Feed-Forward SceneDINO for Unsupervised Semantic Scene Completion

Supplementary Material

Aleksandar Jevtić* [1] Christoph Reich* [1,2,4,5] Felix Wimbauer [1,4]
Oliver Hahn [2] Christian Rupprecht [3] Stefan Roth [2,5,6] Daniel Cremers [1,4,5]
[1]TU Munich [2]TU Darmstadt [3]University of Oxford [4]MCML [5]ELIZA [6]hessian.AI *equal contribution
https://visinf.github.io/scenedino

In this appendix, we provide further implementation details, including dataset properties and an overview of SceneDINO's computational complexity (*cf*. Sec. A). We discuss our multi-view feature consistency evaluation approach (*cf*. Sec. B). Next, we provide additional qualitative and quantitative results (*cf*. Sec. C), including failure cases. Finally, we discuss the limitations of SceneDINO and suggest future research directions (*cf*. Sec. D).

A. Reproducibility

Here, we provide further implementation details, information about the utilized dataset, and computational complexity details to ensure reproducibility. Note that our code is available at https://github.com/tum-vision/scenedino.

A.1. Implementation details

We implement SceneDINO in PyTorch [123] and build on the code of BTS [108], STEGO [33], and S4C [38]. Our encoder-decoder (pre-trained DINO-B/8 and randomly initialized dense prediction decoder) produces per-pixel embeddings of dimensionality $\mathrm{D}_{\mathbf{E}} = 256$. Based on these embeddings, the two-layer MLP ϕ (hidden dimensionality 128) predicts 64 features. As rendering features is expensive, requiring multiple forward passes through the MLP, ϕ predicts 64 features. We employ another MLP to up-project again to the full dimensionality $\mathrm{D} = 768$; this MLP is learned with SceneDINO and can up-project both 3D features and 2D rendered features. We train for 100 k steps with a base learning rate of 10^{-4}, dropping to 10^{-5} after 50 k steps. We train using a batch size of 4, extracting 32 patches of size 8×8 per image. These patches align with the per-patch DINO target features. For our feature field loss formulation (*cf*. Sec. 3.2), we use the loss weights $\lambda_{\mathrm{p}} = 1, \lambda_{\mathrm{s}} = 0.001, \lambda_{\mathrm{f}} = 0.2, \lambda_{\mathrm{fs}} = 0.25$.

The MLP head h (hidden dimensionality 768) produces 64 distilled features K. We perform distillation for 1000 steps with a learning rate of $5 \cdot 10^{-4}$. We train using a batch size of 4, 5 center points, a feature batch of size 576, and cluster with $\mathrm{C} = 19$. For kNN sampling, we use $k = 4$. The feature buffer holds 256 feature batches. The loss term in Eq. (9) is parameterized with $\lambda_{\text{self}} = 0.08$, $\lambda_{k\text{NN}} = 0.43$, and $\lambda_{\text{rand}} = 0.67$. For the similarity thresholds, we use $b_{\text{self}} = 0.44$, $b_{k\text{NN}} = 0.18$, and $b_{\text{rand}} = 0.87$.

We follow standard practice in 2D unsupervised semantic segmentation [17, 31, 33, 51, 78, 92, 95] by applying Hungarian matching [57] to align our pseudo semantics. For SSC validation, we map down to 15 semantic classes while following existing work [31, 33] for 2D validation and map to 19 semantic classes.

A.2. Datasets

We provide additional details about the datasets utilized to train and evaluate SceneDINO.

KITTI-360 [64, 66] provides video sequences from a moving vehicle equipped with a forward-facing stereo camera pair and two side-facing fisheye cameras. In future frames, the fisheye views capture additional geometric and semantic cues of regions occluded in the forward-facing view. For training, we resample the fisheye images into perspective projection. We focus on an area approximately 50 meters ahead of the ego vehicle. Assuming an average velocity of $30 - 50$ km/h, side views are randomly sampled 1 – 4 seconds into the future. Given a frame rate of 10 Hz, this translates to 10 – 40 time steps. Each training sample consists of eight images: four forward-facing views (including the input image) and four side-facing views.

To evaluate our predicted field in SSCBench-KITTI-360, we follow the evaluation procedure of S4C [38]. The voxel predictions are evaluated in three different ranges: $12.8\,\mathrm{m} \times 12.8\,\mathrm{m} \times 6.4\,\mathrm{m}$, $25.6\,\mathrm{m} \times 25.6\,\mathrm{m} \times 6.4\,\mathrm{m}$, and the full range $51.2\,\mathrm{m} \times 51.2\,\mathrm{m} \times 6.4\,\mathrm{m}$. For each voxel, multiple evenly distributed points are sampled from the semantic field. The predictions are aggregated per voxel by taking the maximum occupancy and weighting the class predictions accordingly.

Cityscapes [19] consists of 500 high-resolution and densely annotated validation images of ego-centric driving scenes. For validation, Cityscapes uses a 19-class taxonomy. We leverage the Cityscapes validation samples at a resolution of 640×192 for our domain generalization experiments (2D semantic segmentation).

BDD-100K [115] is a driving scene dataset obtained from urban areas in the US. BDD-100K contains 1000 semantic

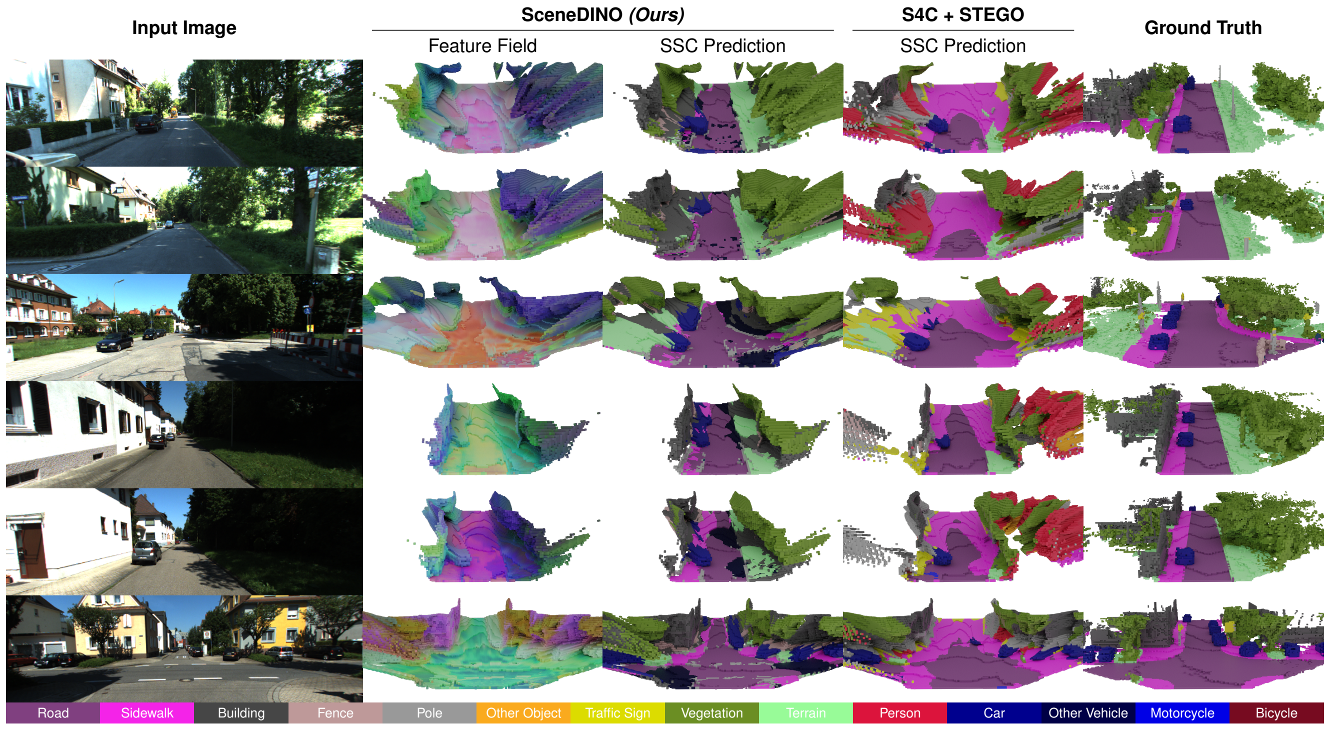


Figure 6. **3D qualitative SSC comparison on KITTI-360.** We provide additional qualitative results, visualizing the input image, SceneDINO's predicted feature field using the first three principal components, and SSC prediction, the SSC prediction of our baseline S4C+STEGO, and the SSC ground truth. We only visualize surface voxels within the field of view for the sake of clarity.

segmentation validation images. The semantic taxonomy follows the 19-class Cityscapes definition. For domain generalization experiments, we utilize BDD-100K images at a resolution of 640×192.

RealEstate10K [119] is a large-scale dataset containing videos of real-world indoor and outdoor scenes, primarily sourced from YouTube. For our experiments, we train with a resolution of 512×288. Each training sample consists of three frames, separated by a randomly sampled time offset. There are no semantic annotations provided with the dataset. We evaluate the multi-view consistency of our model in this setting.

A.3. Computational complexity

SceneDINO requires only a *single* GPU for training and inference. In SSCBench (51.2 m range), SceneDINO requires 0.76 ± 0.1 s to infer a full scene on a V100 GPU. The peak VRAM usage during inference is 11 GB. For reference, S4C requires 0.32 ± 0.13 s. Considering our expressive and high-dimensional feature field and ViT encoder, this is a moderate runtime increase. SceneDINO has 100 M parameters and is trained for approximately 2 days on a *single* V100 32 GB GPU. All results are reported using automatic mixed precision.

B. Multi-View Feature Consistency Evaluation

We aim to assess the multi-view consistency of 2D and 3D features in Tab. 4. Note, we are not aware of any standardized approach for evaluating multi-view feature consistency. To this end, we employ a straightforward approach. Given two video frames with a temporal stride of 3, forward optical flow is computed using RAFT large [99]. We estimate occlusion by forward-backward consistency [125]; for this, we also compute backward optical flow. 2D feature maps obtained using the second frame are backward warped to the 2D features of the first frame. We compute different similarity metrics between the aligned features (L_1, L_2, and cos-sim), ignoring occlusions. While features from DINO, DINOv2, and FiT3D possess a lower resolution than our 2D rendered SceneDINO features, we upscale these features to the image resolution before warping. This evaluation approach utilizes optical flow correspondences and captures both ego motion as well as object motion, offering a simple way to evaluate multi-view feature consistency.

C. Additional Results

Here we provide additional qualitative and quantitative results, extending our results reported in the main paper.

Qualitative results. In Fig. 6, we present additional qualitative results of SceneDINO using our 3D feature distilla-

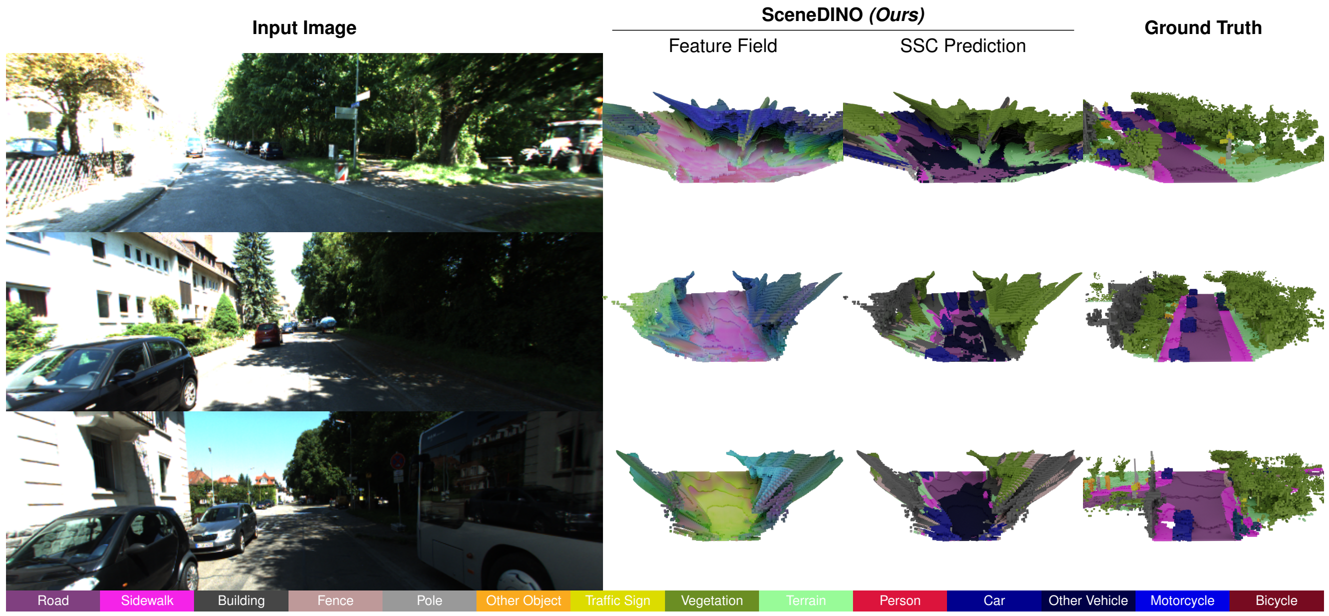


Figure 7. **Failure cases of SceneDINO on KITTI-360.** We provide failure cases of SceneDINO. We visualize the input image, the predicted feature field using the first three principal components, the SSC prediction, and the SSC ground truth. We observe that our semantic predictions struggle in shaded regions. We only visualize surface voxels within the field of view for the sake of clarity.

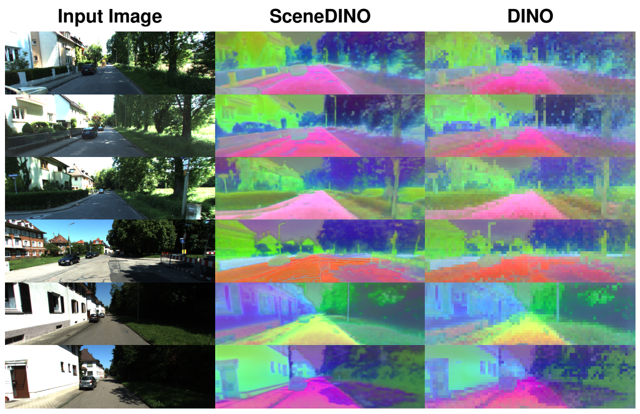


Figure 8. **2D SceneDINO features on KITTI-360.** We visualize our 2D rendered features and DINO features for a given input image *(left)*. We use the first three principal components for feature visualization. Notably, SceneDINO's features *(middle)* are smoother and capture finer structures than DINO *(right)*. Additionally, SceneDINO's features are high-resolution, while DINO generates features with a stride of 8.

tion approach on unsupervised semantic scene completion. We also provide visualizations of our unsupervised SSC baseline, S4C + STEGO. Qualitatively, our approach obtains more accurate SSC results and is able to segment far-away objects, such as cars, better than the S4C + STEGO baseline. This observation aligns with the quantitative results presented in Tab. 1 of the main paper.

Figure 8 qualitatively analyzes our 2D rendered features against DINO. Our features exhibit a smooth appearance for uniform regions, such as sidewalks. Additionally, SceneDINO's features better capture fine structures like poles than DINO features. 2D rendered SceneDINO features are also high resolution in contrast to DINO features that exhibit a lower resolution.

Failure cases. In Fig. 7, we provide failure cases of SceneDINO's SSC predictions. Our predictions exhibit two common failure cases. First, shadowed regions often lead to wrong semantic predictions. Regions affected by significant brightness changes are breaking the brightness consistency, subsequently offering a poor learning signal during training, thus impeding accurate predictions of shadowed regions. Second, objects such as cars can entail tail-like artifacts, not accurately capturing the geometry. As our multi-view image and feature reconstruction training cannot handle dynamic objects, tail-like artifacts could be caused by the poor learning signal for dynamic objects.

Quantitative results. In Tab. 8, we provide additional semantic scene completion results of 3D-supervised approaches as an additional point of comparison. In particular, we report official SSCBench [64] results of VoxFormer-S [63] and OccFormer [118]. Both utilize 3D supervision, including both semantic and geometric annotations. We also report the results of SSCNet [96]. This approach trains using 3D supervision but utilizes a depth image during inference. While SceneDINO achieves state-of-the-art segmentation accuracy in the unsupervised setting, supervised approaches are significantly more accurate.

Table 8. **SSCBench-KITTI-360 results.** Semantic results using mIoU and per class IoU, and geometric results using IoU, Precision, and Recall (all in %, ↑) on SSCBench-KITTI-360 test using three depth ranges. We extend Tab. 1 and compare SceneDINO against our baseline S4C [38] + STEGO [33], 2D-supervised S4C [38], and three 3D-supervised approaches (VoxFormer-S [63], OccFormer [118], and SSCNet [96]). Note that SSCNet uses depth as an additional input during inference, while all other approaches use a single input image.

Method	**S4C + STEGO**			**SceneDINO (Ours)**			**S4C**			**VoxFormer-S**			**OccFormer**			**SSCNet**		
Supervision	**Unsupervised**						**2D supervision**			**3D supervision**						**3D sup. + depth input**		
Range	12.8 m	25.6 m	51.2 m	12.8 m	25.6 m	51.2 m	12.8 m	25.6 m	51.2 m	12.8 m	25.6 m	51.2 m	12.8 m	25.6 m	51.2 m	12.8 m	25.6 m	51.2 m
Semantic validation																		
mIoU	10.53	9.26	6.60	**10.76**	**10.01**	**8.00**	16.94	13.94	10.19	18.17	15.40	11.91	23.04	18.38	13.81	26.64	24.33	19.23
car	18.57	14.09	9.22	21.24	15.94	11.21	22.58	18.64	11.49	29.41	25.08	17.84	40.87	33.10	22.58	52.72	45.93	31.89
bicycle	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	2.73	1.73	1.16	1.94	1.04	0.66	0.00	0.00	0.00
motorcycle	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.97	1.47	0.89	1.03	0.43	0.26	1.41	0.41	0.19
truck	0.11	0.04	0.02	0.00	0.00	0.00	7.51	4.37	2.12	6.08	6.63	4.56	22.40	15.21	9.89	16.91	14.91	10.78
other-v.	0.01	0.05	0.02	0.00	0.00	0.00	0.00	0.01	0.06	3.71	3.56	2.06	8.48	6.12	3.82	1.45	1.00	0.60
person	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	2.06	2.20	1.63	4.54	3.79	2.77	0.36	0.16	0.09
road	61.97	52.47	38.15	51.10	49.12	39.82	69.38	61.46	48.23	66.10	58.58	47.01	73.34	66.53	54.30	87.81	85.42	73.82
sidewalk	18.74	20.95	18.21	20.26	22.31	18.97	45.03	37.12	28.45	38.00	33.63	27.20	49.76	41.30	31.53	67.19	60.34	46.96
building	14.75	24.44	17.81	12.33	18.27	14.32	26.34	28.48	21.36	41.12	38.24	31.18	53.65	44.86	36.42	53.93	54.55	44.67
fence	1.41	0.20	0.11	1.91	0.90	0.58	9.70	6.37	3.64	8.99	7.43	4.97	10.64	7.85	4.80	14.39	10.73	6.42
vegetation	15.83	16.58	11.30	31.22	25.57	19.85	35.78	28.04	21.43	45.68	35.16	28.99	49.91	37.96	31.00	56.66	51.77	43.30
terrain	26.49	9.95	4.17	23.26	18.02	15.22	35.03	22.88	15.08	24.70	18.53	14.69	34.63	24.99	19.51	43.47	36.44	27.83
pole	0.08	0.04	0.04	0.05	0.05	0.05	1.23	0.94	0.65	8.84	8.16	6.51	12.93	10.25	7.77	1.03	1.05	0.62
traffic-sign	0.00	0.00	0.00	0.00	0.00	0.00	1.57	0.83	0.36	9.15	9.02	6.92	14.25	12.37	8.51	1.01	1.22	0.70
other-obj.	0.05	0.04	0.02	0.00	0.00	0.00	0.00	0.00	0.00	4.40	3.27	2.43	8.96	6.71	4.60	1.20	0.97	0.58
Geometric validation																		
IoU	49.32	41.08	36.39	**49.54**	**42.27**	**37.60**	54.64	45.57	39.35	55.45	46.36	38.76	58.71	47.96	40.27	74.93	66.36	55.81
Precision	54.04	46.23	41.91	53.27	46.10	41.59	59.75	50.34	43.59	66.10	61.34	58.52	69.47	62.68	59.70	83.65	77.85	75.41
Recall	84.95	78.69	73.43	87.61	83.59	79.67	86.47	82.79	80.16	77.48	65.48	53.44	79.13	67.12	55.31	87.79	81.80	68.22

Table 9. **SSCBench-KITTI-360 results (DINOv2).** Semantic results using mIoU and per class IoU, and geometric results using IoU, Precision, and Recall (all in %, ↑) on SSCBench-KITTI-360 test using three depth ranges. We compare our baseline S4C + STEGO to SceneDINO, both using DINOv2 features.

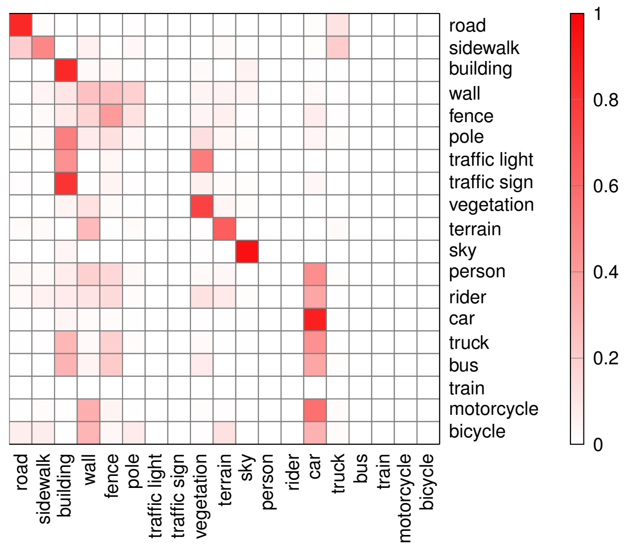
Method	**S4C + STEGO w/ DINOv2**			**SceneDINO w/ DINOv2 (Ours)**		
Supervision	**Unsupervised**					
Range	12.8 m	25.6 m	51.2 m	12.8 m	25.6 m	51.2 m
Semantic validation						
mIoU	11.70	9.27	6.25	**13.76**	**11.78**	**9.08**
car	15.66	10.31	5.84	18.27	13.83	9.51
bicycle	0.00	0.00	0.00	0.00	0.00	0.00
motorcycle	0.00	0.00	0.00	0.00	0.00	0.00
truck	0.00	0.00	0.00	0.00	0.00	0.00
other-v.	0.01	0.01	0.01	0.00	0.00	0.00
person	0.00	0.00	0.00	0.00	0.00	0.00
road	65.81	55.73	35.00	68.04	61.35	46.70
sidewalk	31.78	24.13	19.43	41.63	36.02	27.32
building	0.83	0.41	0.23	15.97	20.87	16.81
fence	0.89	0.57	0.41	0.00	0.00	0.00
vegetation	9.92	11.42	9.24	25.37	17.86	14.82
terrain	33.79	15.96	8.45	37.07	26.81	21.06
pole	16.84	20.43	15.14	0.00	0.00	0.00
traffic-sign	0.00	0.00	0.01	0.00	0.00	0.00
other-obj.	0.01	0.01	0.02	0.00	0.00	0.00
Geometric validation						
IoU	47.51	39.99	35.63	**48.12**	**40.35**	**36.21**
Precision	55.89	47.32	42.36	52.95	45.44	40.92
Recall	76.02	72.06	69.14	84.07	78.29	75.89

Tab. 8 provides additional SSC results of our S4C [38] + STEGO [33] baseline and SceneDINO using DINOv2 features [80]. In particular, we train STEGO with DINOv2 features and lift the resulting unsupervised semantic predictions using S4C. For SceneDINO, we use DINOv2 target features and perform distillation and clustering. Training S4C + STEGO using DINOv2 features leads to improvements for close range (12.8 m) over using DINO features (*cf*. Tab. 8). For larger ranges (*e.g.*, 51.2 m), S4C + STEGO with DINOv2 features drops in accuracy compared to S4C + STEGO with DINO features. We attribute this drop in accuracy to the coarser feature resolution of DINOv2 (larger ViT patch size). This behavior has also been observed for the task of 2D unsupervised semantic segmentation [31]. Note that SceneDINO overcomes the coarse features using a learnable downsampler and multi-view training, learning high-resolution 3D features.

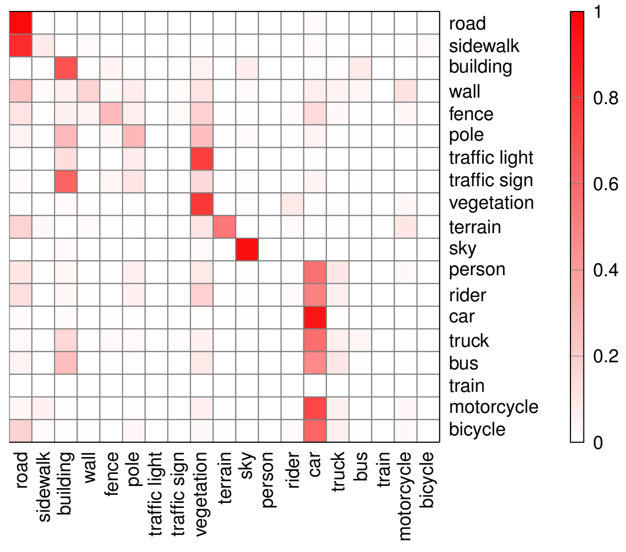
Table 10. **Class-wise 2D unsupervised semantic segmentation results on KITTI-360**. We compare the class-wise IoU scores (all in %, ↑) of SceneDINO against STEGO in 2D on the SSCBench-KITTI-360 test split.

Method	**STEGO**	**SceneDINO**
mIoU	**23.57**	**25.81**
road	63.81	77.73
sidewalk	7.70	44.48
building	65.24	77.67
wall	11.94	3.68
fence	15.36	18.13
pole	11.43	0.93
traffic light	0.00	0.00
traffic sign	0.11	0.00
vegetation	73.35	73.38
terrain	49.31	41.29
sky	69.18	71.72
person	0.00	0.00
rider	0.05	0.00
car	77.72	81.31
truck	2.09	0.04
bus	0.02	0.00
train	0.00	0.00
motorcycle	0.08	0.00
bicycle	0.00	0.00

Class-wise semantic results. To further assess the seg-



(a) SceneDINO



(b) STEGO

Figure 9. **Confusion matrices for 2D unsupervised semantic segmentation on KITTI-360**. Rows represent ground-truth class labels (normalized to 1), while columns correspond to predicted class labels. We report results for *(a)* SceneDINO and *(b)* STEGO on the SSCBench-KITTI-360 test split.

mentation accuracy of SceneDINO, we report the class-wise IoU metric in 3D (*cf*. Tab. 1, 8, and 9) and 2D (*cf*. Tab. 10). We generally observe that SceneDINO performs well in segmenting frequent classes, such as "road", "building", and "sky". Less frequent classes, such as "fence" and "pole", are less well segmented. Classes including very small and fine structures (*e.g*., "pole") are completely missed by SceneDINO. This trend can also be observed for our 3D unsupervised baseline S4C + STEGO and 2D STEGO. We also observe that class-wise metrics strongly correlate between 2D and 3D.

Table 11. **Linear probing results on SSCBench-KITTI-360.** We extend Tab. 7 and report detailed results of SceneDINO using 2D-supervised linear probing. Semantic results using mIoU and class IoU, and geometric results using IoU, Precision, and Recall, and (all in %, ↑) on SSCBench-KITTI-360 test using three depth ranges.

Method	**SceneDINO w/ DINO (Ours)**			**SceneDINO w/ DINOv2 (Ours)**		
Supervision	**Unsupervised**					
Range	12.8 m	25.6 m	51.2 m	12.8 m	25.6 m	51.2 m
Semantic validation						
mIoU	13.63	12.07	9.34	**15.85**	**13.70**	**10.57**
car	16.77	12.37	8.42	20.35	15.04	10.16
bicycle	1.10	0.70	0.47	0.00	0.00	0.00
motorcycle	0.00	0.00	0.00	0.00	0.00	0.00
truck	3.80	2.21	1.52	11.48	7.46	4.63
other-v.	0.13	0.08	0.06	0.00	0.00	0.00
person	0.01	0.00	0.00	0.00	0.00	0.00
road	66.63	62.21	49.99	69.92	63.06	50.49
sidewalk	29.46	25.17	18.85	42.35	37.13	29.13
building	18.64	22.82	17.66	23.03	27.05	21.40
fence	9.29	6.03	3.96	8.82	6.40	4.61
vegetation	32.76	26.49	20.89	30.42	24.96	19.75
terrain	24.80	22.43	18.00	30.73	23.85	17.93
pole	0.25	0.24	0.14	0.46	0.40	0.28
traffic-sign	0.50	0.17	0.09	0.00	0.00	0.00
other-obj.	0.26	0.07	0.04	0.00	0.00	0.00
Geometric validation						
IoU	49.34	42.26	37.61	**49.77**	**43.19**	**38.55**
Precision	52.83	45.95	41.55	52.76	46.46	42.11
Recall	88.21	84.05	79.88	89.76	85.99	82.02

Figure 9 reports confusion matrices of SceneDINO and STEGO for 2D semantic segmentation on KITTI-360. Both approaches share a similar confusion pattern. We attribute this to the fact that both approaches rely on the feature representation of DINO. In particular, we observe confusion between semantically close classes, such as "pole", "traffic light", and "traffic sign". Interestingly, for the semantic classes "person", "rider", "car", "truck", "bus", "motorcycle", and "bicycle", we see a strong confusion. We suspect this correlation is potentially caused by the fact that these classes often appear on the "road" and "sidewalk" and are rare in KITTI-360.

We also provide class-wise SSC results of SceneDINO using 2D-supervised linear probing in Tab. 11. Linear probing provides an upper bound for clustering our features, improving the segmentation accuracy for almost all classes. However, rare classes like "motorcycle" are still not captured using linear probing. This suggests that the DINO feature space fails to express these classes accurately, limiting the segmentation accuracy of SceneDINO. Still, our approach is agnostic to the utilized target features and can potentially profit from better 2D features.

Camera pose analysis. Training SceneDINO requires accurate camera poses. While KITTI-360 offers ground-truth camera poses, these poses are obtained using additional cues, including LiDAR data [66]. To adhere to our fully unsupervised setting, we provide results train-

Table 12. **Camera pose analysis on SSCBench-KITTI-360.** We extend the camera pose analysis in Tab. 5 and report detailed results of SceneDINO with unsupervised camera poses estimated by SOFT2 [122] and ORB-SLAM3 [7]. For reference, we also provide results obtained using the KITTI-360 ground-truth poses. Semantic results using mIoU and class IoU, and geometric results using IoU, Precision, and Recall, and (all in %, ↑) on SSCBench-KITTI-360 test using three depth ranges.

Method	**SceneDINO (Ours)**								
Poses	**SOFT2**			**ORB-SLAM3**			**KITTI-360 (GT)**		
Range	12.8 m	25.6 m	51.2 m	12.8 m	25.6 m	51.2 m	12.8 m	25.6 m	51.2 m
Semantic validation									
mIoU	10.58	9.58	7.72	**10.88**	9.86	7.88	10.76	**10.01**	**8.00**
car	18.47	13.98	10.44	19.37	14.09	9.72	21.24	15.94	11.21
bicycle	0.04	0.03	0.03	0.06	0.03	0.02	0.00	0.00	0.00
motorcycle	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00	0.00
truck	0.00	0.00	0.00	0.05	0.02	0.01	0.00	0.00	0.00
other-v.	0.01	0.02	0.04	0.08	0.06	0.05	0.00	0.00	0.00
person	0.02	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00
road	44.48	44.50	36.06	44.74	40.58	31.86	51.10	49.12	39.82
sidewalk	16.55	16.79	14.38	21.45	23.56	19.88	20.26	22.31	18.97
building	19.40	23.40	18.56	19.19	24.87	20.02	12.33	18.27	14.32
fence	1.79	1.00	0.68	1.62	1.21	0.91	1.91	0.90	0.58
vegetation	32.10	25.65	20.67	32.60	24.91	19.49	31.22	25.57	19.85
terrain	25.59	18.11	14.79	23.98	18.41	16.16	23.26	18.02	15.22
pole	0.18	0.11	0.09	0.00	0.00	0.00	0.05	0.05	0.05
traffic-sign	0.00	0.01	0.00	0.03	0.03	0.02	0.00	0.00	0.00
other-obj.	0.08	0.05	0.03	0.08	0.05	0.03	0.00	0.00	0.00
Geometric validation									
IoU	**49.91**	41.85	37.25	45.42	40.21	36.65	49.54	**42.27**	**37.60**
Precision	54.74	45.66	40.79	54.42	45.54	40.98	53.27	46.10	41.59
Recall	84.98	83.40	81.12	73.33	77.46	77.62	87.61	83.59	79.67

ing with unsupervised camera poses, estimated using stereo visual SLAM. In particular, Tab. 5 reports results of SceneDINO trained using unsupervised camera poses estimated by ORB-SLAM3 [7]. Table 12 extends this and reports detailed SSC results using two different unsupervised stereo visual SLAM approaches—SOFT2 [122] and ORB-SLAM3 [7]. Using unsupervised and visually estimated poses leads to a minor drop in both semantic and geometric SSC validation. While ORB-SLAM3 poses lead to slightly better semantic accuracy than SOFT2 poses, SOFT2 estimated poses result in higher geometric accuracy. Still, both SOFT2 and ORB-SLAM3 provide poses accurate enough for training SceneDINO, reaching a similar accuracy to employing KITTI-360 ground-truth poses.

Out-of-domain results. We illustrate results for out-of-domain prediction in Fig. 10. While our SceneDINO model is trained on the KITTI-360 dataset, we still obtain plausible features when inferring 2D features for vastly different scenes. The 2D rendered features still show a strong correlation with semantically uniform regions, showcasing the generalization of our feature field.

D. Limitations and Future Work

Target features. Our method builds on DINO [11] to obtain target features. While we learn to lift these features into 3D and improve multi-view feature consistency, we cannot improve the discriminative power of the target features

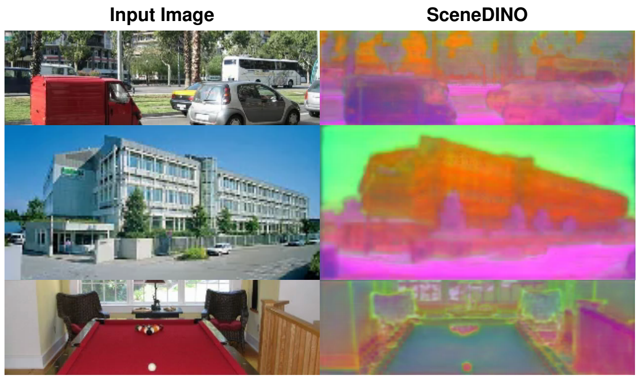


Figure 10. **2D SceneDINO features on out-of-domain images.** We visualize our 2D rendered features *(right)* given an out-of-domain image *(left)* from ADE20K [127]. We use the first three principal components for feature visualization. While not trained on such scenes, SceneDINO still produces plausible feature maps.

per se. However, SceneDINO can be trained using arbitrary 2D target features and can profit from future advances in SSL representations. Note that training SceneDINO requires only 2 days on a single GPU and our training transfers seamlessly to different target features (*e.g.*, DINOv2), thus, utilizing SceneDINO differently is straightforward.

Dynamic objects. Our loss does not model dynamic objects and relies on a static scene assumption. This can potentially cause inaccurate predictions for dynamic classes such as *person* in our experiments. Recent works in depth estimation have explicitly modeled the probability of areas being dynamic [126] and even their motion within the scene [124], which might be extended to SceneDINO.

View sampling and camera poses. For sampling views during training, we rely on the sampling scheme of S4C [38]. This is not directly applicable to other non-driving datasets, where the sampling needs to be tuned. In addition, our approach requires accurate camera poses for each view. We demonstrated that these can be obtained in an unsupervised way for KITTI-360 (*cf*. Tab. 5 & Tab. 12). However, obtaining unsupervised camera poses in more complex scenarios and conditions is still a challenge [121].

Future work. SceneDINO is only trained using a single dataset to be comparable to existing SSC approaches. However, scaling our approach to multiple datasets of more variable scenes could lead to more general feature representations. Ultimately, scaling SceneDINO to internet-scale videos might enable strong zero-shot and cross-domain 3D scene understanding.

References

[121] Lucas R. Agostinho, Nuno M. Ricardo, Maria I. Pereira, Pinto Antoine, and Andry M. Pinto. A practical survey on visual odometry for autonomous driving in challenging

scenarios and conditions. *IEEE Access*, 10:72182-72205, 2022. vi

[122] Igor Cvišić, Ivan Marković, and Ivan Petrović. SOFT2: Stereo visual odometry for road vehicles based on a point-to-epipolar-line metric. *IEEE Trans. Robot.*, 39(1):273-288, 2023. vi

[123] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*2019*, pages 8024–8035. i

[124] Yihong Sun and Bharath Hariharan. Dynamo-Depth: Fixing unsupervised depth estimation for dynamical scenes. In *NeurIPS*2023*, pages 54987–55005. vi

[125] Narayanan Sundaram, Thomas Brox, and Kurt Keutzer. Dense point trajectories by GPU-accelerated large displacement optical flow. In *ECCV*, pages 438—451, 2010. ii

[126] Sungmin Woo, Wonjoon Lee, Woo Woo Jin, Dogyoon Lee, and Sangyoun Lee. ProDepth: Boosting self-supervised multi-frame monocular depth with probabilistic fusion. In *ECCV*, pages 201–217, 2024. vi

[127] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *CVPR*, pages 5122–5130, 2017. vi