Connected Autonomous Vehicle Motion Planning with Video Predictions from Smart, Self-Supervised Infrastructure

Jiankai Sun[1], Shreyas Kousik[2], David Fridovich-Keil[3], and Mac Schwager[1]

***Abstract*— Connected autonomous vehicles (CAVs) promise to enhance safety, efficiency, and sustainability in urban transportation. However, this is contingent upon a CAV correctly predicting the motion of surrounding agents and planning its own motion safely. Doing so is challenging in complex urban environments due to frequent occlusions and interactions among many agents. One solution is to leverage smart infrastructure to augment a CAV's situational awareness; the present work leverages a recently-proposed "Self-Supervised Traffic Advisor" (SSTA) framework of smart sensors that teach themselves to generate and broadcast useful video predictions of road users. In this work, SSTA predictions are modified to predict future occupancy instead of raw video, which reduces the data footprint of broadcast predictions. The resulting predictions are used within a planning framework, demonstrating that this design can effectively aid CAV motion planning. A variety of numerical experiments study the key factors that make SSTA outputs useful for practical CAV planning in crowded urban environments.**

I. Introduction

Connected and autonomous vehicles (CAVs) have the potential to make urban transportation safer, more efficient, and more sustainable. A key challenge towards realizing this goal is that of motion planning; traditional methods rely on static maps and real-time sensor data, and cannot always yield accurate predictions of the future state of the environment due to occlusions or other onboard sensing limitations. Hence, smart infrastructure has been proposed as a way to augment CAV capabilities. However, collecting data to train such smart infrastructure can often require prohibitive amounts of manual labeling. Therefore, there is growing interest in using self-training smart infrastructure to improve CAV safety and performance.

The present work considers a specific type of edge device proposed in our prior work: the Self-Supervised Traffic Advisor (SSTA) [1]. This proposed device automatically processes video data from traffic cameras to support CAV motion planning. SSTAs are a scalable edge solution that avoid extensive hand-labeling and can provide large-scale connected coverage of an urban environment through self-training to share data. Our previous work studied the challenges of self-training and networking, whereas this work assesses the potential for SSTAs to benefit CAV motion planning and control.

Contributions: First, we develop a new output format for SSTAs as an alternative to predicting raw video frames. Specifically, we propose to predict Time to Next Occupancy (T2NO) and Time to Next Departure (T2ND), which describe when each portion of an SSTA's field of view will next be occupied and subsequently unoccupied by a road user. We abbreviate these together as "T2NO/D." Second, we propose an approach to CAV motion planning that leverages our novel T2NO/D outputs from SSTAs. Third, we numerically evaluate the effectiveness of autonomously predicting T2NO/D in comparison to raw video within a CAV motion planning framework. We find that our proposed method can provide an effective smart infrastructure perception output that is directly amenable to CAV planning and control.

[1]Stanford University, [2]Georgia Institute of Technology, [3]University of Texas at Austin. Corresponding author: jksun@stanford.edu.

II. Related Work

SSTAs [1], and this paper, lie at the intersection of CAV infrastructure, networked learning, and video prediction.

a) CAV Infrastructure: To maximize their utility, CAVs should communicate with self-driving and human-driven vehicles, roadside infrastructure, and other road users; many reviews of this literature are available [2]–[5]. Given bandwidth and connectivity limitations, it is critical to decide what, and when, to communicate. For example, Gopalswamy et al. [6] propose to offload CAV responsibilities for situational awareness and information sharing via vehicle-to-infrastructure (V2I) communication. Alternatively, short-range communications allow vehicle-to-vehicle (V2V) transmission of information such as speed, heading, and brake status [7], [8]. Since V2V alone can cause network-wide instabilities as the number of vehicles grows [9], it is critical to explore alternative means of widespread, scalable communication that can improve large-scale traffic metrics such as safety and efficiency [10]. In this work, we propose a scalable infrastructure solution and study its potential for aiding CAVs in an isolated V2I manner; we plan to explore mixing V2V and V2I in future work.

b) Networked Learning: The concept of networked learning has been recently explored in the context of connected and autonomous vehicles (CAVs) [11]. In this paradigm, multiple CAVs can cooperate to improve their individual driving policies through message passing and sharing of information [12]–[14]. We note that CAVs can cooperate to perceive their surroundings [13] and to improve traffic congestion and energy efficiency [15]–[17]. In this work, we instead consider how networked learning in *infrastructure* can be used to improve perception and planning for CAVs.

c) Video Prediction: Video prediction has the potential to impact traffic management, motion planning for autonomous vehicles, and autonomous surveillance [18].

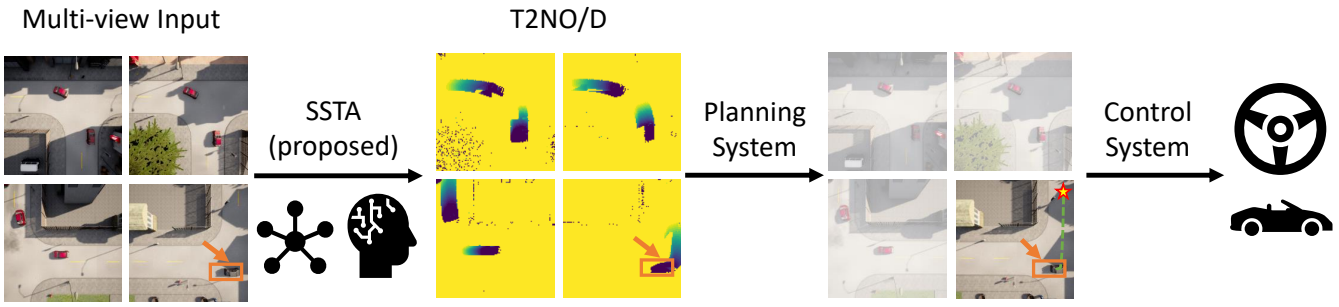


Fig. 1: Our proposed system is a distributed Neural Perception System comprised of a network of static traffic cameras paired with their own neural networks. The location of the ego car is marked by the orange box. The networks predict future pixel-wise occupancy and departure times in each view, which is our first contribution. A planning system then generates a path based on collision checking in the pixel-wise predictions, which is our second contribution. Finally, the path plan is handed to a CAV's control system to track.

This task is convenient for data collection because it can often be achieved without manual data annotation. Deep learning methods can be especially well suited for modeling spatiotemporal relationships in video and generating accurate predictions of future events [19]–[22]. However, most previous video prediction is typically focused on a single camera, aiming to learn visual dynamics [18] or interpolate video spatially and temporally [23]. Our prior work on SSTAs [1] instead addressed the challenge of video prediction across a network of *multiple* static traffic cameras. The key remaining challenge addressed in the present work is that, from a V2I perspective, predicting raw video is not directly amenable to downstream CAV planning and control tasks.

III. Method

In this section, we describe our system architecture (cf. Fig. 1) and algorithms used for video prediction and motion planning. First, we provide an overview of our proposed method. Next, we introduce the Time to Next Occupancy (T2NO) and Time to Next Departure (T2ND) to predict future road user trajectories. We then explain our training loss and neural network architecture used at each SSTA. Finally, we design a collision-checking technique for T2NO/D and utilize it for downstream planning and control.

Remark 1. *In this work, we consider planning only for a single CAV, which we henceforth refer to as the* ego vehicle. *We defer multi-agent SSTA-enabled planning for future work.*

A. Overview of Proposed Method

Our overall goal is CAV motion planning using video predictions generated by SSTAs (Fig. 1). Motion planning requires collision checking, but performing collision checking directly in raw video predictions is challenging. Instead, we propose predicting future occupancy in each pixel of each SSTA view. Our method has three steps: (1) calculating future occupancy using Alg. 1, (2) collision checking pixels occupied by the ego vehicle against pixels with predicted future occupancy, and (3) performing motion planning by modifying the A* algorithm [24] to generate an optimal path that avoids intersecting with obstacles in space and time.

B. Time to Next Occupancy and Time to Next Departure

We leverage the fact that each SSTA is static to capture and use background information. This enables us to compute two quantities that simplify collision checking: Time to Next Occupancy and Time to Next Departure.

Before defining these quantities, we establish some notation, all within the context of a single SSTA. Note, to ease notation, we drop the index i for considering the i^{th} SSTA in this discussion. Suppose the current time is $t = 0$. Let $\mathbb{N}_{256} = \{0, 1, \cdots, 255\}$. Consider a sequence of predicted images $(I_t)_{t=0}^{T}$ with each image in $\mathbb{N}_{256}^{H\times W\times 3}$ (i.e., height H, width W, and RGB channel of dimension 3), and $T \in \mathbb{N}$ as the prediction horizon. Also suppose we also have a *background image* $B \in \mathbb{N}_{256}^{H\times W\times 3}$; in practice we create B for each SSTA by pixel-wise averaging of approximately one minute of video data. Denote indexing into the $(i, j)^{\text{th}}$ pixel of an image I by $I[i, j]$. Let $\mathbb{N}_n$ denote the set $\{0, \cdots, n\}$.

Definition 2. *The* Time to Next Occupancy *(T2NO) is the first time at which a given pixel in an SSTA's image is predicted to be occupied, which we assess by comparison against the background B. We represent T2NO as an array $O \in \mathbb{R}^{H\times W}$.*

On its own, T2NO does not capture all relevant dynamic information in a scene, because, once a pixel is declared occupied at a time t, it is implied to be occupied for all future time $t' > t$, leading to the following definition that alleviates this concern:

Definition 3. *The* Time to Next Departure *(T2ND) measures the first time at which an occupied pixel is no longer occupied. Similar to T2NO, we represent T2ND as an array $D \in \mathbb{R}^{H\times W}$.*

We compute both T2NO and T2ND using Alg. 1. By combining T2NO and T2ND, we can assess when each pixel in an SSTA's field of view will be occupied and then free within the prediction horizon T.

As an example to clarify these concepts, consider a single car passing through pixel (i, j) at $10\ \mathrm{m/s}$, and suppose every pixel corresponds to a $1 \times 1\ \mathrm{m/s^2}$ patch of drivable area.

Algorithm 1: Time to Next Occupancy / Departure

Input: prediction $(I_t)_{t=0}^T$, background B, occupancy threshold τ_O, departure threshold τ_D

$O, D \leftarrow +\infty_{H \times W}$ // preallocate T2NO and T2ND
for $(i,j) \in \mathbb{N}_h \times \mathbb{N}_w$ *(i.e., iterate over each pixel)* **do**
  // first, compute time to next occupancy
  for $t = 0, 1, \cdots, T$ **do**
    $\delta \leftarrow |I_t[i,j] - B[i,j]|$ // error between prediction and background at pixel (i,j)
    if $\delta \geq \tau_O$ **then**
      $O[i,j] \leftarrow t$ // get smallest t for which prediction differs from background
      break
  // second, compute time to next departure
  if $O[i,j] < +\infty$ *(i.e., if there is any future occupancy at the current pixel)* **then**
    for $t = O[i,j], O[i,j]+1, \cdots, T$ *(i.e., for all times after the time to next occupancy)* **do**
      $\delta \leftarrow |I_t[i,j] - B[i,j]|$
      if $\delta \leq \tau_D$ **then**
        $D[i,j] \leftarrow t$ // get smallest t for which prediction is close to background
        break

Output: O (T2NO) and D (T2ND)

Suppose the T2NO is 2.0 s, indicating the earliest time at which the car occupies that pixel. Suppose the car is 5 m long. Then the car will completely pass over the pixel in 0.5 s, meaning that the T2ND at that pixel must be no less than 2.5 s. An adjacent pixel in the direction of travel will have $O[i,j] = 2.1$ s and $D[i,j] \geq 2.6$ s (assuming the car is predicted to maintain its speed and heading).

C. Training to Predict T2NO/D

We follow the networked co-learning procedure outlined in [1, Sec. III.D]. However, we preprocess each i^{th} SSTA's recorded images to generate O_t^i and D_t^i as output by Alg. 1 for each time step t of the training data (i.e. past video frames). Denote the T2NO/D predictions of the i^{th} SSTA at time $\hat{t}$ as $\hat{O}_{\hat{t}}^i$ and $\hat{D}_{\hat{t}}^i$, starting from time t up to a time $t+T$. We minimize the mean-squared error loss between predicted and ground-truth T2NO/D:

$$\mathcal{L}_t = \sum_{i=1}^{N} \left(\sum_{\hat{t}=t}^{t+T} \left(\alpha \left\| O_{\hat{t}}^i - \hat{O}_{\hat{t}}^i \right\|^2 + \beta \left\| D_{\hat{t}}^i - \hat{D}_{\hat{t}}^i \right\|^2 \right) \right), \quad (1)$$

where $\alpha, \beta > 0$ are tunable hyperparameters, and we use Frobenius norms (i.e., pixelwise error squared). Note, the predicted $\hat{O}_{\hat{t}}^i$ and $\hat{D}_{\hat{t}}^i$ are functions of the i^{th} SSTA's hidden state $h_{\hat{t}}^i$, the set of *received* messages $Y_{\hat{t}}^i$ from its connected SSTAs, and its current received video frame $I_{\hat{t}}^i$.

D. Proposed Neural Network Architecture

We use the same architecture as in [1, Sec. III.F]. However, instead of requiring a recursive rollout to generate T2NO/D predictions, we generate the prediction with a single forward pass of the network at each SSTA. So, the outputs of the i^{th} network at timestep $\hat{t}$ are $\hat{O}_{\hat{t}}^i$, $\hat{D}_{\hat{t}}^i$, and the message $y_{\hat{t}}^i$.

Remark 4. *The method proposed in this work reduces the number of neural network evaluations at each SSTA from a T-step recursive rollout to just a single evaluation, since T2NO/D collapse temporal information into a single image.*

E. Collision Checking with T2NO/D

We seek to use SSTA predictions for collision checking within a CAV motion planning framework. First, we explain how to isolate pixels associated with the ego vehicle. Then, we consider collision checking against T2NO/D.

1) Masking the Ego Vehicle's Pixels: Recall that each SSTA outputs a sequence of images $(I_t)_{t=0}^T$, with each $I_t \in \mathbb{N}_{256}^{h \times w \times 3}$. We now introduce an approach to extracting the pixels in an image corresponding to our ego vehicle's pose (2-D position and 1-D heading), which we denote (p_t, θ_t) at time t. Suppose the vehicle has a rectangular body of dimensions $l \times d$. Also suppose that each pixel, with indices (i,j), is associated with a corresponding spatial location $p_{i,j}$, which can be found with the following map:

$$\texttt{sp2px} : p_{i,j} \mapsto (i,j) \quad (2)$$

Since each SSTA is a static camera with a known field of view, it is straightforward to create this map via standard camera calibration techniques. Also, note that we can represent the vehicle's body at the state (p, θ) as an array of vertices arranged counterclockwise:

$$V_t \leftarrow \left(\tfrac{1}{2} \begin{bmatrix} \cos\theta_t & -\sin\theta_t \\ \sin\theta_t & \cos\theta_t \end{bmatrix} \begin{bmatrix} l & -l & -l & l \\ d & d & -d & -d \end{bmatrix} \right) + p_t \quad (3)$$

Recall that our images I_t are of dimension $h \times w \times 3$. We thus create a mask for the given vehicle pose:

$$P_t \leftarrow \texttt{sp2px}(V_t) \quad (4)$$

$$P_t \leftarrow \texttt{fillPoly}(\mathbf{0}_{h \times w \times 3},\ P_t), \quad (5)$$

$$M_t \leftarrow P_t == (255, 255, 255), \quad (6)$$

where P_t is used as an intermediate array. Equation (5) fills in a polygon defined by the (pixel coordinate) vertices V_M, into a black image ($\mathbf{0}_{h \times w \times 3}$), so $P_t \in \mathbb{N}_{256}^{H \times W \times 3}$. Note that `fillPoly` is available in OpenCV [25], and we assume the default fill color is white, or $(255, 255, 255)$. Equation (6) creates the mask $M_t \in \{0,1\}^{H \times W}$ (i.e., a Boolean array) which is `true` for every entry (i.e., pixel) that is white, thereby identifying the pixels corresponding to our vehicle.

2) Collision Checking Against T2NO/D: Suppose we have the T2NO/D arrays $O \in \mathbb{R}^{H \times W}$ and $D \in \mathbb{R}^{h \times w}$ output from Alg. 1. For each time step t, we check:

$$C_t \leftarrow t \cdot M_t \quad (7)$$

$$\texttt{chk}_t \leftarrow \texttt{any}\,(C_t \geq O\ \&\ M_t \leq D) \quad (8)$$

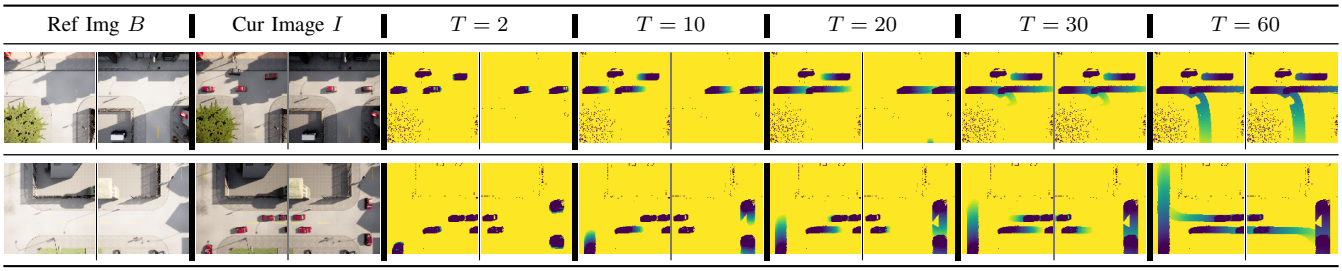


Fig. 2: Visualization of T2NO for different time horizons (T2ND is omitted for ease of understanding). The color scale runs from black at $t = 0$ to yellow as $t \to \infty$. The background image B and the current image I are also shown on the left. We see that T2NO contains little time information when the time horizon is short, whereas T2NO contains ambiguous information when the time horizon is long, and different trajectories even overlap, impeding the effectiveness of CAV motion planning. Even though there are multiple plausible paths for cars to take in the future, depending on the road topology, this ambiguity is not desired and hinders planning.

Equation (7) converts the Boolean mask M_t to a real-valued array, where entries for which M_t is `true` are set to t, and all others are 0. Then, we directly check the entries of $M_t \in \mathbb{R}^{H \times W}$ against O and D. If any of the vehicle's occupied pixels (i.e., `true` pixels) overlap with the time window created by O and D, then $\texttt{chk}_t$ is `true` (i.e., the pose (p_t, θ_t) is in a collision).

F. Planning with T2NO/D

After obtaining T2NO/D, we plan a path for our ego vehicle. In this work, we extend the classical A* algorithm [24] to leverage T2NO/D, but we note that these representations can be readily applied to other planning algorithms.

To avoid collisions with dynamic obstacles, we modify A* to generate a time-optimal path that avoids intersecting with the obstacles in both time and space. Suppose the ego vehicle is at a time and position $z_0 = (0, p_0) \in [0, T] \times \mathbb{R}^2$. Our implementation finds a list of times and positions $z_i = (t_i, p_i)$, where $i \in \mathbb{N}$, which define a trajectory for the ego vehicle to track. We seek to reach a user-specified goal position p_{goal} in the shortest time while avoiding other vehicles. As per the classical A* notation, we use an evaluation function to represent running cost:

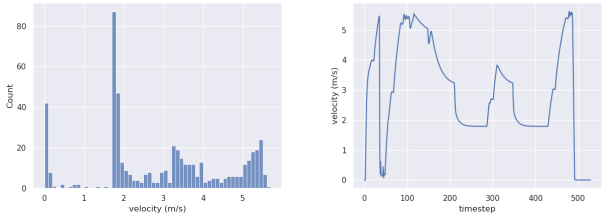
$$f(z_i) = c(z_i) + r_t(z_i) \tag{9}$$

where $c(z_i)$ is the cost of the path from z_0 to z_i and r_t is a time-varying heuristic function that estimates the cost of the most efficient path from p_i to p_{goal} that avoids dynamic obstacles. In particular, we set

$$r_t(z_i) = \|p_i - p_{\text{goal}}\| + K \cdot \texttt{chk}_{t_i},$$

where $K > 0$ is a scalar that penalizes being in collision at time t_i, and $\texttt{chk}_{t_i}$ is the collision check evaluated at time t_i as per (8) (and `true` $= 1$). The final output of A* is a discrete sequence of times and positions. We perform cubic spline interpolation to smooth the sequence before handing it to the ego vehicle to track (in our implementation, the ego vehicle uses a default PD controller from the CARLA simulator [26]).

Next, we evaluate our proposed technical approach via numerical experiments.



(a) Histogram of Ego Vehicle Speed (b) Speed Over Time

Fig. 3: **Ego Car Speed Statistics**

IV. Experiments

We now assess the utility of SSTAs and our proposed T2NO/D output format via a series of numerical experiments. We find that SSTAs can generate predictions which are effective for CAV motion planning.

A. Experiment Setup

1) Implementation Details: We evaluate our approach on a desktop computer with an NVIDIA GeForce RTX 2080Ti and an Intel Core i7-6800K CPU. We design the traffic scenario and evaluate our pipeline with the CARLA 0.9.11 simulator. We train our networks using the ADAM optimizer [27] and employ a mini-batch size of 10 sequences. The learning rate is set to 10^{-3}, and training is stopped after 800 epochs.

2) SSTAs: In each simulated traffic scenario, SSTAs are placed at each intersection and record top-down images. Each SSTA communicates with its neighbors at adjacent intersections, according to the topology of the road graph. Following prior work [1], SSTAs are trained to predict traffic flow (see Fig. 2) using a corpus of simulated traffic scenarios offline. Note that in contrast our prior work explored the effects of online and lifelong learning.

3) Ego Vehicle: We perform planning for an ego vehicle as per Sec. III-F. The overall task is to avoid other vehicles while attempting to reach sub-goals or a global goal location. To ease identification of the ego vehicle, we set its color to grey, and all other vehicles to red for visualization. The outcomes are presented in Tab. I. Since T2NO/D is computed

TABLE I: Performance in 4-view setting over 20 trials (mean $\pm$ standard deviation).

Metrics	Success Rate (%) ↑	Number of Collisions ↓	Total Timesteps to Reach Goals ↓	Total Control Effort ↓	Total Sudden Reversals ↓	Travel Distance (km) ↑
MONOLITH	**65.0**	$15.15 \pm \mathbf{0.86}$	$\mathbf{484.15} \pm 6.72$	$\mathbf{245.49} \pm 5.91$	66.76 ± 1.36	$\mathbf{1.21 \pm 0.34}$
ALLCOMMS	60.0	$\mathbf{15.00} \pm 4.63$	495.50 ± 6.76	$251.47 \pm \mathbf{4.01}$	$\mathbf{66.58 \pm 1.25}$	1.12 ± 0.35
NOCOMMS	50.0	17.50 ± 2.29	$508.40 \pm \mathbf{5.58}$	260.28 ± 4.33	70.50 ± 2.94	1.04 ± 0.38

based on background subtraction, the actual vehicle color does not impact our proposed method's performance, which is an improvement over our prior work.

The ego vehicle is initialized with the same start and goal locations in each experiment, but the traffic pattern is randomly generated each time. An experiment is terminated once the first vehicle reaches its goal location, and we record this time t^* as well as the corresponding distance between each other vehicle and its goal as $\bar{d}$.

The ego vehicle utilizes a replanning frequency of 20 Hz (i.e., it plans in a receding-horizon manner). This necessitates the SSTAs to update their prediction outputs at the same frequency. The maximum speed of the ego vehicle is set to the CARLA default of 8.33 m/s. Please refer to Fig. 3 for a histogram plot displaying the actual speeds achieved by the vehicle in an experiment.

4) Metrics: We evaluate our approach on safety, efficiency, and smoothness with the following metrics:

- *Success Rate* is the percentage of experiments in which the ego car is able to reach the global goal position.
- *Number of Collisions* is how many times the ego car collides with other vehicles or obstacles before reaching its global goal. Note that in our experiments, successful navigation may entail encountering multiple non-fatal collisions to simulate a realistic and challenging environment, consistent with the default setting in CARLA [26]. This approach aims to emulate real-world conditions where occasional collisions may happen without compromising the overall navigation capabilities of the autonomous vehicle.
- *Travel Distance* is the total distance that the ego car is able to travel before experiencing a collision or reaching its goal.
- *Total Timesteps to Reach Goal* measures the duration taken to reach the goal, if it was reached.
- *Total Control Effort* is the sum of the norm of the ego vehicle's 2-D acceleration over the entire experiment. That is, if the vehicle experiences an acceleration $\ddot{x}_t$ at time t, then this metric is $\sum_{t=0}^{t_{\text{final}}} \|\ddot{x}_t\|$, where t_{final} is the final time in the given experiment (i.e., when the vehicle reached the goal or crashed)
- *Total Sudden Reversals* counts how often the vehicle's longitudinal and lateral acceleration changes sign over the course of an experiment.

Note, we only count successful trials (i.e., goal reached) for all metrics except for Success Rate and Travel Distance. This is because failed trials result in smaller Total Control Effort and Total Sudden Reversals due to shorter distances, but this does not mean that failed trials are better than successes. Also, there is no Total Timesteps to Reach Goal for unsuccessful trials.

B. Independent Variables and Hypotheses

1) Influence of Communication: We aim to characterize the effects of SSTA communication upon traffic flow using the metrics above. As such, we compare the performance of our method using the following communication arrangements:

- MONOLITH, in which a single model is trained on all images concatenated together, with no need to communicate, as opposed to multiple models trained at each location and requiring communication.
- ALLCOMMS, in which each SSTA can communicate with *all* others both during training and at run time.
- NOCOMMS, in which SSTAs are trained completely independently and cannot communicate with one another either at training time or run time.

Note, as a control on the experiment, we also evaluated a NOREPLAN baseline, in which agents only receive the first SSTA prediction and hence never replan. This baseline was never able to reach the goal, as expected, and is therefore not included in the results below. *Hypothesis:* performance will increase as communication increases, with the MONOLITH framework performing best.

C. Results and Discussion

1) Influence of Communication: From the results shown in Tab. I, our framework can effectively guide the vehicles to reach the goal for 4-view scenarios. The NOCOMMS and MONOLITH baselines provide lower and upper limits for SSTA performance, respectively. We see that, by communicating with adjacent units, SSTAs' predictions still capture most of the useful patterns in traffic flow; the performance of our approach is only slightly worse than that of MONOLITH, and substantially superior to that of NOCOMMS.

2) Failure Cases: In Fig. 4, we show some examples of the predicted results and corresponding planned path using our pipeline. We find that, for NOCOMMS, the prediction is not accurate enough, which causes two failure modes: (1) misjudging the distance to the vehicle ahead of the ego car, resulting in an unnecessary braking maneuver, and (2) failing to predict the presence of an obstacle and causing a collision.

3) Practical Considerations: In addition, to assess the conservativeness of our method, we analyze the vehicle speeds for all trajectories in a histogram and a velocity profile, both shown in Fig. 3. The vehicle spends a substantial time traveling at a variety of speeds, as expected in dense urban traffic, and is not excessively conservative.

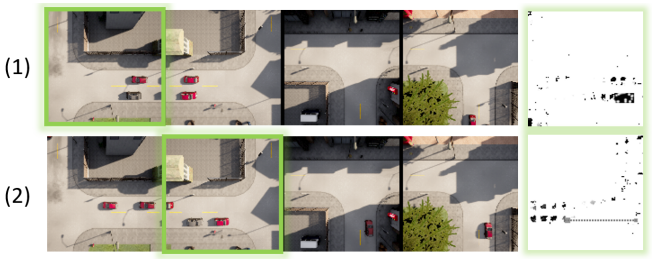


Fig. 4: **Failure Cases**: two failure modes: (1) misjudging the distance to the vehicle ahead of the ego car, resulting in an unnecessary braking maneuver, and (2) failing to predict the presence of an obstacle and causing a collision.

4) Main Takeaways: Overall, our experiments confirm the utility of SSTAs for aiding in CAV motion planning. We note that these experiments relied entirely on the SSTA off-board sensor for ego-motion planning. Hence, we see much more collisions than would be expected for a safe, practical CAV. In a real-world deployment, we would prescribe SSTA outputs as an *augmentation* of CAV sensor data; to isolate the benefits of SSTAs, we ignore all onboard ego vehicle sensors and computation for our experiments.

V. Conclusion

This paper further develops SSTAs [1] by introducing a learning framework for self-supervised traffic prediction paired with planning vehicle motion in a smart city. Our experimental results indicate that the proposed framework makes progress towards three goals: (1) by communicating with one another, SSTAs can make more accurate forecasts of traffic flow, (2) CAVs can use SSTA predictions outputs for downstream motion planning and control, and (3) this pipeline can improve traffic metrics such as efficiency and safety. Of course, limitations remain; future work will improve prediction quality, especially in the transition region between different views, seek to detect anomalous data (i.e., unpredictable traffic events), and explore notions of road user privacy. Our work represents a step forward in the development of CAV motion planning algorithms in partnership with smart infrastructure, with the long-term goal of creating safer, more efficient, and more sustainable urban transportation systems.

Acknowledgment

Toyota Research Institute provided funds to support this work.

References

[1] J. Sun, S. Kousik, D. Fridovich-Keil, and M. Schwager, "Self-supervised traffic advisors: Distributed, multi-view traffic prediction for smart cities," *IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 2022.

[2] M. M. Rana and K. Hossain, "Connected and autonomous vehicles and infrastructures: A literature review," *International Journal of Pavement Research and Technology*, pp. 1–21, 2021.

[3] J. Guanetti, Y. Kim, and F. Borrelli, "Control of connected and automated vehicles: State of the art and future challenges," *Annual reviews in control*, vol. 45, pp. 18–40, 2018.

[4] J. Huang, S. Xie, J. Sun, *et al.*, "Learning a decision module by imitating driver's control behaviors," in *CoRL*, ser. Proceedings of Machine Learning Research, vol. 155, PMLR, 16–18 Nov 2021, pp. 1–10.

[5] J. Sun, H. Sun, T. Han, and B. Zhou, "Neuro-symbolic program search for autonomous driving decision module design," in *CoRL*, ser. Proceedings of Machine Learning Research, vol. 155, PMLR, 16–18 Nov 2021, pp. 21–30.

[6] S. Gopalswamy and S. Rathinam, "Infrastructure enabled autonomy: A distributed intelligence architecture for autonomous vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 986–992.

[7] J. He, H.-H. Chen, T. M. Chen, and W. Cheng, "Adaptive congestion control for dsrc vehicle networks," *IEEE communications letters*, vol. 14, no. 2, pp. 127–129, 2010.

[8] J. He, Z. Tang, X. Fu, *et al.*, "Cooperative connected autonomous vehicles (cav): Research, applications and challenges," in *2019 IEEE 27th International Conference on Network Protocols (ICNP)*, 2019, pp. 1–6.

[9] Y. Zheng, S. E. Li, J. Wang, D. Cao, and K. Li, "Stability and scalability of homogeneous vehicular platoon: Study on the influence of information flow topologies," *IEEE Transactions on intelligent transportation systems*, vol. 17, no. 1, pp. 14–26, 2015.

[10] J. Hou, G. Chen, J. Huang, *et al.*, "Large-scale vehicle platooning: Advances and challenges in scheduling and planning techniques," *Engineering*, 2023.

[11] V. P. Chellapandi, L. Yuan, S. H. Zak, and Z. Wang, "A survey of federated learning for connected and automated vehicles," *arXiv preprint arXiv:2303.10677*, 2023.

[12] C. He, S. Li, J. So, *et al.*, "Fedml: A research library and benchmark for federated machine learning," *preprint arXiv:2007.13518*, 2020.

[13] J. Yu, J. Vincent, and M. Schwager, "Dinno: Distributed neural network optimization for multi-robot collaborative learning," *RA-L*, 2022.

[14] I. Bistritz, A. Mann, and N. Bambos, "Distributed distillation for on-device learning," *NeurIPS*, vol. 33, pp. 22 593–22 604, 2020.

[15] J. Yan, M. Xiong, and L. Zhang, "A cooperative control strategy of connected and autonomous vehicles in ramp merging areas with mixed-autonomy traffic," in *TOCS*, 2021, pp. 214–218.

[16] H. Shi, Y. Zhou, K. Wu, X. Wang, Y. Lin, and B. Ran, "Connected automated vehicle cooperative control with a deep reinforcement learning approach in a mixed traffic environment," *Transportation Research Part C: Emerging Technologies*, vol. 133, p. 103 421, 2021.

[17] A. Kuchiki and T. Namerikawa, "Cooperative control of connected and automated vehicles at signal-free intersections considering passenger comfort," in *SICE*, 2022, pp. 909–914.

[18] J. Walker, C. Doersch, A. Gupta, and M. Hebert, "An uncertain future: Forecasting from static images using variational autoencoders," in *ECCV*, Springer, 2016, pp. 835–851.

[19] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," *preprint arXiv:1511.05440*, 2015.

[20] M. Babaeizadeh, C. Finn, D. Erhan, R. H. Campbell, and S. Levine, "Stochastic variational video prediction," in *ICLR*, 2018.

[21] S. Vyas, Y. S. Rawat, and M. Shah, "Multi-view action recognition using cross-view video prediction," in *ECCV*, Springer, 2020, pp. 427–444.

[22] B. Pan, J. Sun, H. Y. T. Leung, A. Andonian, and B. Zhou, "Cross-view semantic segmentation for sensing surroundings," *RA-L*, vol. 5, no. 3, pp. 4867–4873, 2020.

[23] C. Lu, M. Hirsch, and B. Scholkopf, "Flexible spatio-temporal networks for video prediction," in *CVPR*, 2017, pp. 6523–6531.

[24] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[25] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[26] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *CoRL*, PMLR, 2017, pp. 1–16.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, Y. Bengio and Y. LeCun, Eds., 2015.