Is Anyone There? Learning a Planner Contingent on Perceptual Uncertainty
Supplemental Material

A Decision matrices for each scenario

The potential outcomes of the scenarios in our partially-observed planning benchmark can be denoted using a decision matrix. See Fig. 1 for decision matrices outlining the two-agent scenarios, and Fig. 2 for a decision matrix outlining the three-agent scenario.

	Underconfident	Optimal	Overconfident
Actor exists	Never attempts overtake, stuck behind truck	**Attempts fast overtake**, aborts after detection	**Attempts *slow* overtake**, unable to abort
No actor exists	Never attempts overtake, stuck behind truck	**Attempts fast overtake**, passes truck	**Attempts *slow* overtake**, passes truck

(a) Overtake scenario

	Underconfident	Optimal	Overconfident
Actor exists	**Slow ascent**, brakes and avoids crash	**Faster ascent**, brakes and avoids crash	***Fastest* ascent**, unable to brake to avoid crash
No actor exists	**Slow ascent**, resumes regular speed after hill	**Faster ascent**, resumes regular speed after hill	***Fastest* ascent**, resumes regular speed after hill

(b) Blind summit scenario

	Underconfident	Optimal	Overconfident
Actor exists	Clears intersection **very slowly**	**Slight speed reduction**, yield at intersection	***No slowdown***, unable to brake to avoid crash
No actor exists	Clears intersection **very slowly**	**Slight speed reduction**, resumes speed	***No slowdown***, quickly clears intersection

(c) Intersection scenario

Figure 1: A decision matrix for each scenario specifying the possible outcomes from rolling out human-like policies. This matrix specifies the total number of modes in the expert dataset (used to train the learned generative trajectory model). The columns specify the behavioral policy (an underconfident, overconfident, or optimal driver), and the rows specify the existence of another actor in the scene.

	Underconfident	Optimal	Overconfident
Actor 1 exists Actor 2 exists	Clears intersection **very slowly**	**Slight speed reduction**, yield at intersection	***No slowdown***, **crashes** into actor 1 or 2
Actor 1 exists Actor 2 does not exist	Clears intersection **very slowly**	**Slight speed reduction**, yield at intersection	***No slowdown***, **crashes** into actor 1
Actor 1 does not exist Actor 2 exists	Clears intersection **very slowly**	**Slight speed reduction**, yield at intersection	***No slowdown***, **crashes** into actor 2
Actor 1 does not exist Actor 2 does not exist	Clears intersection **very slowly**	**Slight speed reduction**, resumes speed	***No slowdown***, quickly clears intersection

Figure 2: Decision matrix for a three-agent variant of the original occluded intersection scenario proposed by Zhang and Fisac [1]. In the three agent variant, the occluded agent on the right of the intersection is mirrored on the left-hand side. Because the game-theoretic planner only supports two-player games, the addition of a second agent travelling in the 'danger zone' of the ego vehicle means the game-theoretic planner no longer works and will default to an FRS planner, resulting in suboptimal conservative planning.

B Graphical representation of AVP

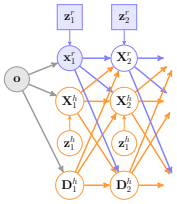

Figure 3: Graphical representation of a plan in AVP applied to a two-agent scenario (a robot r and a human h). AVP is a co-influence model that allows for co-leader planning: optimizing for $\mathbf{z}^r$ plans a policy modelled to influence and be influenced by the stochastic behavior of the human. Importantly, AVP also models the influence of both the human's position $\tilde{\mathbf{X}}^h$ and robot's position $\mathbf{X}^r$ on the human's detection $\mathbf{D}^h$. Modelling this relationship enables planning a future robot position that reduces the robot's uncertainty of the human agent's existence.

C Algorithm for closed-loop control with AVP

Algorithm 1: Receding horizon control with AVP

Require: Trained predictive model M, planning loss criterion $\mathcal{L}$, replanning rate R, fized-size queue Q, P-controller C

$count = 0$
while control is engaged **do**
  if $count = R$ **then**
    # replan
    Collect observed joint trajectory history $\tilde{\mathbf{x}}_{\leq 0}$ from Q
    Collect current perceptual context $\mathbf{i}_0$ from sensor
    Generate optimal trajectory according to the AVP planner using M, $\mathcal{L}$, $\tilde{\mathbf{x}}_{\leq 0}$, and $\mathbf{i}_0$
    $count = 0$
  else
    Apply controls to follow target ego trajectory according to C
  Append current observed joint trajectories to Q
  $count = count + 1$

D Additional model training details

We train a recurrent neural network on a dataset of partially-observed trajectories (positions, if defined) and LiDAR contexts. Positions for all agents in the scene are preprocessed (during training) by shifting and rotating such that the ego is located at (0,0) facing the +X direction. Observed positions are also offset by a small amount of random noise to prevent overfitting. LiDAR point cloud observations are converted to (60,360) range map images to enable simple processing with a convolutional neural network (CNN). The weights of the RNN are optimized to reduce the negative log-likelihood using the Adam optimizer. We use a learning rate of 1e-4 and train for 500 epochs. Our RNN architecture uses a hidden size of 256 with 2 layers, and during training the autoregressive rollouts are generated using teacher forcing. The prediction heads for μ and σ use `tanh` and `exp(tanh)` activations. The context encoder is a CNN with 3 convolutional layers with a kernel size of (3,3) for each layer, 32 filters for the first 2 layers and 8 filters for the final layer (the spatial resolution is flattened after the third convolutional layer using a fully-connected layer).

E Additional data collection details

We collect training data in the CARLA simulator using the `ScenarioRunner` framework. We first specify the initial setup for each scenario by setting the number of vehicles, their terminal destinations, and their spawn points. For vehicles with forking behaviors (i.e., not including non-interactive vehicles such as the truck in the overtake scenario), we script their control code to depend on provided arguments corresponding to the various behaviors described in the decision matrices in Fig. 1. Once the behavioral policies have been tuned to produce the full set of outcomes, we collect training data by running the scenarios in CARLA while logging each vehicle' position and the ego vehicle's LiDAR observation (simulated using Unreal Engine's semantic ray casting). The scenario

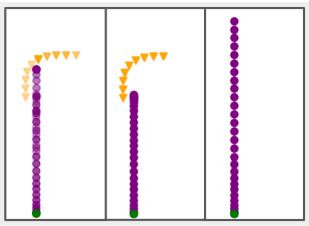

Figure 4: Several visualized planned trajectories from the two-agent intersection scenario. Green indicates a fragment of a past position data (used as input to the model, in addition to perceptual context), purple indicates planned future ego trajectory, and orange indicates predicted sampled non-controllable vehicle positions.

terminates when the ego vehicle reaches its target destination or when a collision occurs. The ground truth vehicle positions are processed into partially observed data by replacing each non-ego vehicles position with a mask token if the vehicle was not detected in the ego's semantic LiDAR point cloud.

F Additional qualitative and quantitative evaluation

F.1 Discussion of additional quantitative metrics for generated plans

Table 1 includes the number of collisions incurred, as well as the average time required to reach the goal (i.e., how fast does the planner complete the objective). We also provide statistics for a lower-bound overly conservative 'safety' policy, which never reaches the goal optimally, but also never results in a crash. We note that only AVP, SOAP, and the safety policy are able to safely complete the two-agent scenarios (SOAP fails on the three-agent scenario). Note that the average time-to-goal is lower for the single-agent and multi-agent baselines because the generated plans are riskier, resulting in frequent crashes. Compared to the safety policy, AVP is able to significantly improve on the average time-to-goal while avoiding additional collisions.

F.2 Videos of generated plans

At the linked website we provide videos illustrating the generated plans for the overly conservative safety policy, AVP, and the overconfident baseline policies (the single-agent and naive multi-agent models). From the videos, we can see that the single-agent and multi-agent baselines generate aggressive plans (right column) that disregard the danger of unseen vehicles interacting with the ego vehicle. On the other hand, the safety policy (left column) is inefficient and far more conservative than most human drivers. AVP (middle column) generates plans that take calculated risks (i.e., contingency plans), resulting in more aggressive driving than the safety policy, without ever allowing the ego vehicle to trigger an inescapable collision with an unseen vehicle.

F.3 Visuals of generated plans at key decision points

Figure 4 visualizes several planned trajectories from the two-agent intersection scenario. All three are candidate trajectories that the AVP imitative model learns to predict using the collected driving data (note that AVP learns to predict another driver, in orange, emerging at a future timestep). Optimizing our planning equation with a collision penalty loss leads to AVP planning to execute the middle trajectory, which balances safety (allowing enough time to yield) while maintaining a reasonable speed. On the other hand, the single-agent and multi-agent baselines are only able to predict the right trajectory: in the single-agent case, the model ignores non-ego vehicles, and for the multi-agent case, because the initial timestep does not have a non-ego agent detected, no future positions for non-ego vehicles are predicted. In both cases, the baseline planners will choose to execute the right trajectory, which is dangerous and leads to a crash 50% of the time.

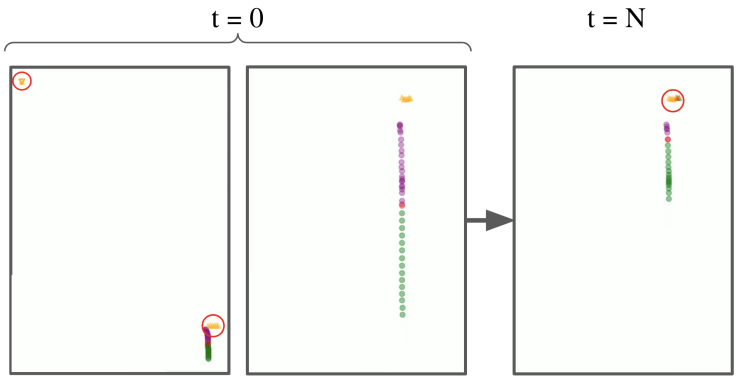

Figure 5: Visualizing a batch of predictions from a trained model at two different timesteps (before and after a non-ego vehicle is detected). Green indicates observed past position data (red indicates observed present position), purple indicates predicted ego positions, and orange indicates predicted non-ego vehicle positions. In the left two panels (timestep $t = 0$, before any non-ego vehicle is observed), the model generates predictions where a vehicle is detected in the future (i.e., an unobserved vehicle exists), and predictions where a vehicle is not detected. The leftmost panel is a zoomed-out version of the middle panel, illustrating the undefined (mask token) position outputs when the predicted detection is 0 (the arbitrary mask outputs are ignored). In the right panel ($t = N$, after a non-ego vehicle has been detected), the model exclusively generates predictions where a vehicle is detected in the future, and there are no undetected outputs (visualized with the mask tokens).

F.4 Visuals of predicted detection at inflection points

In Fig. 5 we illustrate how predicted detection changes once non-ego vehicles have been detected. Prior to the ego detecting any vehicles, AVP will predict a balance of futures with non-ego agents existing (i.e., becoming detected in the future) and non-existing - the precise balance is determined by the distribution in the training data. However, once another vehicle has been detected by the ego, AVP will collapse its belief and consistently predict (correctly) that another vehicle exists, up until the vehicle exits the ego's sensor range.

References

[1] Z. Zhang and J. F. Fisac. Safe occlusion-aware autonomous driving via game-theoretic active perception. In *RSS*, 2021. 1