p5	p15	p25	p50	p75	p95
1.2%	3.0%	3.8%	5.6%	8.0%	12.3%

Table 6. DriveTrack's occlusion error on Kubric (synthetic).

A. Implementation Details

We implement DriveTrack in Python using the Waymo Open Dataset SDK [21]. The dataset has separate tables for each data feature, such as video, bounding boxes, and LiDAR data. Each feature table is further partitioned into Parquet files for each scene. We use Dask Distributed to merge the necessary tables one scene at a time before processing the annotations, using 8 workers with 2 threads each.

We preprocess the scene video, depth maps, and 3D LiDAR point clouds before building the individual annotation point tracks. We use the Waymo SDK to convert the provided range images to 3D point clouds, and then project the 3D point clouds to the image plane to generate depth maps (§4.4). For processing nearest-neighbor depth maps, we parallelize frames across CPUs. Using 32 CPU cores, processing 200 depth maps takes around 25 seconds. For depth maps generated using CompletionFormer [27], we spawn a process for each available GPU and use a multiprocessing queue to pass jobs to each process. On 6 NVIDIA V100 GPUs, processing 200 depth maps takes around 35 seconds.

After preprocessing the scene and caching the videos, depth maps, and 3D point clouds, we process each annotation (object) in parallel as described in (App. 4). In total using 32 CPU cores, each annotation takes about 10 minutes to complete.

B. DriveTrack Annotations

B.1. Video animation

We recommend watching the supplementary video to see several qualitative visualizations of DriveTrack's annotations. Our video includes 11 scenes, covering a variety of lighting conditions, weather patterns, and occlusion configurations. For each scene, we show 10 randomly-sampled points from DriveTrack's annotations; each point disappears whenever DriveTrack labels it as occluded.

B.2. Examples

Fig. 10 visualizes the annotations computed by DriveTrack for several scenes in the Waymo dataset [21]. The top two rows illustrate how DriveTrack robustly detects different types of occlusions, from both cars and signposts. The middle two rows demonstrate a nighttime scene with vehicle occlusions. The bottom two rows demonstrate occlusions from pedestrians in the scene.

B.3. Pseudo-ground-truth evaluation

Assessing the quality of DriveTrack is challenging; in §5, we do a proxy evaluation with vehicle speeds. App. B.1 and App. B.2 show visual quality through figures and a supplementary video. We also evaluated DriveTrack's labeling workflow against Kubric, a synthetic benchmark for which we have ground truth. Using Kubric's perfect bounding box annotations, we found that the tracks generated by DriveTrack are 100% accurate. To evaluate occlusion accuracy, we subsampled 50% of the ground-truth depth map and used nearest-neighbor depth completion to fill in the masked values. Table 6 shows percent occlusion error at various percentiles.

B.4. Limitations

While most of DriveTrack's annotations are high-quality, there are a few edge cases where DriveTrack fails. Fig. 11 illustrates one example, where DriveTrack fails to track a point on a sliding door on a parked van. Due to the rigid body assumption, DriveTrack selects points on the open door and then tracks them through all frames. However, as the door closes, the points should follow the door and eventually become occluded once the door closes. Instead, DriveTrack leaves such points dangling in open space.

A simple way to mitigate these errors is to filter out points whose speeds deviate significantly from the annotated ones (§4.5). A more robust method is to estimate surface normals from the 3D point cloud, which would help better track the depth contours of the vehicle. We leave this to future work.

Another source of labeling error stems from noisy bounding box labels in the autonomous driving dataset. For instance, the bounding box may not be perfectly centered around the target vehicle in each frame of a track. For the Waymo dataset [21] that we used to build DriveTrack, we found that most bounding box labels are visually accurate. For other datasets, one proposal is to smooth the bounding box tracks to suppress labeling error; we leave this to future work.

C. Additional Fine-tuning Results

Fig. 12 shows additional results from fine-tuning TAPIR on DriveTrack. Each row shows three frames from a scene in DriveTrack, and visualizes tracking performance before and after fine-tuning. Fine-tuning on DriveTrack consistently improves tracking performance.

Fig. 13 shows the transfer potential of TAPIR on more scenes from the DAVIS dataset. We find that fine-tuning in particular improves the tracking accuracy of outliers.

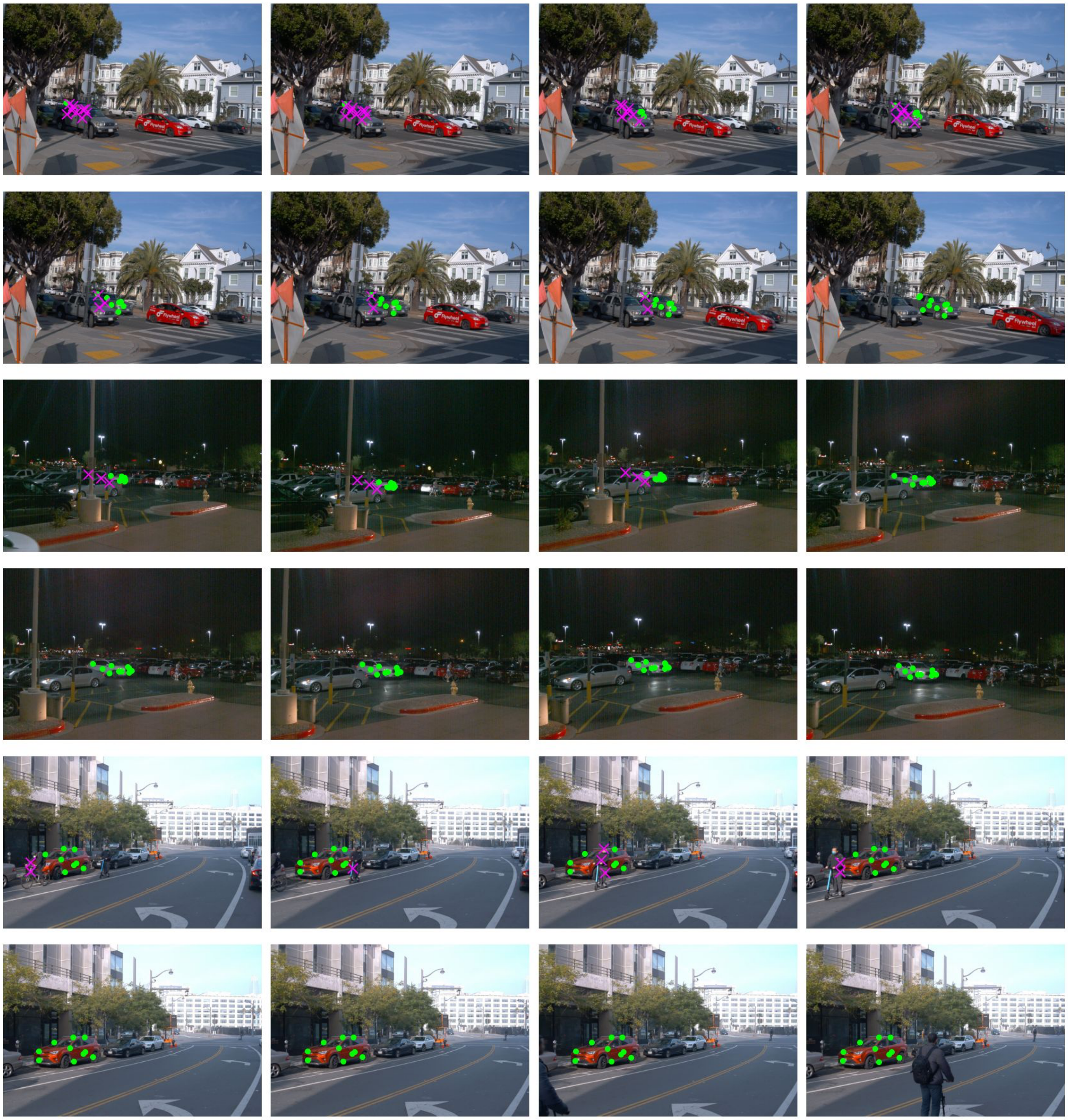

Figure 10. Example annotations from DriveTrack on three scenes, spanning a variety of lighting and weather conditions. Each batch of two rows corresponds to a new scene, spaced four frames apart on a 30-frame subsection of the scene video.

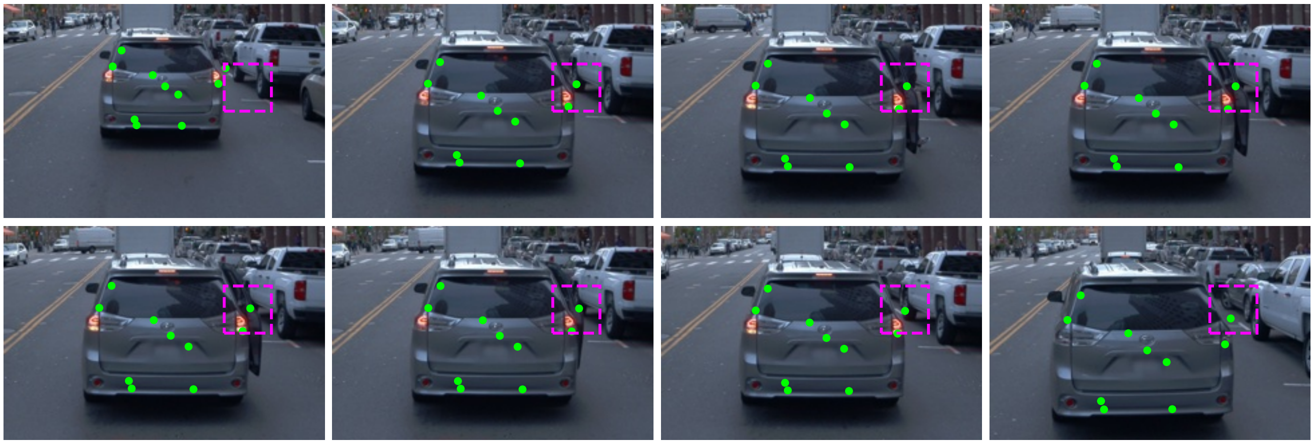

Figure 11. Example of a failure in DriveTrack's annotations, highlighted by the magenta box. When the van door closes, the point selected on the door frame becomes dangled in space instead of tracking the door as it closes.

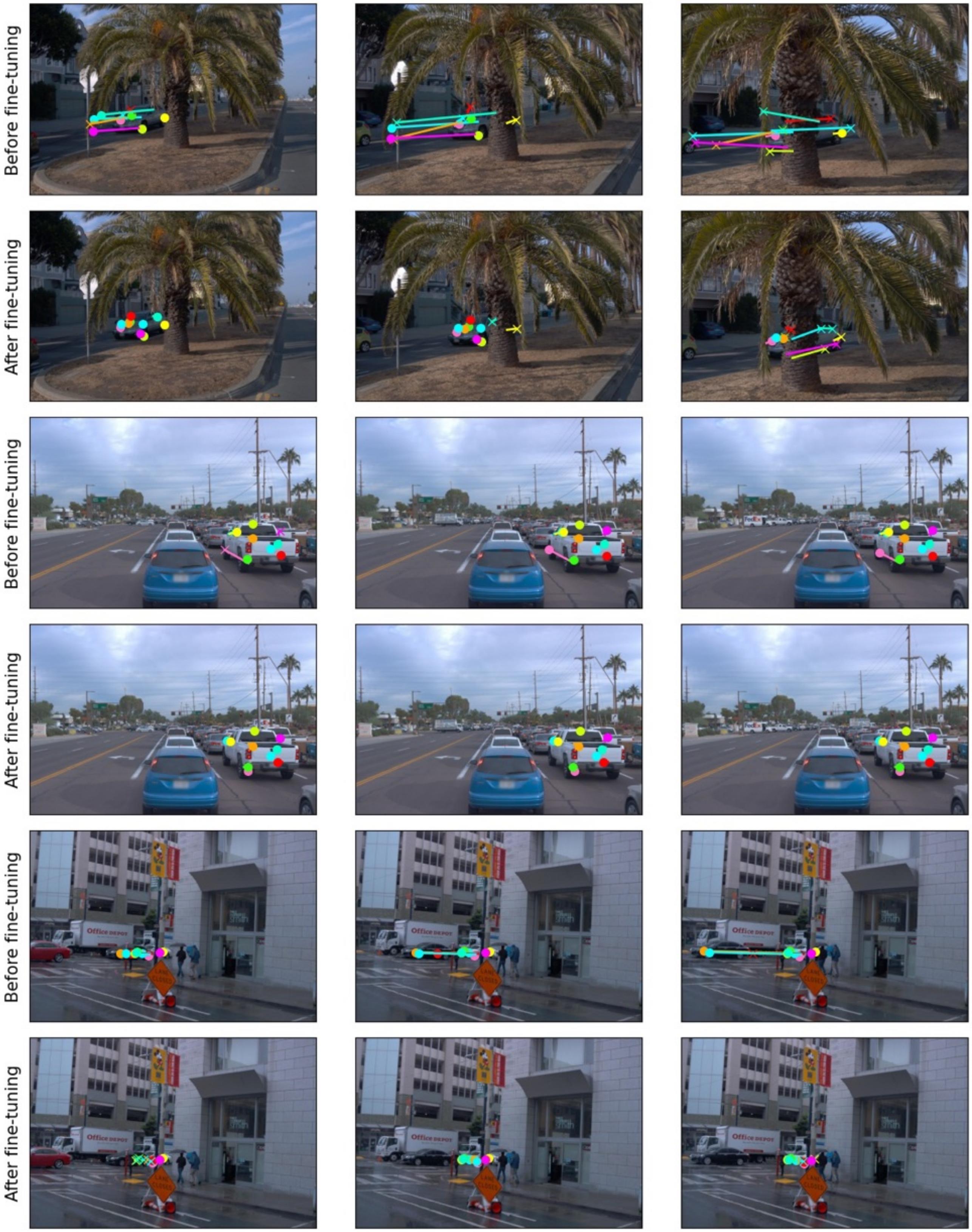

Figure 12. Point tracks predicted by TAPIR before and after fine-tuning. The markers indicate the locations predicted by each model, and the line segments lead to their respective ground-truth locations. ● denotes points predicted as visible, and × denotes points predicted as occluded.

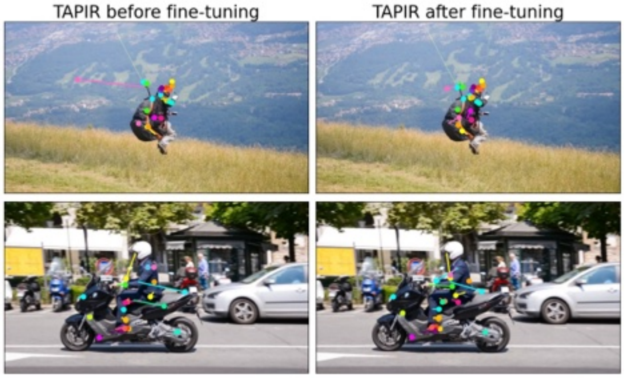

Figure 13. Point tracks predicted by TAPIR on scenes from DAVIS.