Drive with the Flow

Enrico Mannocci Matteo Poggi Stefano Mattoccia
University of Bologna

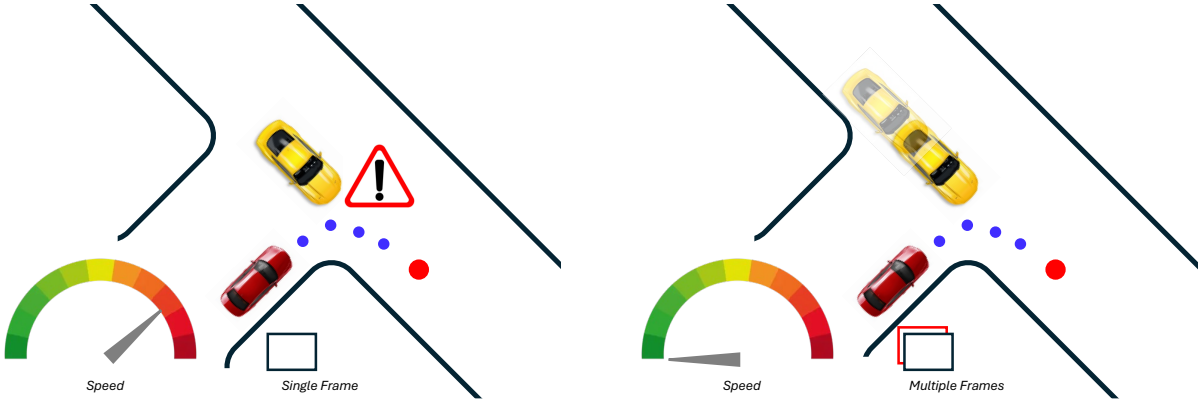

Fig. 1. **Enriched perception with explicit temporal dynamics.** Existing end-to-end autonomous driving frameworks ignore temporal information within the perception module, as it only processes a single frame at once, dampening the collision avoidance capability of the model. By explicitly modeling such cue, the model can better understand the dynamics of other agents such as cars and pedestrians, and better avoid collisions with them.

***Abstract*— End-to-end autonomous driving systems have recently made rapid progress, thanks to simulators such as CARLA. They can drive without infraction of common driving rules on uncongested roads but are still struggling with dense traffic scenarios. We conjecture that this occurs because it lacks understanding of the dynamics of the surrounding vehicles, caused by the absence of explicit short-term memory within the perception path of end-to-end models. To address this challenge, we revise the perception module to explicitly model temporal information, by extending it with an auxiliary task that is well-known in computer vision research: optical flow. We generate a novel benchmark using the CARLA simulator to train our model, FlowFuser, and prove its superior ability to avoid collisions with other agents on the road.**

I. Introduction

Achieving end-to-end autonomous driving has been one of the most ambitious goals pursued by the research community since the last decade. To this end, a suite of sensors ranging from color cameras to LiDARs are used to perceive the surrounding environment, whereas a neural network is deployed to predict direct vehicle control, thus casting *perception* and *planning* in a full-stack manner [2], [4], [23], [28].

The latter task was tackled using two main approaches: imitating an expert (Imitation Learning) [3], [14], [15], [24] or completely data-driven mainly using Reinforcement Learning [13], [29]. Independently on the approach, planning represents the ultimate goal for any of these frameworks, thus attracting most of the community's efforts regarding evaluation paradigms and network designs. For instance, from a testing perspective, planning performance can be measured both in an open-loop or closed-loop manner. The former protocol compares the decisions of the trained model with those of an expert driver: these are possibly encoded within the driving routes followed to collect any driving datasets, both real or synthetic. Although scalable, this protocol biases the evaluation to fewer possible conditions occurring during driving. The latter, instead, consists of running the trained model inside a simulation engine [9], where it can draw its own driving path, and external agents can react according to the specific decisions made by the model. While it allows for more comprehensive testing, this protocol is hard to implement with real data, thus making the evaluated model sensitive to synthetic to real domain shifts.

For what concerns network designs, the common trend consists of identifying meaningful representations to model the perceived environments – e.g., Bird-Eye View (BEV) maps [18], [19], [24] – that are then processed by a specific network module in charge of predicting car controls, for instance, to reach the next waypoint along the driving path. As this latter embodies the planning strategy of the whole model, most of the focus of recent, state-of-the-art solutions has been on improving it [18], [19] while taking perception for granted. However, we argue that the design of the perception component of an autonomous driving model plays a fundamental role. Indeed, the ability to extract richer cues from the surrounding environment has the potential to improve planning itself, even in the absence of further architectural modifications. Specifically, the temporal dimension has been ignored so far by the perception modules of most existing solutions [18], [19], [24]. Whereas some of them take this into account directly in the planner [16], [24], by embedding it in the hidden state of a recurrent neural network, we

argue that explicit modeling of temporal dynamics within the perception part has the potential to deliver vital information – e.g., the motion of vehicles or pedestrians in front of us – to further improve the planning performance, as well as the overall explainability of the system.

In this paper, we take a step back and focus on designing a perception module capable of explicitly processing temporal information. Specifically, we extend the input stem to process multiple, consecutive frames to perform optical flow estimation [33] as an auxiliary task within the perception head. This will provide the downstream planning module with richer features encoding *explicit* short-term memory, crucial to better understanding the behavior of other, independent agents moving in the surrounding environment – such as other vehicles and pedestrians, for instance. This knowledge, although cheap to obtain from images, combined with the *implicit* memory modeled by the recurrent structure of the planning module will allow for predicting safer trajectories and reducing collisions with moving agents, as shown in Figure 1. To prove the effectiveness of our proposal, we extend the perception module of TransFuser [16], [24] – a well-known and widely used baseline on the CARLA autonomous driving benchmark – implementing FlowFuser.

Furthermore, training our model requires additional, optical flow annotations that are usually not provided by the common training datasets used in the field [16], [24]. Purposely, we generate a brand new benchmark using the CARLA engine, involving eight cities from the Leaderboard v2.0 under very different weather conditions to maximize the variety and amount of training samples. For this purpose, we adapted the codebase used to generate data for Leaderboard v1 and developed a data generation pipeline, as well as a closed-loop evaluation process. To improve the scalability and the efficiency of the generation process, we investigate an alternative annotation technique to obtain ground truth BEV semantic segmentation maps, simply relying on a CARLA semantic camera to replace the expensive and storage-consuming offline procedure carried out by the previews works [16], [24], [34].

Our main contributions can be resumed as follows:

- we investigate, for the first time, the explicit modeling of temporal information within the perception module of end-to-end autonomous driving models, in the form of optical flow estimation from images.
- we generate a new benchmark tailored to our study, based on CARLA Leaderboard v2.0, providing additional ground truth annotations for optical flow and annotating BEV semantic segmentation images in a cheaper and more scalable way
- to validate our claims, we extend TransFuser [16], [24] into FlowFuser. Our experiments prove the effectiveness of our proposal, as well as show the pivotal rule of optical flow within the perception module compared to other auxiliary tasks such as depth estimation and semantic segmentation.

Our project page is available at https://noce99.github.io/FlowFuser/.

II. Related work

The idea of teaching an autonomous driving system to imitate humans has a long history [22]. This paradigm is called Imitational Learning, and it uses data from an expert to teach a system how to drive [3] [14] [15] [17]. Its main counterpart is represented by Reinforcement Learning, which has recently started to achieve promising results [13] [29]. The two differ in the way of approaching the problem: while the former paradigm lets the model imitate the humans' driving style, assuming it as the solution to the task, the latter encourages the model to find its own solution to the task, without explicit guidance about it. Recently, the former paradigm became very popular, thanks to the availability of closed-loop driving simulators such as CARLA [9].

Pioneering works propose using neural networks to elaborate cameras and LiDAR information [1] to produce a BEV image as the output, starting from which classical control methods are applied to steering, accelerating, and braking. More recent works moved toward end-to-end models, capable of directly predicting the steering, accelerator, and brake position as their output [12], followed by frameworks predicting discrete navigation commands (straight on, turn right, etc.) [6] [7] and later on, waypoints starting with TransFuser [24] [16]. Specifically, this latter [16] proposed and evaluated two different strategies to predict waypoints as the future positions of the vehicle, being equally distanced in time or in space, respectively.

Lately, foundation models are emerging as an alternative paradigm to autonomous driving [5], [26], [27], [32], integrating the ability of large language models (LLMs) to reason about longtail scenarios. However, any of these approaches ignore the explicit temporal information available to the perception module: this latter has been modeled some approaches distilling a readily available optical flow map [31], thus not in an end-to-end manner, or by implicitly learning good spatio-temporal features [14], yet by requiring prior knowledge of the camera poses.

III. Drive with the Flow

In this section, we introduce our framework designed for improved temporal awareness in end-to-end autonomous driving. As we argue the perception module of modern models [16], [24] lack explicit processing of this cue, we propose the introduction of an auxiliary task – optical flow estimation [33] – among those already carried out. We will show how this simple solution can greatly improve the overall planning process by the model and improve its ability to avoid moving obstacles.

A. FlowFuser Architecture

Figure 2 shows an overview of our multi-modal framework, *FlowFuser*, which extends TransFuser [16], [24], a popular baseline. It processes two main input modalities, respectively images and LiDAR point clouds. The latter are converted into 2-bin histograms over a 2D BEV grid of 32×32 meters, discretized into an image of fixed resolution equal to 256×256 pixels. The current speed and goal location

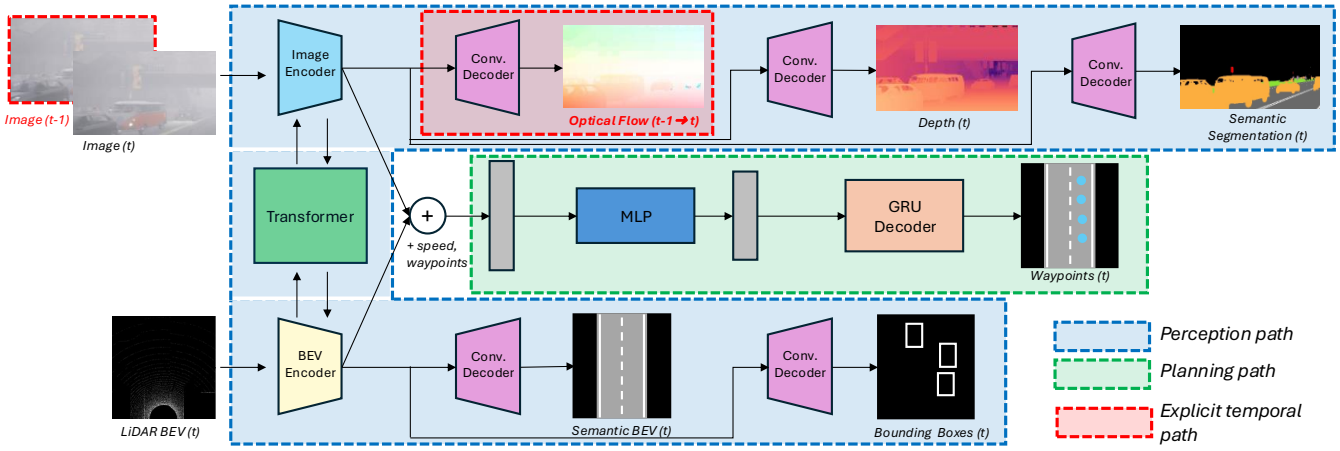

Fig. 2: **FlowFuser architecture.** Following TransFuser [16], [24], our model deploys a multi-modal perception module (blue) and a planning module (green). FlowFuser extends the perception module to process an additional image at time $(t-1)$, as well as with a convolutional decoder predicting the optical flow between the two images (red). The supplementary supervision used to train this latter provides the planning module with richer features, encoding explicit temporal information.

are also inputs to the neural network. The whole architecture is composed of two, main sub-networks: *perception* and *planning* modules, respectively highlighted in blue and green in the figure.

At any time stamp t, the former processes a single color image and LiDAR BEV with two distinct instances of a RegNetY-3.2GF backbone [11], [20], pre-trained on ImageNet [25]. This backbone is composed of four stages. Features extracted at different stages are processed by a Transformer [30], thanks to the self-attention mechanism. The encoders yield two sets of features, that are concatenated and forwarded to the planning module, which will process them to predict a set of waypoints using a decoder, composed of Gated Recurrent Units (GRUs) implementing a short-term memory mechanism. In parallel, they are also processed by several decoders for estimating intermediate outputs, that are used as entry points for supplying auxiliary supervision to the overall model. Specifically, image features are used to estimate depth and semantic segmentation maps (top right in the figure), whereas BEV features are processed to predict a semantic BEV and bounding boxes related to the presence of obstacles on the path (at the bottom in the figure).

Although the supervision provided for these preliminary perception tasks is crucial for training the original TransFuser [16], [24], we argue it lacks temporal awareness, which is offloaded entirely to the planning module within the hidden state of GRUs. Accordingly, we extend the RegNetY backbone processing the color image to support a second image as input – i.e., the frame at time $(t-1)$ – by replicating the weights of the first layer and normalizing them according to [10]. Then, we added a third decoder in charge of predicting optical flow out of the features yielded by the image encoder – highlighted in red in Figure 2. By supervising the decoder during the first stage of the training, we encourage the RegNetY backbone to extract features encoding the temporal information contained in the two subsequential frames, which may be crucial to better understand the behavior and motion of other agents navigating on the road such as cars, pedestrians or cyclists.

A reader familiar with the optical flow literature [33] may argue that a simple encoder-decoder network processing concatenating images [8] does not represent the best choice to predict accurate flow maps. However, it is worth stressing that optical flow only represents an auxiliary task to FlowFuser and that a simple design like the one we propose is sufficient for improving the temporal awareness of the model and its ability to avoid collisions.

B. Dataset Creation

In order to train FlowFuser with the necessary, additional supervision, we created a new dataset using the CARLA Simulator [9]. We generate multiple runs in the eight cities provided by the simulator, covering 1 square kilometer each on average. We spawn autonomous agents such as vehicles and pedestrians, moved by CARLA using privileged information to optimize the resource usage during simulation. During the process, we spawn a car equipped with the sensor suite shown in Figure 3 on the left.

Data collection. To gather the training data, we set the simulation frequency to 20 Hz. We then collect data from all sensors simultaneously every five simulation frames – except for the RGB camera, from which we also capture the following frame, as it is needed to compute optical flow. We choose a five-frame time step as a trade-off between the decrease in the similarity of consecutive data frames and the length of runs.

To increase data variety, we simulate diverse weather conditions thanks to the precise control enabled by CARLA according to *cloudiness*, *precipitation*, *wind*, and *fog* intensities. We vary these values for each run according to a Beta PDF (Probability Distribution Function), defined in Equation 1, whose parameters are shown qualitatively in Figure 3 on

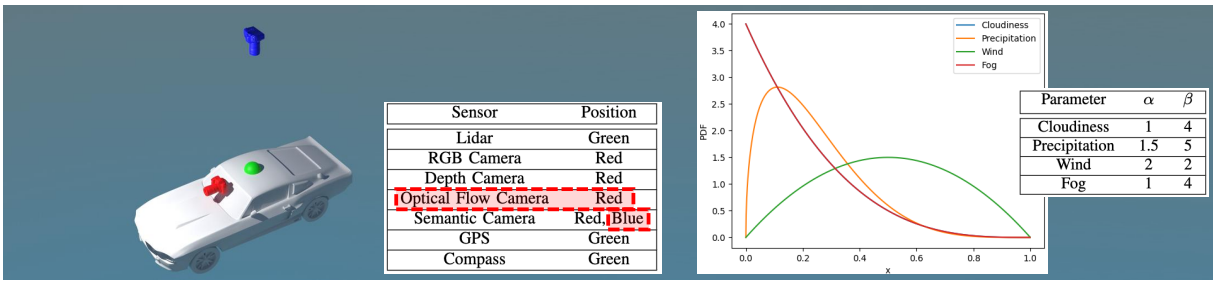

Fig. 3: **CARLA acquisition and weather generation settings.** Our data generation engine extends the established sensors suite (left) with an optical flow camera and a further semantic camera (both highlighted in red). For modeling weather variety, we sample *cloudiness*, *precipitation*, *wind*, and *fog* intensities according to some PDFs (right).

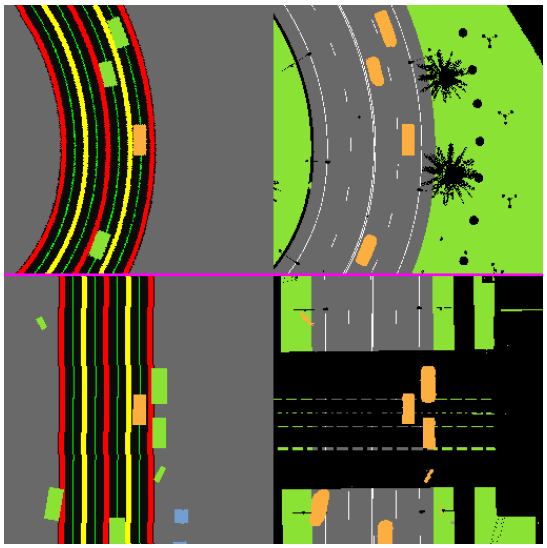

Fig. 4: **Ground truth semantic BEVs comparison.** Maps obtained during offline processing (left) or captured on the fly by the CARLA semantic camera (right).

the right.

$$f(x;\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{K(\alpha,\beta)}$$
$$\text{with} \quad K(\alpha,\beta) = \int_0^1 x^{\alpha-1}(1-x)^{\beta-1}\,dx \tag{1}$$

The final dataset combines 90 simulations, each collecting 2000 data frames for a total of 12.5 simulated hours.

Data Processing. After collection, we processed data to make it suitable for training FlowFuser. We replace the ground truth semantic BEV obtained after offline processing [21], [34] with the segmentation maps collected by the additional semantic camera we attached to the sensor suite. The major shortcoming of replacing the offline procedure with a camera from CARLA is in the occlusions derived by the presence of objects higher over the ground, thus appearing in place of the road in the BEV map. Figure 4 shows two examples obtained according to the two methodologies, highlighting on the right how the CARLA camera is sensitive to the presence of trees or bridges occluding the road. Nonetheless, we will show in our experiments how this does not affect significantly the performance of FlowFuser.

Figure 5 provides an overview of the information stored within a single data frame in our dataset. Specifically, (A) and (B) are the outputs of the RGB camera in two consecutive frames – previous and current timestamp – (C) is the output of the depth camera (log-depth), (D) is the optical flow camera output, (E) is the output of the front semantic segmentation camera, covering the following seven classes: *roads*, *lanes*, *pedestrians*, *vehicles*, *traffic lights*, *signs*, and *unknown*; (F) is the LiDAR BEV, (G) shows the ground truth waypoints and (H) the target point the vehicle is heading to; (J) and (K) shows a further example of ground truth semantic BEV according to the two sources described before, (L) shows ground truth bounding boxes for vehicles (green) and pedestrians (red); finally, (M) and (N) show the actual and target speed.

IV. Experimental Results

In this section, we report the outcome of our experiments.

A. Implementation Details

We implement FlowFuser starting from the CARLA Garage codebase [16]. Following the protocol used for TransFuser [16], [24], we run two training phases. During the first stage, we train the backbone and the decoders in charge of predicting the intermediate outputs – respectively depth, semantic segmentation, and optical flow maps, together with semantic BEV and bounding boxes. During the second phase, the planner module is trained to predict instant speed and waypoints. We use the same loss functions adopted by TransFuser, and add an L2 loss function between predicted and ground truth optical flow, both normalized in [-1, 1]. The flow loss is multiplied by $1e^3$ to balance the contribution of the different terms. During both stages, we run 30 epochs of training, with a total batch size of 180. Both require between 18 and 22 hours to complete, over 4 A100 GPUs.

At inference time, we follow [16], [24] and calculate the steering angle, acceleration, and brake using a PID controller. To minimize human intervention, we remove any additional hardcoded checks in the inference code aimed at stopping the vehicle – except a gentle push to the model in case it remains stuck in the absence of obstacles. The same policies

Fig. 5: **Input and output modalities in a single data frame.** We show the multi-modal input processed by FlowFuser (A,B,F,H,M), as well as the ground truth annotations used for training (C,D,E,G,J,K,L,N).

TABLE I: **Experimental results.** We compare the results achieved by TransFuser and FlowFuser on evaluation routes, and report the impact of using the ground truth semantic BEVs generated online rather than offline. All models are evaluated over 3 independent trainings, reporting mean and standard deviation. Best overall results in **bold**.

	TransFuser [16], [24]			FlowFuser (ours)			FlowFuser (ours) + online BEV masks		
Town	Compl. [%] ↑	Coll. ↓	Time [s] ↓	Compl. [%] ↑	Coll. ↓	Time [s] ↓	Comp. [%] ↑	Coll. ↓	Time [s] ↓
Town 1	99.1 ± 0.8	0.10 ± 0.02	136.0 ± 7.3	93.1 ± 0.3	0.10 ± 0.10	163.2 ± 14.5	95.2 ± 2.1	0.12 ± 0.07	140.3 ± 16.2
Town 2	96.1 ± 5.5	0.12 ± 0.03	158.6 ± 7.1	89.2 ± 8.5	0.20 ± 0.03	104.9 ± 6.1	93.2 ± 1.5	0.41 ± 0.08	114.8 ± 6.4
Town 3	82.4 ± 6.3	1.50 ± 0.02	272.3 ± 12.3	84.9 ± 2.0	0.65 ± 0.15	275.1 ± 1.9	91.4 ± 4.4	0.27 ± 0.18	257.2 ± 2.5
Town 4	55.2 ± 8.5	1.03 ± 0.63	233.3 ± 2.3	57.08 ± 3.5	0.55 ± 0.05	276.7 ± 5.4	51.3 ± 6.2	0.47 ± 0.09	278.9 ± 3.8
Town 5	97.8 ± 2.2	0.79 ± 0.09	156.2 ± 5.0	92.25 ± 7.8	0.06 ± 0.02	117.2 ± 16.5	95.6 ± 4.5	0.24 ± 0.05	177.5 ± 20.2
Town 6	95.4 ± 4.6	1.55 ± 0.45	209.6 ± 5.8	96.2 ± 3.6	1.02 ± 0.29	198.2 ± 21.3	91.7 ± 5.5	0.81 ± 0.18	217.4 ± 13.5
Town 7	18.6 ± 0.7	0.45 ± 0.05	264.5 ± 1.5	22.3 ± 1.5	2.02 ± 0.17	233.4 ± 4.5	21.3 ± 4.5	1.65 ± 0.17	295.3 ± 3.5
Town 10	97.8 ± 0.3	1.88 ± 0.38	114.1 ± 1.5	97.3 ± 2.5	3.02 ± 0.06	117.1 ± 1.8	94.1 ± 4.7	2.63 ± 0.17	111.5 ± 1.7
Average	87.1 ± 0.3	0.67 ± 0.01	194.8 ± 0.9	**90.9 ± 0.2**	**0.35 ± 0.01**	195.0 ± 1.3	88.7 ± 1.0	0.41 ± 0.03	**184.9 ± 1.2**

are applied in any experiment, both when using TransFuser or FlowFuser.

B. Evaluation

For testing our framework, we use CARLA Scenario Runner, a plugin generating the same starting traffic condition multiple times over the same route, allowing for a fair evaluation of different models. We have generated ten evaluation routes per city, each a few kilometers long, not overlapping with the training routes. A single evaluation run can terminate both if the model gets stuck or if it fully completes the route. We have selected three evaluation metrics: the route completion percentage (Compl., the higher the better), and the number of collisions with other vehicles, pedestrians, or objects, normalized over the number of routes (Coll., the lower the better). We also report the time needed to complete the route, although not entirely meaningful in the case of models achieving different completion percentages.

TransFuser vs FlowFuser. Table I collects the results achieved by both the original TransFuser model [16], [24] and our FlowFuser on the eight cities, by averaging over the ten routes in each. We report metrics on the single cities, as well as the overall average. Overall, FlowFuser dramatically reduces the amount of collisions with other vehicles and pedestrians, from 0.67 to 0.35 on average. The improvement is particularly evident in Towns 3, 5, and 6, where the traffic is denser and more agents are moving around. This comes with a small improvement in completion, from 87.1 to 90.9%, and an average increase of 0.2 seconds to the time required to complete routes. Figure 6 shows a qualitative example with FlowFuser, thanks to the temporal information modeled within its perception path, avoiding a collision where TransFuser cannot.

Offline vs online ground truth BEVs. In the right-most part of Table I we report the results achieved by training FlowFuser with the ground truth semantic BEV obtained from the top-view CARLA semantic camera as an alternative to those generated offline [21], [34]. FlowFuser performance slightly decreases with this easier-to-obtain ground truth replacement. A faster time in route completion is achieved.

C. Ablation Studies

We conclude our evaluation by conducting some ablation experiments, aimed at assessing the importance of the different tasks carried out by the perception module on images, as well as how the different weather conditions observed at training time allow FlowFuser for more robust trajectory planning in terms of collision avoidance.

Impact of the perception tasks. In Table II, we report the results achieved by different variants of FlowFuser, obtained by removing one of the image decoders in charge of predicting optical flow – i.e., the original TransFuser – depth, semantic segmentation. Overall, we can further appreciate how optical flow plays a crucial role in enabling effective collision avoidance. Indeed, by either removing depth or semantics, we can notice a moderate increase in the average number of collisions – respectively 0.10 and 0.06 compared to the results shown in the mid column of Table I, versus the 0.20 increase achieved by TransFuser when not enhanced by the optical flow task. Interestingly, yet not surprisingly, depth has a higher impact on collisions compared to semantic segmentation, whose removal yields the lowest increase in the number of collisions.

Impact of weather conditions. Table III compares the results achieved by two FlowFuser variants, trained respectively under different weather conditions (w/ wth.) or using

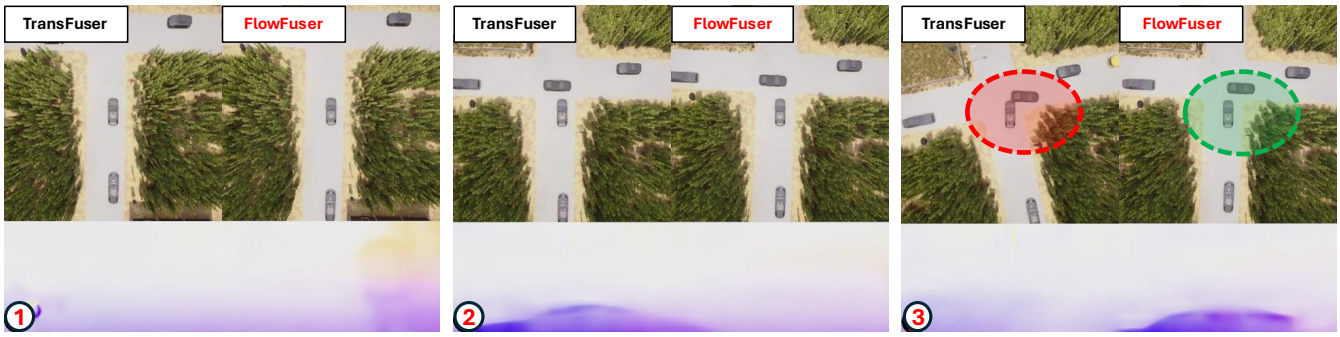

Fig. 6: **Qualitative results.** From left to right, visualization of three consecutive frames observed by TransFuser [16], [24] and FlowFuser when running on the same route. While the former collides with another car, our model avoids it thanks to the explicit temporal information processed by the perception module – whose flow predictions are shown at the bottom.

TABLE II: **Ablation study – impact of perception tasks.** Results by FlowFormer when trained without supervision about depth estimation or semantic segmentation.

		FlowFuser		
Town	Setting	Compl. [%] ↑	Coll. ↓	Time [s] ↓
Town 1	w/o flow [16], [24]	100.0	0.00	130.9
	w/o depth	100.0	0.00	151.1
	w/o semantics	100.0	0.00	152.6
Town 2	w/o flow [16], [24]	92.2	0.10	162.1
	w/o depth	91.1	0.40	109.9
	w/o semantics	99.9	0.30	121.8
Town 3	w/o flow [16], [24]	86.9	1.50	281.0
	w/o depth	86.9	0.50	281.3
	w/o semantics	82.9	0.40	266.6
Town 4	w/o flow [16], [24]	58.6	0.40	235.5
	w/o depth	52.7	0.40	252.7
	w/o semantics	63.6	0.60	261.7
Town 5	w/o flow [16], [24]	95.6	0.70	157.1
	w/o depth	100.0	0.80	127.4
	w/o semantics	100.0	0.60	134.1
Town 6	w/o flow [16], [24]	90.8	1.10	215.4
	w/o depth	100.0	0.80	240.8
	w/o semantics	100.0	0.40	216.4
Town 7	w/o flow [16], [24]	19.1	0.50	263.3
	w/o depth	45.3	0.80	211.1
	w/o semantics	32.2	0.70	229.1
Town 10	w/o flow [16], [24]	100.0	0.10	112.9
	w/o depth	100.0	0.00	121.1
	w/o semantics	100.0	0.30	118.7
Average	w/o flow [16], [24]	80.4	0.55	194.8
	w/o depth	84.5	0.46	**186.9**
	w/o semantics	**84.8**	**0.41**	187.6

TABLE III: **Ablation study – weather conditions during training.** Results by FlowFormer when trained with a variety of weather conditions or sunny weather only.

		FlowFuser		
Town	Setting	Compl. [%] ↑	Coll. ↓	Time [s] ↓
Town 1	w/ wth.	92.8	0.20	177.7
	w/o wth.	100.0	0.10	147.7
Town 2	w/ wth.	99.9	0.20	111.0
	w/o wth.	100.0	0.30	103.3
Town 3	w/ wth.	82.9	0.50	272.1
	w/o wth.	66.3	1.30	247.0
Town 4	w/ wth.	51.3	0.50	302.8
	w/o wth.	53.4	0.60	321.9
Town 5	w/ wth.	84.5	0.00	138.0
	w/o wth.	94.2	1.70	186.2
Town 6	w/ wth.	100.0	0.90	219.5
	w/o wth.	100.0	0.70	257.2
Town 7	w/ wth.	24.9	0.40	245.8
	w/o wth.	38.5	1.60	260.0
Town 10	w/ wth.	100.0	0.10	118.5
	w/o wth.	100.0	0.60	117.9
Average	w/ wth.	79.5	**0.35**	**198.8**
	w/o wth.	**81.5**	0.86	205.2

a dataset of equivalent size and variety, yet limited to sunny weather only. Interestingly, although not impacting the overall completion percentage significantly, a variegate set of weather conditions observed during training plays a crucial role as well in minimizing the number of collisions occurring during FlowFuser driving. We conjecture that the absence of weather events such as rain during training severely affects optical flow estimation when these are met during the evaluation, whereas their occurrence at training time allows FlowFuser to focus only on the dynamics of the other vehicles and pedestrians and to ignore raindrops.

V. Conclusions

This paper proved that explicit modeling of temporal dynamics is beneficial to end-to-end autonomous driving frameworks. We achieved this by introducing optical flow as an auxiliary task within the perception model, building our FlowFuser on top of the TransFuser baseline. We generated a new dataset for training and evaluating our proposal, whose results support our conjectures.

Future directions. Temporal information could also be modeled on the LiDAR side of the perception module, by adding a further decoder to perform 3D scene flow [33] estimation. Unfortunately, obtaining LiDAR scene flow labels from CARLA is not trivial, as it does not provide a custom sensor for this purpose. Future work will explore this possibility.

Acknowledgment

This study was carried out within the MOST – Sustainable Mobility National Research Center and received funding from the European Union Next-GenerationEU – PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.4 – D.D. 1033 17/06/2022, CN00000023. This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

We acknowledge the CINECA award under the ISCRA initiative, for the availability of high-performance computing resources and support.

REFERENCES

[1] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018.

[2] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[3] Sergio Casas, Abbas Sadat, and Raquel Urtasun. Mp3: A unified model to map, perceive, predict and plan. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[4] Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. End-to-end autonomous driving: Challenges and frontiers. *arXiv preprint arXiv:2306.16927*, 2023.

[5] Long Chen, Oleg Sinavski, Jan Hünermann, Alice Karnsund, Andrew James Willmott, Danny Birch, Daniel Maund, and Jamie Shotton. Driving with llms: Fusing object-level vector modality for explainable autonomous driving. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[6] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. *arXiv preprint arXiv:1710.02410*, 2018.

[7] Felipe Codevilla, Eder Santana, Antonio M. López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. *arXiv preprint arXiv:1904.08980*, 2019.

[8] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[9] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[10] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth prediction. In *The International Conference on Computer Vision (ICCV)*, October 2019.

[11] Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin, and Piotr Bojanowski. Self-supervised pretraining of visual features in the wild. *arXiv preprint arXiv:2103.01988*, 2021.

[12] Imtiaz Ul Hassan, Huma Zia, H. Sundus Fatima, Syed Adnan Yusuf, and Muhammad Khurram. A lightweight convolutional neural network to predict steering angle for autonomous driving using carla simulator. *Modelling and Simulation in Engineering*, 2022(1):5716820, 2022.

[13] Jumman Hossain. Autonomous driving with deep reinforcement learning in carla simulation. *arXiv preprint arXiv:2306.11217*, 2023.

[14] Shengchao Hu, Li Chen, Penghao Wu, Hongyang Li, Junchi Yan, and Dacheng Tao. St-p3: End-to-end vision-based autonomous driving via spatial-temporal feature learning. In *European Conference on Computer Vision*, pages 533–549. Springer, 2022.

[15] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, et al. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17853–17862, 2023.

[16] Bernhard Jaeger, Kashyap Chitta, and Andreas Geiger. Hidden biases of end-to-end driving models. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2023.

[17] Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. Vad: Vectorized scene representation for efficient autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8340–8350, 2023.

[18] Zhenxin Li, Kailin Li, Shihao Wang, Shiyi Lan, Zhiding Yu, Yishen Ji, Zhiqi Li, Ziyue Zhu, Jan Kautz, Zuxuan Wu, et al. Hydra-mdp: End-to-end multimodal planning with multi-target hydra-distillation. *arXiv preprint arXiv:2406.06978*, 2024.

[19] Zhiqi Li, Zhiding Yu, Shiyi Lan, Jiahan Li, Jan Kautz, Tong Lu, and Jose M Alvarez. Is ego status all you need for open-loop end-to-end autonomous driving? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14864–14873, 2024.

[20] Raghav Mehta, Vítor Albiero, Li Chen, Ivan Evtimov, Tamar Glaser, Zhiheng Li, and Tal Hassner. You only need a good embeddings extractor to fix spurious correlations. *arXiv preprint arXiv:2212.06254*, 2022.

[21] Błażej Osiński, Adam Jakubowski, Piotr Miłoś, Paweł Ziecina, Christopher Galias, Silviu Homoceanu, and Henryk Michalewski. Simulation-based reinforcement learning for real-world autonomous driving. *arXiv preprint arXiv:1911.12905*, 2024.

[22] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

[23] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7077–7087, 2021.

[24] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[25] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[26] Hao Shao, Yuxuan Hu, Letian Wang, Guanglu Song, Steven L Waslander, Yu Liu, and Hongsheng Li. Lmdrive: Closed-loop end-to-end driving with large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15120–15130, 2024.

[27] Chonghao Sima, Katrin Renz, Kashyap Chitta, Li Chen, Hanxue Zhang, Chengen Xie, Jens Beißwenger, Ping Luo, Andreas Geiger, and Hongyang Li. Drivelm: Driving with graph visual question answering. In *European Conference on Computer Vision (ECCV)*, 2024.

[28] Ardi Tampuu, Tambet Matiisen, Maksym Semikin, Dmytro Fishman, and Naveed Muhammad. A survey of end-to-end driving: Architectures and training methods. *IEEE Transactions on Neural Networks and Learning Systems*, 33(4):1364–1384, 2020.

[29] Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. End-to-end model-free reinforcement learning for urban driving using implicit affordances. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7153–7162, 2020.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[31] Hengli Wang, Peide Cai, Yuxiang Sun, Lujia Wang, and Ming Liu. Learning interpretable end-to-end vision-based motion planning for autonomous driving with optical flow distillation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13731–13737, 2021.

[32] Zhenhua Xu, Yujia Zhang, Enze Xie, Zhen Zhao, Yong Guo, Kwan-Yee K Wong, Zhenguo Li, and Hengshuang Zhao. Drivegpt4: Interpretable end-to-end autonomous driving via large language model. *IEEE Robotics and Automation Letters*, 2024.

[33] Mingliang Zhai, Xuezhi Xiang, Ning Lv, and Xiangdong Kong. Optical flow and scene flow estimation: A survey. *Pattern Recognition*, 114:107861, 2021.

[34] Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, and Luc Van Gool. End-to-end urban driving by imitating a reinforcement learning coach. *arXiv preprint arXiv:2108.08265*, 2021.