Supplementary Material for TransFuser: Imitation with Transformer-Based Sensor Fusion for Autonomous Driving

Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger

1 Architecture Details

In this section, we provide additional details regarding the multi-scale feature fusion component of the TransFuser and the low-level PID controller that converts waypoints to steer, throttle, and brake of the vehicle.

1.1 Multi-Scale Feature Fusion

TransFuser takes in RGB images and a LiDAR BEV representation as inputs and processes them with two RegNetY-3.2GF [1] modules respectively. It uses several transformers for the fusion of intermediate feature maps between both modalities. For the camera input of resolution 160×704 pixels, the RegNet module produces intermediate feature maps of dimensions $40\times176\times72$, $20\times88\times216$, $10\times44\times576$ and $5 \times 22 \times 1512$ in different RegNet blocks. For the LiDAR input of resolution 256×256 pixels, the RegNet module produces intermediate feature maps of dimensions $64 \times 64 \times 72$, $32 \times 32 \times 216$, $16 \times 16 \times 576$ and $8 \times 8 \times 1512$ in different RegNet blocks. Since processing feature maps at high spatial resolutions is computationally expensive, we downsample higher resolution feature maps from the early encoder blocks using average pooling to a fixed resolution of 5×22 for the camera and 8×8 for the LiDAR before passing them as inputs to the fusion transformer.

Next, we describe the fusion mechanism between the RGB and LiDAR BEV feature maps. The height and width channels of the RGB and LiDAR BEV feature maps are collapsed into one dimension and stacked together to form a tensor of size $(8 * 8 + 5 * 22) \times 72/216/576/1512$ for the 4 different fusion resolutions. This tensor is passed as input to the transformer, which outputs a feature of the same dimension. It is then reshaped back into 2 tensors of dimension $5\times22\times72/216/576/1512$ and $8\times8\times72/216/576/1512$ each for the camera and LiDAR branches. Each of these 2 tensors is then upsampled using bilinear interpolation to match the original feature resolution. They are then combined with the existing feature maps via element-wise summation. We pass the fused features back into the feature extraction branches since this provides the deeper RegNet layers access to the global context of the 3D scene encoded in the earlier layers. After the final fusion layer, we use 1×1 convolution to reduce the number of features to 512 so that we can keep the rest of the network the same, independent of the chosen vision architecture, which can have varying channel dimensions. The resulting 512-dimensional feature vector output from both the image and LiDAR BEV stream is combined via element-wise summation. This 512-dimensional feature vector constitutes a compact representation of the environment which incorporates global contextual reasoning.

- *K. Chitta, B. Jaeger, Z. Yu, K. Renz and A. Geiger are with the Autonomous Vision Group, University of Tübingen and Max Planck Institute for Intelligent Systems, Tübingen. E-mail: kashyap.chitta@tue.mpg.de, bernhard.jaeger@uni-tuebingen.de, zehao.yu@uni-tuebingen.de, katrin.renz@uni-tuebingen.de, a.geiger@uni-tuebingen.de.*
- *A. Prakash is with the University of Illinois Urbana-Champaign. Work done while at the Max Planck Institute for Intelligent Systems, Tübingen. E-mail: adityap9@illinois.edu*

1.2 PID Controller

To convert waypoints to vehicle controls, we use two PID controllers. This process is summarized in Algorithm 1. We first compute the $T-1$ vectors between waypoints of consecutive time-steps, with $T = 4$ in our configuration. The longitudinal controller (LonPID in Algorithm 1) takes in a weighted average of these vectors and tries to match the vehicle velocity v to the desired velocity γ as much as possible. We find the weights $\lambda = \{1, 0, 0\}$, for the $T-1$ vectors, prioritizing the closest pair of waypoints, give the best empirical results. The lateral PID controller (LatPID in Algorithm 1) orients the vehicle along the aim direction α which is computed as the orientation of the midpoint of $\mathbf{w}_1$ and $\mathbf{w}_2$. Each controller has 3 gain parameters, which are tuned on a subset of our training routes. For the longitudinal controller, we set $K_p = 5.0, K_i = 0.5, K_d = 1.0$ and for the lateral controller, we set $K_p = 1.25, K_i = 0.75, K_d = 0.3$. Both controllers use a buffer of size 20 to approximate the integral term as a running average. The other parameters used are a brake threshold speed β_{min} and brake speed ratio β_{ratio}, which we tune jointly with the gain parameters, and set to 0.4 and 1.1 respectively. If the speed is smaller than 0.01 we set the lateral angle to 0. This is to avoid error accumulation in the integral while the car is stopped (cars have to move forward to reduce lateral error).

Algorithm 1: Generating an action from waypoints.

input : $v, \{\mathbf{w}_t\}_{t=1}^{T}$; `// velocity and waypoints`
output: steer $\in (-1, 1)$, throttle $\in (0, 1)$, brake $\in (0, 1)$; `// vehicle control`
$\gamma = 0$
for $t \leftarrow 1$ **to** $T - 1$ **do**
  $\gamma \mathrel{+}= \lambda_t ||\mathbf{w}_{t+1} - \mathbf{w}_t||$; `// desired velocity`
$\omega = \frac{(\mathbf{w}_1 + \mathbf{w}_2)}{2}$
$\alpha = \tan^{-1}\left(\frac{\omega[1]}{\omega[0]}\right)$; `// aim direction`
steer $=$ LatPID(α)
if $\gamma < \beta_{min}$ **or** $\gamma < v\beta_{ratio}$ **then**
  throttle $= 0$
  brake $= 1$
else
  throttle $=$ LonPID$(\gamma - v)$
  brake $= 0$

2 Datasets

In this section, we describe our data generation pipeline with details about the sensor configuration, routes and scenarios used in the CARLA simulator.

2.1 Sensor Configuration

We use three front cameras with a field of view (FOV) of 120° for the RGB image input rotated by $-60°$, 0° and 60° around the upward yaw axis. The cameras are mounted at a height of 2.3m from the ground level of the ego-vehicle and located 1.3m in front of the centroid of the ego-vehicle. We offset the cameras forward by 1.3m to avoid the hood of the ego-vehicle from occluding a portion of the rendered images. We extract images from this camera at a resolution of 960 $\times$ 480 pixels. In the preprocessing stage, we crop the sides of the images to remove image distortions. Finally, they are concatenated, resulting in a single image width height 160 and width 704. For the LiDAR point cloud, we use a ray-cast-based Velodyne 64 LiDAR with an 85m range and rotation frequency of 10 FPS. The upper FOV of the LiDAR is set at 10° and the lower FOV is set at -30°. The LiDAR sensor is mounted at a height of 2.5m from the ground level of the ego-vehicle and located 1.3m in front of the ego-vehicle. Since the simulation is running at 20 FPS the sensor is returning either the front or the backside at each simulation step. We only use the front side of the LiDAR, since this is where most of the relevant information is located, and perform action repeat at each second time step. We further use additional sensors such as IMU to get the orientation, GPS for localization, and speedometer to get the current speed of the ego-vehicle.

The CARLA leaderboard client adds independent Gaussian noise with a standard deviation of 50 cm to all 3 GPS axis. This makes the localization of the vehicle (and consequently of the target position) unreliable. To reduce this noise, we use a model based denoising algorithm. We keep a buffer of the last 100 observed GPS positions. At each time step, we estimate our own expected relative motion using a kinematic bicycle model. We add this motion to all observed GPS points, effectively turning them into an estimate of the next position. We then average all updated GPS positions to obtain a Monte Carlo estimate of the true position. This Monte Carlo estimate has a lower variance than the original as can be seen in figure Fig. 1.

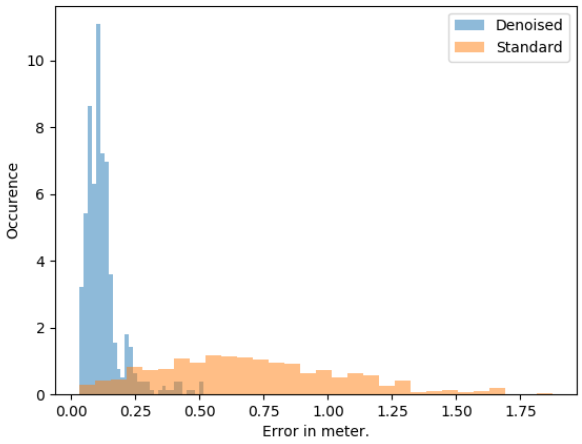

Fig. 1: **Histogram of localization error.** Error is measured in absolute meters compared to the ground truth position. The default GPS signal sent by the simulator is compared with the position estimated by the denoising algorithm. Data was collected in a short route where the car waits at a red light and then performs a left turn.

2.2 Routes and Scenarios

We generate around 3500 training routes in CARLA to train our models. Each route consists of one unique scenario. A scenario is defined by a trigger 'transform' which indicates the spawn location and the orientation of that scenario in a particular town. Specifically, we consider Scenarios[1] 1, 3, 4, 7, 8, 9, and 10 in our training dataset. Since we focus on safety-critical scenarios, e.g. pedestrians emerging from occluded regions to cross the road at random locations, vehicles running red lights, and unprotected turns, we do not include Scenarios 5 and 6, which involve lane changing. We also do not include Scenario 2 since it occurs naturally during the driving in the presence of other dynamic agents.

Scenarios 4, 7, 8, 9, and 10 can only occur at intersections. We sample all the intersections present in all CARLA town maps, and extract every possible traversal through these intersections. We then check for all traversals where the spawned scenario is valid, and include these for training. Across the 8 public town maps, we obtain around 2500 such routes through intersections. The average length of these routes is 100m.

Scenarios 1 and 3 can occur anywhere on a CARLA map. Since we have an abundance of intersections in our dataset due to the other scenarios, we collect data for these scenarios along curved highways. There are around 500 such routes. The average route length is 400m, and the scenario is spawned at the middle of the route.

1. https://leaderboard.carla.org/scenarios/

Route	Town	Weather	Daytime	Length
0	1	MidRain	Dawn	1130
1	1	Cloudy	Dawn	1014
2	1	Cloudy	Morning	893
3	1	HardRain	Noon	731
4	1	HardRain	Twilight	936
5	1	HardRain	Morning	685
6	2	Wet	Noon	1010
7	2	Cloudy	Night	974
8	2	Cloudy	Twilight	820
9	2	WetCloudy	Noon	820
10	2	HardRain	Sunset	972
11	2	MidRain	Night	872
12	3	SoftRain	Twilight	2303
13	3	MidRain	Twilight	1748
14	3	WetCloudy	Night	1436
15	3	MidRain	Noon	1870
16	3	Wet	Night	1869
17	3	HardRain	Dawn	1556
18	4	WetCloudy	Twilight	2069
19	4	SoftRain	Dawn	2058
20	4	SoftRain	Night	1862
21	4	HardRain	Night	1863
22	4	Cloudy	Sunset	2319
23	4	Wet	Sunset	2440
24	5	MidRain	Sunset	2101
25	5	SoftRain	Noon	1554
26	5	SoftRain	Morning	1271
27	5	WetCloudy	Morning	1018
28	5	MidRain	Morning	1071
29	5	WetCloudy	Dawn	1651
30	6	SoftRain	Sunset	2525
31	6	Wet	Dawn	1859
32	6	Wet	Morning	2842
33	6	Wet	Twilight	2270
34	6	Cloudy	Noon	1442
35	6	WetCloudy	Sunset	1760

TABLE 1: **Longest6 Routes.** Environmental condition and length (in meters) of each of the 36 evaluation routes.

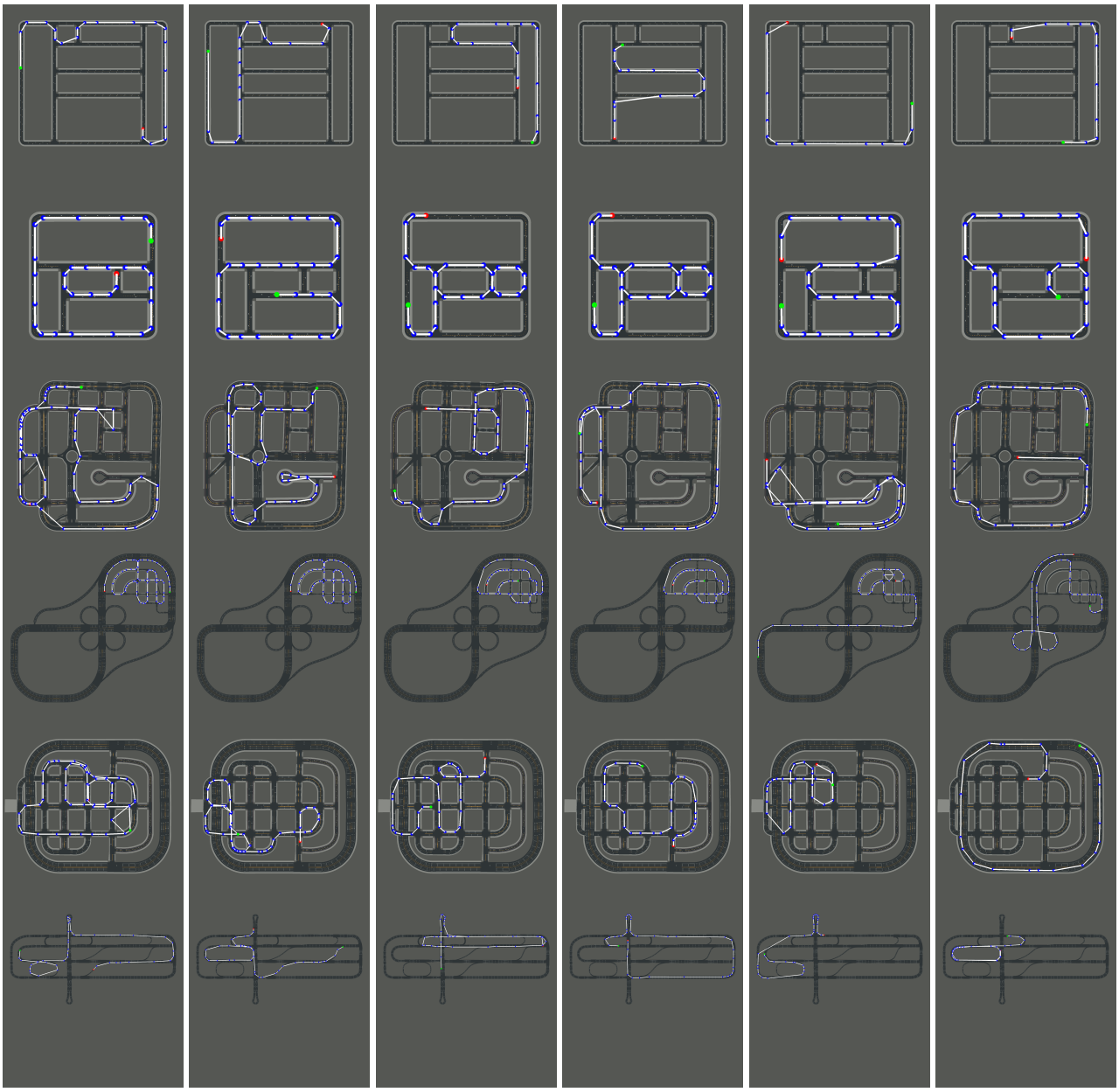

Fig. 2: **Visualization of Longest6 Routes.** We show the 36 unique routes on their corresponding town map. The starting point is shown in red, the destination in green and target points along the route are highlighted in blue.

Finally, since changing lanes is challenging, we extracted curved routes similar to those with Scenarios 1 and 3, but include lane changes along the route instead of the CARLA scenario. There are around 500 lane change routes with an average length of 400m. Each such route has 2 lane changes, one at the beginning and one at the half-way point.

2.3 Longest6 Routes

Our proposed evaluation setting consists of 36 routes from 6 different CARLA towns (Town01-Town06). Each route has a unique environmental condition obtained by combining one of 6 weather conditions (Cloudy, Wet, MidRain, WetCloudy, HardRain, SoftRain) with one of 6 daylight conditions (Night, Twilight, Dawn, Morning, Noon, Sunset). We provide details regarding the weather, daytime, and length of the 36 evaluation routes in Table 1 and visualizations of the routes in Fig. 2.

3 Baselines

In this section, we describe Late Fusion and Geometric Fusion in detail.

3.1 Late Fusion

We implement a version of our architecture where the RGB image and the LiDAR BEV streams are processed independently of each other. The image features are extracted using RegNetY-3.2GF, which is initialized from ImageNet pretrained weights, whereas the LiDAR BEV features are extracted using RegNetY-3.2GF, which is trained from scratch. These features are then combined via element-wise summation and passed to the waypoint prediction network. This architecture is equivalent to removing the transformer modules from the TransFuser. This is similar to the encoder of [2] which also processed the image feature and LiDAR features separately. While Sobh et al. [2] used a learned controller module conditioned on discrete navigational commands (similar to CILRS), we use our auto-regressive waypoint prediction network since it performs better empirically.

3.2 Geometric Fusion

We implement a multi-scale geometry-based fusion method, inspired by [3], [4], involving both image-to-LiDAR and LiDAR-to-image feature fusion. We unproject each 0.125m $\times$ 0.125m block in our LiDAR BEV representation into 3D space, resulting in a 3D volume. Then, we randomly select 5 points from the LiDAR point cloud lying in this 3D volume and project them into the image space. We aggregate the image features of these points via element-wise summation before passing to a 3-layer MLP, consisting of 512 units each. The output of the MLP is then combined with the LiDAR BEV feature of the corresponding 0.125m $\times$ 0.125m block at multiple resolutions throughout the feature extractor.

Similarly, for each image pixel, we randomly select 5 points in the LiDAR point cloud which project to that pixel in image space, then project these points into the BEV space, aggregate their features via element-wise summation before passing to a 3-layer MLP, consisting of 512 units each. The output of the MLP is then combined with the features of the corresponding image pixels at multiple resolutions throughout the feature extractor.

Similar to the TransFuser, we downsample the higher spatial resolution feature maps to a fixed resolution of 5 $\times$ 22 for the camera and 8 $\times$ 8 for the LiDAR branch. After the features are combined, they are upsampled to the original resolution using bilinear interpolation, before being fed into the individual feature extractors of the two modalities. This baseline is equivalent to replacing the transformers in our architecture with geometry-based feature projections.

Method	LiDAR?	Map?	DS ↑	RC ↑	IS ↑
CaRINA [5]	✓	-	4.56	23.80	0.41
CIRLS [6]	-	-	5.37	14.40	0.55
LBC [7]	-	-	8.94	17.54	**0.73**
CaRINA [5]	✓	✓	15.55	40.63	0.47
Pylot [8]	✓	✓	16.70	48.63	0.50
TF CVPR [9]	✓	-	16.93	51.82	0.42
AIM-MT [10]	-	-	19.38	67.02	0.39
NEAT [10]	-	-	21.83	41.71	0.65
MaRLn [11]	-	-	24.98	46.97	0.52
Late Fusion	✓	-	26.07	64.67	0.47
WOR [12]	-	-	31.37	57.65	0.56
TF+ [13]	✓	-	34.58	69.84	0.56
GRIAD [14]	-	-	36.79	61.86	0.60
Geometric Fusion	✓	-	41.70	87.85	0.47
Latent TF (Ours)	-	-	45.20	66.31	0.72
TF (Ours)	✓	-	61.18	86.69	0.71
LAV [15]	✓	-	**61.85**	**94.46**	0.64

TABLE 2: **CARLA Leaderboard Snapshot.** The LiDAR and Map column indicate whether a submission uses a LiDAR sensor or an HD map. Methods are ranked by the main metric, driving score (DS). Additionally, we report the route completion (RC) and infraction score (IS).

4 Additional Results

In this section, we provide additional results and analysis to supplement our findings in the main paper.

4.1 Leaderboard Snapshot

For completeness, we report a snapshot of all publicly available methods with a report on the CARLA leaderboard in Table 2, taken on 24.05.2022. If a method has multiple submissions in the same setting, we report the best result. Progress on the CARLA leaderboard has been rapid over the last years. It is important to note that the leaderboard gives maximum flexibility to researchers on how to solve the problem of self-driving. It only places restrictions on the maximum numbers of sensors used. As a result, evaluated systems vary among many axes: the amount of training data, the kind of training labels, sensors inputs, pre-training, training method, run time, architecture and engineering. The progress of evaluated systems often comes from improvements along multiple axes. While comparisons can only be made at a system level in this setting the resulting newer systems do drive much better. Methods like CILRS and LBC were popular baselines [9], [10], [12], [16] in the past. Their performance does get dwarfed now, so we encourage future work to compare to more recent baselines.

4.2 Latent TransFuser

To investigate the importance of using transformers to fuse the BEV branch with the perspective branch in Latent TransFuser we conduct an ablation where we replace the transformers with simple MLPs. Specifically, we flatten the same input features as the transformers use and process them with 2 layer MLPs with a hidden dimension of 100 and a ReLU activation function. We use one MLP for each fusion

Parameter	Value	DS ↑	RC ↑	IS ↑
Fusion Method	MLP	38.66	86.24	0.47
Default Configuration	Worst Seed	49.68	91.97	0.54
	Best Seed	**50.48**	**92.28**	**0.55**

TABLE 3: **Latent TransFuser Ablation.** We conduct an ablation where we replace the transformers in Latent TransFuser with 2 layer MLPs. Fusing the BEV branch with the perspective image branch using transformers leads to a significant improvement in all three metrics.

direction at each of the 4 resolutions, giving 8 in total. The result is shown in Table 3. Using Transformers to fuse the perspective and BEV branch leads to a clear improvement of more than 10 DS.

4.3 Attention Map Visualizations

As described in Sec 4.10 of the main paper, we visualize the attention maps from the final attention layer for each head for each transformer. Specifically, we consider intersection scenarios from Town03 and Town05 and examine the top-5 attention weights for the tokens in image feature map and LiDAR feature map. From the visualizations in Fig. 7 and Fig. 8, we observe a common trend in attention maps: TransFuser attends to areas near vehicles and traffic lights at intersections, albeit at a slightly different location in the image and LiDAR feature maps.

In addition, we also provide statistics on cross-modal attention for image and LiDAR feature tokens. We report the % of tokens for which atleast one of the top-5 attended tokens belong to the other modality for each of the 4 transformers (T1, T2, T3, T4) in Table 5 of the main paper. The results indicate distinct behavior for different transformers. Next, we provide attention map visualizations for each transformer to study their behavior in detail.

T1 exhibits significant image to LiDAR cross-attention but negligible LiDAR to image cross-attention. Moreover, we observe that for image tokens, T1 always attends to nearly the same location in the LiDAR, i.e., just in front of the ego-vehicle as shown in Fig. 3.

T2 shows similar behavior to T1 in terms of negligible LiDAR to image cross-attention. However, it differs from T1 in image to LiDAR cross-attention, i.e., it attends to different regions in the LiDAR input (Fig. 4), not just in front of the ego-vehicle. We also observe significant image to image self-attention in head 3 and have provide 2 such visualizations in Fig. 5.

T3 exhibits strong LiDAR to image cross-attention, as opposed to T1 and T2 which have negligible LiDAR to image cross-attention. However, it always attends to a region near the bottom of the image.

T4 shows extensive cross-attention for both image and LiDAR tokens, which indicates that it is effective in aggregating information across the two modalities. We have provided its visualizations in Fig. 7 and Fig. 8.

References

[1] I. Radosavovic, R. P. Kosaraju, R. B. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[2] I. Sobh, L. Amin, S. Abdelkarim, K. Elmadawy, M. Saeed, O. Abdeltawab, M. Gamal, and A. E. Sallab, "End-to-end multi-modal sensors fusion system for urban automated driving," in *Advances in Neural Information Processing Systems (NIPS) Workshops*, 2018. 4

[3] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3d object detection," in *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 4

[4] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, "Multi-task multi-sensor fusion for 3d object detection," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 4

[5] L. A. Rosero, I. P. Gomes, J. A. R. da Silva, T. C. dos Santos, A. T. M. Nakamura, J. Amaro, D. F. Wolf, and F. S. Osório, "A software architecture for autonomous vehicles: Team lrm-b entry in the first carla autonomous driving challenge," *arXiv.org*, vol. 2010.12598, 2020. 4

[6] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 4

[7] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *Proc. Conf. on Robot Learning (CoRL)*, 2019. 4

[8] I. Gog, S. Kalra, P. Schafhalter, M. A. Wright, J. E. Gonzalez, and I. Stoica, "Pylot: A modular platform for exploring latency-accuracy tradeoffs in autonomous vehicles," in *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2021. 4

[9] A. Prakash, K. Chitta, and A. Geiger, "Multi-modal fusion transformer for end-to-end autonomous driving," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 4

[10] K. Chitta, A. Prakash, and A. Geiger, "Neat: Neural attention fields for end-to-end autonomous driving," in *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 4

[11] M. Toromanoff, E. Wirbel, and F. Moutarde, "End-to-end model-free reinforcement learning for urban driving using implicit affordances," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 4

[12] D. Chen, V. Koltun, and P. Krähenbühl, "Learning to drive from a world on rails," in *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 4

[13] B. Jaeger, "Expert drivers for autonomous driving," Master's thesis, University of Tübingen, 2021. 4

[14] R. Chekroun, M. Toromanoff, S. Hornauer, and F. Moutarde, "GRI: general reinforced imitation and its application to vision-based autonomous driving," *arXiv.org*, vol. 2111.08575, 2021. 4

[15] X. Chen, T. Jiang, J. Song, J. Yang, M. Black, A. Geiger, and O. Hilliges, "gdna: Towards generative detailed neural avatars," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 4

[16] Z. Zhang, A. Liniger, D. Dai, F. Yu, and L. Van Gool, "End-to-end urban driving by imitating a reinforcement learning coach," in *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 4

Fig. 3: **T1 Cross-Modal Attention Maps.** For the red query token in the image feature map, we show the top-5 attended tokens in green. For image tokens, T1 always attends to the region just in front of the ego-vehicle in the LiDAR input.

Fig. 4: **T2 Cross-Modal Attention Maps.** For the red query token in the image feature map, we show the top-5 attended tokens in green and highlight the presence of vehicles in the LiDAR point cloud in yellow T2 attends to different regions in the LiDAR input, as opposed to T1.

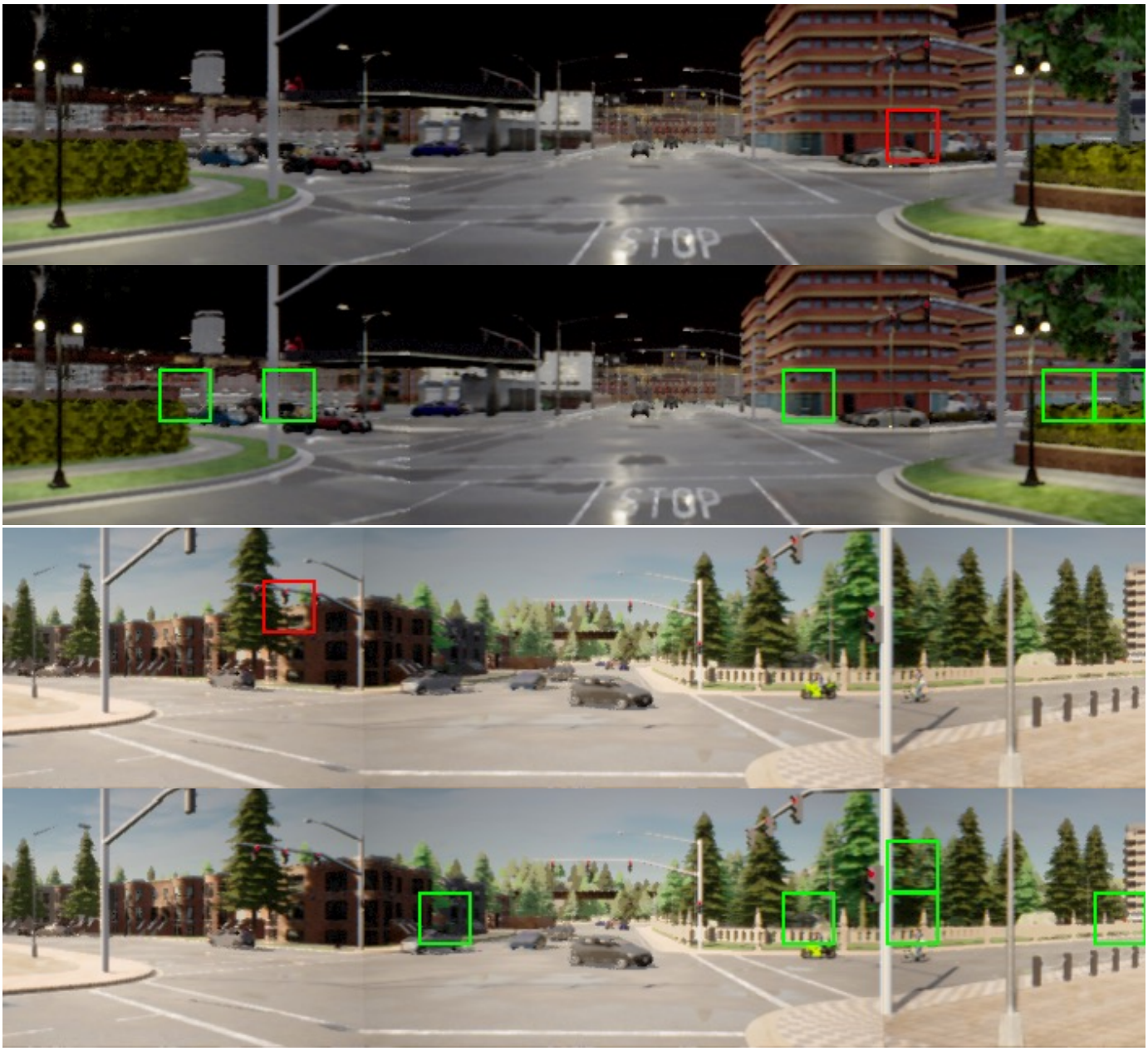

Fig. 5: **T2 Self-Attention Maps.** For the red query token in the image feature map, we show the top-5 attended tokens in green. T2 exhibits significant image to image self-attention in head 3.

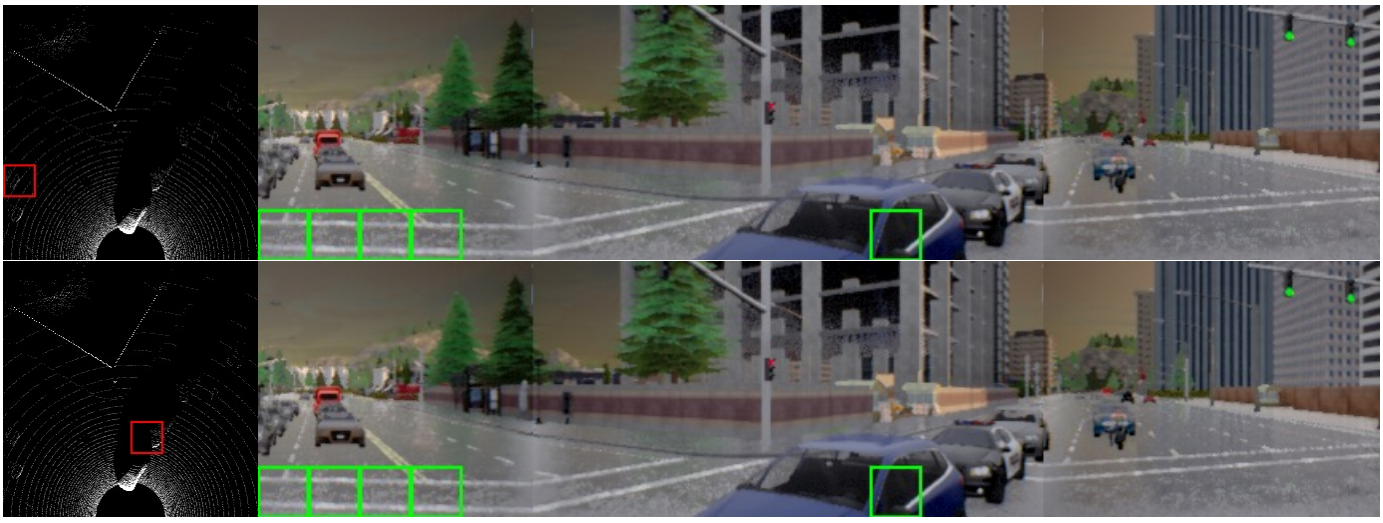

Fig. 6: **T3 Cross-Modal Attention Maps.** For the red query token in the LiDAR feature map, we show the top-5 attended tokens in green. In contrast to T1 and T2, T3 shows strong LiDAR to image cross-attention but it always attends to a region at the bottom of the image.

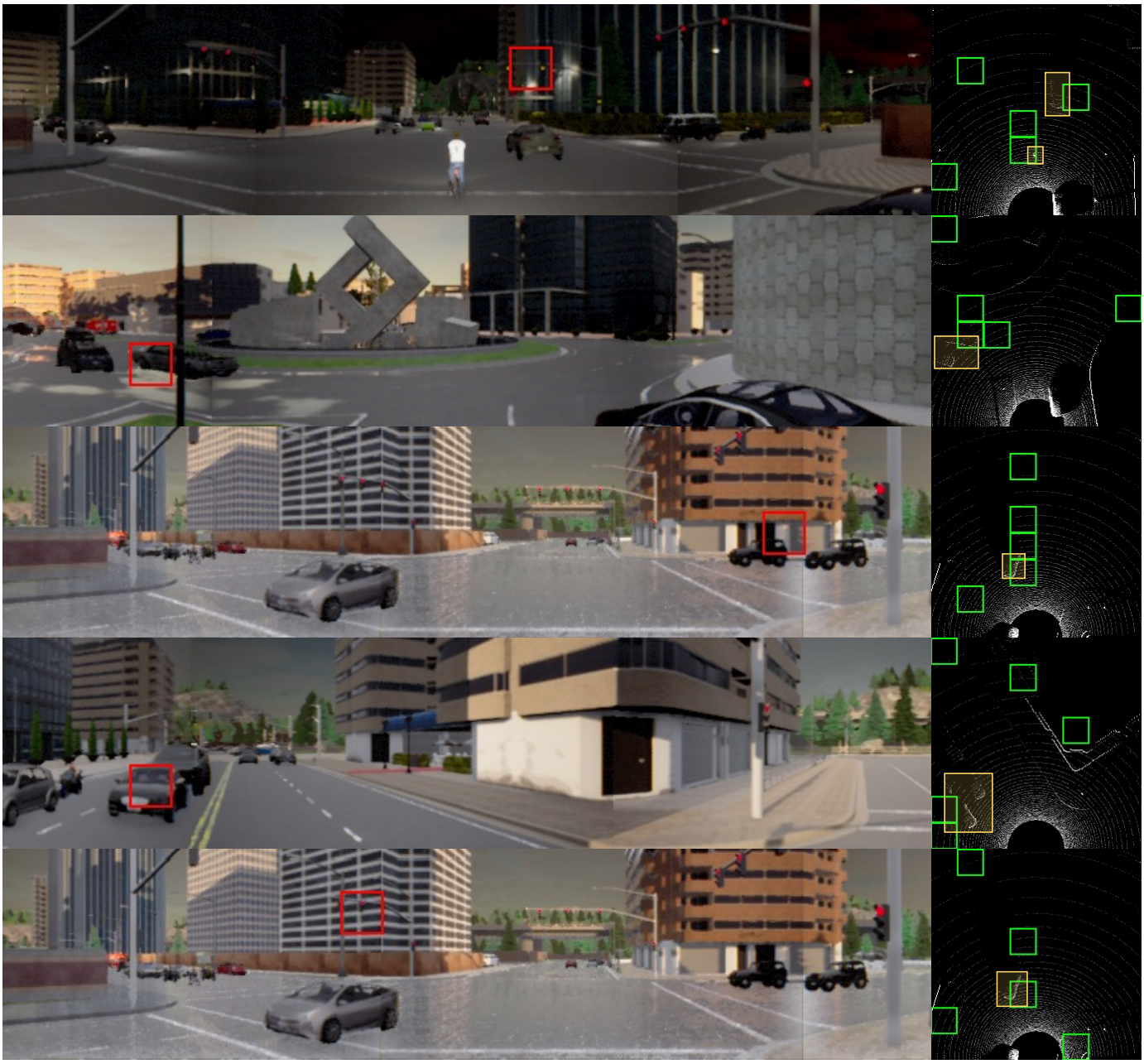

Fig. 7: **T4 Attention Maps.** For the red query token in the image feature map, we show the top-5 attended tokens in green and highlight the presence of vehicles in the LiDAR point cloud in yellow. TransFuser attends to areas near vehicles and traffic lights at intersections.

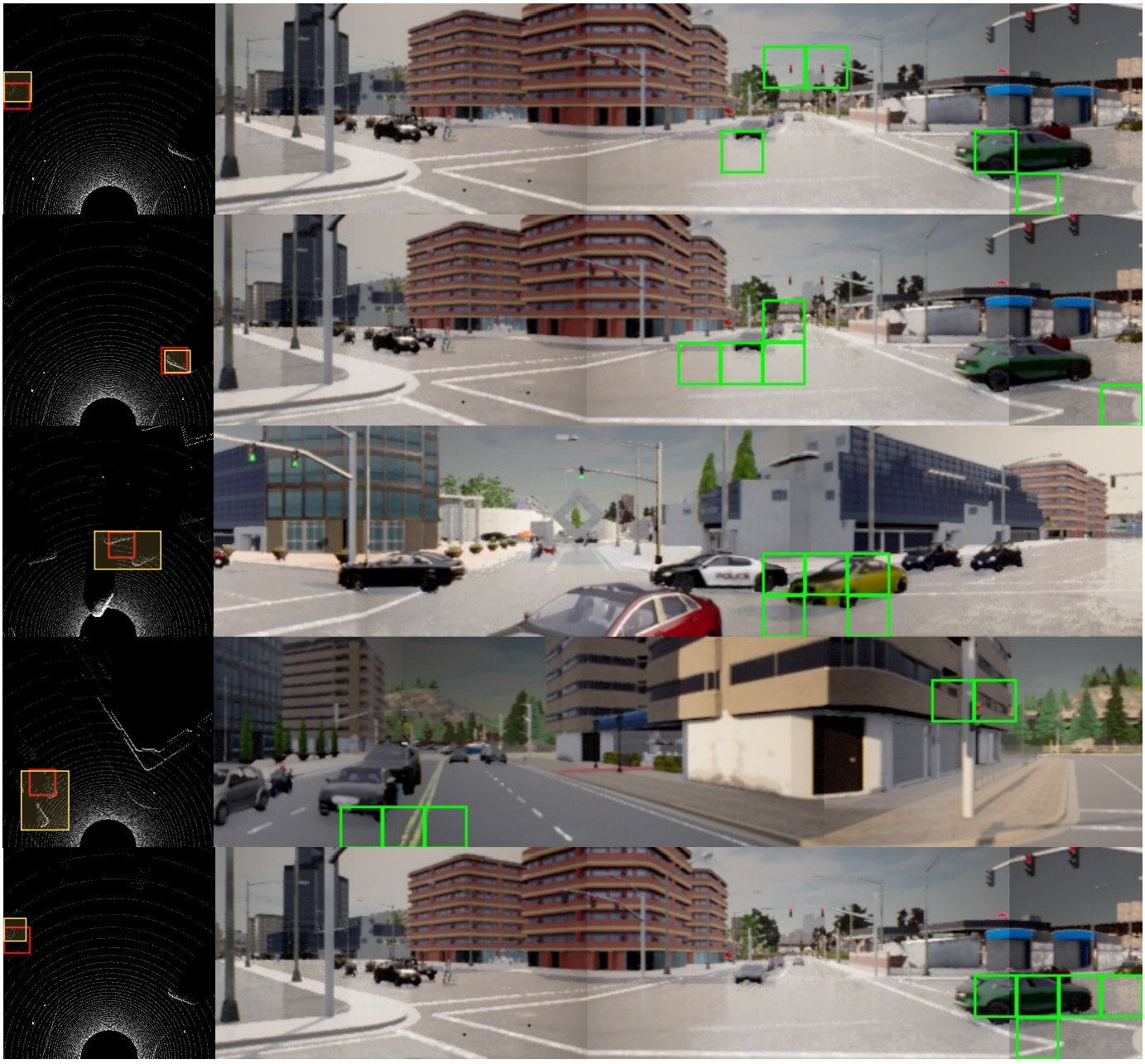

Fig. 8: **T4 Attention Maps.** For the red query token in the LiDAR feature map, we show the top-5 attended tokens in green and highlight the presence of vehicles in the LiDAR point cloud in yellow. TransFuser attends to areas near vehicles and traffic lights at intersections.