Supplementary Materials – Perceive, Predict, and Plan: Safe Motion Planning Through Interpretable Semantic Representations

Abbas Sadat*[1], Sergio Casas*[1,2],
Mengye Ren[1,2], Xinyu Wu[1], Pranaab Dhawan[1], Raquel Urtasun[1,2]

Uber ATG[1], University of Toronto[2]
{asadat, sergio.casas, mren3, xinyuw, pdhawan, urtasun}@uber.com

In this supplementary material, we present the trajectory parameterization and sampling procedure in more details. Additionally, we give an overview of all the planning cost-functions. Further details about training our models are included as well as more qualitative results.

1 Trajectory Parametrization and Sampling

The output of the planner is a trajectory that consists of a sequence of bicycle model states $\tau = p_t$, $p_t = (x, y, \theta, v, \kappa, a)$, where (x, y) is the position of the center of the rear axle of the vehicle, θ is the heading, v and a are the forward velocity and acceleration, and κ is the curvature of the vehicle path which can be converted to steering angle. Candidate trajectories can be generated by sampling various curvature and acceleration values and using the kinematic bicycle model to obtain the other states (position, heading, velocity) [1]. However, this approach will be very inefficient as most of the sampled trajectories will not exhibit proper lane-based driving. Therefore we adopt an alternative approach which uses the lanes structures to generate higher quality trajectories. Specifically, we sample trajectories in *Frenet Frame* of the driving-path of the desired lane [2], i.e. instead of kinematic bicycle-model state, we use longitudinal position and lateral offset (and their higher order derivatives) relative to a driving-path to represent a trajectory. Figure 1 demonstrates such parametrization, where r is the driving path parametrized by arc-length s, $s(t)$ is the longitudinal position of the vehicle parametrized by time t, and $d(s)$ is the lateral offset from the driving-path parametrized by arc-length s. Each pair of $s(t)$ and $d(s)$ can describe a bicycle model trajectory. Specifically, the Frenet state defined as $[s, \dot{s}, \ddot{s}, d, d', d'']$ can be transformed to bicycle model state $(x, y, \theta, v, \kappa, a)$ [2]. Note that $\dot{(.)} := \frac{\partial}{\partial t}$, and $(.)' := \frac{\partial}{\partial s}$ denote the derivatives with respect to time and arc-length.

Our trajectory sampling procedure is as follows: we first sample a set of longitudinal trajectories that convert various velocity profiles such as stopping, accelerating to a specific-velocity, maintaining the current velocity. Then, for each longitudinal trajectory, we sample lateral trajectories that include maneuvers such as nudging, changing-lane, and following the driving-path. Combining

* Denotes equal contribution

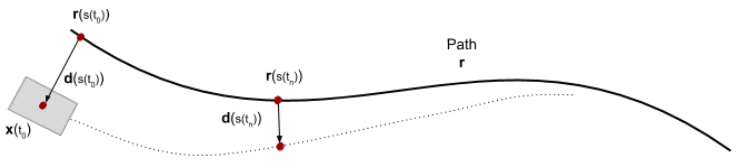


Fig. 1: **Frenet Frame**

the two sets results in bicycle model trajectories that are proper lane-based trajectories including lateral maneuver variations to handle challenging scenarios.

We use quintic and quartic polynomials to represent longitudinal and lateral trajectories. Specifically, the set of longitudinal trajectories $\mathcal{S} = \{s(t)\}$ are sampled by using a large set of mid-conditions $[\dot{s}(t_1), t_1]$ and end-conditions $[\dot{s}(T), T]$ and solving for two quartic polynomials that are stitched together. The acceleration ($\ddot{s}$) at t_1 and T are fixed at 0. We parameterize lateral trajectories $[d(s), d'(s), d''(s)]$ in terms of the longitudinal distance s. We generate a set of mid-conditions $[d(s_1), s_1]$ and fix $d'(s_1)$ and $d''(s_1)$ to be 0. We require the lateral trajectories to approach the driving path and hence the end-conditions $[0, 0, 0]$. The initial, mid- and end-conditions are used to obtain two quintic polynomials that specify the lateral offsets for each longitudinal trajectory. Each pair of sampled longitudinal and lateral trajectories $[s(t), d(s)]$ are transformed back to a bicycle model trajectory.

2 Motion Planner Cost Functions

In this section we present the details of the cost functions that are used to evaluate trajectories in the motion planner. Figure 2 shows a subset of the subcosts.

Collision, Safety-margin, and Headway: The trajectory of the SDV should be collision-free and at a safe distance from surrounding objects. We use collision and safety-distance costs (Fig. 2(a)) to penalize trajectories that have spatio-temporal overlap with the predicted trajectories of other actors or violate a safety margin. This is achieved by computing the distance between SDV polygon and the predicted polygon of all the other actors at each timestep. Furthermore, the SDV should maintain a safe headway to the leading vehicle, such that if the lead vehicle applies a hard break, the SDV can slow-down smoothly to avoid collision and uncomfortable breaking (Fig. 2(b)). This cost is computed using the relative longitudinal distance of the SDV and the lead vehicle as well as their velocities. Note that the above costs are defined when the prediction module generates bounding-boxes and trajectories for the actors. The collision and safety subcosts using occupancy representations are described in the paper. Also, if the prediction module forecasts multiple modes of trajectories for each actor, the above subcosts are computed for each predicted trajectory and are weighted by the probability of that trajectory mode.

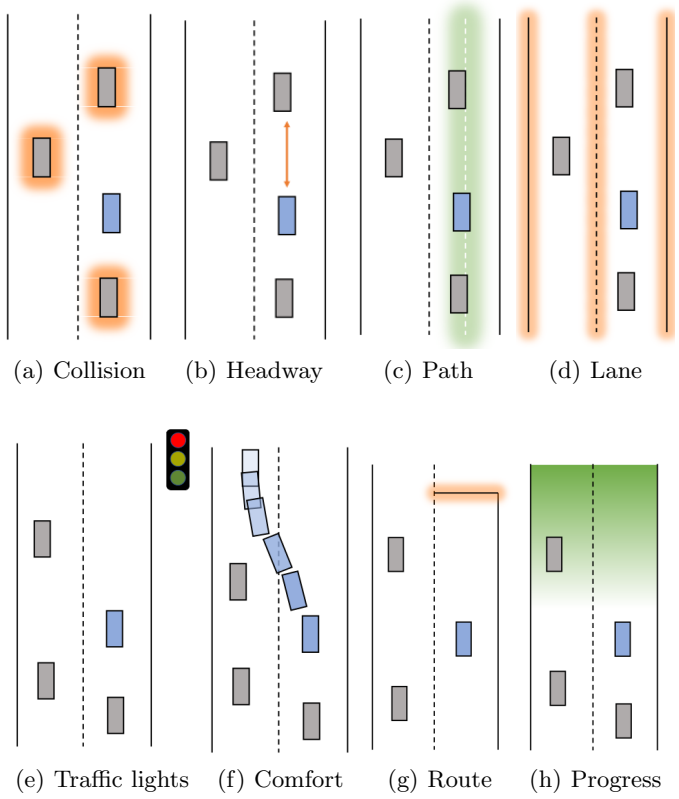


Fig. 2: **Examples of the motion planner cost functions**

Driving-path, Lane and Road Boundaries: The SDV should adhere to the structure of the lanes and roads. For example, it is expected that vehicles stay close to the center of the lane and not move over the boundaries of the lane and roads. For this purpose, we introduce subcosts that penalize the violation of lane and road boundaries as well as the distance to the driving-path of the lane. These subcosts are demonstrated in Fig. 2(c) and 2(d).

Speed-limit, Traffic Lights, and Stop Signs: We penalize the violation of speed-limit at each trajectory step to promote driving at the regulated speed-limit. Furthermore, for each red traffic-light or stop sign, the SDV needs to come to a stop at the intersection stop-line, represented by a longitudinal position along the lane. Therefore, we introduce cost functions where the violation of each stop-line by a trajectory is penalized.

Route, Progress, and Cost-to-go: The SDV is given a high-level route represented as a sequence of lanes. Although the SDV can use other lanes that are adjacent to the route lanes, we penalize the number of lane-changes that is required to return back to the route lanes. Furthermore, if the SDV is using a dead-end lane (i.e., lanes that diverge from the route), we penalize violation of a distance-threshold to the end of that lane such that the SDV is forced to change the lane (Fig. 2(g)). Trajectories are also rewarded (negative cost) by the distance they move along the lane to promote progress in the route (Fig. 2(h)).

We also introduce a cost that captures what comes beyond the planning horizon. Specifically, for each trajectory we penalize the deceleration that is needed to reduce the SDV speed, from the value specified by the last trajectory point to an acceptable lower value, due to upcoming speed-limits, stop-signs, or red traffic light.

Dynamics and Comfort: We prune trajectories that violate vehicle constraints such as maximum acceleration or curvature to only allow executable trajectories for the SDV. Furthermore, since rapid changes in acceleration or steering lead to uncomfortable rides, we penalize such aggressive motions. Specifically, we penalize jerk and violation thereof, acceleration and violation thereof, lateral acceleration and violation thereof, curvature and its first and second derivatives. All the violations above are computed based on a predefined threshold.

3 Training details

Optimizer: We use Adam optimizer to update the weights in the perception backbone and occupancy forecasting networks, with a base learning rate of $1e{-}5$. We use exponentiated gradient descent to optimize the planning parameters such that the subcosts' weights remain greater than zero after each iteration i, with a planning base learning rate $\alpha = 1e-3$:

$$w^{(i+1)} = w^{(i)} \exp(-\alpha \mathbf{g})$$

Here α is the learning rate and $\mathbf{g}$ denotes the gradient of w. We employ linear scaling to both learning rates with respect to the batch size.

Hyperparameters: In our multi-task learning setting, we use a weight of $\lambda_S = 1$ for the semantic occupancy cross entropy loss and $\lambda_M = 1e-3$ for the motion planning max-margin loss. In the semantic occupancy cross entropy we employ hard negative mining with a ratio of 10 negative examples for each positive one. Note that originally the classification problem is much more imbalanced, with the vast majority of the grid cells being not occupied. More precisely, we first define a subset of negative pixels $\text{Neg}^{t,c}$ over time t and classes c, which include a random 10% of the non-occupied grid cells. Then we pick all the positive examples $\text{Pos}^{t,c}$ and the hardest $10 \cdot |\text{Pos}^{t,c}|$ from $\text{Neg}^{t,c}$ (the ones with the highest loss). Finally

we combine the positives and the subset of negatives to from the final subset of pixels $\mathcal{S}^{t,c}$.

4 Architecture details

In this section we give more details about the architecture of the recurrent occupancy updates. The backbone network was already explained in details in the paper.

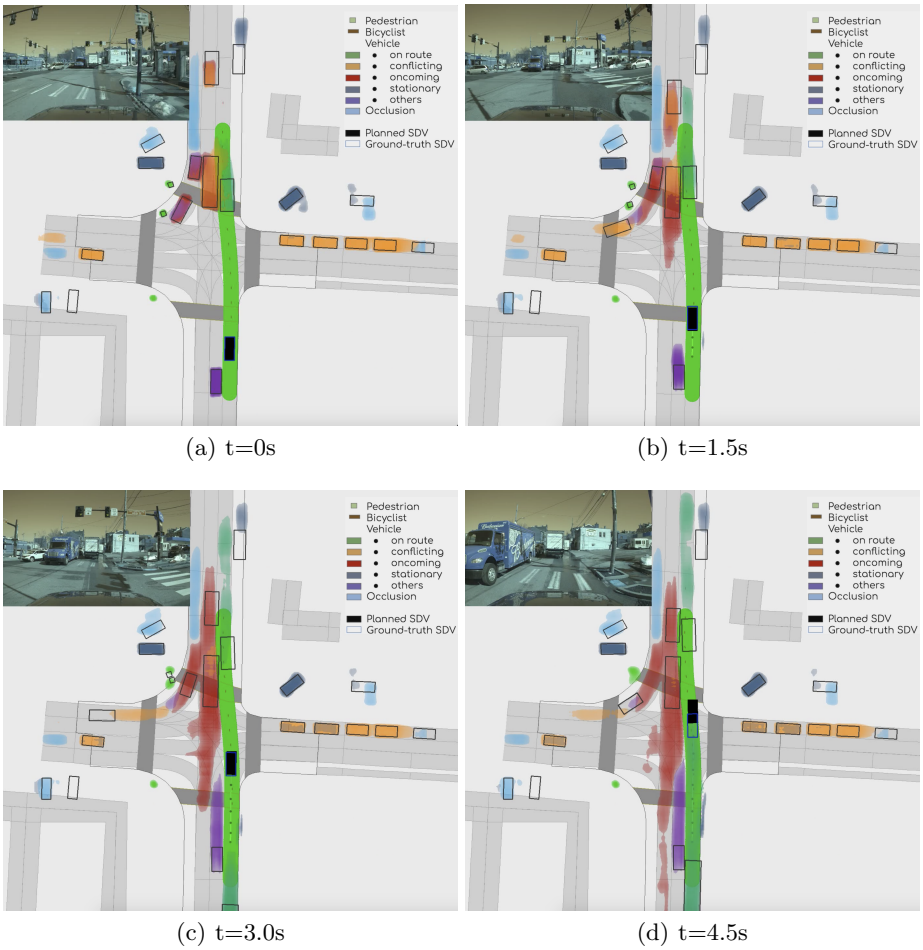
Recurrent Occupancy Update: We employ a multi-scale context fusion by performing two parallel fully convolutional networks with different dilation rates. One stream performs regular 2d convolutions over $\mathcal{F}_{2x}$ with a 2-layer CNN with dilation of 1, using 128 feature channels. The other stream takes the coarser features $\mathcal{F}$ and processes it with another 2-layer CNN with dilation of 2 at both layers, using 128 feature channels. We then apply bilinear interpolation to the feature tensor at 2x downsampling to bring it to the lower resolution (4x downsampling), and concatenate these two tensors along the channel dimension to obtain the context $\mathcal{F}_{\text{occ}}$. Our approach then predicts the occupancy over time in a recurrent fashion, from the context. Note that the context $\mathcal{F}_{\text{occ}}$ is downsampled 4 times from the input (0.8 m/pixel), but this is too coarse for motion planning (e.g., when trying to turn into tight spaces, it could look like the space is occupied by a parked car when it is not, just due to discretization). Thus, we seek to produce the occupancies at 0.4 m/pixel. Our proposed recurrence looks as follows:

$$l^{t,c} = l^{t-1,c} + \mathcal{U}_\theta(\mathcal{F}_{\text{occ}} \otimes \mathcal{I}(l^{0:t-1,c}))$$

$l^{t,c}$ are the logits for root class c at timestep t into the future. $\mathcal{I}$ is a 2x bilinear interpolation to bring the previous occupancy logits into a resolution of 0.8 m/pixel. $\otimes$ represents feature-wise concatenation. $\mathcal{U}_\theta$ is 2-layer CNN with a hidden dimension of 256, where the first convolution is transposed to upsample the resolution by 2, and the second layer is a regular convolution. The initial occupancy at t=0 $l^{0,c}$ is predicted by a small 2-layer CNN from $\mathcal{F}_{\text{occ}}$ and upsampled using $\mathcal{U}_\theta$. All the aforementioned convolutions have a filter size of 3, stride of 1 and no max pooling. Because all the tensors in this recurrence are spatial, this design choice of using interpolation and transposed convolutions to perform the hidden computations at a lower spatial resolution is important to keep the GPU memory requirements low.

5 Additional Qualitative Results

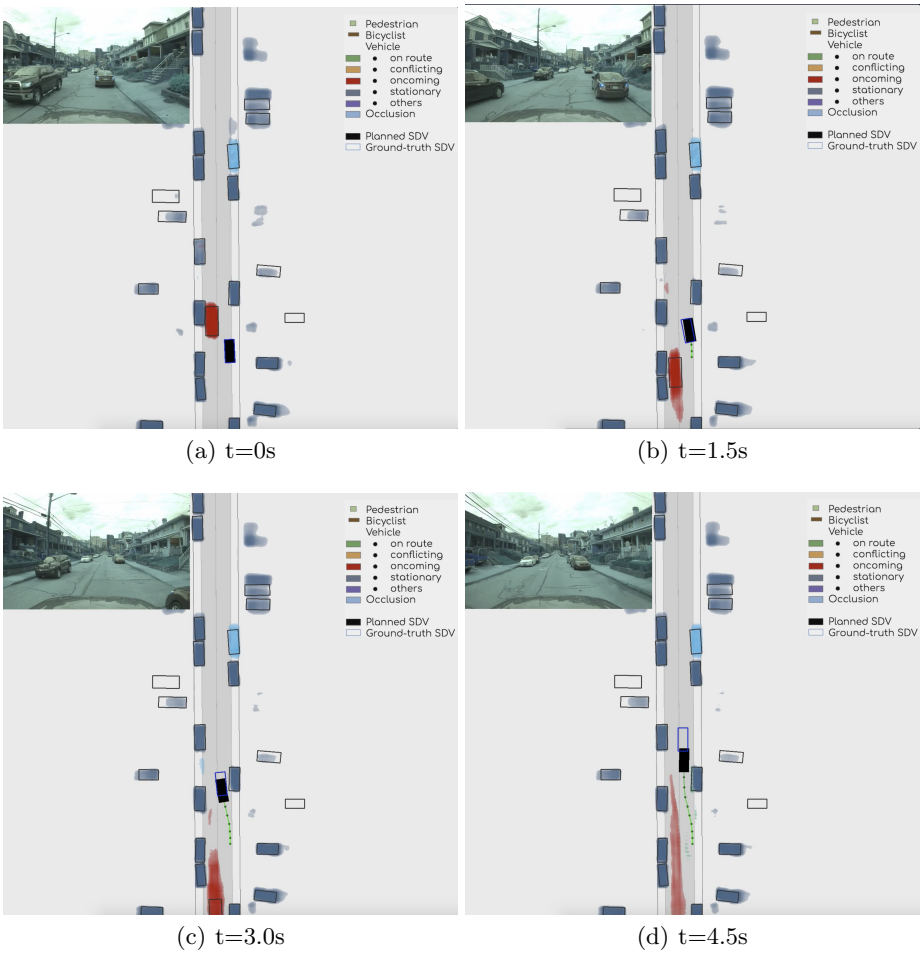
Figures 4-8 show additional qualitative results. Each figure include the occupancy at the current time as well as the forecasts for future time-steps.



(a) t=0s (b) t=1.5s

(c) t=3.0s (d) t=4.5s

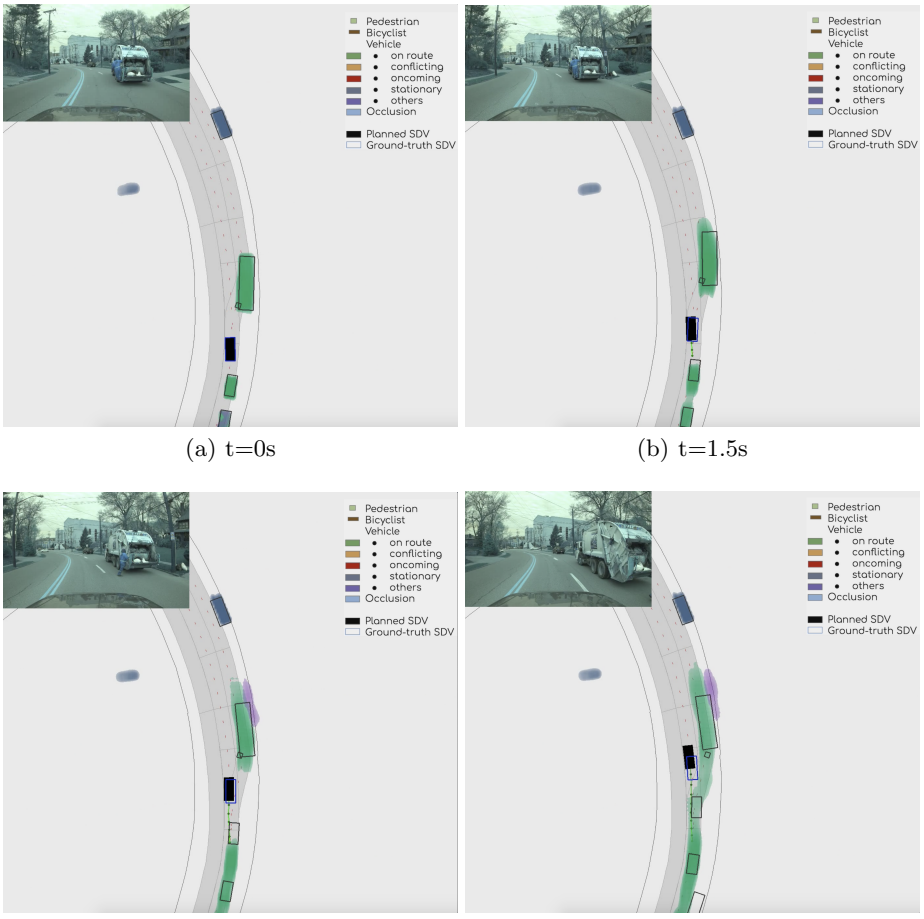
Fig. 3: **Qualitative results**: The legend on the top-right shows the color representing each subcategories of actors. On top-left, we show the image captured at the time by a front-view camera on the SDV. In this scenario, we can see regions on the lane (top-middle) that is occluded due to the obstruction by the oncoming truck.



(a) t=0s (b) t=1.5s

(c) t=3.0s (d) t=4.5s

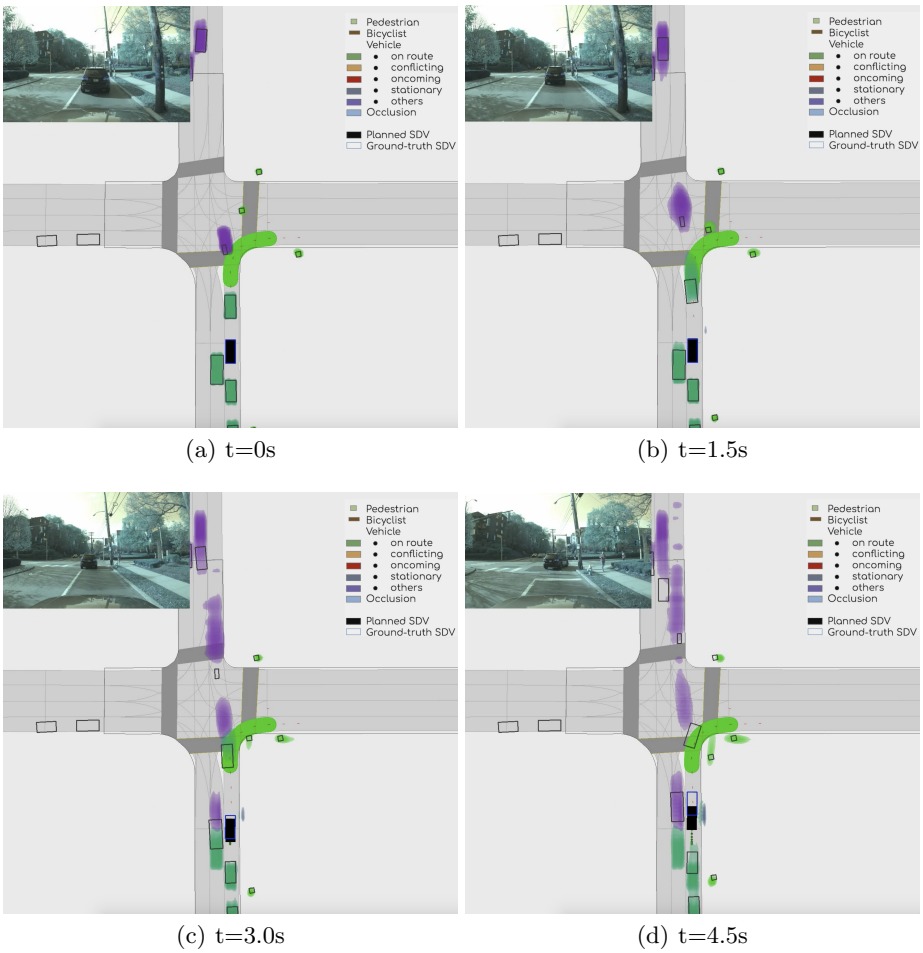
Fig. 4: **Qualitative results**: This figure demonstrates a scenario with a n oncoming truck as well as many stationary vehicles. The SDV is able to nudge around the parked vehicle adn continue in the route.



(a) t=0s (b) t=1.5s

(c) t=3.0s (d) t=4.5s

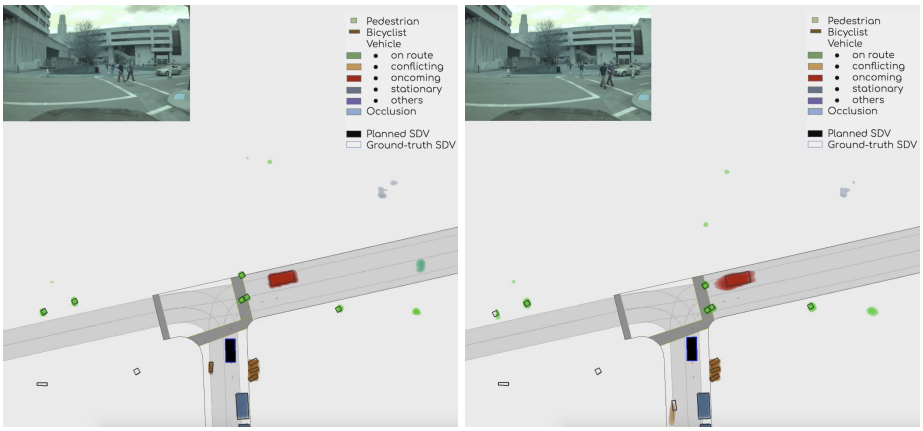
Fig. 5: **Qualitative results**: This example shows a garbage truck. As the truck is entering the right lane, and the SDV is able to nudge around it and continue in the route. Note that the occupancy representation of the truck is covering the person that on the side of the truck too.



(a) t=0s (b) t=1.5s

(c) t=3.0s (d) t=4.5s

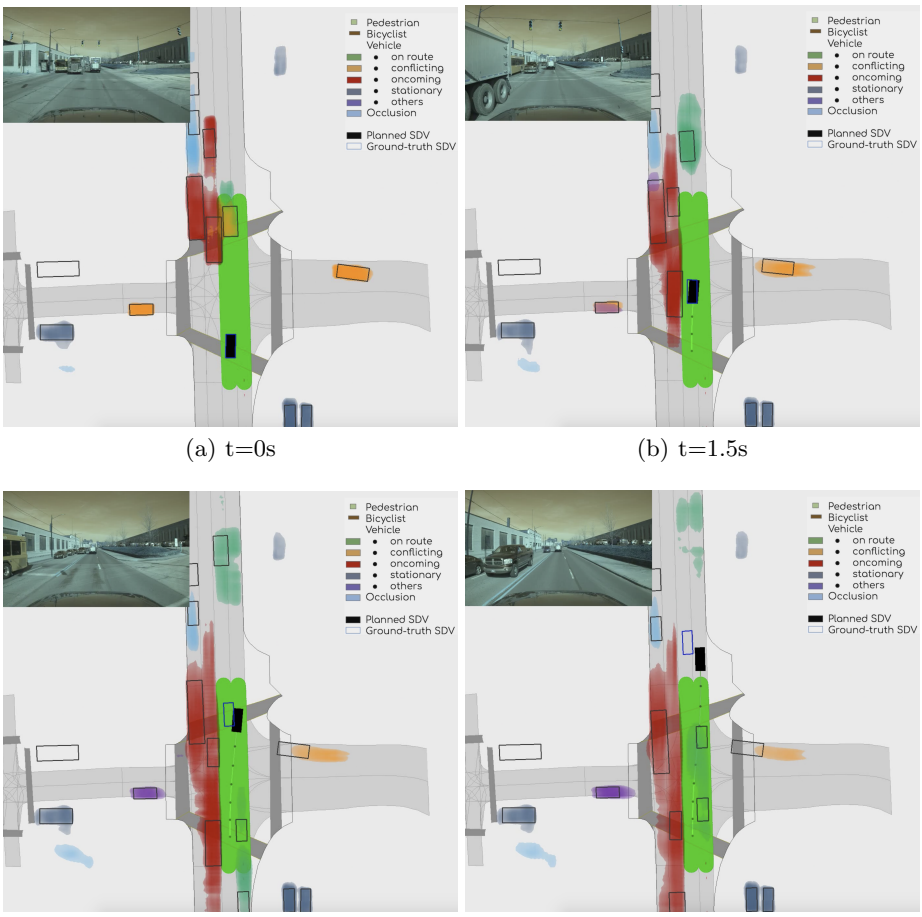
Fig. 6: **Qualitative results**: This figure show a vehicle on the left of the SDV at t=0s. As the vehicle approaches the intersection, it is categorized as others (i.e. not relevant to the planner) as it is located on a left-turn lane.



(a) t=0s (b) t=1.5s

(c) t=3.0s (d) t=4.5s

Fig. 7: **Qualitative results**: This example show cautious behavior od the SDV as some pedestrians are crossing the street.



(a) t=0s (b) t=1.5s

(c) t=3.0s (d) t=4.5s

Fig. 8: **Qualitative results**: In this scenario large vehicles are occupying the oncoming lane, and a truck is encroaching the SDV lane a little bit. The planner chooses to lane-change to the right, contrary to the human driver that continued driving on the same lane.

References

1. Kong, J., Pfeiffer, M., Schildbach, G., Borrelli, F.: Kinematic and dynamic vehicle models for autonomous driving control design. In: 2015 IEEE Intelligent Vehicles Symposium (IV), IEEE (2015) 1094–1099
2. Werling, M., Ziegler, J., Kammel, S., Thrun, S.: Optimal trajectory generation for dynamic street scenarios in a frenet frame. In: ICRA. (2010)