GASP: Unifying Geometric and Semantic Self-Supervised Pre-trained for Autonomous Driving

Appendix

A. Baseline reimplementation

We base our work on the method described in [2]. However, since their implementation is closed-source we reimplemented their method according to their paper, their predecessor [1] as well as personal correspondence with the authors [46]. To verify our reimplementation, we report the number of parameters in the original implementation and our version in Tab. 5. Note that the decoder head matches perfectly (up to the value number reported in the original paper) and that the encoder is within a 3% margin.

Parameter count	Uno encoder	Uno decoder
Reported	17.4M	0.06M
Reimplementation	16.9M	0.06M

Table 5. Parameter count in original UnO (as reported in the paper) and our reimplementation.

In addition, we also verify our reimplementation by running identical experiments and comparing to the results reported in the paper. In Fig. 10, we show the *Average Precision* for semantic forecasting across number of training samples for fine-tuning with both frozen and unfrozen sensor encoder.

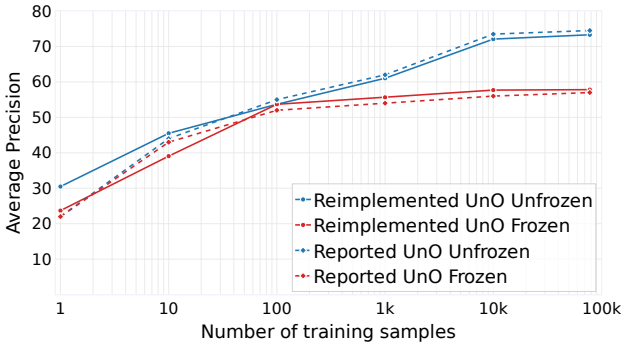

Figure 10. Semantic forecasting AP across number of training samples for our reimplemented baseline (solid) and the results reported in [2] (dashed).

Lastly, we show the results for 4D-occupancy in Tab. 6 using recall at precision 70% as the metric, following the original paper [2]. Our results show higher performance numbers, which may be due to either improvements in our set-up or slight differences in evaluation settings. Moreover, applying our training recipe from Sec. 4 further enhances performance, while additional training improvements, such as rotation augmentation (Sec. 3.2) and handling missing rays (Sec. 3.2), provide an extra boost. For a fair comparison of our main contributions, we refer to the best-performing version as *UnO*.

Name	Details	R@P 70%
UnO†	reported in [2]	67.0
UnO^{r}	reimplemented	72.0
UnO^{+}	our training recipe	77.6
UnO	our improvements	79.4
GASP	proposed method	81.6

Table 6. 4D occupancy recall at precision 70% for our reimplementation and original implementation [2].

B. Additional results

Here, we show additional results from our evaluation. For completeness we also report the numbers visualized as graphs in the main manuscript.

B.1. Semantic BEV forecasting

First, in Tab. 7, we show the performance of GASP and UnO on the semantic BEV forecasting task using the Soft-IoU metric. The results follow the same trend as the Average Precision metrics shown in Tab. 8.

Enc.	PT	Labeled data samples 1	10	10^2	10^3	10^4	10^5
❄	UnO	$11.6^{\pm 1.8}$	$14.8^{\pm 2.0}$	$21.3^{\pm 1.4}$	21.3	22.3	22.6
	GASP	$19.0^{\pm 2.6}$	$23.1^{\pm 4.5}$	$28.0^{\pm 0.9}$	30.5	29.4	31.4
🔥	UnO	$16.6^{\pm 2.4}$	$22.5^{\pm 3.9}$	$25.6^{\pm 2.2}$	27.4	39.9	39.5
	GASP	$26.8^{\pm 2.2}$	$27.9^{\pm 1.4}$	$29.0^{\pm 2.8}$	36.0	39.6	41.0

Table 7. Semantic BEV forecasting performance (Soft-IoU) for GASP and UnO across different number of fine-tuning samples with frozen (❄) and unfrozen (🔥) sensor encoder.

For completeness, we show the detailed numbers from Fig. 5 in Tab. 8

Enc.	PT	Labeled data samples 1	10	10^2	10^3	10^4	10^5
❄	UnO	$17.8^{\pm 3.2}$	$34.8^{\pm 2.1}$	$52.4^{\pm 0.8}$	55.6	57.7	57.8
	GASP	$30.0^{\pm 7.3}$	$52.7^{\pm 2.0}$	$61.2^{\pm 0.4}$	65.7	66.2	66.1
🔥	UnO	$25.6^{\pm 3.7}$	$41.4^{\pm 4.2}$	$52.4^{\pm 0.8}$	61.1	72.1	73.3
	GASP	$45.7^{\pm 2.3}$	$56.0^{\pm 0.9}$	$61.1^{\pm 0.9}$	68.8	73.6	74.7

Table 8. BEV semantic forecasting performance (AP) showed across number of fine-tuning samples with frozen (❄) and unfrozen (🔥) sensor encoder.

B.2. Map segmentation

In Tab. 9 we report detailed results for the map segmentation task showed in Sec. 4.2.

B.3. Semantic 4D occupancy

As an extension to the 3D semantic occupancy (BEV forecasting) in Sec. 4.3 we can post-train the model on also including the height in the prediction, namely 4D (3D+time) occupancy. Tab. 10 reports the Average Precision performance for UnO and GASP on the vehicle class. We note that GASP outperforms the baseline for all number of labeled samples.

Enc.	PT	Labeled data samples 1	10	10^2	10^3	10^4	10^5
❄	UnO	$4.7^{\pm 2.2}$	$4.9^{\pm 1.9}$	$17.6^{\pm 0.7}$	26.4	31.0	34.1
	GASP	$9.6^{\pm 3.1}$	$12.1^{\pm 3.7}$	$28.7^{\pm 0.8}$	35.2	39.1	40.0
🔥	✗	$3.3^{\pm 0.2}$	$4.1^{\pm 0.7}$	$10.6^{\pm 3.6}$	34.1	42.7	45.6
	UnO	$4.8^{\pm 2.4}$	$5.0^{\pm 1.8}$	$17.9^{\pm 1.0}$	34.2	37.5	44.9
	GASP	$9.2^{\pm 3.4}$	$12.5^{\pm 3.0}$	$28.6^{\pm 0.9}$	40.0	43.2	45.7

Table 9. Map segmentation mIoU (mean and std. dev) across a number of labeled samples with frozen (❄) and unfrozen (🔥) sensor encoder. GASP outperforms the baseline across all amounts of training samples indicating that the BEV features contain a richer BEV representation.

Enc.	PT	Labeled data samples 100	10^3	10^4	10^5
🔥	Scratch	18.4	38.7	44.6	54.7
	UnO	29.1	43.4	45.6	59.9
	GASP	41.3	47.1	49.0	60.9

Table 10. Average Precision (AP) for Semantic 4D Occupancy Forecasting. We evaluate performance across different numbers of fine-tuning samples and report results for vehicle segmentation.

B.4. nuScenes Dataset

To verify the generalizability of GASP, we conduct experiments on the nuScenes dataset [9]. The results make for the same conclusions as in the main manuscript (see Sec. 4).

Enc.	Method	4D-occ ↑ n/a	Sem. Forecasting ↑ Labeled samples 10^2	10^3	10^4	Map Seg. ↑ Labeled samples 10^2	10^3	10^4
❄	UnO	93.9	24.2	34.2	41.0	8.0	12.7	13.8
	GASP	95.1	33.0	40.6	48.8	11.1	15.5	16.4
🔥	None	n/a	10.0	40.2	54.0	6.7	15.7	20.6
	UnO	n/a	28.7	45.7	56.2	8.2	15.3	20.4
	GASP	n/a	35.2	49.4	59.0	12.0	18.3	20.7

Table 11. Results on nuScenes show the generalization of GASP. We show performance directly obtained from pre-training (4D occ. P@R70) and its generalization to downstream tasks (Semantic BEV Forecasting and Map Segmentation) when finetuned on different amounts of labeled samples. We ablate each component added to UnO with the sensor encoder frozen (❄) and unfrozen (🔥) as well as performance with no pre-training.

B.5. Computational efficiency

To assess the computational complexity we measure the time for executing a forward pass for our components. We evaluate on a NVIDIA A100 GPU, measuring runtime over 5 training samples and can be seen in Tab. 12. The majority of the computations is spent on encoding the raw sensor data to a BEV representation. At test-time inference this is the only part that we keep from the pre-training protocol, and additional compute will depend on specific downstream tasks that are chosen and not evaluated here. During pre-training we also generate the queries and their target labels, as well as decoding the output. Both sampling and decoding the 1.8M queries have minor impact on runtime compared to the BEV encoder. Note that the large number of queries makes it essential to sample the queries on the GPU, enabled by provided code, rather than the CPU. It is also worth noting that the decoder head Sec. 3.1 is deliberately designed to be lightweight, ensuring that the main expressivity lies on the BEV encoder. Nevertheless, if one wishes to use the occupancy, VFM, or ego-path heads employed during pre-training also at test-time, the queries can be pre-computed (*e.g.* uniformly sampled within a region of interest), such that only the decoding step needs to be executed.

Block	**Mean & Std (ms)**
Encode BEV	124.345 ± 7.264
Sampling occupancy (CPU)	443.320 ± 108.979
Sampling occupancy (GPU)	7.391 ± 0.041
Sampling VFM (GPU)	6.839 ± 0.522
Decode occupancy	7.664 ± 0.755
Decode VFM	6.958 ± 0.187

Table 12. Per-block runtime summary (mean and sample std over $n = 5$ runs).

C. Additional ablations

For completeness, we present the full results of our ablation studies, highlighting the contribution of each component to the final performance.

C.1. VFM feature dimensions

We vary the number of the principal component analysis reduced components (8, 16, and 32) from DINOv2 that we train to predict. Tab. 13 reports the impact on performance. We conclude that learning to predict the 16 most important components yields good results across all three tasks.

Enc.	dim	**4D-occ ↑**	**Sem. Forecasting ↑** Labeled samples			**Map Seg. ↑** Labeled samples		
			10^2	10^3	10^5	10^2	10^3	10^5
❄	8	81.1	50.4	54.7	54.4	28.5	36.8	40.7
	16	81.6	59.3	64.5	64.1	30.6	35.2	40.0
	32	80.2	59.3	62.5	63.0	26.9	34.7	38.6
🔥	8	n/a	53.4	62.7	76.4	29.7	39.3	44.7
	16	n/a	59.8	67.3	77.0	29.1	40.0	45.7
	32	n/a	60.4	67.1	76.7	29.1	38.8	45.3

Table 13. Performance across different downstream task when varying the number of DINOv2 dimensions used in the regression objective.

C.2. Missing rays

Here, we ablate the use of inferred missing rays during pre-training. We measure performance on geometric 4D occupancy using the recall at precision 70% and report the numbers in Tab. 14. As noted in the main manuscript, we do not see any quantitative improvements, but rather qualitative ones. We hypothesize that this is because the metric inherently disregards the regions where this supervision helps.

Missing rays	✗	✓
GASP	81.6	81.0

Table 14. Missing rays. Recall at precision 70%

C.3. Augmentation

In addition to the rotation augmentation outlined in the main manuscript, we also experiment with translation, jitter augmentations, and the number of feature dimensions in the DINOv2 features. We measure geometric 4D occupancy performance as measured by the recall at precision 70%.

Rotation augmentation: For completeness, we, apart from GASP, also show that UnO benefits from rotation augmentation in Tab. 15.

Translation augmentation: Similarly to the rotation augmentation, we can augment the training data such that the vehicle is translated in x and y directions. We ablate the effects of adding such translation augmentation, and Tab. 16 displayed the recall at precision 70%. We don't see any major improvements using this augmentation and opt to use the more impactful rotation augmentation. We hypothesize that the data already includes a large variety of lateral and longitudinal shifts.

Jitter augmentation: We also experiment with jitter augmentation, which aims to up-sample negative queries close to the positive queries by adding a jitter parameter τ to the negative query equation:

$$\mathbf{q}_i^- = \mathbf{o}_i + (\mathbf{p}_i - \mathbf{o}_i)d^\tau \tag{7}$$

where $d \sim \mathcal{U}(0, 1)$.

PT	Rotation augmentation (4D-occupancy ↑)					
	±0°	±5°	±10°	±20°	±45°	±90°
UnO	77.6	79.2	78.3	79.4	78.7	74.1
GASP	78.1	80.1	81.7	81.6	78.1	77.2

Table 15. Rotation augmentation. Recall at precision 70%

PT	Translation augmentation			
	–	±0.5m	±1.5m	±3.0m
GASP	78.1	79.0	78.3	76.0

Table 16. Translation augmentation. Recall at precision 70%

Our initial intuition—that increasing jitter would lead to sharper geometry learning—did not hold, as performance declines with higher jitter values. Conversely, reducing jitter also appears to negatively impact model performance.

PT	Jitter					
	0.6	0.8	1.0	1.2	2.0	3.0
GASP	80.0	80.1	81.6	80.7	75.0	68.5

Table 17. Jitter. Recall at precision 70%

C.4. VFM loss

We ablate the loss function used to learn our VFM features in Tab. 18. We again compare performance on geometric 4D occupancy using the recall at precision 70% metric. The Smooth-L1 loss reduces the performance and we opt to use the L1-loss.

PT	L1	Smooth-L1 (β)		
		0.1	0.5	1.0
GASP	81.6	78.9	79.3	79.3

Table 18. VFM-loss using DINOv2 features. Smooth L1 does not give any additional benefit compared to regular L1. Recall at precision 70%

C.5. Ego-path vs Ego-trajectory

Full trajectory requires reasoning about traffic signals, interactions with other agents, vehicle dynamics, and exact speed profiles, while ego-path focuses on the feasible spatial corridors the ego vehicle could take. As such, ego path provides a high-level inductive bias for learning drivability without committing to specific timing, accelerations, or policy-dependent decisions while also being more efficient to generate training data (queries) for. However, by adding a temporal dimension to the ego-path query generation we can study the difference in modeling ego-path vs ego-trajectory during pre-training. As shown in Tab. 19, simplifying the learning task by using ego-path instead of ego-trajectory has negligible effect on downstream performance. We therefore view ego-path supervision as an efficient way to encourage models to learn where it is possible to drive in the future, while reserving full trajectory prediction as a downstream evaluation task to test planning-relevant reasoning.

Enc.	**Method**	**4D-occ ↑**	**Sem. Forecasting ↑**			**Map Seg. ↑**		
		n/a	Labeled samples			Labeled samples		
			10^2	10^3	10^5	10^2	10^3	10^5
❄	Path	81.6	61.2	65.7	66.2	28.7	35.2	39.1
	Trajectory	81.7	61.0	65.3	65.9	28.9	34.9	39.4

Table 19. Performance of GASP when trained with ego-path vs ego-trajectory. Simplifying the supervision signal has negligible impact.

D. Qualitative analysis

This section aims to provide additional qualitative results. Firstly, we showcase a failure case of naively projecting lidar points into camera images, which can produce artifacts where points behind foreground objects (e.g., vehicles) are incorrectly associated with image features. As the lidar sensor is typically mounted higher than the cameras, its rays may pass over occluders and strike surfaces that are invisible to the camera view, leading to noisy supervision when used for training. Our depth-filtering mechanism mitigates this issue by enforcing per-pixel minimum-depth consistency, ensuring that only the closest visible points contribute features for learning. This reduces the impact of spurious "shadow" points and provides cleaner training targets for semantic representation learning.

Model predictions after pre-training can be seen in Figs. 12 to 15. Notably, these predictions are smooth and continuous, without the shadow artifacts. This indicates that the model has learned to produce coherent semantic features and to ignore residual noise introduced by projection errors.

In Fig. 12 one can see an overview of both semantic and occupancy-field information, and in Fig. 13 a bird's-eye view of point-cloud inputs, features, and occupancy is provided. One can clearly see that the VFM features are different across *e.g.* vehicles, trees, buildings, and the road. These examples not only show the quality of the information that the learned representation embeds, but also highlighting interesting emergent properties of our pre-training strategy. For instance, in Fig. 14, the multimodal outputs on path probabilities in a three-way intersection are depicted. Further, in Fig. 15, we show that feature prediction in GASP has the potential of capturing fine-grained, and important, scene details such as the distinction between road and sidewalk. The VFM features of the lane divider is also slightly different than the center of the lane.

For the BEV semantic forecasting task, Sec. 4.3, complementary qualitative results are provided in Fig. 16 for three scenes, a common case, a case with unusual objects, and a more complex case.

Outputs from the map segmentation task are shown in Fig. 17, for different number of samples in the training set given a frozen GASP encoder. Here, we can see the models ability to predict the online map well despite using an order of magnitude less supervision data.

Qualitative results connected to the ego-trajectory prediction task can be viewed in Fig. 18.

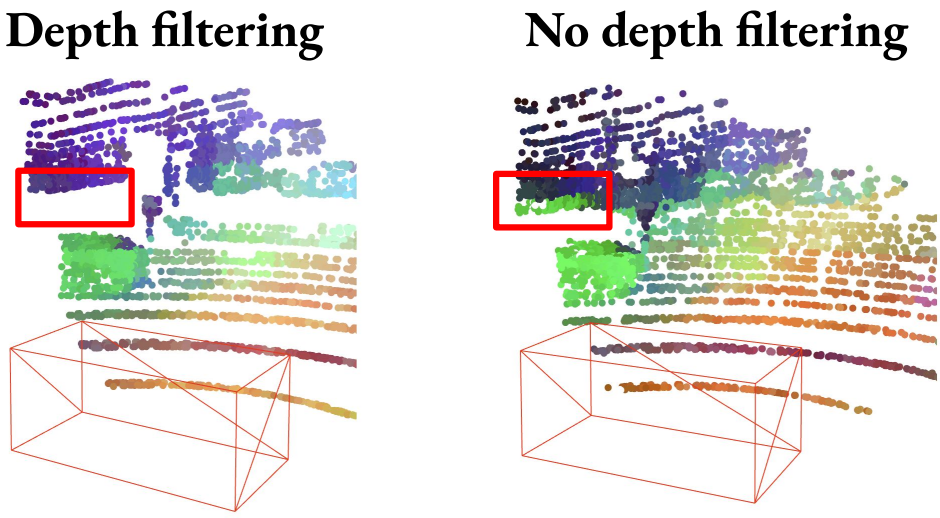

Figure 11. Naively projecting LiDAR points into the image creates "shadow" artifacts, where points behind foreground objects (e.g., vehicles) are incorrectly assigned image features that are occluded from the camera's view. To address this, we apply per-pixel minimum-depth filtering and retain only the closest visible points, $\mathcal{P}_{\text{vis}}$. The bounding box marks the ego vehicle, and colors show vision foundation model features reduced to 3D via PCA. The red square highlights where vehicle features (green) have a shadow on the wall behind it (purple) when not using depth filtering, and that these can be removed using the depth filtering from Sec. 3.2.

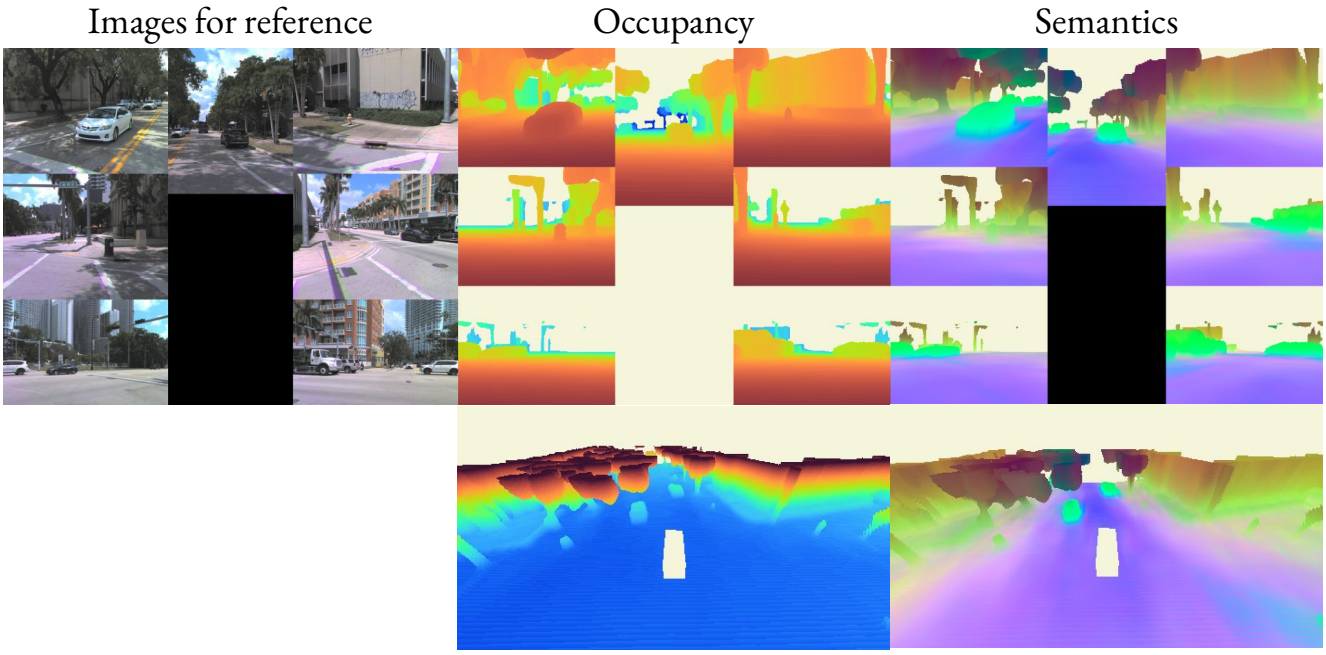

Figure 12. Occupancy and VFM features projected into the camera view. The white box representing the ego vehicle has been added for illustrative purposes. The predicted VFM features have been reduced to three components via PCA and visualized as RGB. Semantic structure is evident in the feature space: for example, vehicles are grouped together and shown in green, while roads appear in purple. This demonstrates that the model has learned to separate different semantic classes in its feature representation.

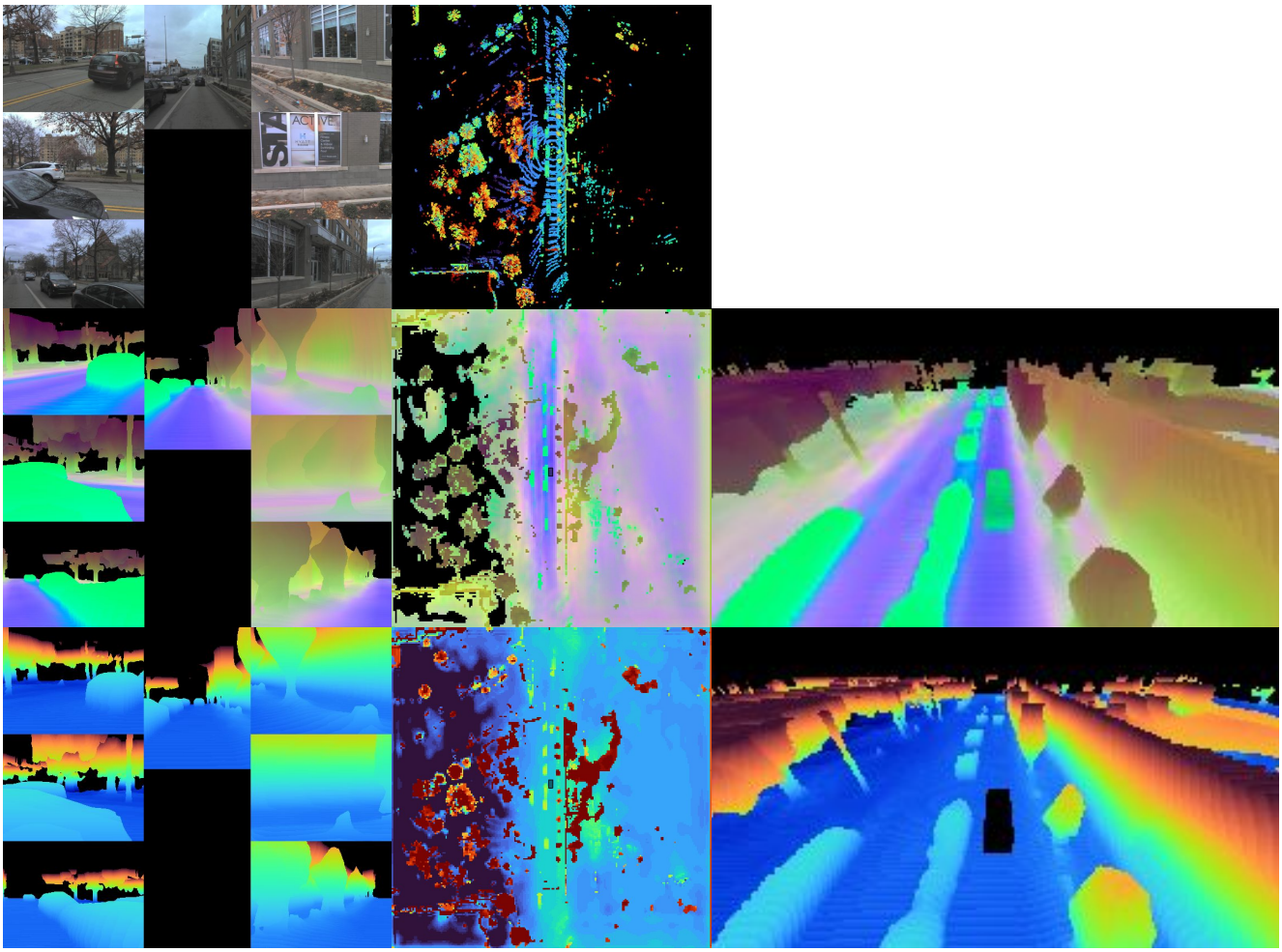

Figure 13. Occupancy and VFM features projected into a camera view, a view from above the ego vehicle, and a bird's-eye view. *E.g.* road users are clearly separated in the feature space from buildings, trees, and the road.

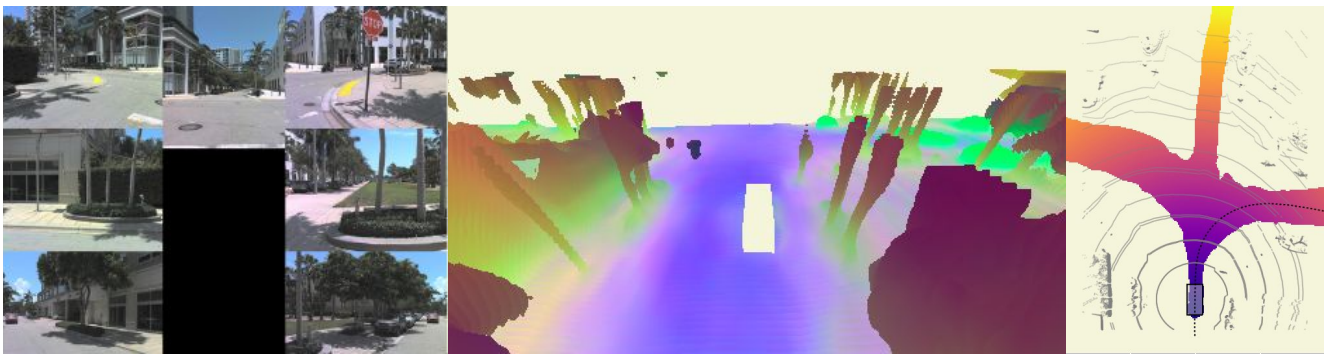

Figure 14. VFM features and predicted ego paths at a three-way intersection. Despite being supervised with only a single recorded path, the model outputs multiple plausible paths, capturing the inherent multimodality of the scene.

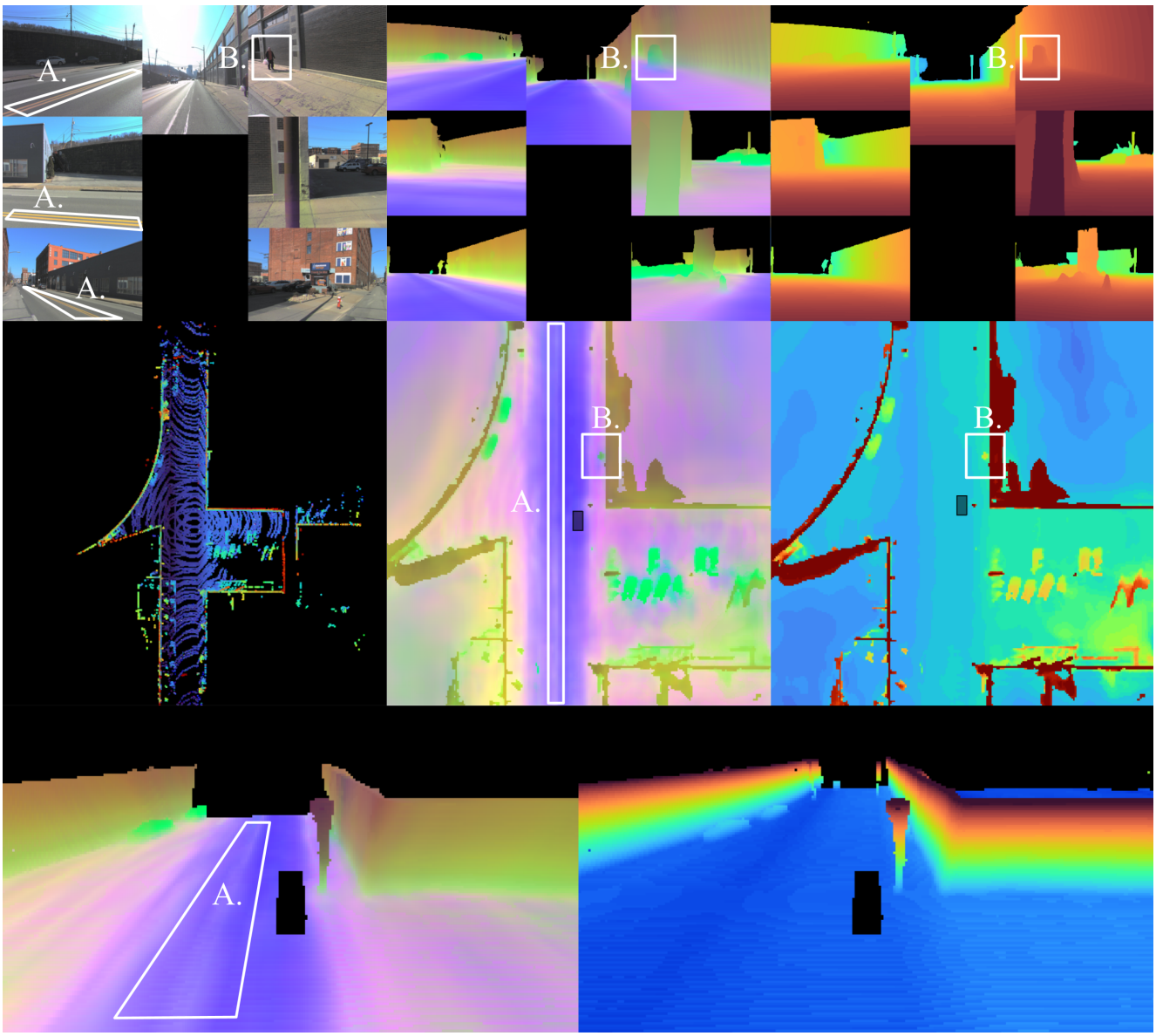

Figure 15. A qualitative example of the feature-level information predicted by the representation produced by GASP, showcasing its capability of contrasting otherwise diffuse scene elements such as the lane-dividers (marked A.) or the person carrying a bag (marked B.) from the background.

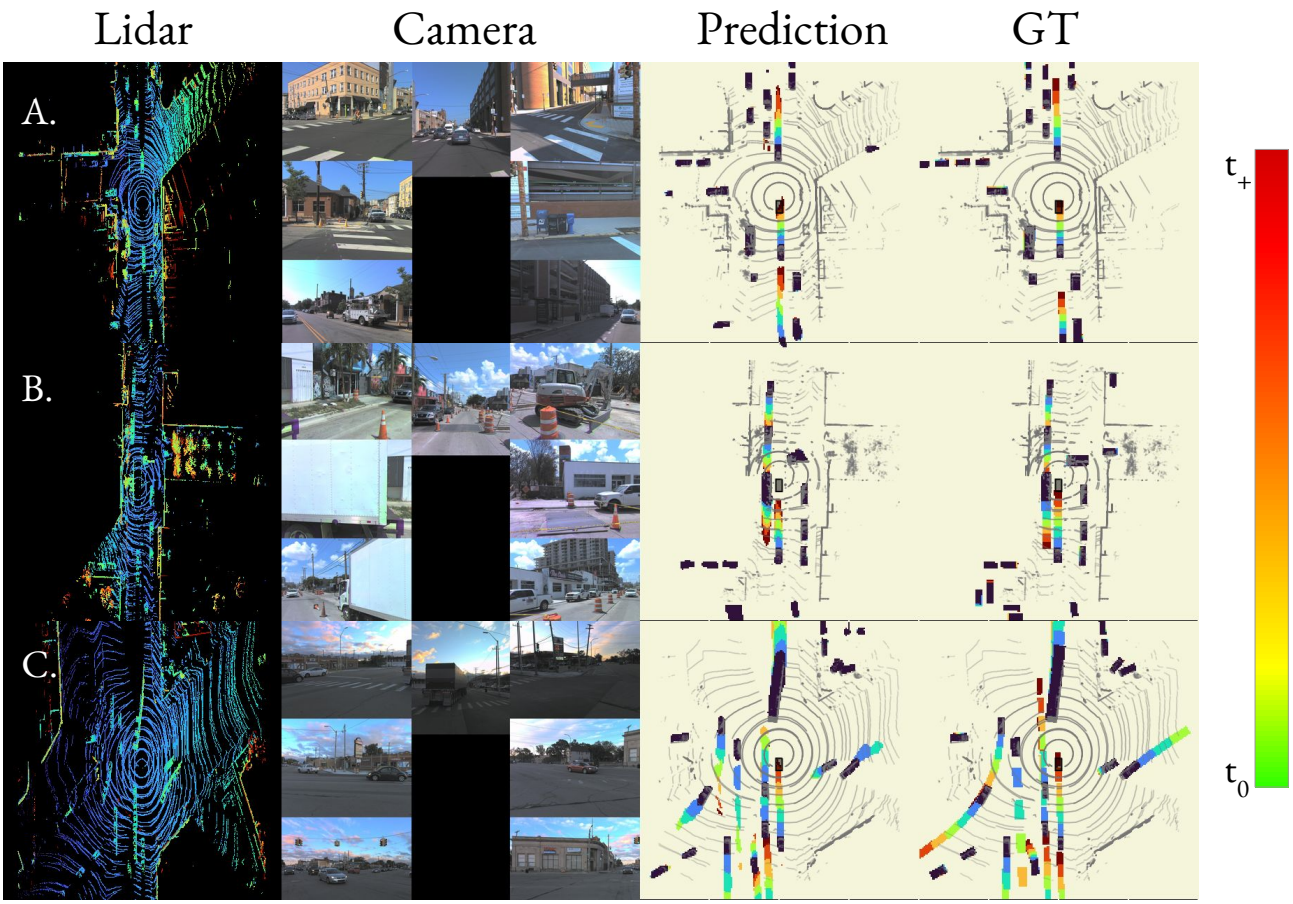

Figure 16. BEV segmentation forecasting results showing A) a typical scenario, B) a scenario with uncommon road users (in this case an excavator), and C) a more complex scenario. An unfrozen GASP representation with 100 samples available in the post-training task is used.

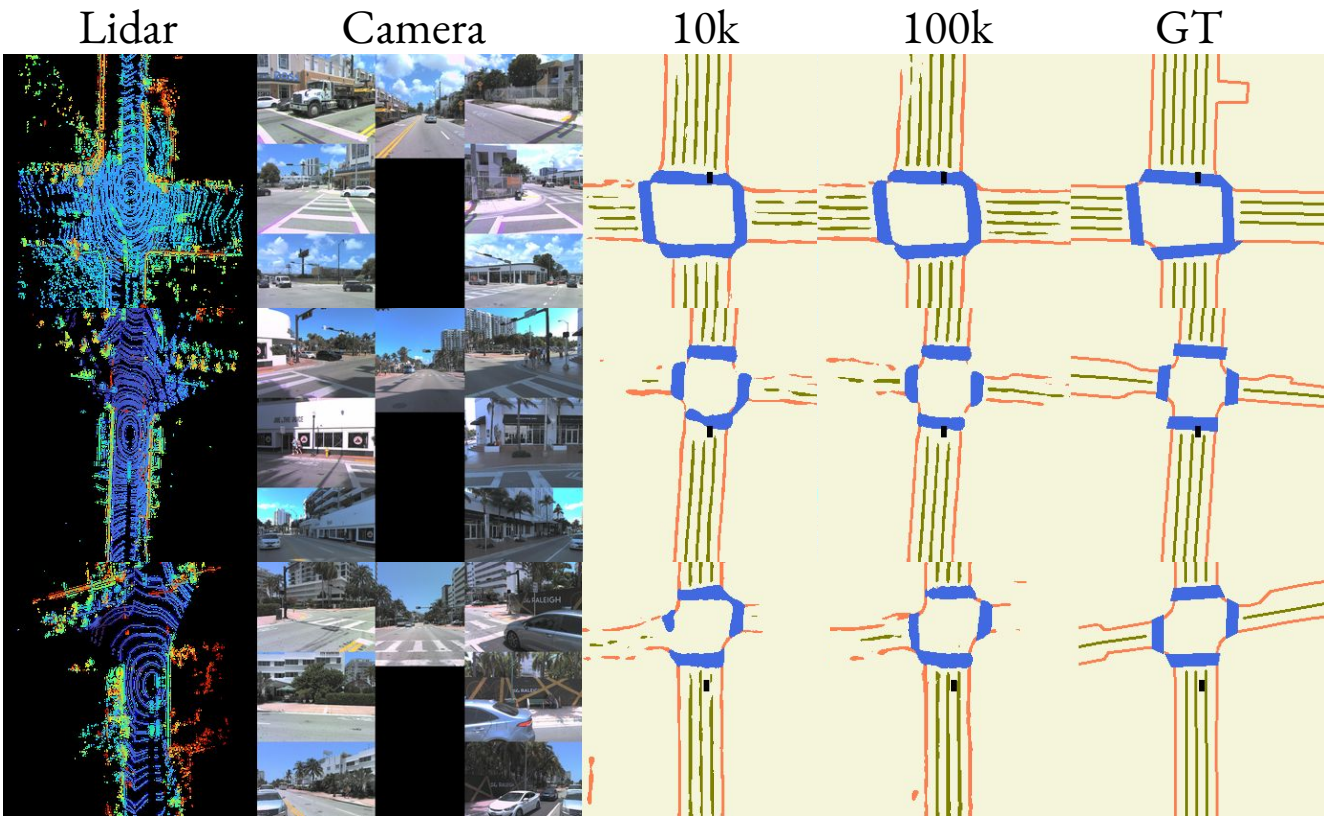

Figure 17. Prediction results from the map segmentation post-training task of GASP with frozen encoder. Note that predictions are only made based on lidar input. Camera images are only provided as visual clarity for the reader regarding what scene is being predicted.

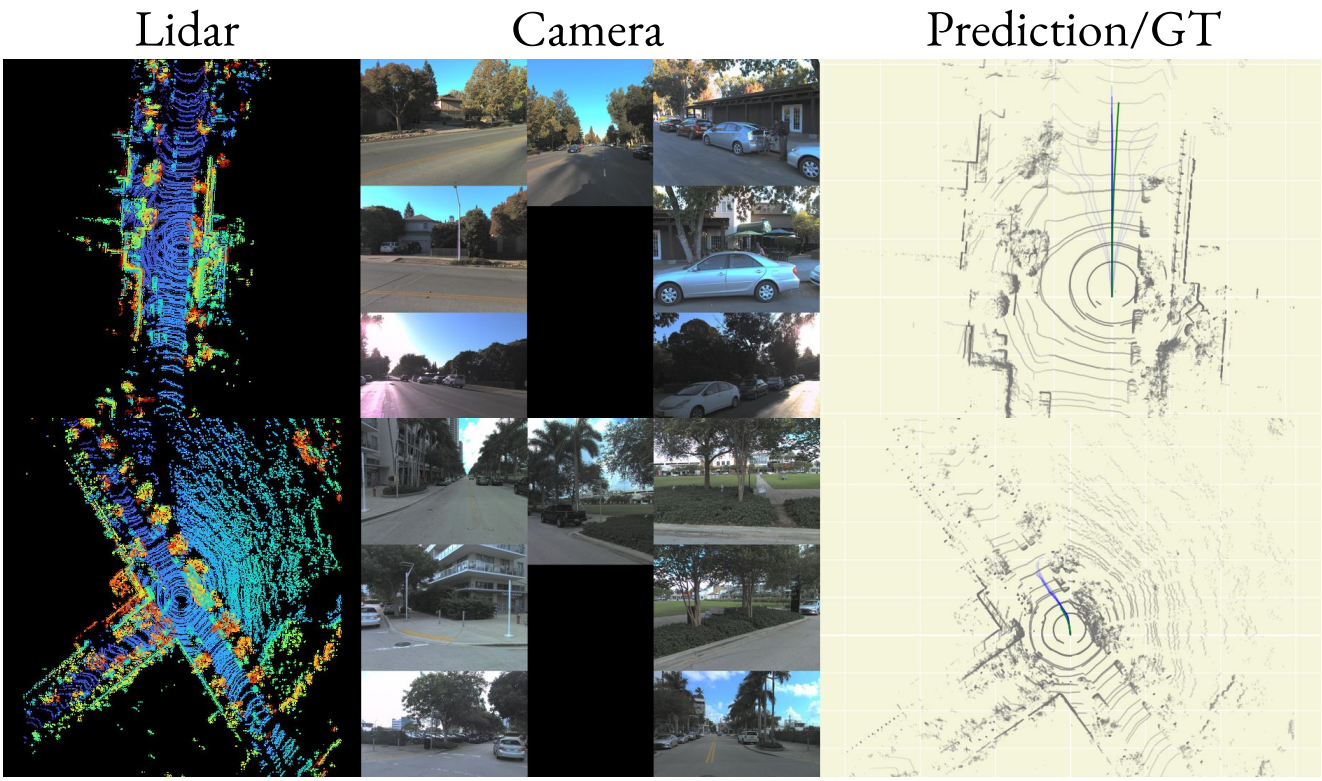

Figure 18. Ego-trajectory prediction results using a frozen GASP representation. Expert trajectories, the groundtruth, are shown in green while predictions are shown in blue. Note that camera inputs are only provided as visual support for the reader and are not part of the prediction.