

SAM4D: Segment Anything in Camera and LiDAR Streams

Supplementary Material

In this document, we further provide the following materials to support the findings and conclusions drawn in the main body of this paper.

- Section A: Model and training details;
- Section B: Details on data engine and dataset;
- Section C: Details on the experimental settings;
- Section D: Discussions on limitation and future work;
- Section E: Acknowledgements to public resources.

A. Model and Training Details

A.1. Model Architecture

Image Encoder. The image encoder in SAM4D follows the same architecture as SAM2 [7], but given the smaller scale of our dataset compared to SAM2, we adopt the Hiera-S variant [8] as the default configuration to balance performance and efficiency.

LiDAR Encoder. We adopt MinkUNet [3], implemented with TorchSparse [10, 11], as the LiDAR encoder. Inspired by ResNet [4], we define Mink34 and Mink50 as backbone structures, with Mink34 as the default choice. The encoder downsamples the input to stride 32, with feature dimensions [32, 32, 64, 128, 256], and then upsamples to stride 4, extracting voxel features at strides 16, 8, and 4. The stride 16 features are primarily used by the memory module and mask decoder, while the stride 8 and 4 features assist the mask decoder in recovering high-resolution segmentation details, similar to SAM2. To improve the generalization across various datasets, we exclude the original xyz coordinates and do not use intensity and elongation features provided by Waymo Open Dataset [9]. Instead, we assign a binary occupancy value to occupied voxels, ensuring the LiDAR encoder remains dataset-agnostic.

Mask Decoder. Sparse prompt tokens from image and LiDAR are concatenated and used as queries for mask prediction, with a shared Transformer module across both modalities. The query token configuration follows SAM2, consisting of mask queries, sparse prompt tokens, IoU tokens, and object pointers stored in memory. This design enables efficient cross-modal segmentation while ensuring consistency between image and LiDAR-based queries.

Model Parameters. SAM4D comprises 119.88M parameters, distributed across its core components. The image encoder has 34.32M parameters, while the LiDAR encoder contains 26.94M parameters. The memory module, responsible for temporal feature aggregation and cross-modal attention, is the largest component with 53.96M parameters. The mask decoder, which processes prompt-based queries for segmentation, accounts for 4.66M parameters. This de-

Config	Value
data	Waymo-4DSEg
steps	~44k
resolution	Camera: 768×768 , LiDAR: 0.15 m
precision	bfloat16
optimizer	AdamW
optimizer momentum	$\beta_1 = 0.9, \beta_2 = 0.999$
gradient clipping	type: ℓ_2 , max: 0.1
weight decay	0.1
learning rate (lr)	OneCycleLR, init: $5e^{-6}$, max: $5e^{-5}$, anneal strategy: cos, pct_start: 0.4
warmup	linear, 7.5k iters
layer-wise decay	0.9
image augmentation	hflip, resize to 768×768 (square)
video augmentation	hflip, affine (deg: 25), colorjitter, grayscale, per-frame colorjitter, mosaic- 2×2
LiDAR augmentation	rotation-Z (deg: 45), hflip, vflip
drop path	0.2
mask losses (weight)	Focal (20), Dice (1)
IoU loss (weight)	ℓ_1 (1)
occlusion loss (weight)	Cross-entropy (1)
global attn. blocks	12 - 16 - 20

Table A1. Hyperparameters for SAM4D full training.

sign balances multimodal fusion, temporal reasoning, and segmentation efficiency.

A.2. Training Details

Without loss of generality, we use the front-view camera and LiDAR from the Waymo dataset to validate the feasibility of the proposed solution, which can later be extended to multi-camera and LiDAR setups. Since our data is constructed through 4D reconstruction using a 5-camera and LiDAR system, we cannot guarantee that objects appear in both modalities in every frame. Therefore, during the training process, to enable parallel imitation of interaction logic for multiple targets, the following rules are applied when selecting targets for each step in the data pipeline: there is a 0.5 probability that the target exists in both modalities, a 0.25 probability that it appears only in the camera, and a 0.25 probability that it appears only in the LiDAR. During training, the iterative modification logic for mimicking targets is as follows: if the target belongs to both modalities, each prompt randomly selects one modality; if the target belongs to only one modality, the modality in which the target appears is chosen for the prompt.

SAM4D is trained on Waymo-4DSEg for 44k steps with a 768×768 image resolution and a LiDAR voxel size of 0.15. Specifically, we sample 8-frame sequences and randomly select up to 2 frames to receive prompts. During training, corrective clicks are probabilistically sampled

based on both ground-truth masks and model predictions. The initial prompts are assigned with probabilities of 0.5 for ground-truth masks, 0.25 for points, and 0.25 for bounding boxes, respectively. The loss consists of a combination of focal loss and dice loss for mask prediction and mean absolute error (MAE) loss for IoU prediction. If an object is missing in a given modality, we do not apply supervision to the prediction of that modality. AdamW with OneCycleLR are utilized to optimize the network. Image and video augmentations follow SAM2 (excluding shear), while LiDAR-specific augmentations include Z-axis rotation, hflip, and vflip. Other hyperparameters align with SAM2, which are provided in Tab. A1.

B. Details on Data Engine and Dataset

B.1. Data Engine Details

In this section, we provide a more detailed description of the implementation details of the three steps in our data engine to supplement Sec. 5 in our main paper.

Step 1: Generation of VFM-based image masklets

In this step, we utilize vision foundation models (VFM) [2, 6] to generate initial annotations including boxes and masks in keyframes firstly. In each new keyframe, we redetect scene objects and match them with the propagated masks from the previous keyframe, merging them as the segmentation result of the current keyframe. Newly detected objects, which were not present in the previous masklets, are first propagated backward to the start of the sequence, and then the merged masks (both new and existing objects) are propagated forward to the next keyframe, continuing this process until the end of the sequence.

This iterative approach produces masklets for the entire image sequence, ensuring consistent object categories and instance IDs across time, laying the foundation for LiDAR ground truth generation in subsequent stages.

Step 2: 4D Reconstruction and Ray Casting

Transferring masklets from video to LiDAR frames requires establishing correspondences between pixels and LiDAR points, which is challenging due to the large number of pixels and 3D points in the sequence. To address this challenge, we preemptively perform 4D LiDAR reconstruction using VDBFusion [12], which generates a more efficient representation of spatial occupancy, since the voxel count depends only on scene size and is independent of the number of frames.

The 4D reconstruction comprises multiple foreground components and a single background component. The background consists of static objects and is maintained as a single instance in the world coordinate system. The foreground includes potentially moving entities such as vehicles, cyclists, and pedestrians, each with its own motion trajectory. We leverage the pre-annotated 3D bounding boxes

of these foreground objects to obtain their relative poses in each frame with respect to the world, and subsequently perform individual 4D reconstruction within each object’s body coordinate system. This allows the voxels occupied by the foreground objects to remain unchanged even as they move, with only the overall position shifting.

Following this, we generate a dense pixel-voxel mapping table via ray casting. We first compute image poses in the world coordinate system by solving the PnP(Perspective-n-Point) problems, based on the given single-frame point cloud and pixel correspondences. Then, for each image, we construct multiple rays starting from the camera position and ending at the center points of the voxels within the viewing frustum. Each ray falls into an image pixel, and we match the pixel with the voxel that the ray intersects together to build pixel-voxel the mapping table.

Step 3: Cross-Modal Masklet Fusion

By querying the pixel-voxel mapping table established in Step 2, we can identify the voxels corresponding to the pixels masked in Step 1, thereby transferring the mask to the voxels. In an ideal scenario, SAM2 ensures consistency of masklets between frames in the video, and we can directly merge the voxel masks from the video stream to obtain an accumulated voxel masklet.

However, we found that both the video masklets and the mapping table are often noisy. A common issue is that SAM2 mis-matches objects, confusing two similar objects appearing in close positions across different frames as the same object. This issue occurs for both background and foreground objects. Additionally, because the pixel-to-voxel correspondence is not always accurate and the edges of 2D masks on images are not perfectly precise, the resulting masks projected onto the voxels are prone to contain noise. Finally, the image segmentation model occasionally misclassifies artifacts such as light spots or other visual anomalies in images as actual objects.

To mitigate these problems, we implemented a clustering approach for noise filtering in the voxel masklets. We employed the DBSCAN algorithm to cluster voxels based on their BEV positions and selected the cluster with the highest average quality as the main cluster, filtering out the rest as noise. Assuming that voxels associated with a single object are adjacent in BEV space, we utilized the DBSCAN algorithm to cluster the voxels based on their BEV positions. We also counted the frequency with which each voxel was mapped to the current object and computed the vote rate as the ratio of this frequency to the total observations in the current image sequence. Ultimately, we selected the cluster with the highest average vote rate as the main cluster, and leave out rest as noise. Fig A2 provides examples of the issues mentioned above and demonstrates the effectiveness of our filter in addressing these problems.

After filtering, we expect significant overlap between

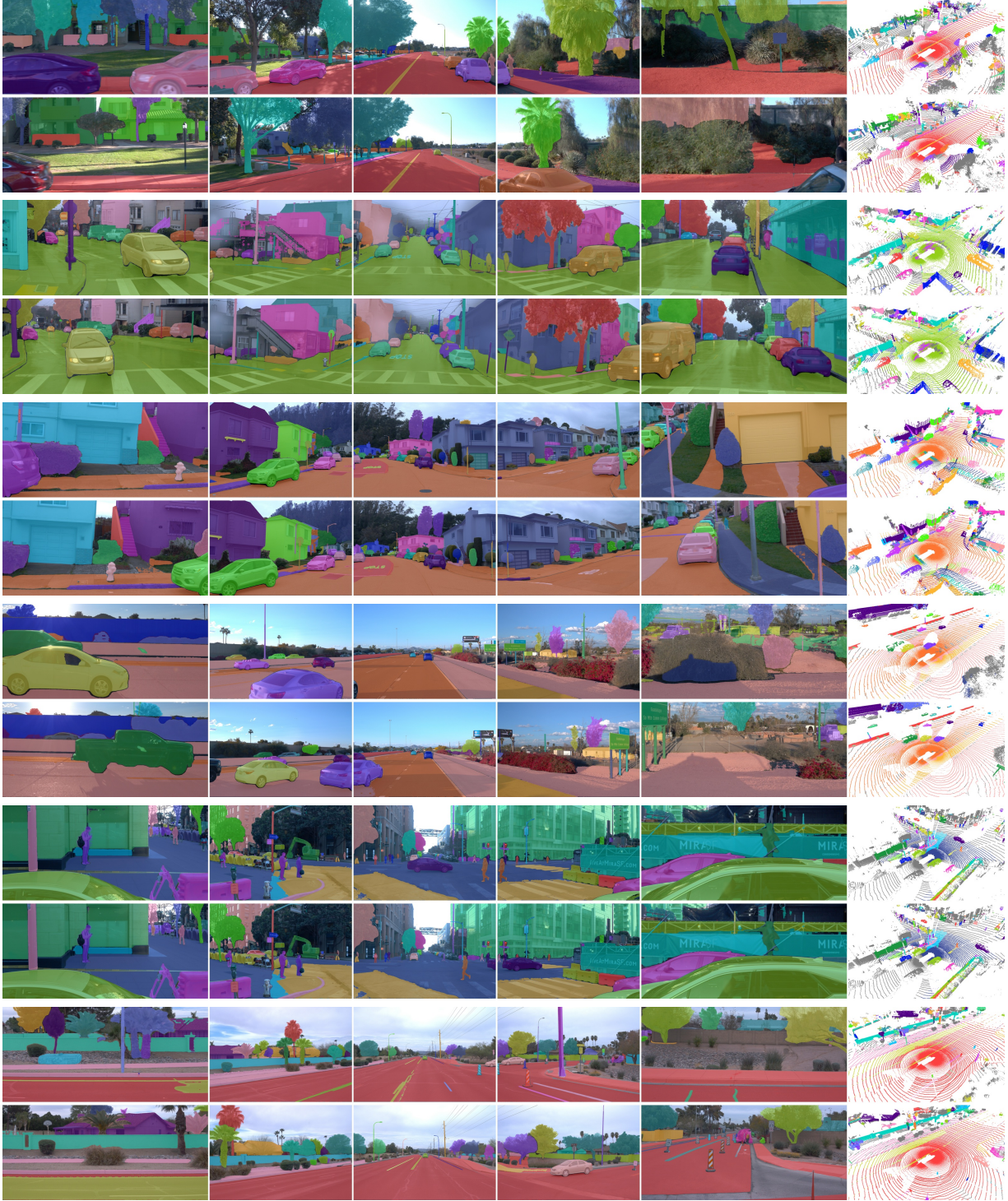


Figure A1. Examples of the Waymo-4Dseg Dataset: In this figure, we visualize the masklets from selected frames of 6 clips. For each clip, we present two frames in two rows, where each row displays the masklets from the side-left, front-left, front, front-right, and side-right images, from left to right, along with the masklets from the LiDAR frames. Through both horizontal and vertical comparisons, the consistency of the masklets across different video streams, modalities, and frames can be observed.

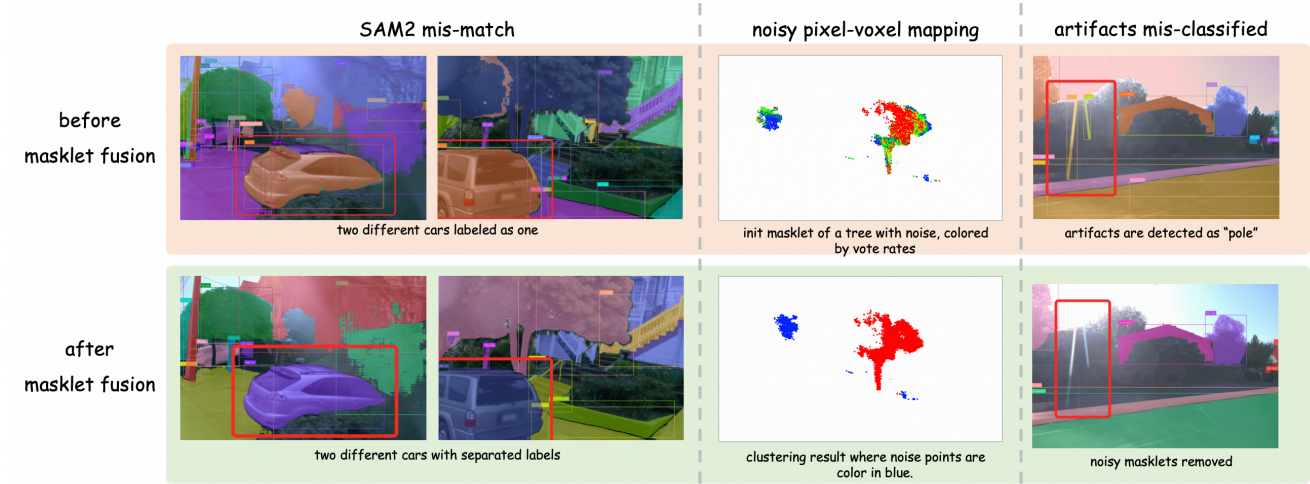


Figure A2. This figure illustrates three representative problems that may arise before masklet fusion, as well as how our fusion process addresses these issues.

voxel masklets from different videos corresponding to the same object. Overlaps between masklets from two videos were assessed, leading to the merging of those with substantial overlap into a single unified voxel masklet. Finally, we created a mapping table between points from LiDAR frames and voxels based on their 3D spatial distances, facilitating the transfer of the final voxel masklet to the LiDAR frames.

We evaluated the quality of the unified voxel masklets using cross-modal IoU. Assuming that a masklet is visible for image i , we calculated the IoU between the voxels mapped by masklet in image i and the visible part of the unified voxel masklet. The average IoU across all images represents one masklet’s overall score. The mean score of the masklets in our dataset is 0.56, with a 10th percentile of 0.24. Throughout this process, human annotators play a crucial role in adjusting the parameters based on mask quality and conducting frame-by-frame verification of the final labels in both the image and LiDAR.

B.2. Dataset Statistical Information

Here, we provide additional information about the dataset and details about our data engine. In Figure A3, we provide detailed statistical information about the masklets in our dataset, including volume distribution, area distribution, the proportion of frames in which cross-modal masklets co-occur, and score distribution. The distributions of volume and area reflect the diversity and richness of the annotated objects in our dataset. Additionally, we calculate the proportion of frames in which cross-modal masklets are present in both modalities, a metric of interest to users. Common scenarios include objects exiting the video frame as the vehicle moves forward while still being detectable by LiDAR, or objects being beyond the scanning range of LiDAR but still visible in the video. Such cases require special han-

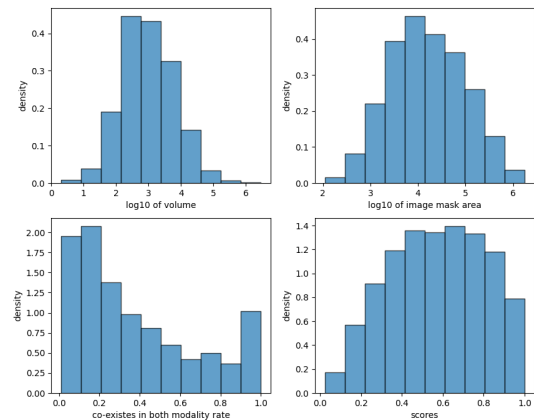


Figure A3. Statistical Information of Masklets in the Dataset

dling during model prompting and training, which is why we provide the proportion of frames where cross-modal masklets coexist. We also present the score distribution of the masklets. Although denoising has been applied during processing, a small number of low-quality masklets remain. Fortunately, we can quantify their quality through scores and filter them out during experimentation. In Figure A1, we present the visualized cross-modal masklets from several clips.

C. Details on the Experimental Settings

C.1. About the Training and Validation Data.

As mentioned above, when the data engine generates pseudo labels, it creates voxels in 4D for each object, allowing us to obtain the volume of the object in space. Additionally, multi-modal consistency checks are performed

to calculate the IoU between each frame’s image and point cloud. The average IoU over the sequence serves as a reference for the quality of the pseudo-labels, which we refer to as the “score.” During SAM4D training, targets with a volume greater than 10 and a score greater than 0.3 are used. For testing, to further ensure the reliability of the ground truth, the volume filtering threshold is increased to 50, and the score threshold is raised to 0.5. Furthermore, there is currently significant ambiguity in the pseudo-labels for ground regions. To ensure better convergence of the LiDAR branch, we temporarily exclude instances near the ground during both training and evaluation. Despite these settings, the number of targets evaluated in each sequence still exceeds 100, resulting in slow evaluation speeds. To accelerate the evaluation, we filter and evaluate only those objects that appear in at least one frame in both the front-view camera and LiDAR.

C.2. About the Generalization Experiments.

The generalization experiments are conducted on nuScenes dataset [1] with nuInsSeg [5]. The nuInsSeg dataset [5], built on nuScenes [1], provides 2D instance segmentation annotations for foreground objects, with instance IDs corresponding to 3D point cloud segmentation labels.

D. Limitations and Further Discussions

D.1. Limitations

While SAM4D effectively integrates multimodal and temporal segmentation, the domain gap across LiDAR sensors remains a challenge, as variations in sensor configurations and point cloud density limit generalization compared to images. Moreover, the spatial representation on point clouds is inherently constrained by single-frame sparsity, occlusions, and blind spots, which may hinder object completeness in certain scenarios. Additionally, while Waymo-4Dseg provides high-quality multimodal labels, the size of the data set can be expanded to cover a broader range of driving conditions, weather variations, and rare long-tail scenarios. Increasing data set diversity would improve the generalizability of the model, particularly in corner cases where data sparsity remains a challenge.

D.2. Future Work

Currently, SAM4D is trained on pseudo-labels generated by an automated data engine. Although the data labels have undergone multi-modal consistency verification, ambiguities and inaccuracies still persist. Future work will focus on improving SAM4D’s data strategy, model adaptability, and scalability. To enhance label quality, we plan to expand dataset scale using our automated data engine and integrate human-annotated subsets for fine-tuning. A confidence-based filtering mechanism will further refine pseudo-labels

iteratively. Additionally, extending SAM4D to incorporate natural language descriptions will enable multimodal segmentation conditioned on text, leveraging LLMs for semantic guidance to reduce reliance on human annotations. Exploring weakly supervised and self-supervised learning will further enhance adaptability while minimizing manual labeling. Beyond data efficiency, improving memory attention and computational efficiency will enable scaling to multi-camera and multi-sensor systems, enhancing 4D spatiotemporal perception in complex environments.

E. License and Consent with Public Resources

E.1. Public Datasets

We utilize the Waymo Open Dataset [1] to construct our Waymo-4Dseg dataset. nuScenes [1] and the corresponding nuInsSeg [5] are adopted to further evaluate our model:

- Waymo Open Dataset¹ Waymo Dataset License
- nuScenes² CC BY-NC-SA 4.0
- nuScenes-devkit³ Apache License 2.0
- nuInsSeg⁴ MIT License

E.2. Public Implementation

We leverage publicly available pre-trained models and source codes to investigate the promptable segmentation in the multimodal domain:

- SAM2⁵ Apache License 2.0
- TorchSparse⁶ MIT License
- GroundingDINO⁷ Apache License 2.0
- Grounded-SAM-2⁸ Apache License 2.0
- VDBFusion⁹ MIT License
- Mask-Propagation¹⁰ MIT License

References

- [1] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nusenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 5
- [2] Tianrun Chen, Ankang Lu, Lanyun Zhu, Chaotao Ding, Chunyan Yu, Deyi Ji, Zejian Li, Lingyun Sun, Papa Mao, and Ying Zang. Sam2-adaptor: Evaluating & adapting segment anything 2 in downstream tasks: Camouflage, shadow,

¹<https://waymo.com/open>.

²<https://www.nuscenes.org/nuscenes>.

³<https://github.com/nutonomy/nuscenes-devkit>.

⁴<https://github.com/Serenos/nuInsSeg>.

⁵<https://github.com/facebookresearch/sam2>.

⁶<https://github.com/mit-han-lab/torchsparse>.

⁷<https://github.com/IDEA-Research/GroundingDINO>.

⁸<https://github.com/IDEA-Research/Grounded-SAM-2>.

⁹<https://github.com/PRBonn/vdbfusion>.

¹⁰<https://github.com/hkchengrex/Mask-Propagation>.

- medical image segmentation, and more. *arXiv preprint arXiv:2408.04579*, 2024. 2
- [3] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3075–3084, 2019. 1
- [4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1
- [5] Xiang Li, Junbo Yin, Botian Shi, Yikang Li, Ruigang Yang, and Jianbing Shen. Lwsis: Lidar-guided weakly supervised instance segmentation for autonomous driving. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1433–1441, 2023. 5
- [6] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European Conference on Computer Vision*, pages 38–55. Springer, 2024. 2
- [7] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. 1
- [8] Chaitanya Ryali, Yuan-Ting Hu, Daniel Bolya, Chen Wei, Haoqi Fan, Po-Yao Huang, Vaibhav Aggarwal, Arkabandhu Chowdhury, Omid Poursaeed, Judy Hoffman, et al. Hiera: A hierarchical vision transformer without the bells-and-whistles. In *International conference on machine learning*, pages 29441–29454. PMLR, 2023. 1
- [9] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020. 1
- [10] Haotian Tang, Zhijian Liu, Xiuyu Li, Yujun Lin, and Song Han. TorchSparse: Efficient Point Cloud Inference Engine. In *Conference on Machine Learning and Systems (MLSys)*, 2022. 1
- [11] Haotian Tang, Shang Yang, Zhijian Liu, Ke Hong, Zhongming Yu, Xiuyu Li, Guohao Dai, Yu Wang, and Song Han. TorchSparse++: Efficient Point Cloud Engine. In *Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2023. 1
- [12] Ignacio Vizzo, Tiziano Guadagnino, Jens Behley, and Cyrill Stachniss. Vdbfusion: Flexible and efficient tsdf integration of range sensor data. *Sensors*, 22(3):1296, 2022. 2