1. More Implementation Details

1.1. Deformable Attention

In the Progressive Correspondence Aligner, we employ deformable attention to enable adaptive local feature aggregation. Given an input feature map $x \in \mathbb{R}^{C\times H\times W}$, a query element q is associated with a feature vector $z_q \in \mathbb{R}^C$ and a 2D reference position $p_q \in \mathbb{R}^2$. The deformable attention feature at query q is formulated as:

$$\text{DefAttn}(z_q, p_q, x) = \sum_{m=1}^{M} W_m \Bigg(\sum_{k=1}^{K} A_{mqk} W'_m x(p_q + \Delta p_{mqk}) \Bigg), \tag{1}$$

where M is the number of attention heads and K is the number of sampled keypoints per head ($K \ll HW$). For each head m, Δp_{mqk} and A_{mqk} are the predicted sampling offsets and attention weights for the k-th point, respectively, and $W_m, W'_m \in \mathbb{R}^{C\times C}$ are learnable projection matrices.

This mechanism allows each query to dynamically select a sparse set of relevant features around its reference position p_q, enhancing robustness to local sparsity and geometric distortions commonly present in LiDAR point clouds.

1.2. MLP-Mixer

In the Confidence-Aware Structural Localization module, the confidence-weighted keypoint pairs F_{pair} are processed by a lightweight MLP-Mixer, which captures interactions among all correspondences and aggregates them into a compact structural representation. Specifically, the MLP-Mixer iteratively models dependencies across both spatial and channel dimensions using cascaded 1D convolution layers:

$$F_{\text{fusion}} = \left[\text{Conv1D}([\text{Conv1D}(F_{\text{pair}})]^\top)\right]^\top. \tag{2}$$

This design effectively encodes structural relations while maintaining computational efficiency.

1.3. Training Details

We set the number of refinement stages to $S = 3$ and use $N_q = 27$ query keypoints ($n_x = n_y = n_z = 3$). We employ $L = 3$ layers in the Adaptive Keypoint Extractor. The Transformer uses 8 attention heads with a feature dimension of $C = 128$ and an FFN dimension of 1024. For fair evaluation across datasets, we adopt consistent training settings with dataset-specific configurations. For NuScenes, we use a batch size of 128 and a point cloud range of $[-9.6, -9.6, -3.0, 9.6, 9.6, 3.0]$. The voxelization module discretizes points into a grid with a voxel size of $[0.15, 0.15, 0.3]$ and $[21, 128, 128]$ cells, followed by a VoxelNet backbone. For KITTI, we use the same batch size but a smaller point cloud range $[-4.8, -4.8, -1.5, 4.8, 4.8, 1.5]$ and a finer voxel size $[0.075, 0.075, 0.15]$ to preserve geometric details better.

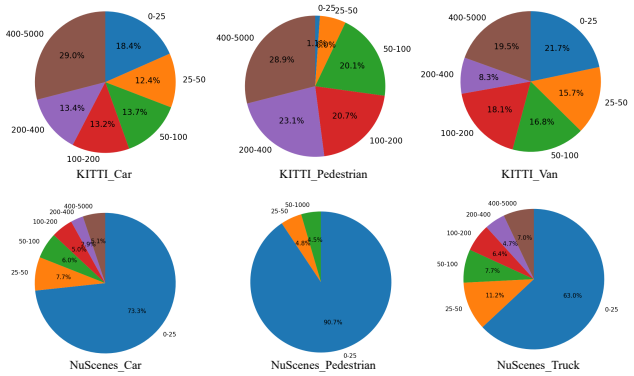

Figure 1. The per-frame distribution of LiDAR points across different object categories in the KITTI and nuScenes test datasets.

1.4. Dateset Details

KITTI consists of 21 training and 29 test video sequences captured in urban driving scenes. Sequences 0–16 are used for training, 17–18 for validation, and 19–20 for testing. The number of LiDAR points falling on each tracked object varies significantly across frames (see Fig. 1). For cars, roughly half of the instances contain fewer than 100 points, while for vans the majority of instances are represented by fewer than 100 points. Thus, a large portion of objects in KITTI are observed with relatively few points.

NuScenes contains 1000 scenes, split into 700/150/150 for train/val/test. We follow the official "train track" split for training and report results on the validation set. Compared to KITTI, LiDAR observations in nuScenes are much sparser. As shown in Fig. 1, 73.3% of car instances and 63.0% of truck instances are covered by fewer than 25 points. The sparsity is most extreme for pedestrians, where 90.7% of frames contain fewer than 25 points and almost no instances exceed 100 points. This severe sparsity makes nuScenes a particularly challenging benchmark for 3D single-object tracking.

2. More Visualization

2.1. Failure Cases

Representative examples with predicted keypoints and low confidence scores are shown in Fig. 2. We analyze failure modes and identify two dominant scenarios: (1) **extreme sparsity**, where the target is represented by very few points (e.g., distant vehicles), leading to significant loss of geometric structure and making it difficult to establish reliable keypoint correspondences; (2) **severe occlusion**, where most parts of the target are occluded, rendering the structurally

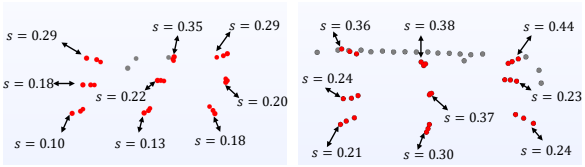

Figure 2. Examples of extreme sparsity and severe occlusion. Predicted keypoints and their confidence scores are shown.

informative regions for stable keypoint extraction unobservable.

In both cases, the lack of reliable geometric correspondences results in low-confidence predictions of keypoints. This observation further indicates that the predicted confidence scores effectively reflect the reliability of the established correspondences.

3. More Ablation Studies

3.1. Efficiency Analysis

As shown in Tab. 1, we compare two state-of-the-art category-specific and three unified trackers using reported results from the original papers ("–" indicates not reported). Overall, UniKPT achieves a favorable efficiency–accuracy trade-off as a unified model.

Table 1. Efficiency and accuracy comparison

Method	**Succ./Prec.**	**FLOPs**	**FPS**	**Type**
StreamTrack	55.75 / 69.22	–	40	Specific
P2P	59.84 / 72.13	1.23G	71	Specific
MoCUT	51.19 / 64.63	3.27G	48	Unified
TrackAny3D	54.57 / 66.25	–	28	Unified
UniKPT (Ours)	**64.21 / 77.29**	**0.55G**	**37**	**Unified**

Table 2. Sensitivity study on the number of layers L.

L	1	2	3	4
Succ./Prec.	63.0/75.9	63.6/76.8	64.2/77.3	64.1/77.3

3.2. Ablation Study on the Number of Layers L

Table 2 presents the ablation study on the number of layers L in the Adaptive Keypoint Extractor. As L increases from 1 to 3, both Success and Precision consistently improve, indicating that a deeper network can better model feature interactions among template keypoints and enhance structural representation capability. When further increasing L to 4, the performance gain becomes marginal, suggesting a saturation effect. Therefore, we set $L = 3$ as the default.