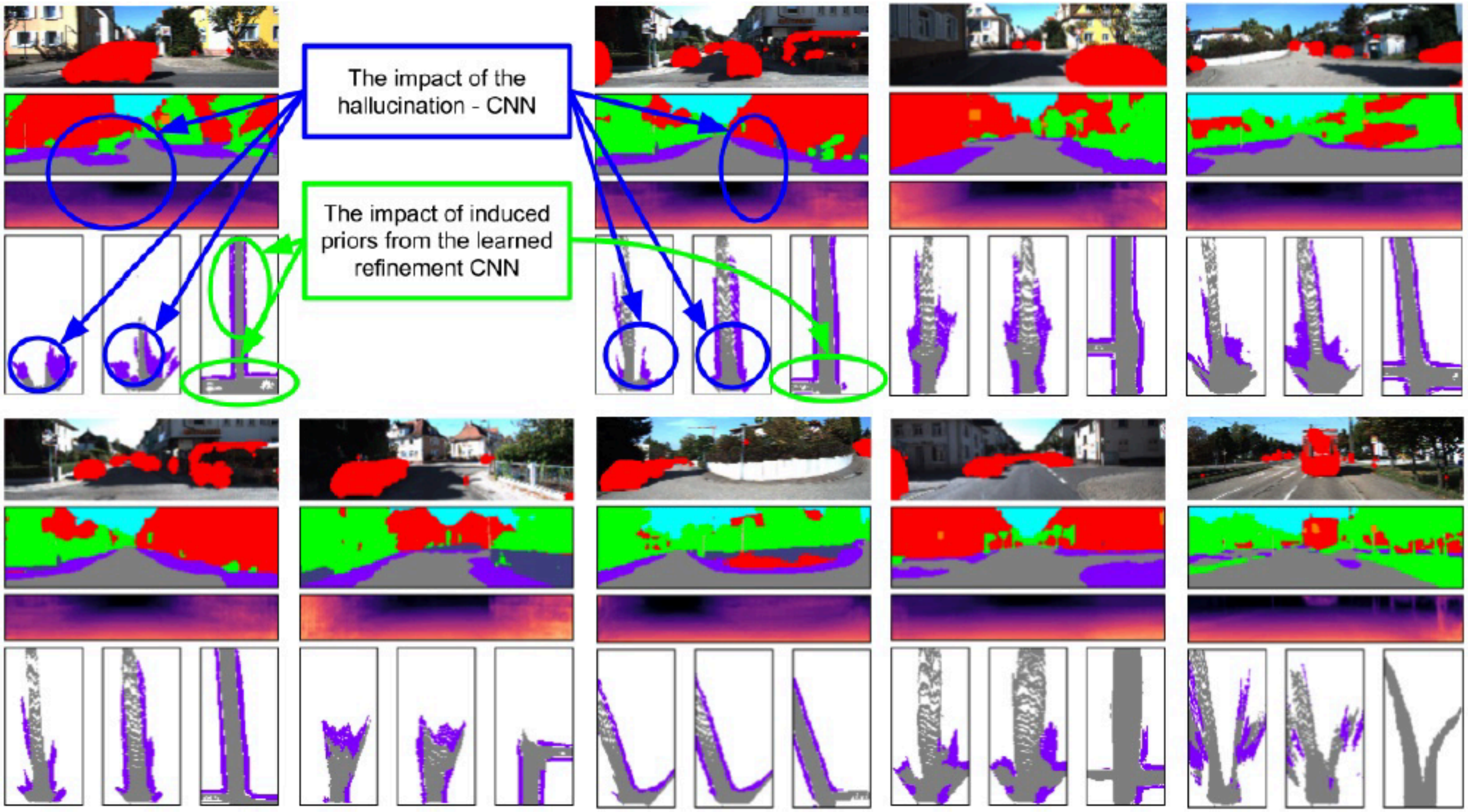

The impact of the hallucination - CNN
The impact of induced priors from the learned refinement CNN

Fig. 8: **Examples of our BEV representation**. Each one shows the masked RGB input, the hallucinated semantics and depth, as well as three BEV maps, which are (from left to right), The BEV map without hallucination, with hallucination, and after refinement. The last example depicts a failure case.

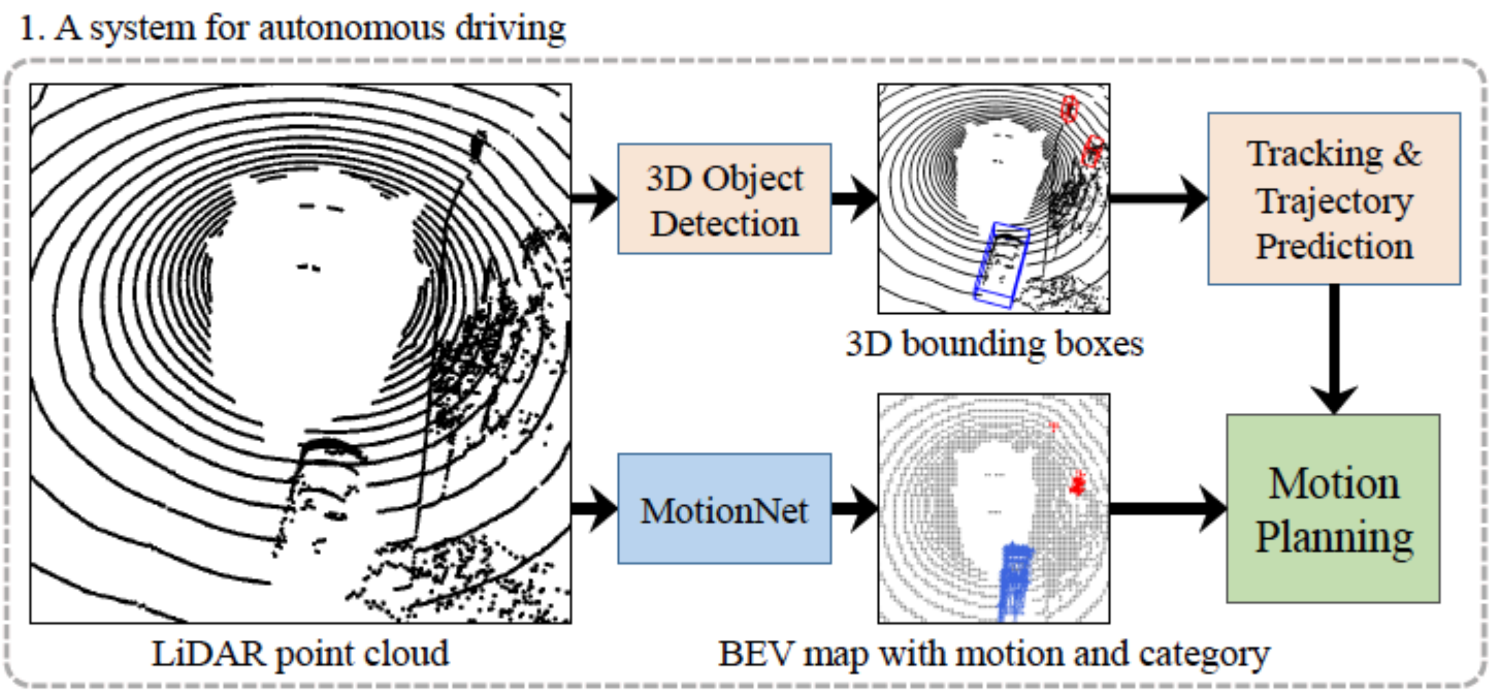
Schulter et al, 2018

Issues

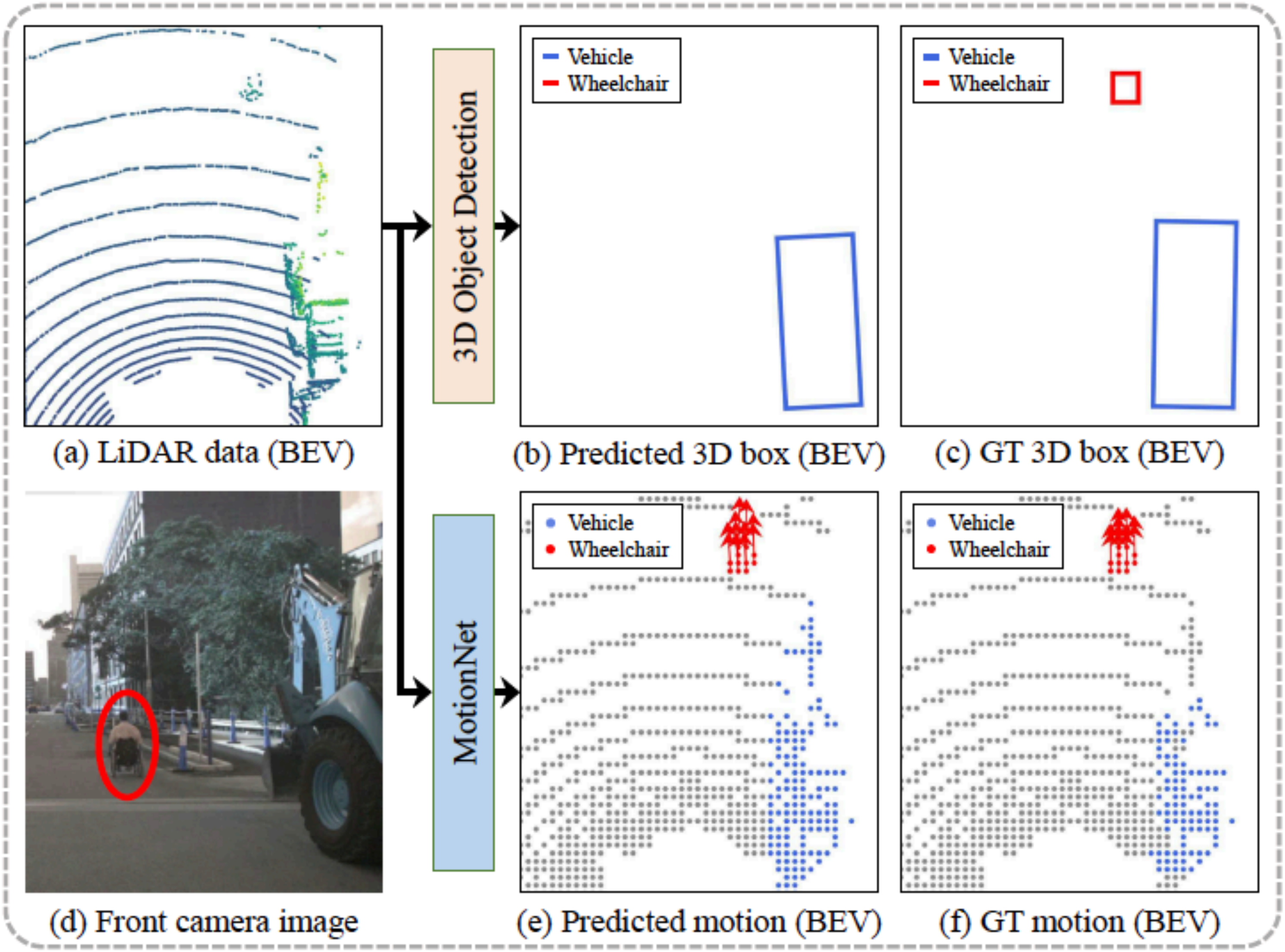
- Something of a jumble of parts
  - each trained separately
- Results aren’t super
  - why doesn’t registration help more?
- Shouldn’t temporal consistency help?
  - why only per frame?
- Depth prediction then “dropping” seems like a bad idea
  - the ground depth map is pretty much a plane, so why not camera model?
  - why not more sophisticated ground plane model?
    - plane+relief?
- BIG Q: could you use this to plan?

Building BEVs from LIDAR alone

- Build feature map by:
  - cut space into voxels
  - each voxel gets 1 (return) or 0 (no return)
  - now interpret voxel grid as a feature map over the ground
- Classify/predict motion using these features


1. A system for autonomous driving
3D Object Detection
3D bounding boxes
Tracking & Trajectory Prediction
MotionNet
Motion Planning
LiDAR point cloud
BEV map with motion and category

2. Example: disabled person in a wheelchair



Wu, 20

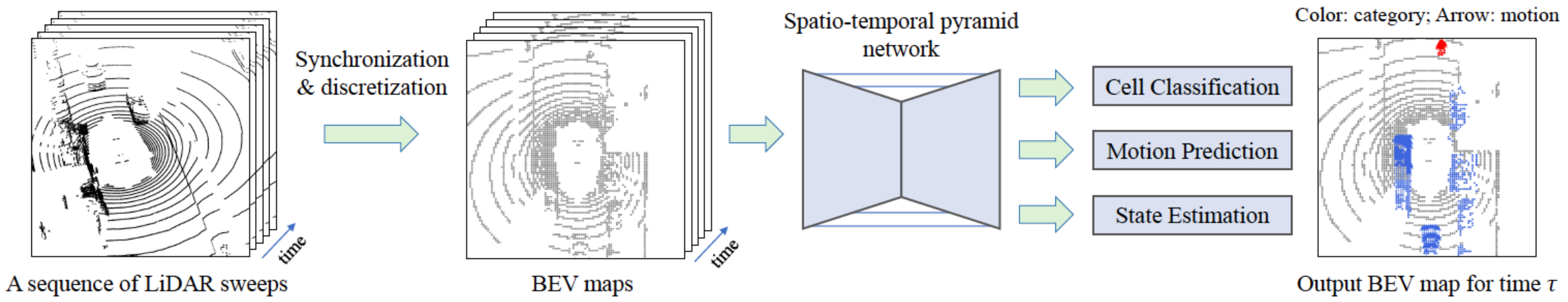

Synchronization & discretization
Spatio-temporal pyramid network
Color: category; Arrow: motion
Cell Classification
Motion Prediction
State Estimation
time
time
A sequence of LiDAR sweeps
BEV maps
Output BEV map for time τ

Figure 2. Overview of **MotionNet**. Given a sequence of LiDAR sweeps, we first represent the raw point clouds into BEV maps, which are essentially 2D images with multiple channels. Each pixel (cell) in a BEV map is associated with a feature vector along the height dimension. We then feed the BEV maps into the spatio-temporal pyramid network (STPN) for feature extraction. The output of STPN is finally delivered to three heads: (1) cell classification, which perceives the category of each cell, such as vehicle, pedestrian or background; (2) motion prediction, which predicts the future trajectory of each cell; (3) state estimation, which estimates the current motion status of each cell, such as static or moving. The final output is a BEV map, which includes both perception and motion prediction information.

Wu, 20

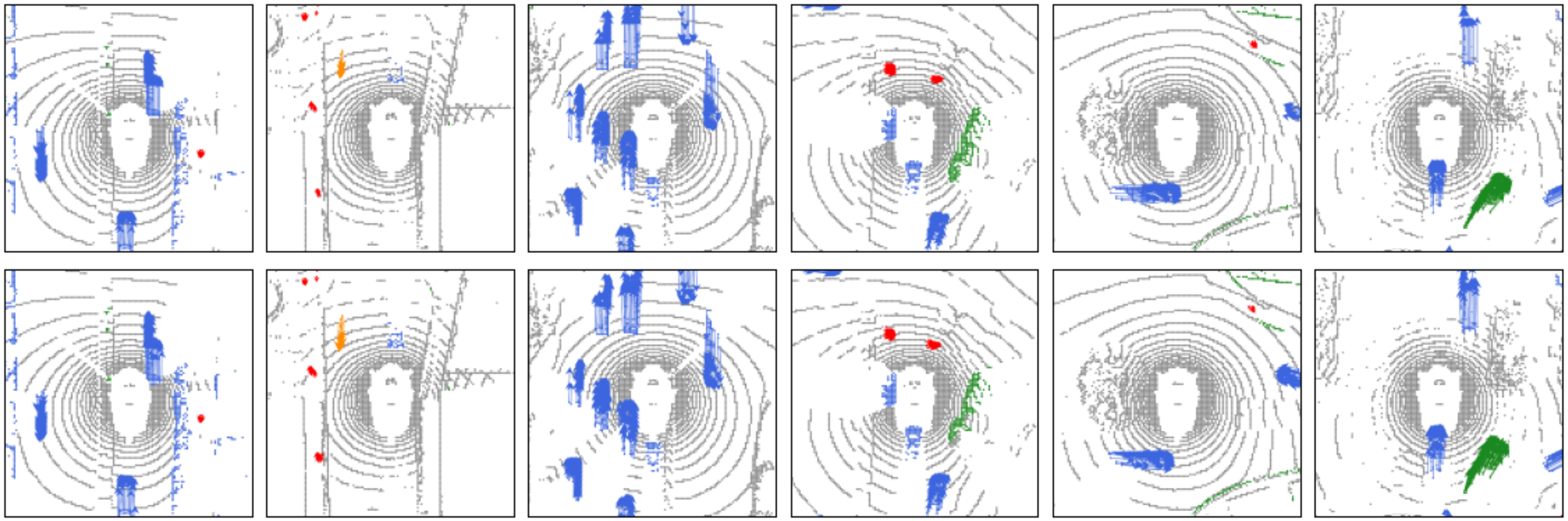


Figure 5. Qualitative results show that MotionNet produces both high-quality classification and motion prediction. Top row: ground-truth. Bottom: MotionNet predictions. Gray: background; blue: vehicle; red: pedestrian; orange: bicycle; green: others. (Zoom in for best view.)

Wu, 20

Issues

- Neat to do BEV with LIDAR
- Not shocking that this feature construction beats det’n
  - above the ground and moving - wheelchair eg
- Why not use camera info as well?
  - we’ll see riffs on this
- BIG Q: could you use this to plan?
-

Lagniappe: Neat idea

- Neat idea I heard from Killian Weinberger
  - repeated drives around an area reveal partial labels

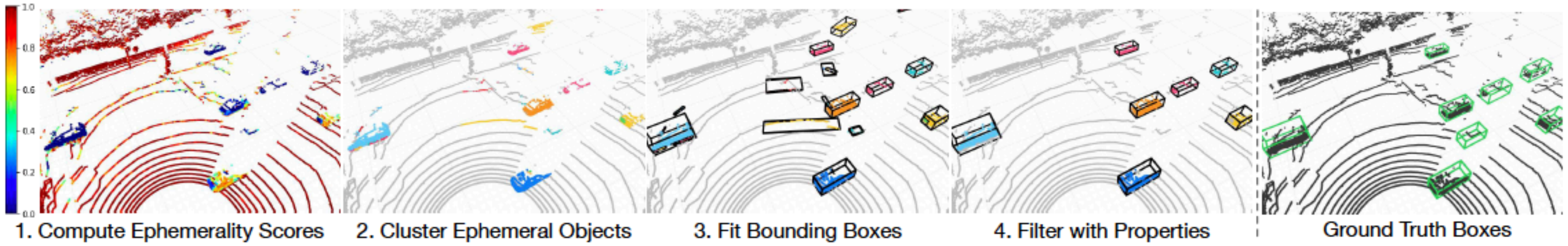

1. Compute Ephemerality Scores
2. Cluster Ephemeral Objects
3. Fit Bounding Boxes
4. Filter with Properties
Ground Truth Boxes

Figure 2. **Generation of seed labels.** Seed labels from object discovery are used to train downstream detectors. We begin by computing the PP score for each point. Then we segment out clusters that are non-persistent and apply a box-fitting algorithm to each cluster. We filter out superfluous bounding boxes using our common-sense assumptions. This entire process is supervision-free.

You et al, 22

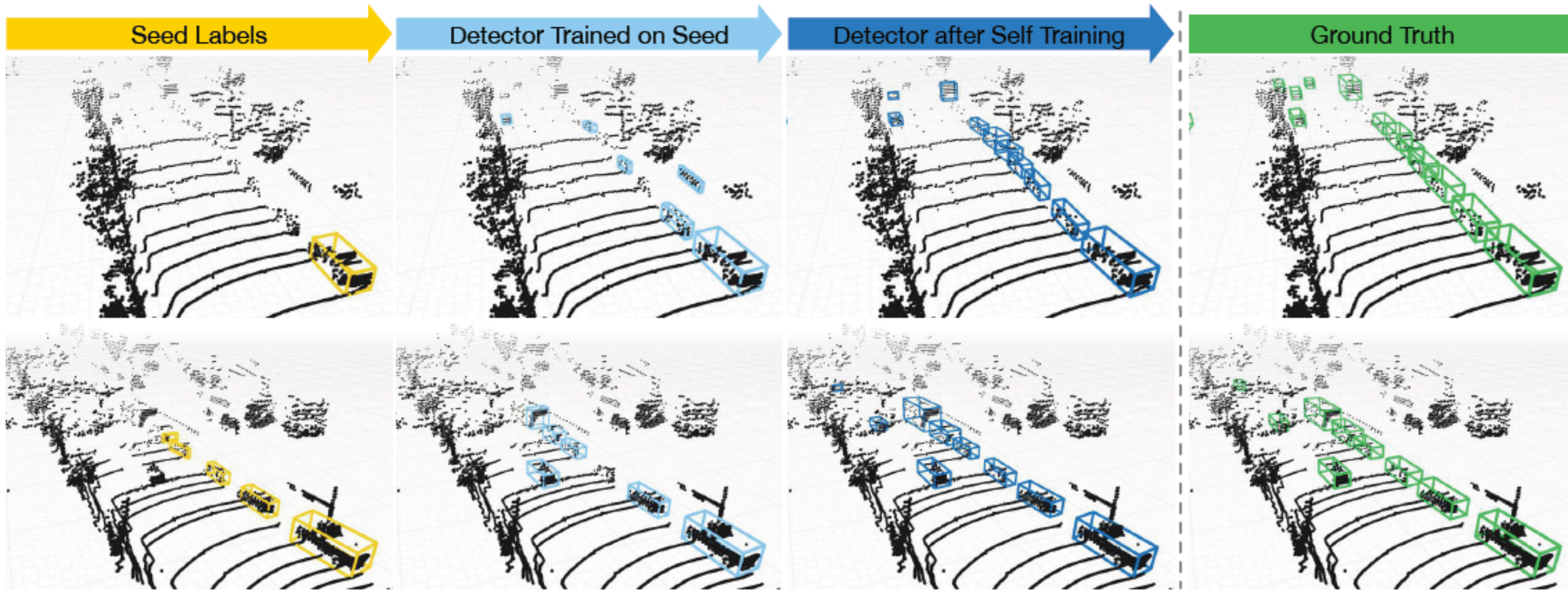


Figure 1. **Visualizations of MODEST outputs.** We show LiDAR scans from two scenes in the Lyft dataset in two rows. From *zero labels* our method is able to bootstrap a detector that achieves results close to the ground truth. The key insight is to utilize noisy "seed" label produced from an ephemerality score and filtered with common-sense properties, and self-train upon them to obtain high quality results.

Monolayout

- Building a BEV as an amodal scene completion problem
  - amodal
    - we impute location/labels of stuff we can’t see
      - by exploiting properties of shape, priors
    - eg, we know how big cars are, and can guess road is behind
- Idea
  - Predict on ground plane directly
    - rather than predict, then map

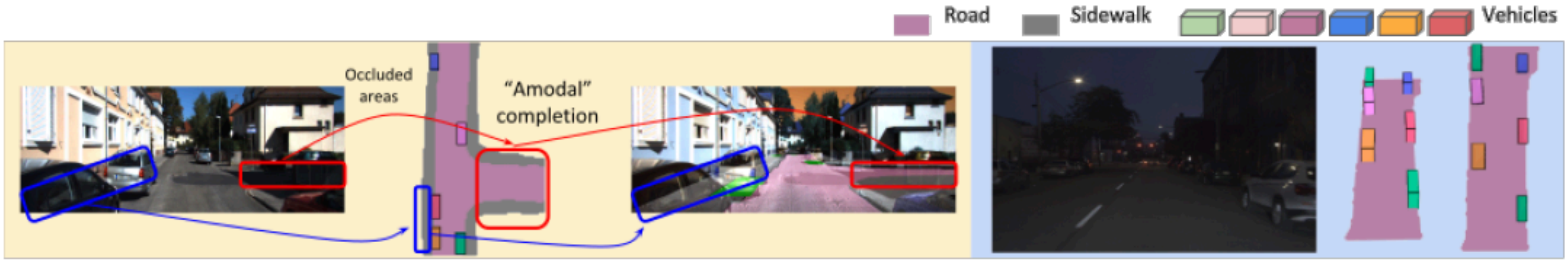


Figure 1: ***MonoLayout***: Given only a single image of a road scene, we propose a neural network architecture that reasons about the *amodal* scene layout in bird's eye view in *real-time* (30 fps). Our approach, dubbed *MonoLayout* can *hallucinate* regions of the static scene (road, sidewalks)—and traffic participants—that do not even project to the visible regime of the image plane. Shown above are example images from the KITTI [10] (left) and Argoverse [5] (right) datasets. *MonoLayout* outperforms prior art (by more than a 20% margin) on hallucinating occluded regions.

Mani et al 20

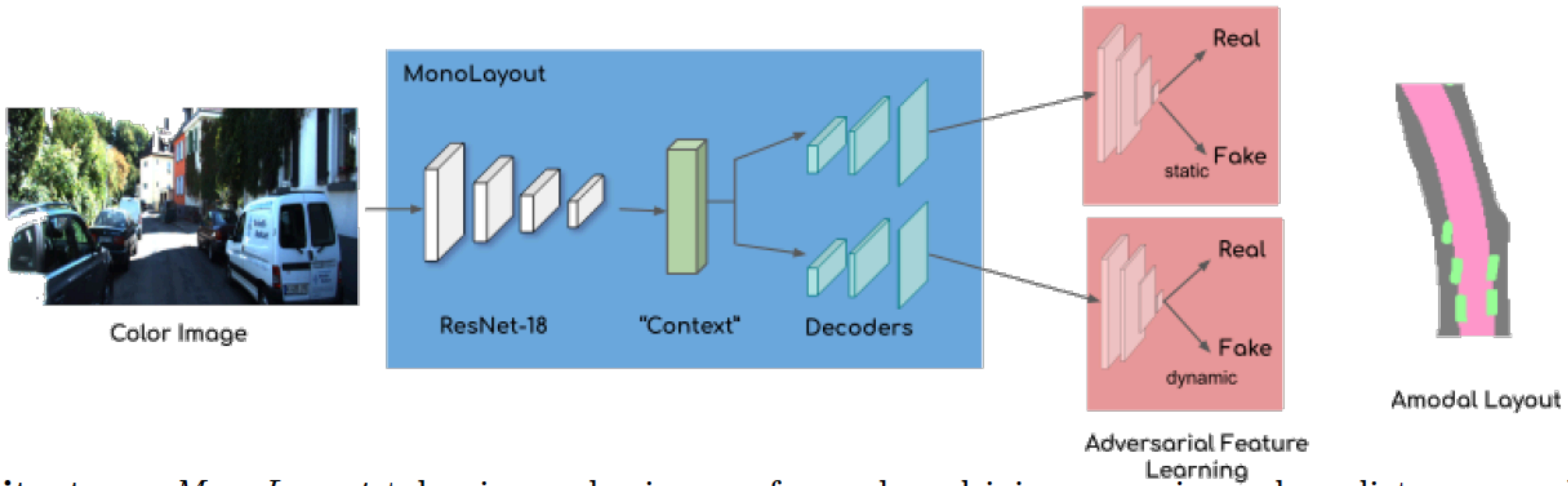


Figure 2: **Architecture**: *MonoLayout* takes in a color image of an urban driving scenario, and predicts an amodal scene layout in bird's eye view. The architecture comprises a *context encoder*, *amodal layout decoders*, and *two discriminators*.

Mani et al 20

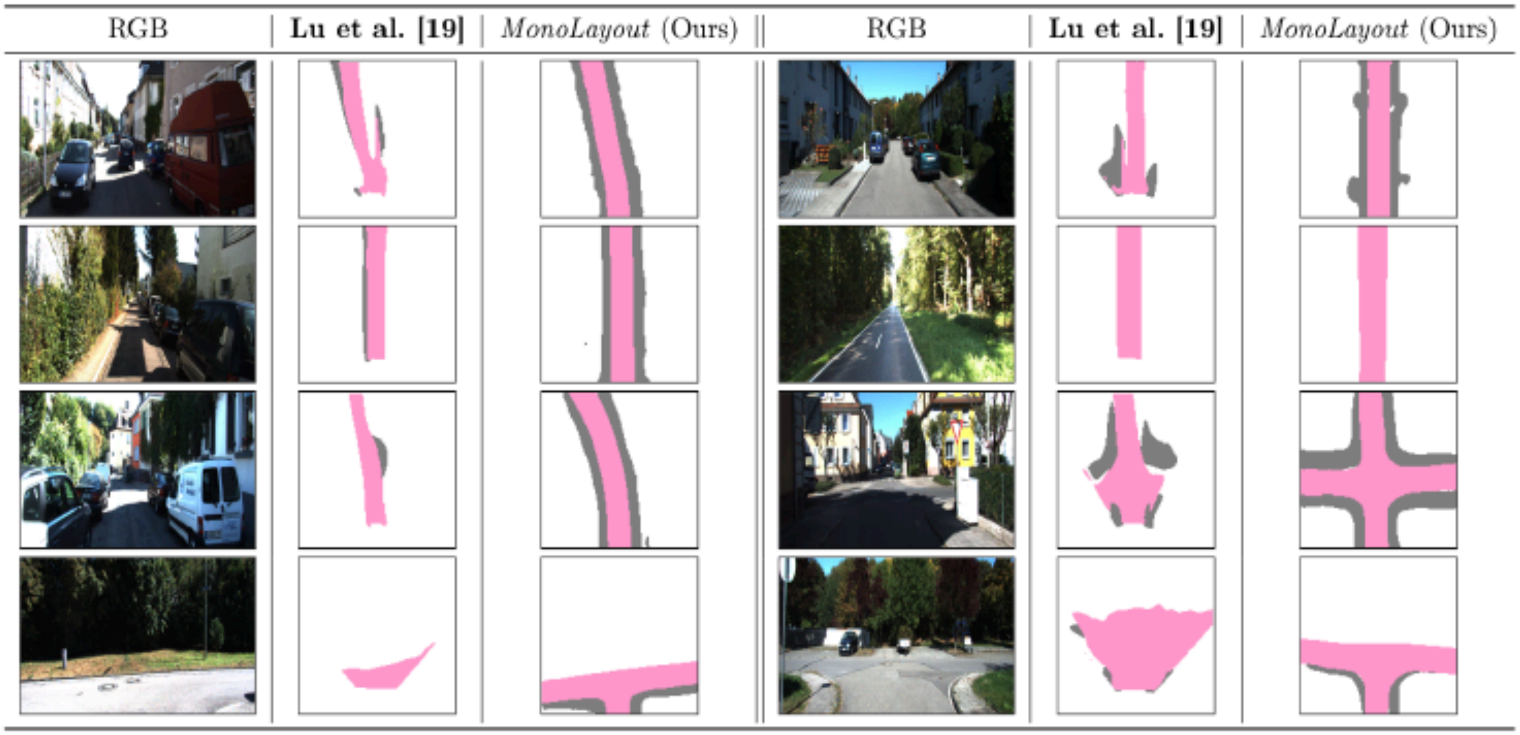


Figure 3: **Static layout estimation**: Observe how *MonoLayout* performs amodal completion of the static scene (road shown in pink, sidewalk shown in gray. MonoOccupancy [24] fails to reason beyond occluding objects (top row), and does not hallucinate large missing patches (bottom row), while *MonoLayout*(Ours) is accurately able to do so. Furthermore, even in cases where there is no occlusion (row 2), *MonoLayout*(Ours) generates road layouts of much sharper quality. Row 3 show extremely challenging scenarios where most of the view is blocked by vehicles, and the scenes exhibit high-dynamic range (HDR) and shadows.

Mani et al 20

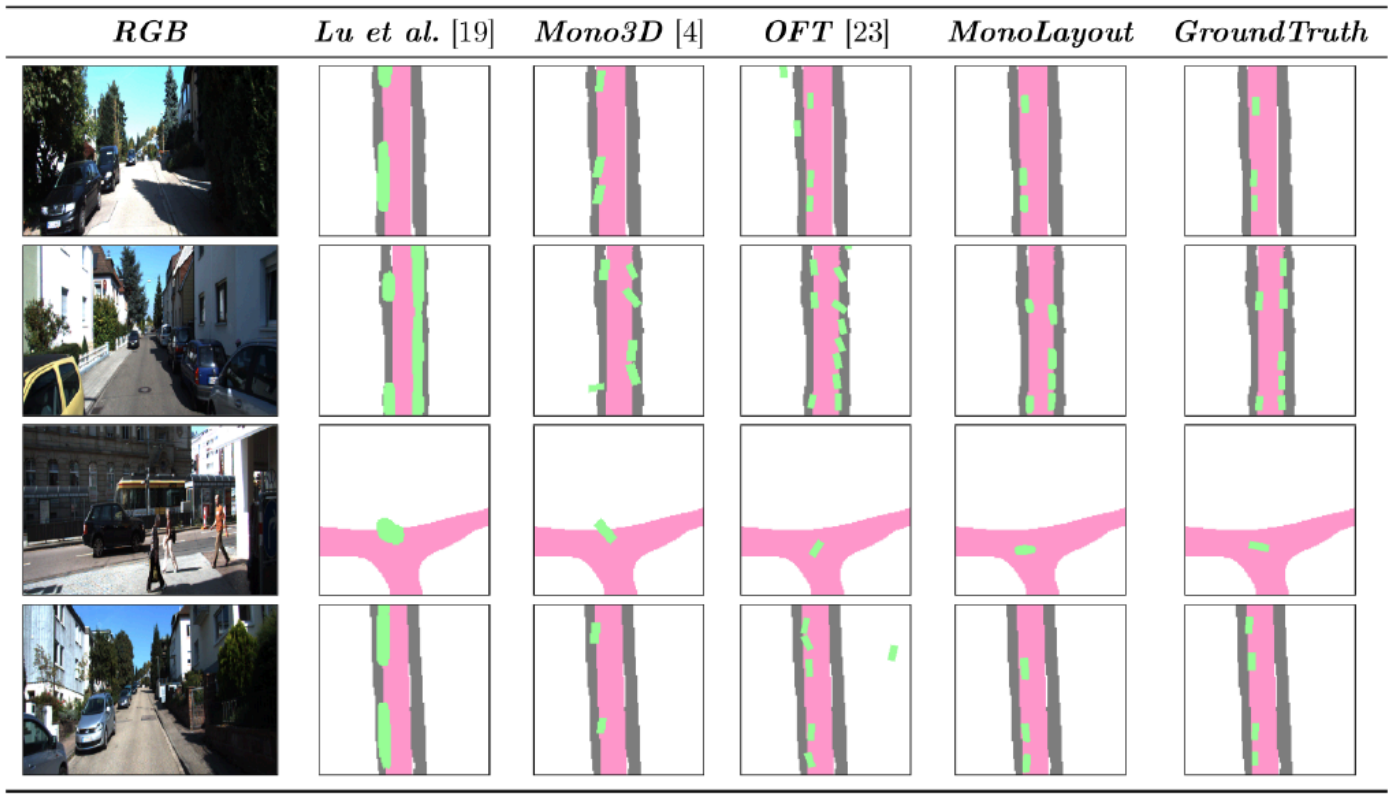


Figure 4: **Dynamic layout estimation**: We show vehicle occupancy estimation results on the KITTI [10] 3D Object detection benchmark. From left to right, the column corresponds to the input image, MonoOccupancy [24], Mono3D [6], OFT [30], *MonoLayout* (Ours), and ground-truth respectively. While the other approaches miss out on detecting cars (top row), or split a vehicle detection into two (second row), or stray detections off road (third row), *MonoLayout* (Ours) produces crisp object boundaries while respecting vehicle and road geometries.

Mani et al 20

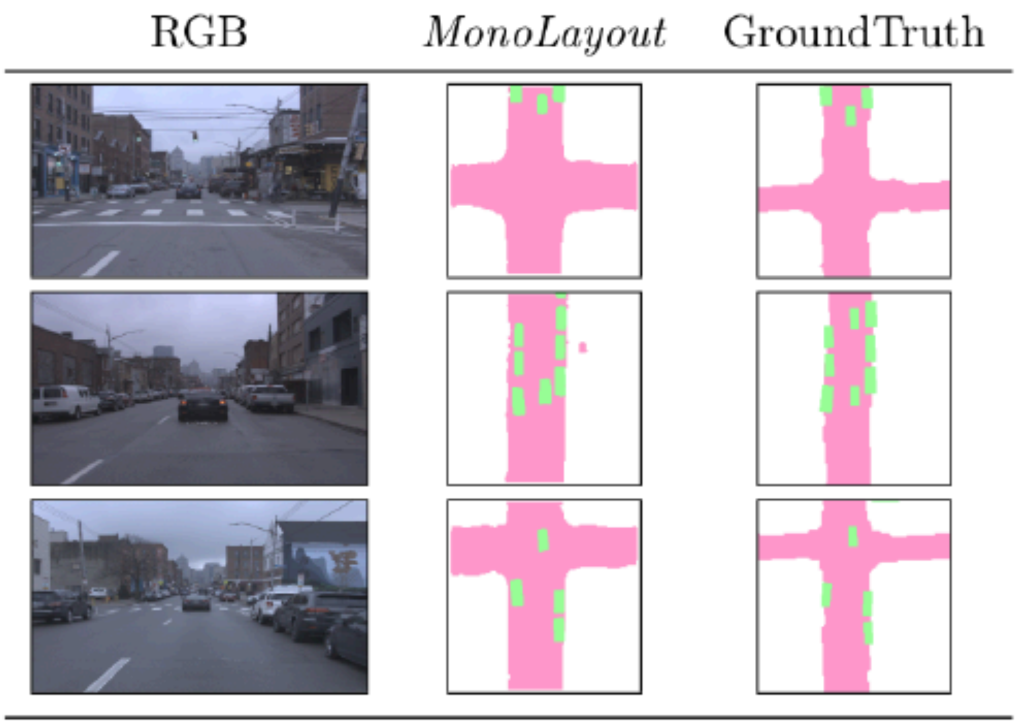


Figure 6: **Amodal scene layout estimation** on the Argoverse [5] dataset. The dataset comprises multiple challenging scenarios, with low illumination, large number of vehicles. *MonoLayout* is accurately able to produce sharp estimates of vehicles and road layouts. (Sidewalks are not predicted here, as they aren't annotated in Argoverse).

Mani et al 20

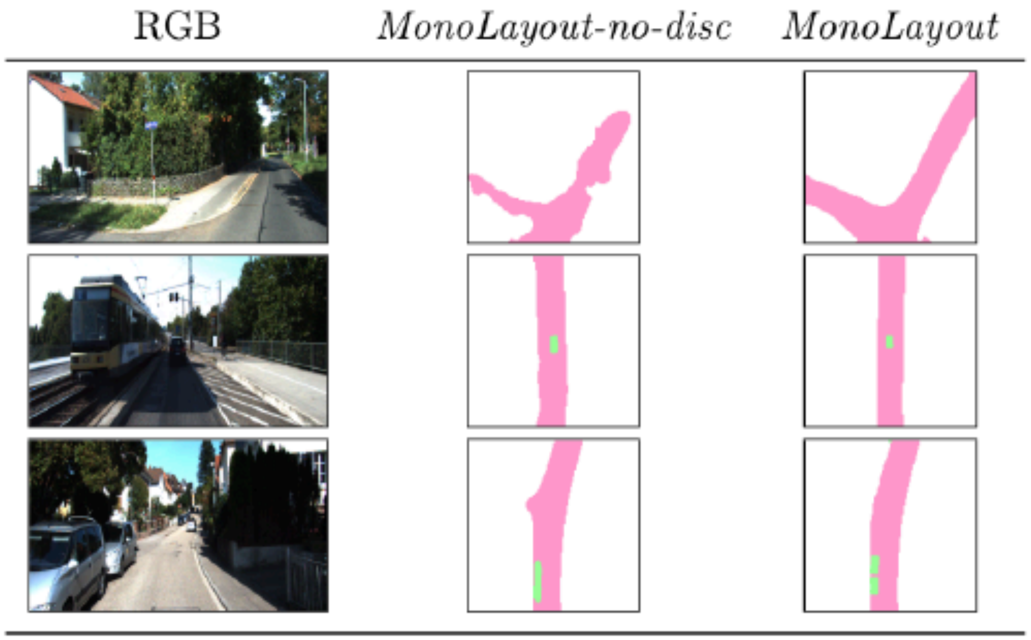


Figure 7: **Effect of adversarial learning**: As can be clearly seen here, the discriminators help enhance both the static (road) layout estimation (top and middle rows), as well as produce sharper vehicle boundaries (bottom row). While this translates to performance gains in static layout estimation (*c.f*. Table 3), the gains in dynamic layout estimation are more cosmetic in nature.

Mani et al 20

Issues

- Shouldn’t temporal consistency help?
  - why only per frame?
- Depth prediction then “dropping” seems like a bad idea
  - the ground depth map is pretty much a plane, so why not camera model?
  - why not more sophisticated ground plane model?
    - plane+relief?
- BIG Q: could you use this to plan?

Temporal consistency

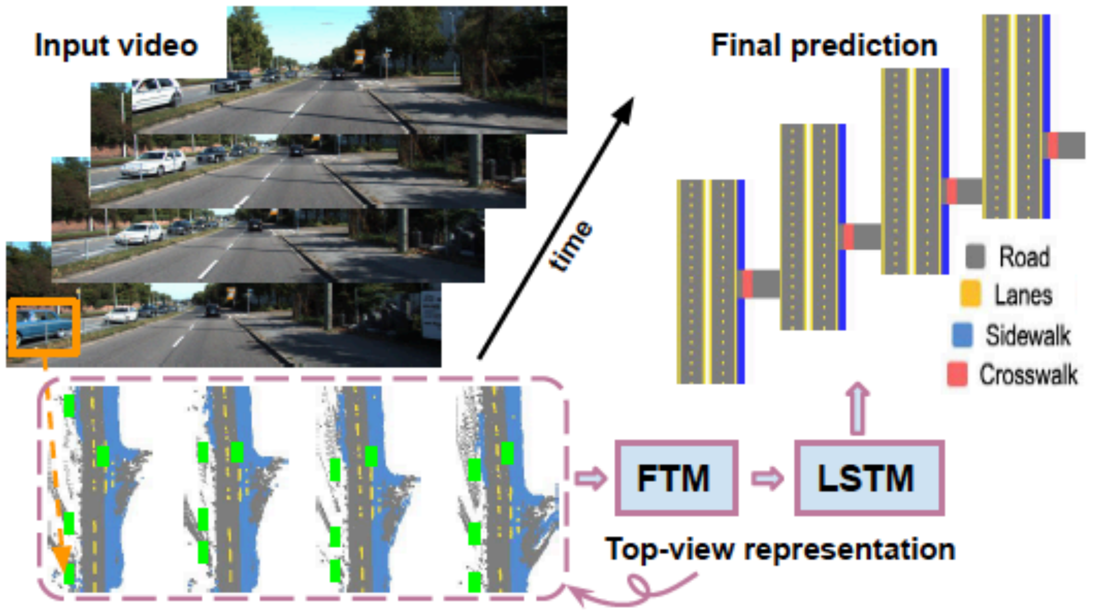


Figure 1: Given perspective images (top left) that captures a 3D scene, our goal is to predict the layout of complex driving scenes in top-view both accurately and coherently.

Liu et al 20

Temporal consistency

Estimates camera motion

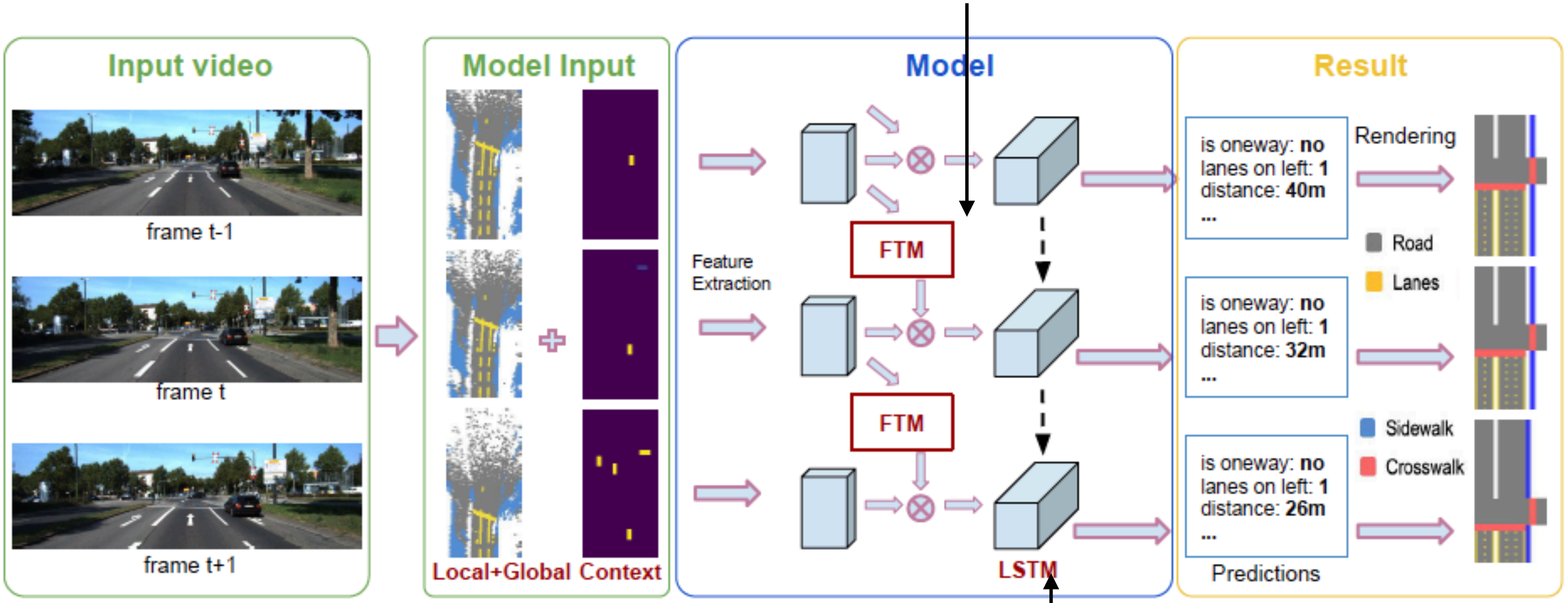


Figure 2: **Overview of our proposed framework:** Given videos as input, top-view maps aggregated from local, global and context information, are fed into our FTM/LSTM network to predict the parametric road scene layout.

Temporal smoothing

Liu et al 20

End to end

- Train three modules, using
    - parametric annotations
    - a renderer
- Then train the lot end-to-end

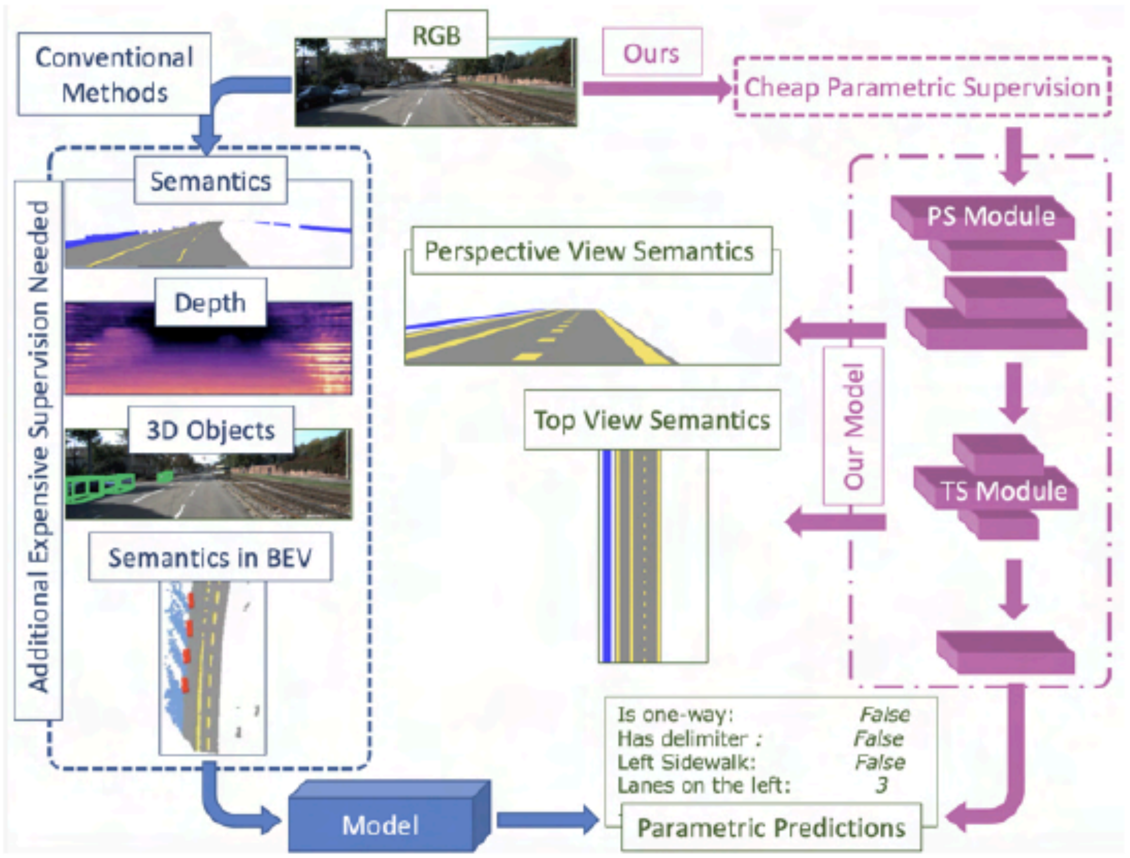


Figure 1. We propose an end-to-end model that inputs perspective image and outputs parametric layouts in top-view. Compared to existing methods, ours requires only the parametric layout annotations during training and achieves SOTA performance under complex road scenarios. Moreover, it generates occlusion-reasoned (see the predicted semantics on regions occluded by cars) pixel-level semantics in both perspective and top view.

Liu et al 22

Data

Have

Want

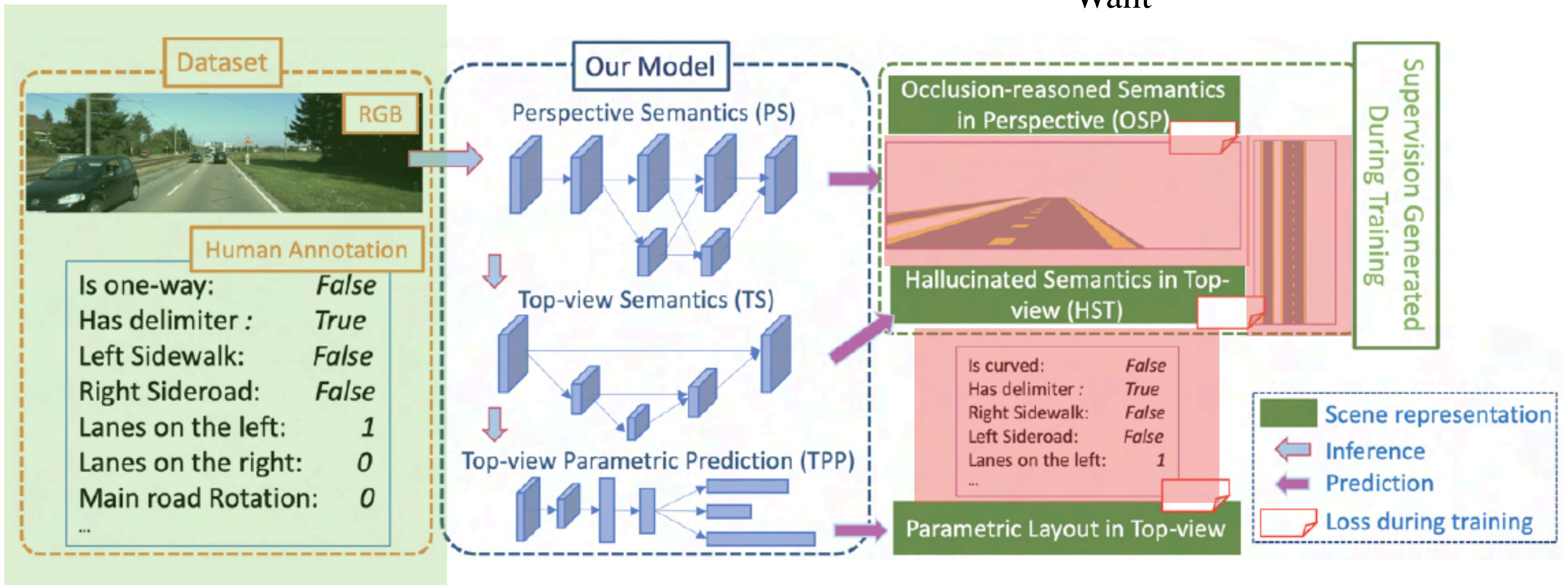
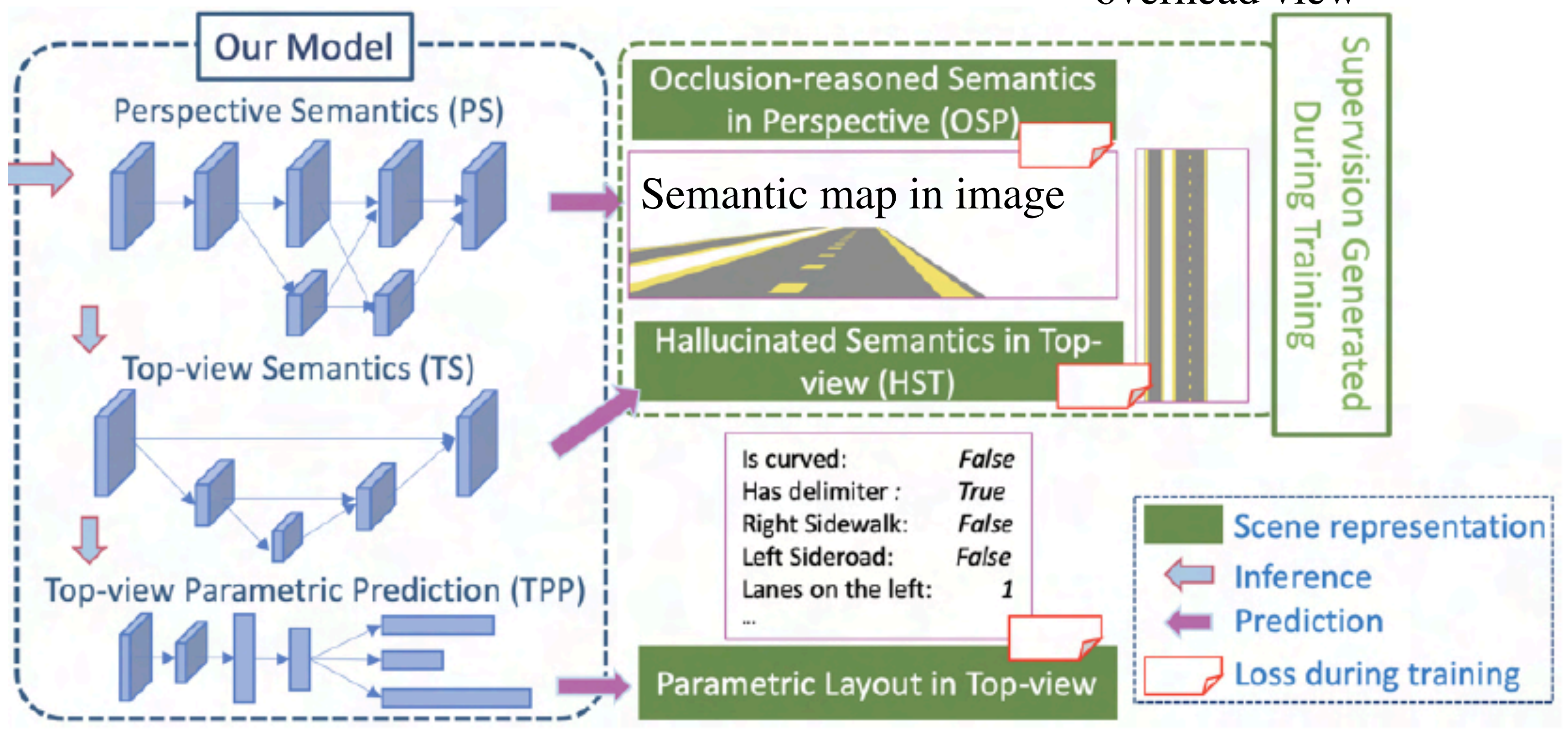


Figure 2. **Overview of our proposed framework:** Taking a single RGB as input, our model predicts (1) occlusion-reasoned semantics in perspective view, (2) hallucinated semantics in top-view and (3) parametric layout predictions in top-view, with only *attribute-level* annotations in top-view. This is achieved with multiple intermediate modules and deeply supervised training.

Liu et al 22

Structure

Semantic map overhead view



Our Model
Perspective Semantics (PS)
Top-view Semantics (TS)
Top-view Parametric Prediction (TPP)
Occlusion-reasoned Semantics in Perspective (OSP)
Hallucinated Semantics in Top-view (HST)
Supervision Generated During Training
Parametric Layout in Top-view
Scene representation
Inference
Prediction
Loss during training

Semantic map in image

Is curved:	*False*
Has delimiter :	*True*
Right Sidewalk:	*False*
Left Sideroad:	*False*
Lanes on the left:	*1*
...	

Semantic parameters overhead view

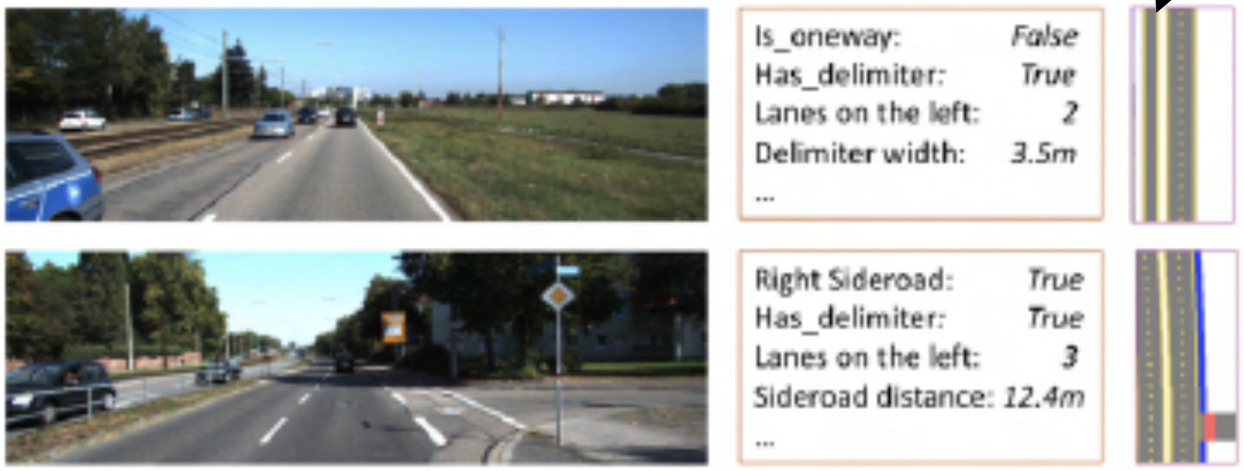
Liu et al 22

Training

- Train each separately, then end-to-end

Actual labels, from human labeller

Ground rendering, predicted from labels



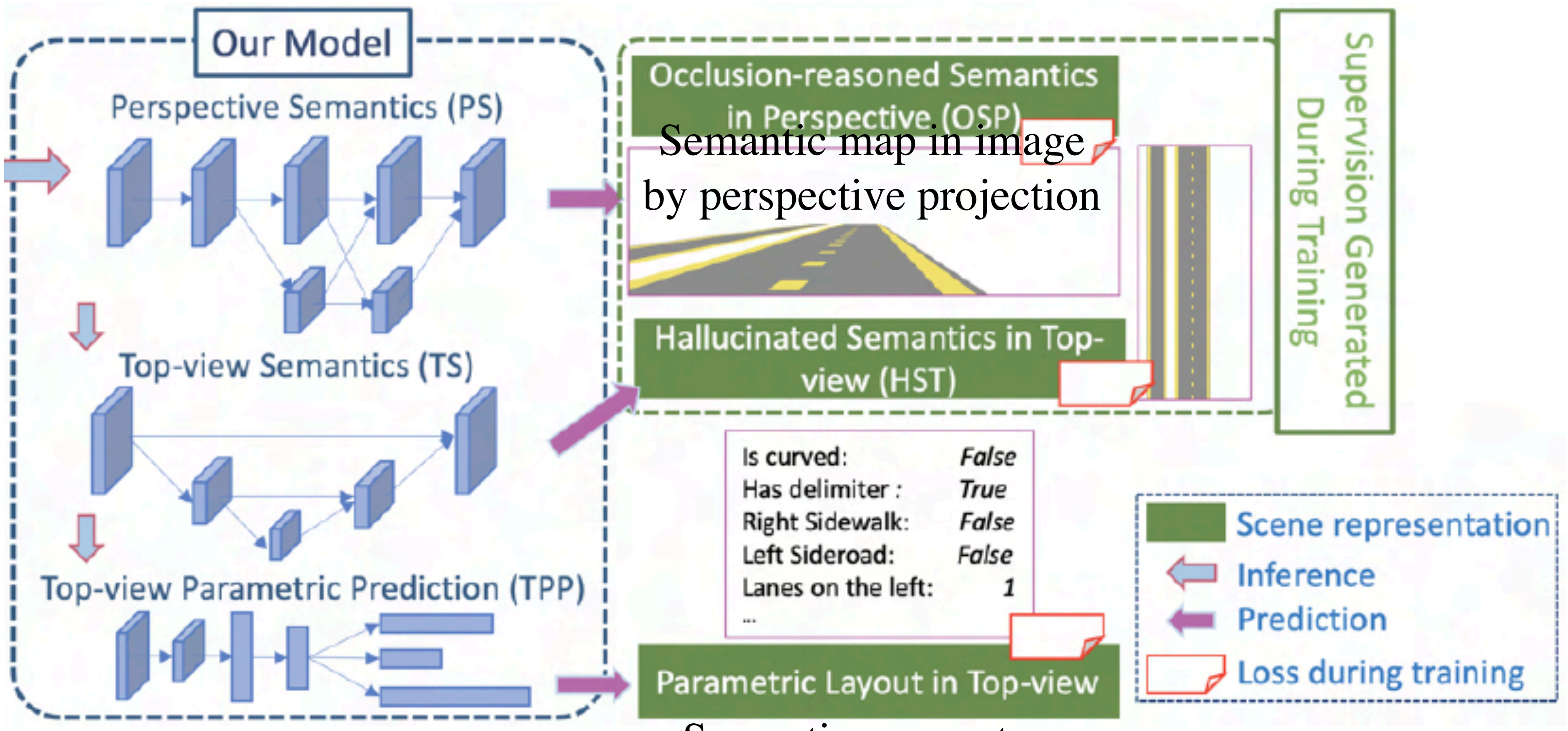
Is_oneway:	*False*
Has_delimiter:	*True*
Lanes on the left:	*2*
Delimiter width:	*3.5m*
...	

Right Sideroad:	*True*
Has_delimiter:	*True*
Lanes on the left:	*3*
Sideroad distance:	*12.4m*
...	

Figure 5. Examples of rendered ground-truth for TS module. From left to right: RGB, parametric human annotations and rendered pixel-level semantics in top-view.

Data

Semantic map
overhead view
from renderer


Our Model
Perspective Semantics (PS)
Top-view Semantics (TS)
Top-view Parametric Prediction (TPP)
Occlusion-reasoned Semantics in Perspective (OSP)
Hallucinated Semantics in Top-view (HST)
Supervision Generated During Training
Is curved: False
Has delimiter : True
Right Sidewalk: False
Left Sideroad: False
Lanes on the left: 1
...
Parametric Layout in Top-view
Scene representation
Inference
Prediction
Loss during training

Semantic map in image
by perspective projection

Semantic parameters
overhead view
from human annotation

Liu et al 22

Does it work?

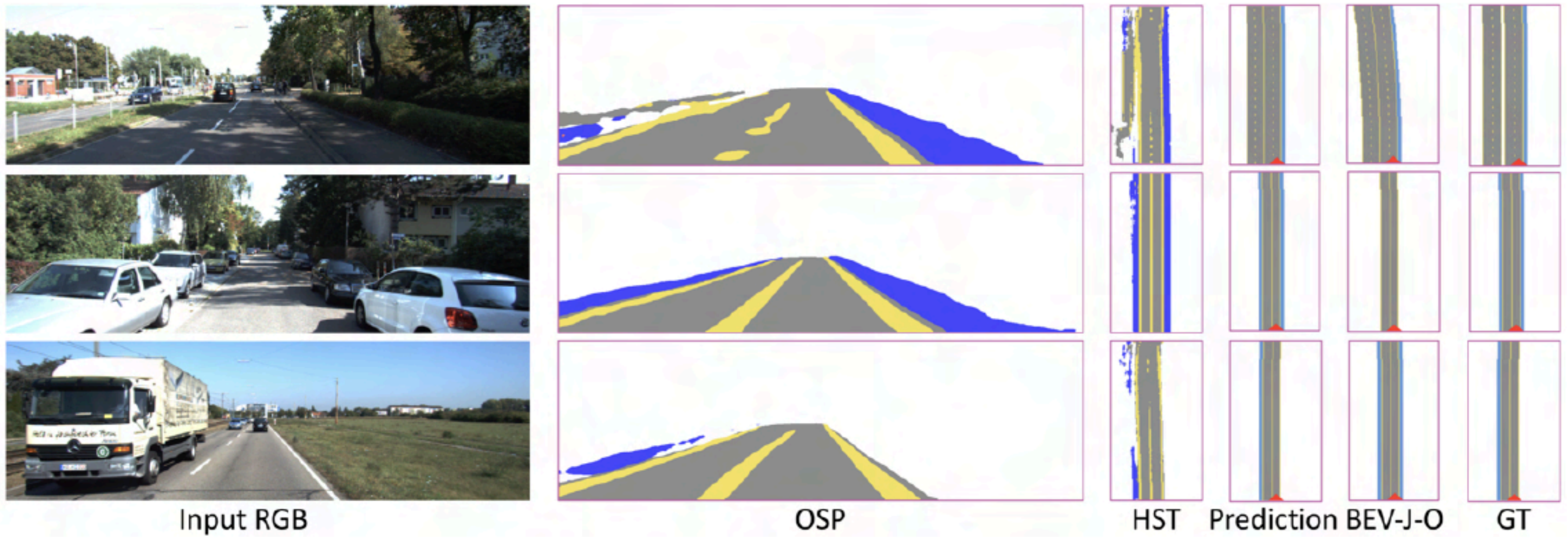


Figure 6. Full predictions of our proposed model. From left to right: input RGB, OSP, HST, image rendered from parametric predictions, results from [23] and image rendered from ground-truth attributes.

23 is Liu, 20!

Liu et al 22

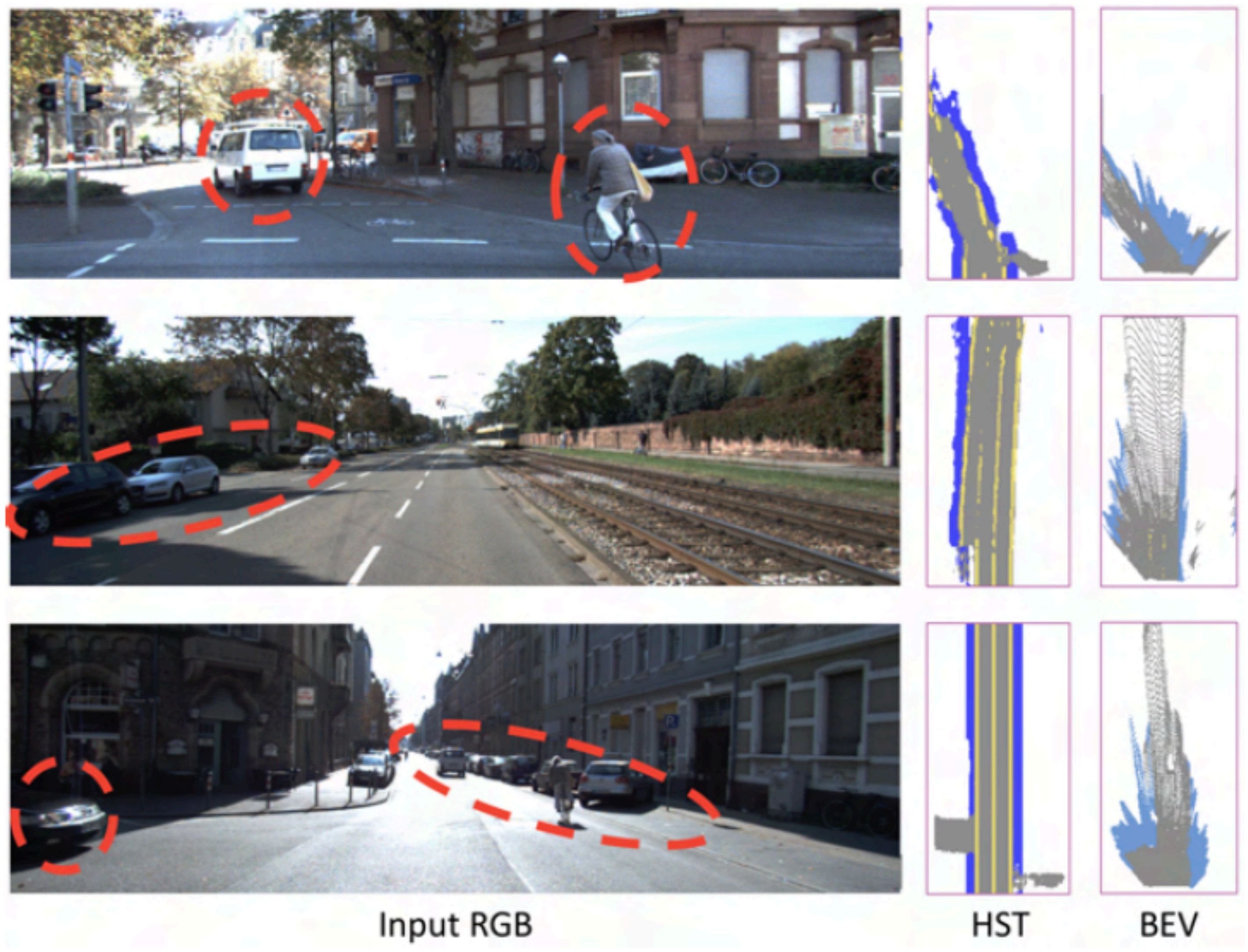


Figure 8. We demonstrate input RGB, predicted HST as well as BEV of [46], which is trained with thousands of pixel-level annotated images and LiDAR images. As can be seen in these examples, our model is able to hallucinate far away regions in a realistic manner, even on curved road, with NO pixel-level human annotations.

Liu et al 22

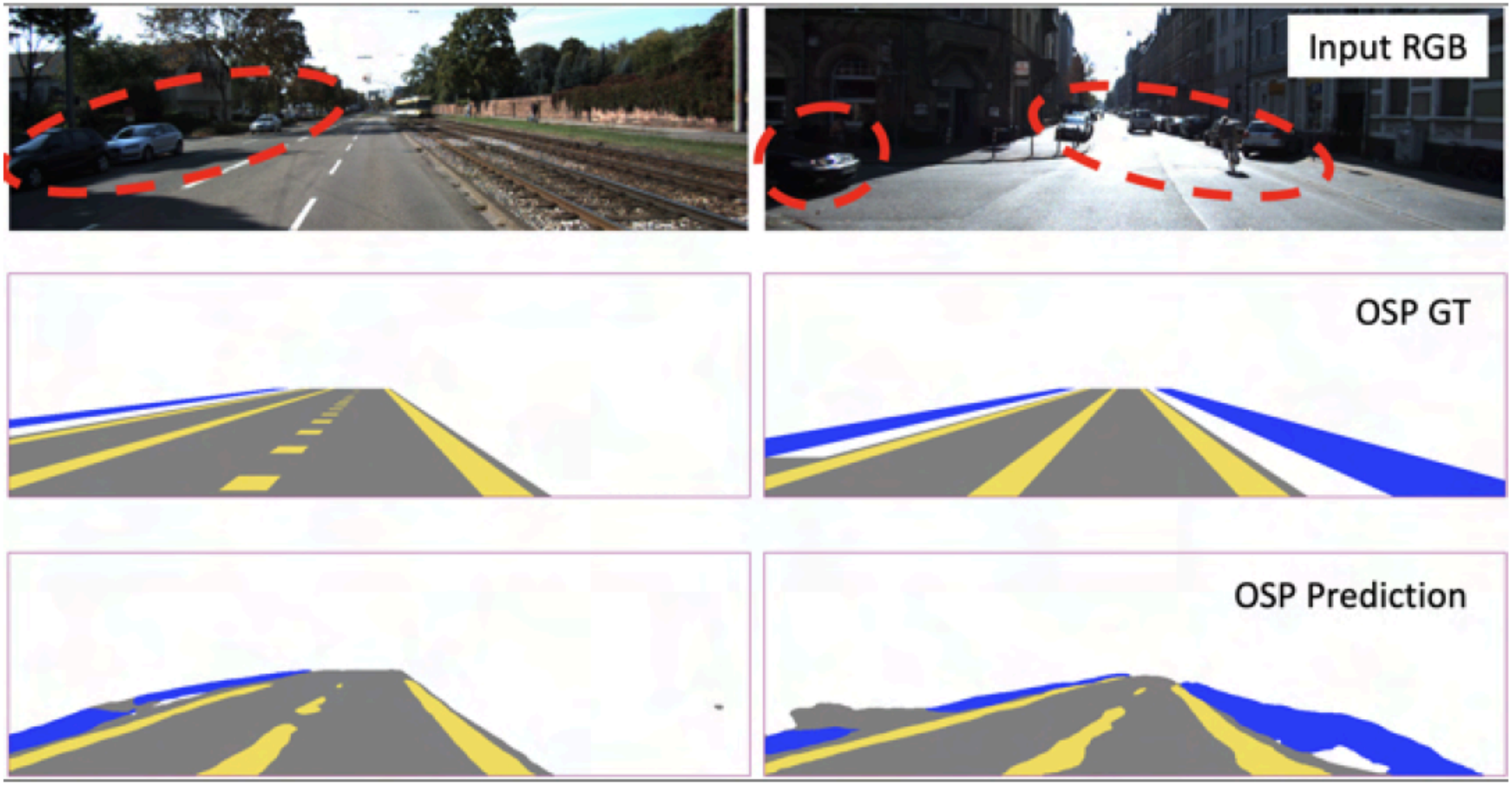


Figure 9. Input image, generated ground-truth pixel-level semantics and predicted semantics from top to bottom row. Our model is able to predict the semantics quite well despite occlusions.

Liu et al 22

Issues

- Shouldn’t temporal consistency help?
  - why only per frame?
- Why not more sophisticated ground plane model?
    - plane+relief?
    - this is very likely do-able
- BIG Q: could you use this to plan?

AutoLay - using Lidar as well

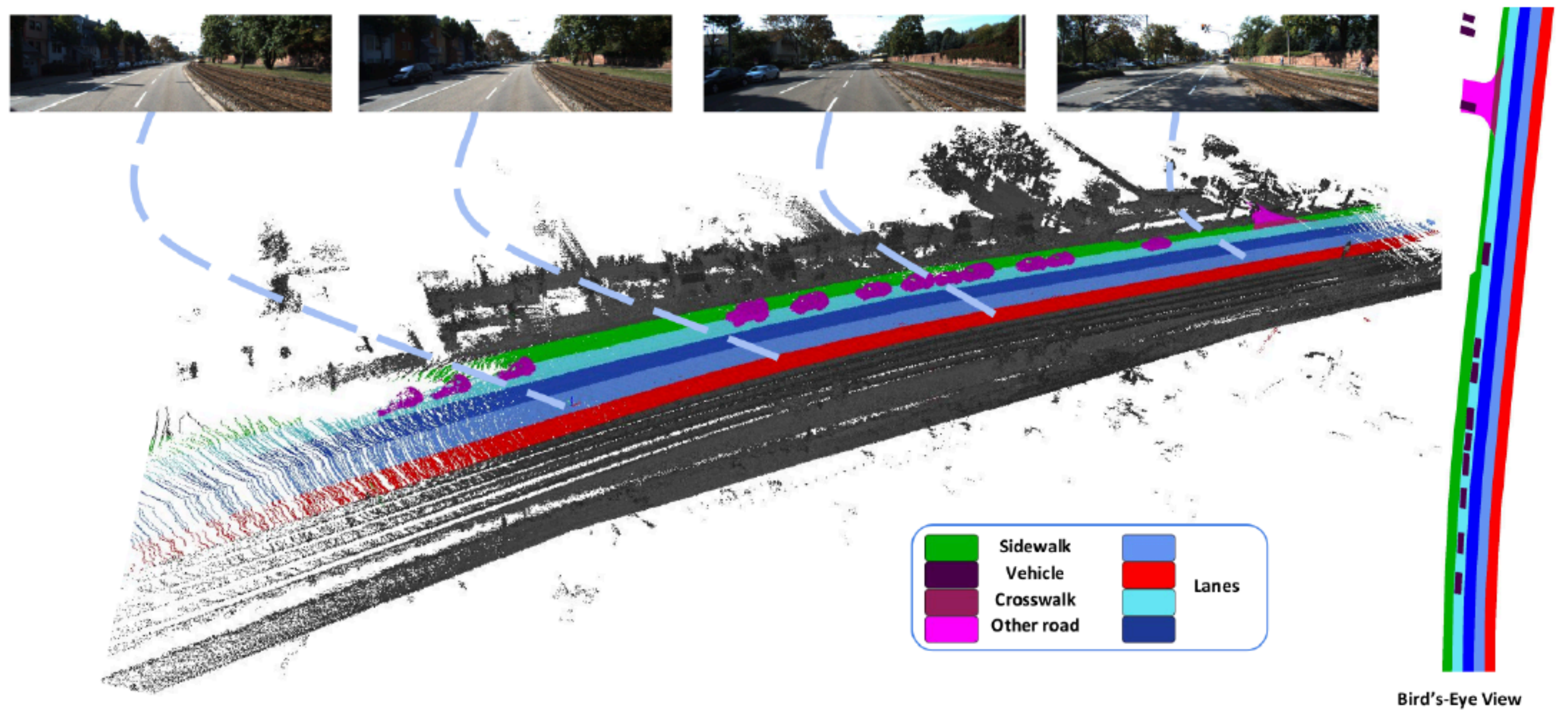


Fig. 1: **Amodal layout estimation** is the task of estimating a semantic occupancy map in bird's eye view, given a monocular image or video. The term *amodal* implies that we estimate occupancy and semantic labels even for parts of the world that are occluded in image space. In this work, we introduce ***AutoLay***, a new dataset and benchmark for this task. *AutoLay* provides annotations in 3D, in bird's eye view, and in image space. A sample annotated sequence (from the KITTI dataset [1]) is shown below. We provide high quality labels for sidewalks, vehicles, crosswalks, and lanes. We evaluate several approaches on sequences from the KITTI [1] and Argoverse [2] datasets.

Mani et al, 21

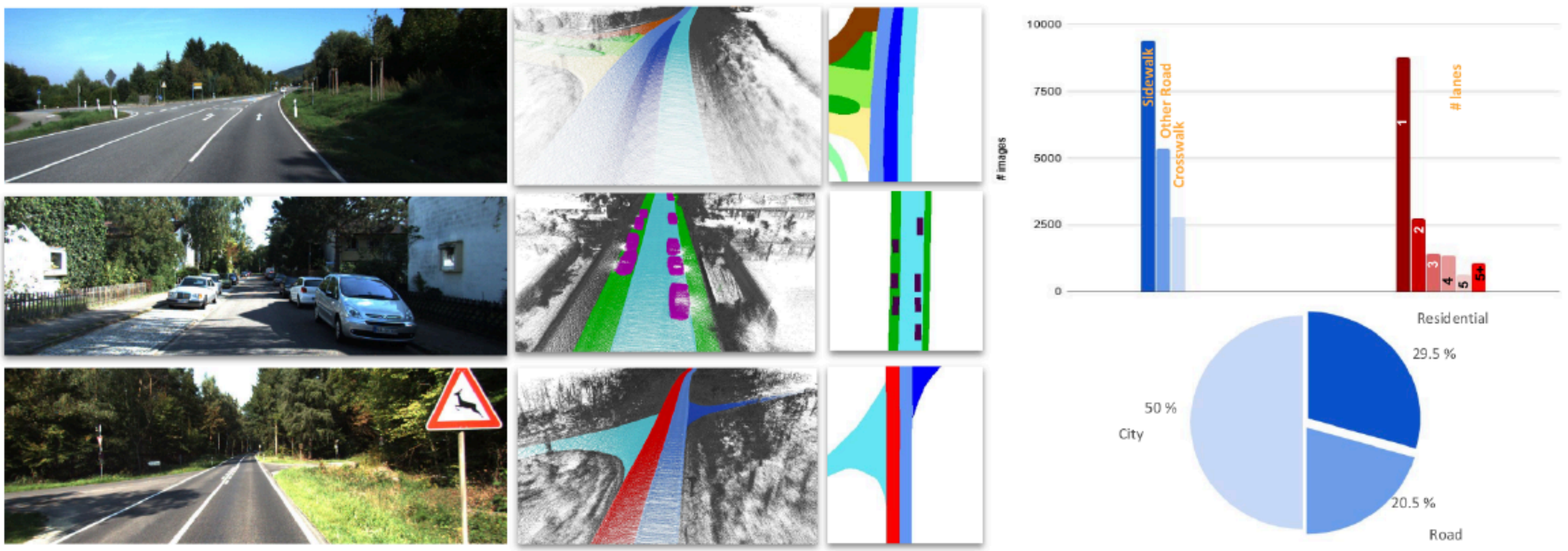


Fig. 2: **Dataset**: (*Left to right*) Sample images from the *KITTI* split of *AutoLay*. Corresponding annotated lidar pointclouds. Amodal scene layout in bird's eye view. (*Last column*) Distribution of semantic classes (bar plot), and scene types (pie chart).

Mani et al, 21

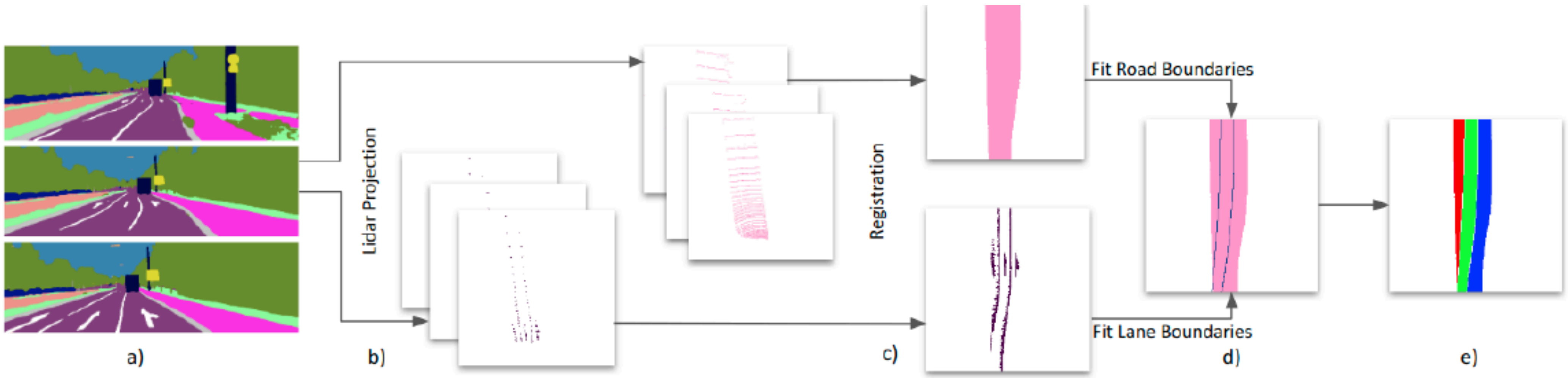


Fig. 3: **Weak Supervision setup**: We show our automated (noisy) label generation scheme herein. Points from the lidar frame are projected down to semantically segmented images (a), to obtain sparse static layouts (b). These sparse layouts are stacked across an image sequence to generate dense static layouts (c). In the next step (d), road and lane boundaries are extracted and combined to obtain lane layouts (e).

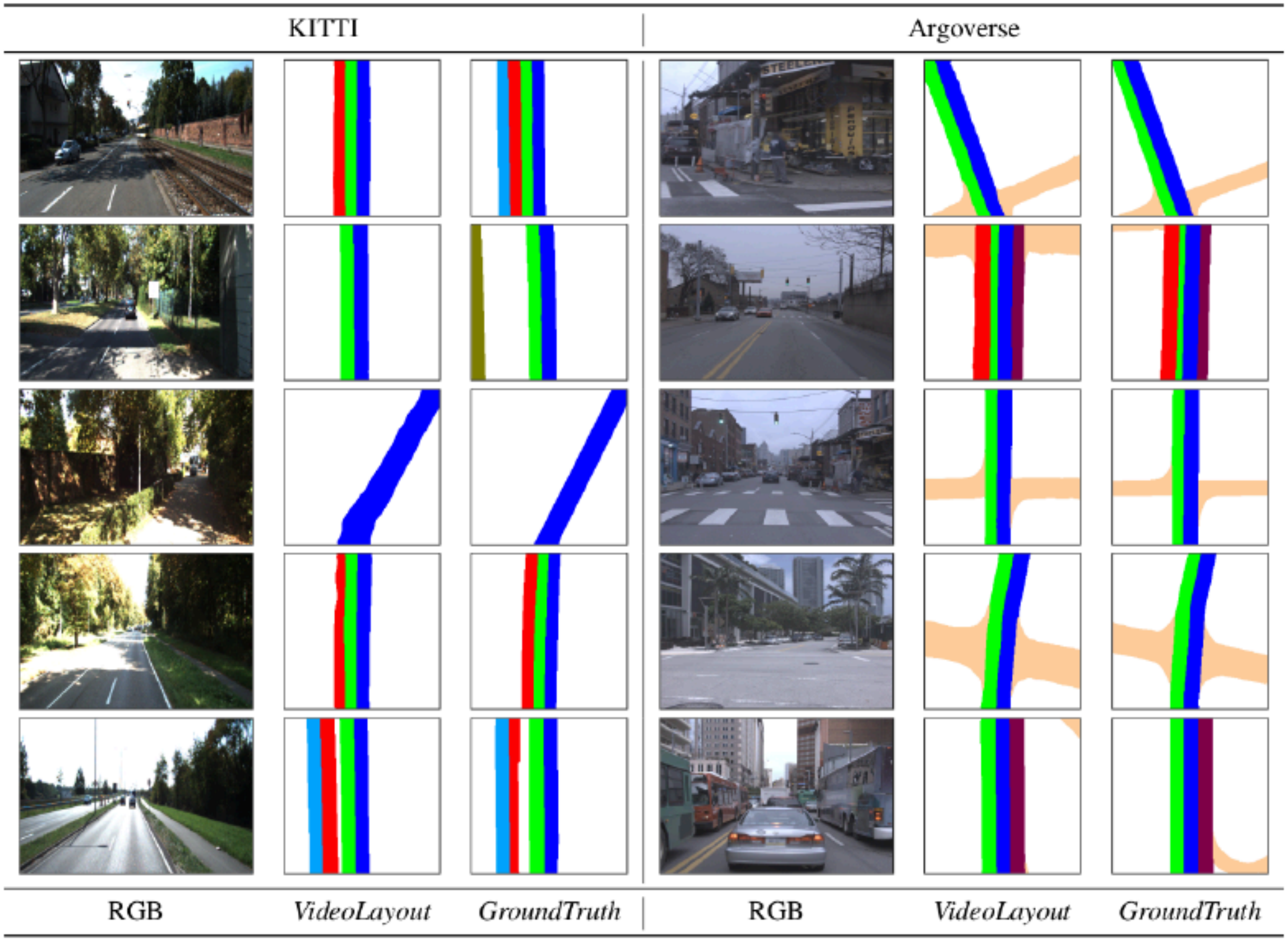


Fig. 4: **Static layout estimation**: *VideoLayout* predicts fine-grained attributes of the road scene including lanes, side roads and ego-lane. Each individual color represents a single lane. Ego-lane is shown in blue and side-roads are shown in light orange. *VideoLayout* produces decent static layout estimates for both AutoLay and Argoverse datasets. Its able to hallucinate occluded regions in the scene very reliably specially for Argoverse dataset.

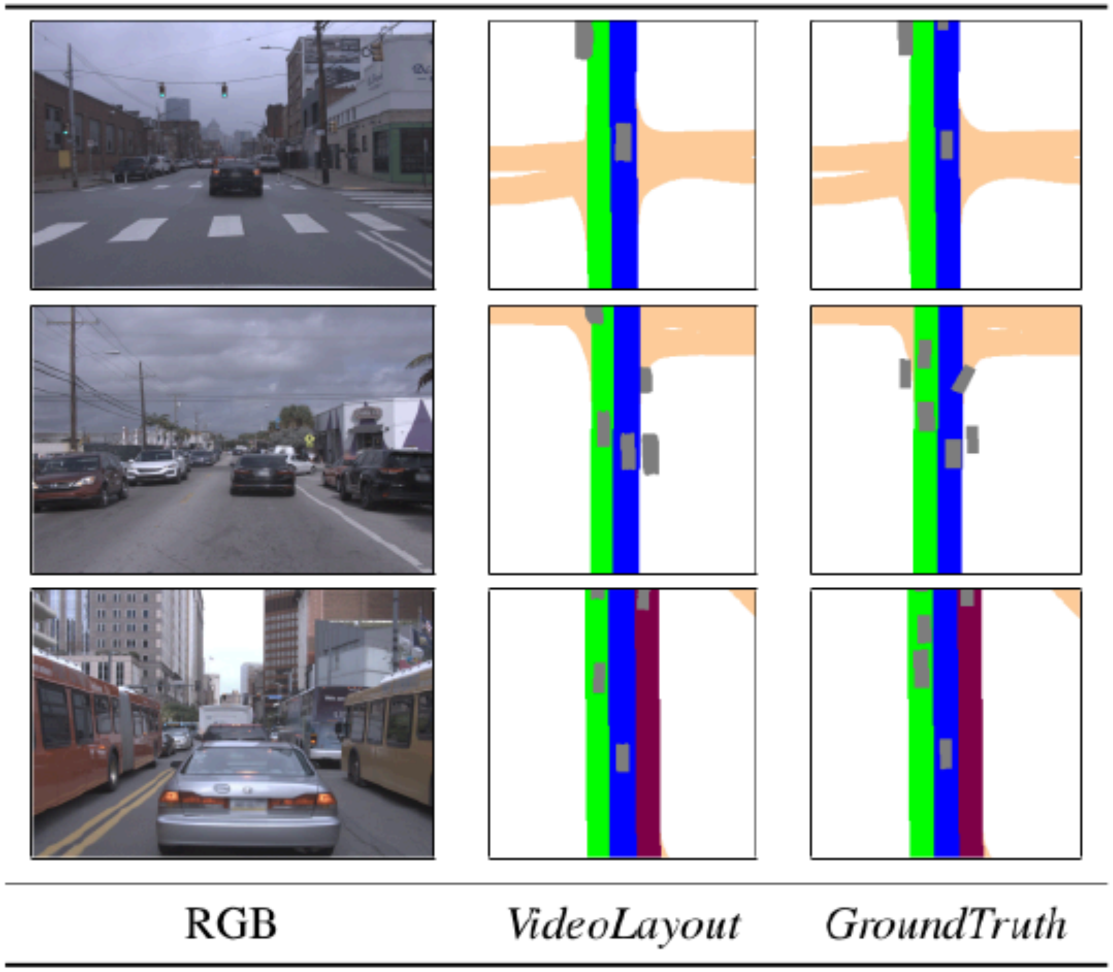


Fig. 5: **Dynamic Layout Estimation:**. *VideoLayout* provides crisp and accurate vehicle occupancies. The grey boxes indicate vehicle occupancies. Observe the ability of *VideoLayout* to precisely localize vehicles that are distant and partially occluded.

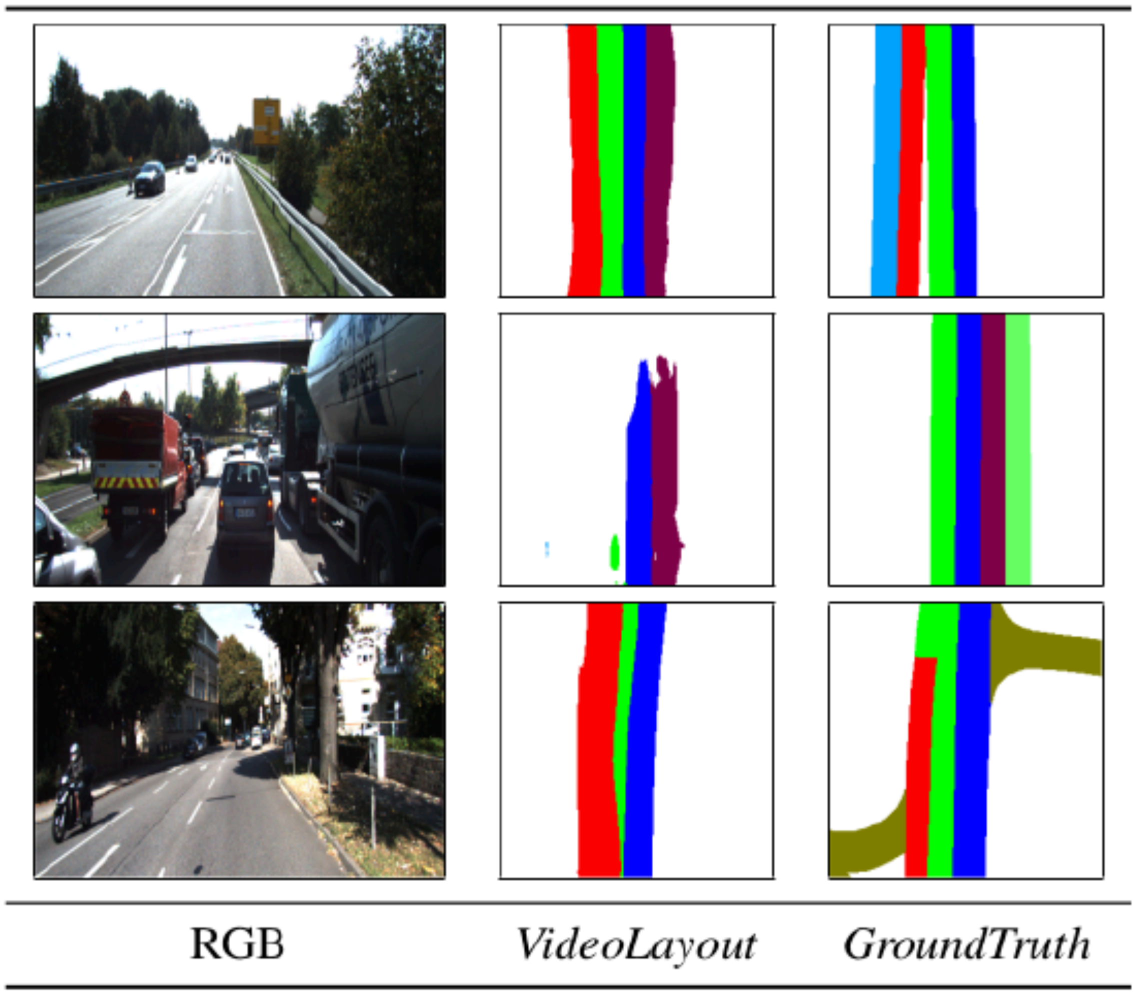


Fig. 6: **Failure Cases** of *VideoLayout*. (*Top row*) Failure to predict ego-lane. (*Middle row*) Failure in heavy traffic scenarios. (*Bottom row*) Failure in predicting lane width.

Direct prediction of BEVs

- rather than predict bits and pieces, then project

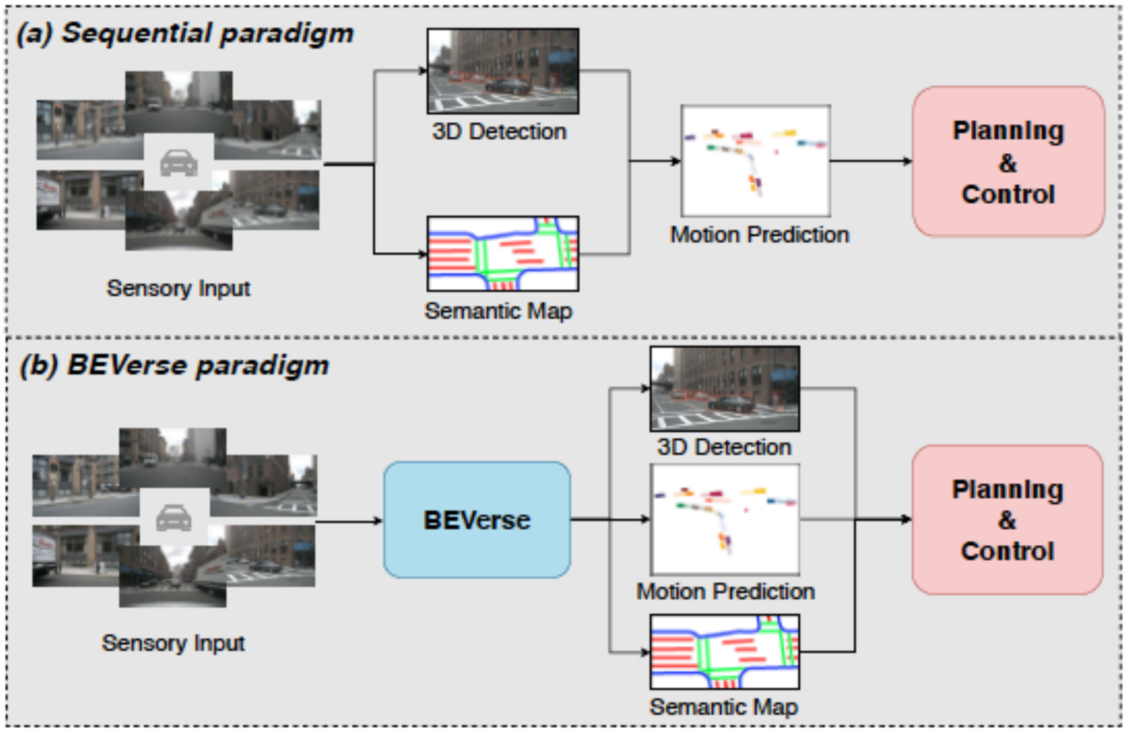


Figure 1. The traditional paradigm (a) follows the sequential design, where the perception, prediction, and planning&control are conducted one by one. Note that the perception includes the detection for dynamic objects and the map construction for static environments. Since the sequential paradigm inevitably suffers from repeated feature extraction and severe error propagation, we propose BEVerse (b) for joint perception and prediction. With shared feature extraction and parallel multi-task inference, BEVerse achieves a better trade-off between performance and efficiency.

Zhang, 22

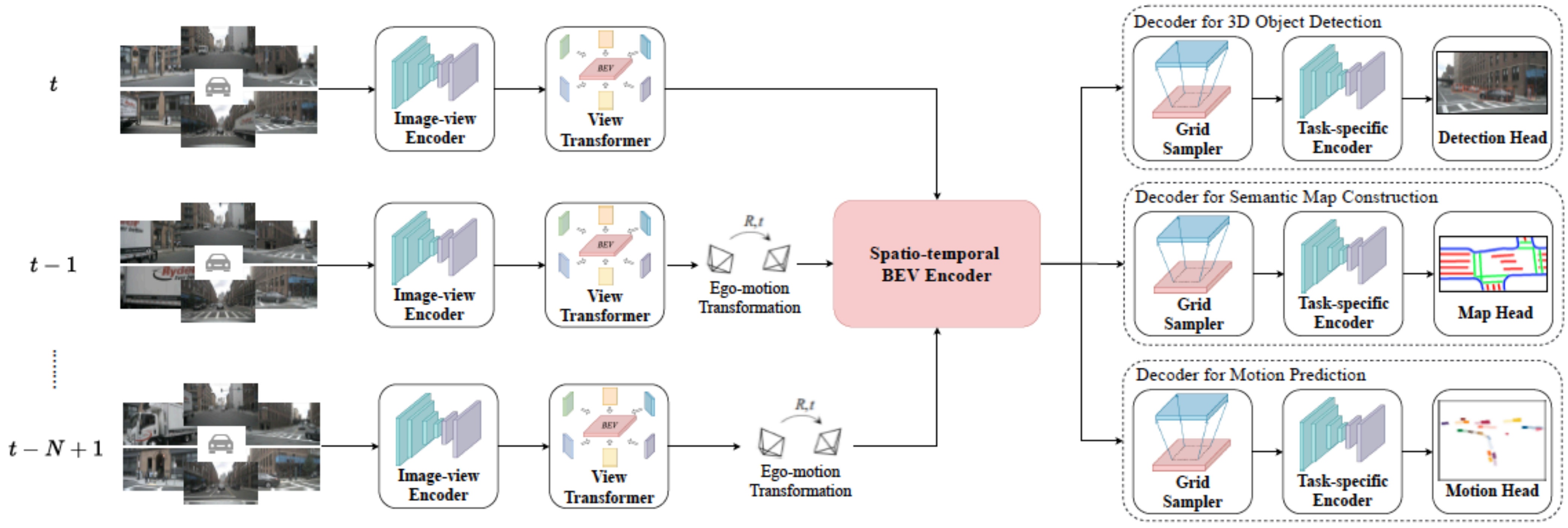

t
$t-1$
$t-N+1$
Image-view Encoder
View Transformer
BEV
R, t
Ego-motion Transformation
Spatio-temporal BEV Encoder
Decoder for 3D Object Detection
Decoder for Semantic Map Construction
Decoder for Motion Prediction
Grid Sampler
Task-specific Encoder
Detection Head
Map Head
Motion Head

Figure 2. The framework of BEVerse. With consecutive frames from surrounding cameras as input, BEVerse first constructs the BEV feature representation for each timestamp. The process includes the image-view feature extraction and the view transformation. Then, the BEV features from past frames are aligned to remove ego-motions and processed by the temporal model. Finally, the well-established BEV feature with both spatial and temporal information is sent to multiple task decoders for joint reasoning of perception and prediction.

Zhang, 22

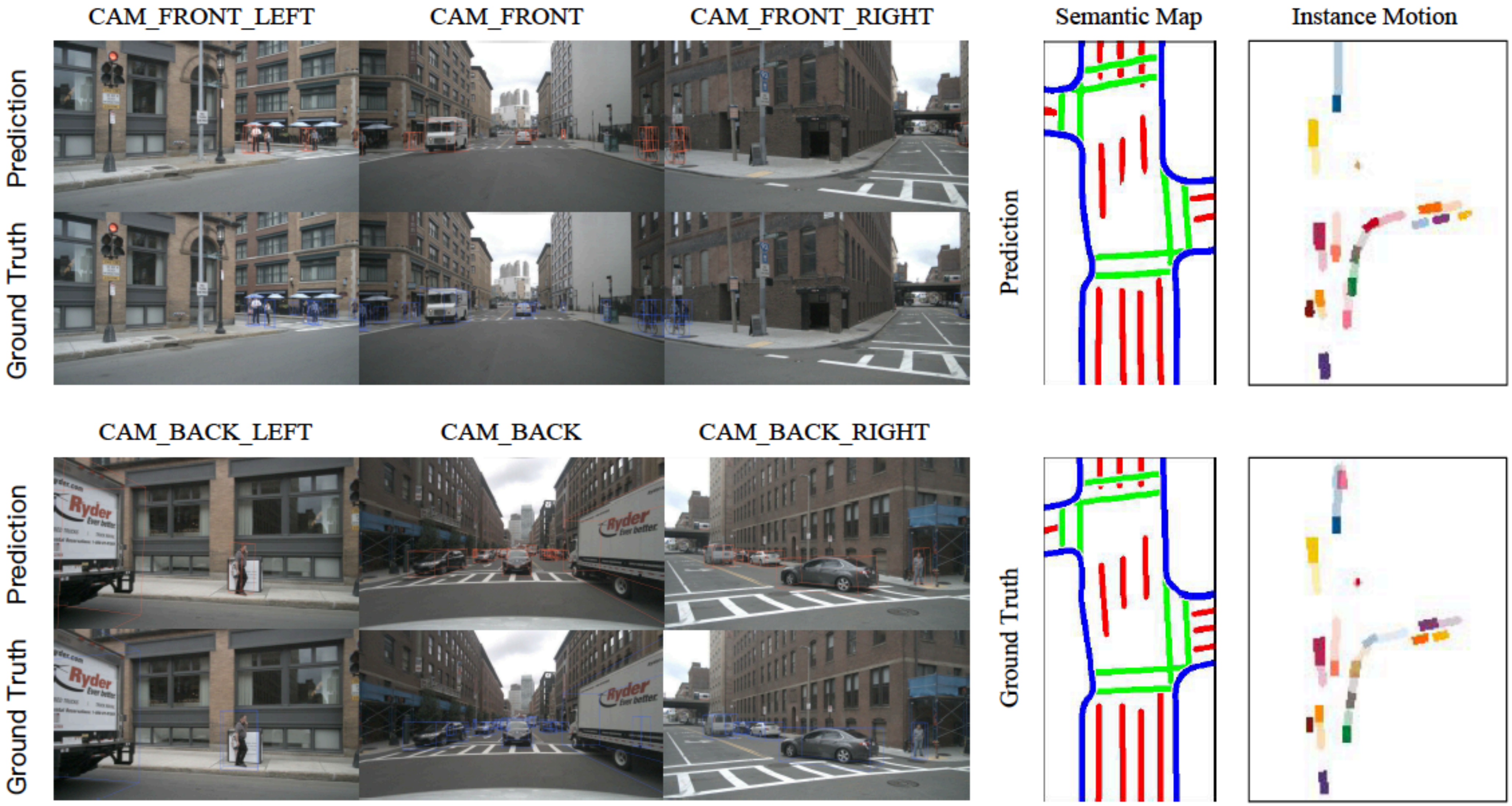


Figure 4. The qualitative results for 3D object detection, semantic map construction, and instance motion prediction. For the visualization of semantic maps, we represent lane dividers, pedestrian crossings, and lane boundaries with red, green, and blue colors. For the visualization of motions, the future trajectories of road agents are shown with transparent paths. Best viewed in color.

Zhang, 22