OV-SCAN: Semantically Consistent Alignment for Novel Object Discovery in Open-Vocabulary 3D Object Detection

Supplementary Material

6. Additional Implementation Details

6.1. Novel Object Proposals

OV-SCAN relies on Grounding DINO [21] and SAM [17] to detect 2D proposals from a given list of (user-defined) novel classes. Tab. 5 outlines the novel classes we used for our experiments for both the nuScenes [2] and KITTI [12] datasets. We adhere to the original taxonomy of each dataset, except that "cyclist" is split into "bicycle" and "motorcycle" for KITTI.

Table 5. **Novel classes used to identify novel object proposals for each dataset.** These classes are referred to as "novel" since their ground truth labels, although available in the respective datasets, are not used in training OV-SCAN.

Dataset	Novel Classes
nuScenes	car, truck, pedestrian, bicycle, motorcycle, bus, traffic cone, barrier block, construction vehicle
KITTI	car, van, truck, tram, bicycle, motorcycle, pedestrian, person sitting

In post-processing, our method adopts a similar approach to UP-VL [30] for removing false positives. We treat each novel class as a positive class while including a set of background classes. To refine the results, we cross-reference each image crop from Grounding DINO with CLIP [33], filtering out background classes such as "vegetation", "fence", "gate", "curb", "sidewalk", "wall", "building", "railing", and "rail guard."

6.2. Cross-Modal Association

As discussed in Sec. 3.2, the primary objective of cross-modal association is to accurately pair each 2D proposal with its corresponding 3D object cluster. Our detailed implementation is outlined in Algorithm 1, where each 2D proposal is first projected into a 3D frustum defined by (d_{min}, d_{max}). Ideally, each proposal can then be matched to the cluster containing the point closest to the center frustum ray, provided this distance is within a matching threshold τ_{match}. However, challenges arise when a single 2D proposal corresponds to multiple object clusters due to fragmentation from misclustering, partial occlusion, or sparsity. To address this, we allow *one-to-many* matching, enabling a single 2D proposal to generate multiple competing cross-modal proposals. We resolve such conflicts during cross-modal target preparation by optimizing 3D box parameters for each candidate proposal, ultimately retaining the optimal proposal based on the objective defined in Eq. (1). Additionally, we address scenarios involving *many-to-one* matching, where larger objects (e.g., transit buses) span multiple camera views. In these situations, applying 3D NMS at the conclusion of cross-modal target preparation effectively resolves potential redundancies. Our conflict resolution strategy is detailed in Algorithm 2.

Algorithm 1 Cross-Modal Association

```
def cross_modal_association(3D_object_clusters,
                            2D_novel_object_proposals,
                            calibrations):
    """
    Input:
    - 3D_object_clusters: Set of 3D object clusters
    - 2D_novel_object_proposals: Set of 2D novel
                                 object proposals
    - calibrations: Intrinsic and extrinsic
                    calibration parameters

    Output:
    - cross_modal_proposals:
            List of (2D proposal, 3D cluster) pairs
    """

    # Initialize proposal list
    cross_modal_proposals = []

    for proposal in 2D_novel_object_proposals:
        # Compute frustum from 2D proposal
        frustum = get_frustum(proposal, calibration,
                              d_min, d_max)

        # Compute frustum center ray
        frustum_center_ray
            = get_frustum_center_ray(frustum)

        # Find 3D clusters with tau distance
        # from center frustum ray
        matched_clusters
            = get_clusters_near_ray(3D_object_clusters,
                                    frustum_center_ray,
                                    tau_match)

        # Store matching pairs
        for cluster in matched_clusters:
            pair = (proposal, cluster)
            cross_modal_proposals.append(pair)

    return cross_modal_proposals
```

6.3. Adaptive 3D Box Search

To solve the continuous nonlinear optimization problem from Eq. (1), OV-SCAN employs particle swarm optimization (PSO) [16] to search for the 3D annotation parameterized by $\theta = (x, y, z, l, w, h, r_y)$. [10], for which the control hyperparameters are reported in Tab. 6.

For each PSO search, the position values (x, y, z) of each

Algorithm 2 Cross-Modal Target Preparation

```
def cross_modal_target_preparation(cross_modal_proposals,
                                   novel_object_bank,
                                   calibrations):
    """
    Input:
    - cross_modal_proposals:
        List of (2D proposal, 3D cluster) pairs
    - novel_object_bank:
        Set of novel objects for training
    - calibrations: Intrinsic and extrinsic
                    calibration parameters
    """

    cross_modal_targets = []
    box_search_cost_costs = []

    for pair in cross_modal_proposals:

        2d_proposal, 3d_object_cluster = pair

        # Fit a 3D bounding box to the proposal.
        # 3D_box_params = (x,y,z,l,w,h,ry)
        3D_box_params, box_search_cost =
            adaptive_3D_box_search(3d_object_cluster,
                                   2d_proposal,
                                   PSO_params)

        # Get instance mask (from SAM)
        instance_mask = get_instance_mask(2d_proposal)

        # Selective Alignment Filters
        is_not_occluded = occlusion_filter(2d_proposal,
                                           instance_mask)
        is_high_res = resolution_filter(2d_proposal)
        is_aligned_mv =
            multi_view_alignment_filter(2d_proposal,
                                        3D_box_params,
                                        calibrations)
        fit_for_alignment =
          is_not_occluded & is_high_res & is_aligned_mv

        # Get 2D Embedding from CLIP
        2D_image_embed = CLIP(2d_proposal)
        high_level_novel_class = classify(
            2D_image_embed,
            set_of_novel_classes
        )

        # Prepare novel object target
        novel_object_target = (
            3D_box_params,
            2D_image_embed,
            high_level_novel_class,
            fit_for_alignment
        )

        cross_modal_targets.append(novel_object_target)
        box_search_costs.append(box_search_cost)

    # Solve conflicts from one-to-many matching in CMA
    cross_modal_targets = resolve_CMA_conflicts(
        cross_modal_targets,
        box_search_costs
    )

    # Perform NMS to remove duplicates
    cross_modal_targets =
        NMS(cross_modal_targets, box_search_costs)

    # Update novel object bank
    update(novel_object_bank, cross_modal_targets)

    return
```

Table 6. PSO hyperparameters used in adaptive 3D box search.

Parameter	Description	Value
N_{swarm}	Swarm size	50
N_{iter}	Iterations per particle	3000
w_{init}	Initial inertia weight	10.0
w_{end}	End inertia weight	0.1
c_1	Cognitive coefficient	1.0
c_2	Social coefficient	1.0
C_{noise}	Initialization noise	0.1

candidate are initialized in areas with a high likelihood of corresponding to the true object center. In particular, half of the candidates are initialized at the closest point to the center frustum ray, and the rest are initialized at the mean of the object point cluster $\mathcal{P}_{\text{obj}}$. To accommodate larger objects that require a broader search space, noise proportional to the anchor's size is sampled from $\mathcal{N}(0, C_{\text{noise}}(\frac{\mathbf{A}_{\text{max}}+\mathbf{A}_{\text{min}}}{2}))$ and added to the initialized position. The dimension and orientation parameters are initialized uniformly across candidates. The inertia weight w follows a cosine annealing schedule to balance exploration with exploitation. We follow Find n' Propagate's [10] class anchors.

6.4. Selective Alignment

During selective alignment, OV-SCAN uses class-specific thresholds τ_{occ} for the occlusion filter since objects naturally occupy different proportions of space within a 2D proposal. As illustrated in Fig. 8, a car generally occupies a larger area in its instance mask compared to a pedestrian, resulting in a significantly higher proportion of pixels classified as instance pixels. Consequently, the pixel distribution within instance masks varies considerably across classes, making a uniform threshold for determining occlusion level inadequate.

Figure 8. **Percentage of instance pixels across object classes.** Vehicles, such as cars, typically occupy a greater proportion of pixels within their 2D proposals compared to objects like pedestrians and traffic cones.

To address this variability, reasonable threshold values for $\boldsymbol{\tau_{occ}}$ are manually determined for each class, as shown in Tab. 7, ensuring more robust estimation of highly occluded objects.

Table 7. Parameterization of τ_{occ} for different novel classes.

Novel Class	τ_{occ}
car	0.5
truck	0.5
pedestrian	0.25
bicycle	0.4
motorcycle	0.4
bus	0.5
traffic cone	0.25
barrier block	0.35
construction vehicle	0.5

6.5. Hierarchical Two-Stage Alignment Head

In Stage 1, H2SA employs classification as an auxiliary task to generate high-level text prompts for alignment. Following TransFusion-L [1], it regresses class-specific heatmaps to jointly localize and classify object proposals. We compute the class-based text prompt embeddings $\hat{A}_{\text{text}}$ ahead of time for retrieval. The top K proposals are then passed through an object decoder to produce the set of features $\mathcal{O}_{\text{3D}}$. In stage two, H2SA aligns each 3D object embedding $\mathcal{O}_{\text{3D}}$ with its 2D counterpart $\mathcal{A}_{\text{2D}}$. H2SA passes $\hat{A}_{\text{text}}$ through a set of linear layers to generate multi-scale text prototypes $\{W_H, W_{2H}, W_{4H}\}$. The Cross-Modal Distillation Block (CMDB) refines and upscales these prototypes, distilling $\mathcal{O}_{\text{3D}}$ into multi-scale representations. The first-step CMDB operation is defined as:

$$W'_H = \text{LN}(\text{MHA}(W_H, \mathcal{O}_{\text{3D}}, \mathcal{O}_{\text{3D}}) + W_H), \tag{11}$$

where LN is layer normalization and MHA is multi-head attention. Next, CMDB fuses the refined text prototype W'_H with the 3D object embedding $\mathcal{O}_{\text{3D}}$ to produce a unified feature U_{2H} for the next step. This fusion is achieved through channel-wise concatenation, followed by a feed-forward network:

$$U_{2H} = \text{LN}(\text{FFN}(\text{concat}(W'_H, \mathcal{O}_{\text{3D}}))). \tag{12}$$

This process iteratively integrates features across different scales, enabling robust cross-modal alignment to higher-dimension alignment targets.

6.6. Prompt-based Classification

As mentioned in Sec. 3.5, OV-SCAN employs a specific-to-broad strategy, relying exclusively on H2SA for classification. First objects are classified into fine-grained subclasses before being mapped to their respective novel classes for evaluation. To achieve this, we utilize the frozen CLIP text encoder to generate text embeddings using the template `"a type of {SUBCLASS}."` as the text prompt. For each object proposal generated by the detector, the fine-grained subclass is determined by selecting the subclass with the highest object-text similarity. The corresponding label $\hat{c}_{\text{fg}}$ for each object proposal is computed as:

$$\hat{c}_{\text{fg}} = \arg\max_{c_i \in C_{\text{fg}}} S_C(\text{H2SA}(\mathcal{O}_{\text{3D}}, \hat{\mathcal{A}}_{\text{text}}), t_{c_i}), \tag{13}$$

where C_{fg} denotes the set of fine-grained subclasses and t_{c_i} is the text embedding corresponding to subclass c_i. For the nuScenes dataset, the fine-grained subclasses used are outlined in Sec. 6.6. OV-SCAN follows the same procedure for the KITTI dataset.

Table 8. **Fine-grained subclasses for each high-level novel class in the nuScenes dataset.** Con.V. refers to construction vehicle.

Novel Class	Fine-grained Subclasses
car	sedan, van, minivan, hatchback, suv, coupe, police car, sprinter van, taxi
truck	pickup truck, tow truck, semi-truck, gasoline truck, delivery truck, garbage truck, fire truck, flatbed truck, ambulance, cement truck, dump truck
bus	school bus, coach bus, double-decker bus, transit bus, shuttle bus, minibus
trailer	portable message board trailer, flatbed trailer, freight trailer, cargo trailer
Con.V.	excavator, bulldozer, forklift, construction loader, construction lift
pedestrian	adult, construction worker, police officer, child
motorcycle	cruiser motorcycle, sport motorcycle, touring motorcycle, moped
bicycle	bicycle
traffic cone	traffic cone, traffic drum, traffic delineator post
barrier	plastic jersey barrier, concrete jersey barrier

7. Extending to Additional Novel Classes

This section outlines steps to expand the open set of novel classes:

1. Extend the existing set of novel classes provided to Grounding DINO and SAM. Regenerate the set of novel object proposals for the given dataset.
2. For each additional novel class, add additional anchor boxes for the adaptive 3D box search and regenerate the novel object bank for the dataset.
3. Update the BEV encoder in TransFusion-L to incorporate heatmaps for the newly added high-level novel classes and train OV-SCAN.
4. For prompt-based classification, provide the set fine-grained subclasses for each additional novel class.

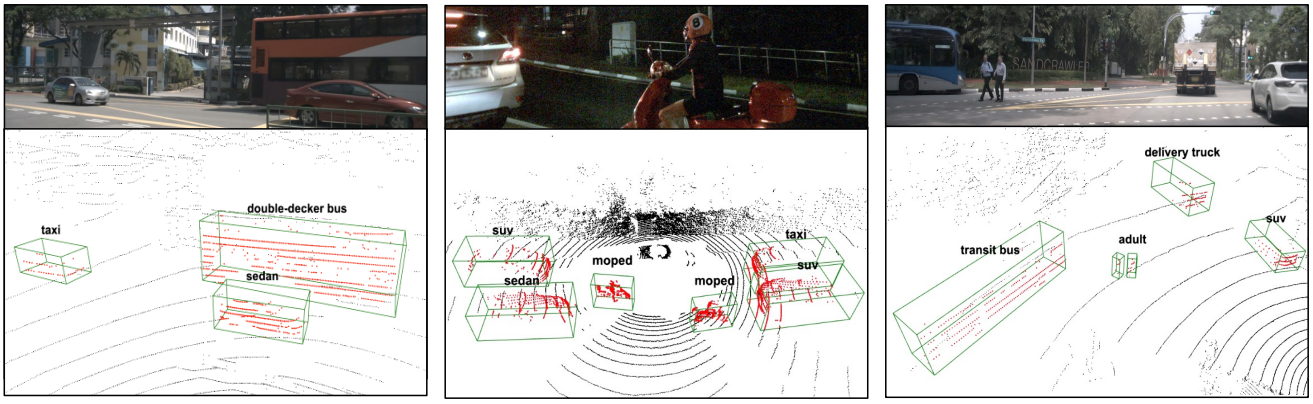

Figure 9. **Visualization results on various traffic scenarios.** OV-SCAN detects novel objects in scenarios including oncoming traffic (left), vehicles stopped at a traffic light (center), and objects encountered at a busy intersection (right).

Table 9. **Results for various weather conditions on the nuScenes validation set.**

Method	Day	Night	Dry	Wet
OV-SCAN	31.1	23.4	31.8	25.6
OV-SCAN-Fusion	34.0	20.7	34.4	29.6

8. Robustness of Multi-Sensor Fusion

In Tab. 9, we observe that OV-SCAN-Fusion achieves improved detection performance over OV-SCAN across all weather conditions except at night. While the fusion strategy generally enhances robustness by leveraging complementary modalities, its effectiveness diminishes in low-light conditions. The reduced reliability of image features at night introduces noise into the model, highlighting a limitation in sensor fusion under varying lighting conditions.

9. Additional Visualizations

Cross-Modal Alignment. Fig. 9 presents additional qualitative results to highlight the cross-modal alignment performance. OV-SCAN detects two cars passing a parked double-decker bus in oncoming traffic. It also accurately identifies a pair of mopeds stopped at a traffic light. Finally, a variety of objects are detected at a busy intersection.

3D Box Search. In Fig. 10, we perform a side-by-side comparison of the box regressed when fueled by SC-NOD vs. Greedy Box Seeker from Find n' Propagate. Both methods use Transfusion-L [1] off-the-shelf, while displaying detections with confidence scores over 0.05. However, during training, Find n' Propagate natively follows its predefined *Setting 2*, treating three classes ("car," "pedestrian," and "bicycle") as base and leaving the remaining classes as novel. In contrast, OV-SCAN treats every class as novel. OV-SCAN regresses notably more precise bounding boxes for novel classes. Additionally, Find n' Propagate tends to produce more false positives due to its increased recall strategy, whereas OV-SCAN maintains better precision and localization accuracy.

10. Annotation Cost

Our adaptive box search was implemented as a multi-threaded CPU process and was run on an enterprise server (two Xeon Gold 6140 CPUs, 768 GB RAM, 8 V100 GPUs). Labeling the full nuScenes train and validation sets took three to four days, while model training required an additional one and a half days. This offline cost was reasonable considering the effort required to manually annotate large-scale 3D datasets from scratch. The approach is scalable and configurable—users can adjust the number of iterations or extend to new object types with no extra human effort. A GPU implementation could further reduce cost.

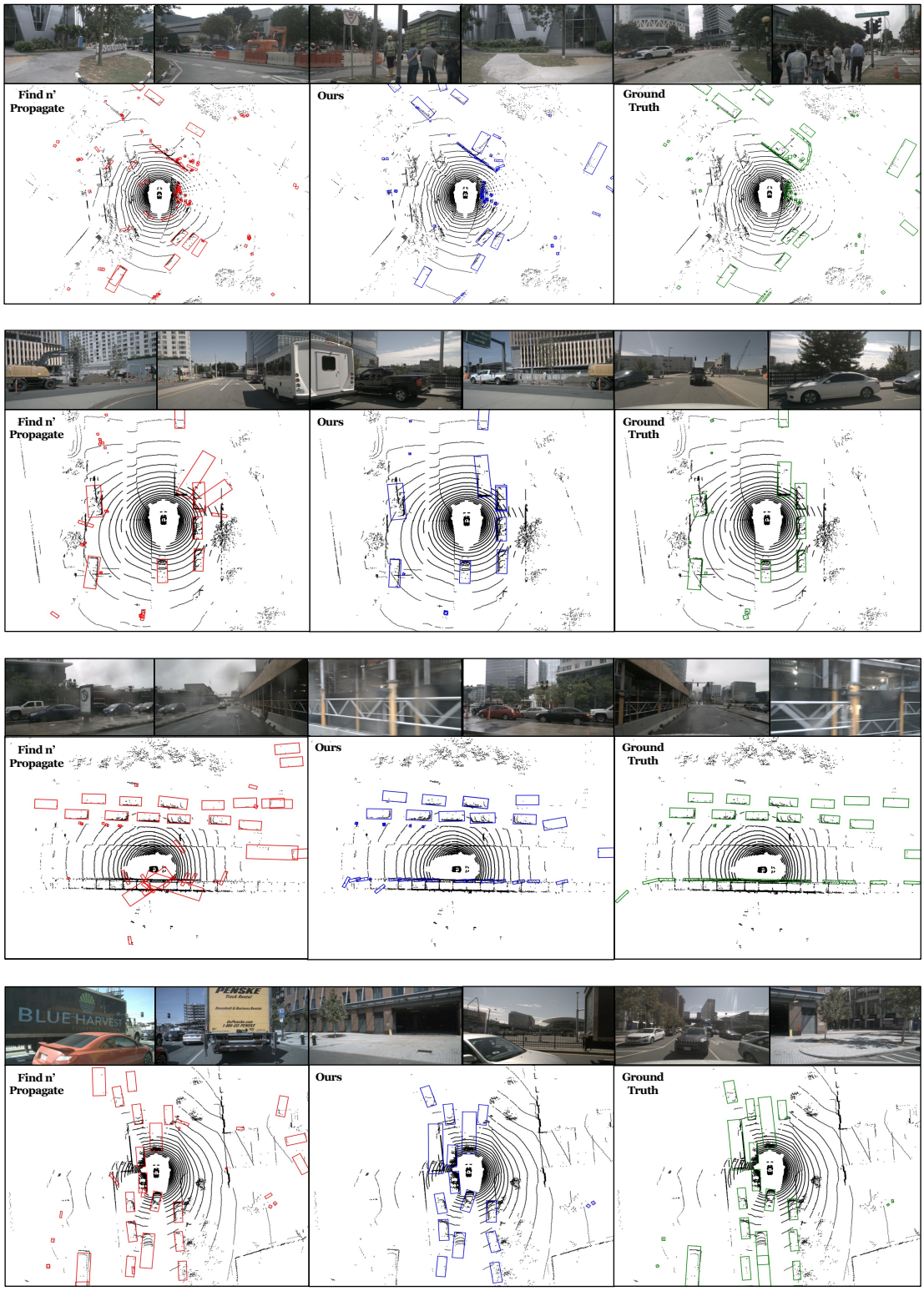

Figure 10. **Comparison between OV-SCAN and Find n' Propagate [10].** OV-SCAN regresses more precise bounding boxes than Find n' Propagate without requiring any human-annotated labels. We compare to Find n' Propagate in *Setting 2*, which uses three base classes ("car", "pedestrian", and "bicycle"), leaving the rest as novel.