DriveTrack: A Benchmark for Long-Range Point Tracking in Real-World Videos

Arjun Balasingam
MIT CSAIL

Joseph Chandler
MIT CSAIL

Chenning Li
MIT CSAIL

Zhoutong Zhang
Adobe Systems

Hari Balakrishnan
MIT CSAIL

Abstract

This paper presents DriveTrack, a new benchmark and data generation framework for long-range keypoint tracking in real-world videos. DriveTrack is motivated by the observation that the accuracy of state-of-the-art trackers depends strongly on visual attributes around the selected keypoints, such as texture and lighting. The problem is that these artifacts are especially pronounced in real-world videos, but these trackers are unable to train on such scenes due to a dearth of annotations. DriveTrack bridges this gap by building a framework to automatically annotate point tracks on autonomous driving datasets. We release a dataset consisting of 1 billion point tracks across 24 hours of video, which is seven orders of magnitude greater than prior real-world benchmarks and on par with the scale of synthetic benchmarks. DriveTrack unlocks new use cases for point tracking in real-world videos. First, we show that fine-tuning keypoint trackers on DriveTrack improves accuracy on real-world scenes by up to 7%. Second, we analyze the sensitivity of trackers to visual artifacts in real scenes and motivate the idea of running assistive keypoint selectors alongside trackers.

1. Introduction

Long-range keypoint tracking in videos underpins many computer vision applications, including autonomous driving [14], robotics [24], pose estimation [8], 3D reconstruction [16], and medical imaging [26]. Each of these applications involves moving objects and moving cameras. Keypoint tracking—whose goal is to track unique points in the presence of mobility and occlusions—is an active area of research [4, 5, 10, 29].

Most proposals follow the Track Any Point (TAP) [4] formulation: given a video and a set of query points, the algorithm must estimate the locations of those points in all other frames where they are visible. The underlying tracking algorithms vary significantly. TAPIR [4, 5] is an end-to-end method that predicts correspondences using feature maps and cost volumes. By contrast, PIPs++ [10, 29] stitches optical flow vectors together to construct long-range trajectories. These are two recent methods that improve the state-of-the-art, adding to a number of techniques proposed over the last two decades.

This paper observes that the accuracy of state-of-the-art trackers suffers on real videos. In particular, noisy visual characteristics—such as texture, lighting variations, occlusions, and motion-induced image distortions—can hinder tracking performance (§3). The key problem is that modern trackers train on vast *synthetic* datasets [9, 13, 29] whose scenes do not exhibit these imperfections. We are aware of only two benchmarks for the TAP task on real-world videos [12, 19], with each offering (only) tens of human-labeled annotations per scene.

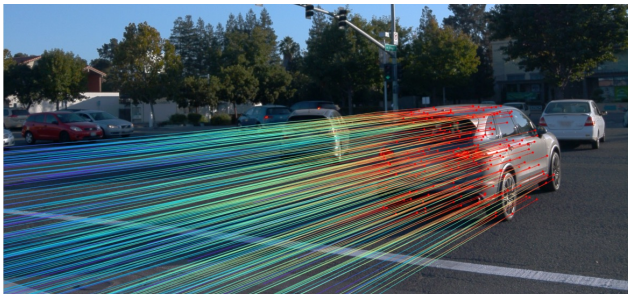

Figure 1. DriveTrack automatically generates dense, accurate, and long-range point track annotations for autonomous driving videos.

To overcome this shortcoming, we propose *DriveTrack*, a large-scale benchmark for long-range point tracking tasks. DriveTrack brings to real-world videos the density and fidelity of annotations available only for synthetic benchmarks today. By using camera feeds from cars driven in urban areas, DriveTrack captures realistic motion, noisy visual attributes, and occlusions, which synthetic [9, 13] or rendered [29] datasets do not model. Fig. 1 shows the annotations computed by DriveTrack for a driving scene. Although DriveTrack is built on autonomous driving videos, it captures the wide variety of visual artifacts typical in real-world scenes.

To generate point tracks, we adapt methods used by synthetic benchmarks [9, 29] that use rendering software to precisely annotate the motion of simulated trajectories. However, real-world videos do not have the luxury of a simulator. To overcome this challenge, DriveTrack uses timestamped LiDAR point clouds, object bounding box annotations, and camera poses and orientations [2, 7, 21, 25]. Since LiDAR point sweeps do not have 1:1 correspondence over time [21], DriveTrack cannot compute correspondences between adjacent point clouds, as synthetic benchmarks are able to. DriveTrack instead transforms each timestamped point cloud according

	FlyingThings++ [10]	Kubric [9]	Kinetics [12]	DAVIS [19]	PointOdyssey [29]	**DriveTrack**
Resolution	540×960	256×256	≥720×1080	1080×1920	540×960	1280×1920
Frame rate (Hz)	8	8	25	25	30	10
Avg. trajectory count	1024	Arbitrary	26.3	21.7	18,700	100,800
Avg. frames per video	8	24	250	67	2,035	84
# of training frames	21,818	Arbitrary	–	–	166,000	672,000
# of validation frames	4,248	Arbitrary	–	–	24,000	84,000
# of test frames	2,247	Arbitrary	297,000	1,999	26,000	84,000
# of point annotations	300M	Arbitrary	80M	400K	49B	84B
# of point tracks	16K	Arbitrary	32K	650	11K	1B
Real-world scenes	✗	✗	✓	✓	✗	✓
Depth maps	✓	✓	✗	✗	✓	✓
Object masks	✓	✓	✓	✓	✓	✓
Multiple views	✗	✓	✗	✗	✓	✓

Table 1. DriveTrack is the first benchmark on real-world videos to offer annotations that match the scale and fidelity of synthetic datasets.

to the camera pose and bounding box annotations to generate hundreds of thousands of highly accurate point annotations per object. We also implement several refinements to ensure that our annotations are robust to noise in hand-labeled bounding boxes.

DriveTrack can annotate point tracks for rigid bodies in any dataset of real-world videos that includes point clouds, 3D object segmentations, and camera poses. With this paper, we release point tracking annotations for the Waymo Open Dataset [21]. Our dataset contains 1 billion point tracks for over 10,000 distinct objects across 24 hours of video. Table 1 compares DriveTrack to other point tracking benchmarks.

DriveTrack makes long-range tracking practical for real-world scenes. This paper presents results for two use cases:

- **Fine-tuning keypoint trackers**. We fine-tune TAP-Net [4], TAPIR [5], PIPs [10], and PIPs++ [29] on DriveTrack, showing an improvement of 4-7% on DriveTrack's test set and 1-2% on DAVIS [19] (§6).
- **Keypoint sensitivity.** Visual imperfections make keypoint tracking in real videos more challenging. We use the scale of annotations made available by DriveTrack to quantify the sensitivity of tracking accuracy to visual imperfections. From this analysis, we motivate how DriveTrack can be used to build keypoint selectors, which can recommend robust keypoints to use with trackers (§7).

DriveTrack's data generation code and annotations are available at `drivetrack.csail.mit.edu`.

2. Related Work

Real-world datasets. The TAP-Vid benchmark [4] released annotations for DAVIS [19] and Kinetics [12], two real-world video datasets. TAP-Vid introduces a workflow that helps a human annotate keypoint tracks in a sequence of video frames. State-of-the-art keypoint trackers [4, 5, 10] evaluate on the DAVIS and Kinetics benchmarks to quantify performance on real-world scenes. However, since the scale of annotations is on the order of tens of trajectories per scene, these benchmarks are viable only for evaluation, and not for fine-tuning models to be robust to visual artifacts seen in real-world scenes.

The KITTI dataset's Segmenting and Tracking Every Pixel (STEP) benchmark [7] tags every pixel (or 3D point) with a semantic label (*e.g.* car, truck, pedestrian, *etc.*) and a unique track ID. It constructs tracks by propagating segmentation masks using RAFT [22]. However, STEP does not annotate point correspondences over a long-range, *e.g.* the exact trajectory of a unique pixel over a video sequence. Even earlier, the Middlebury dataset [1] was the de-facto benchmark for optical flow and motion estimation tasks. This dataset consists of a mix of real-world and synthetic scenes. While the annotation quality is high, the sheer number of annotations, as with DAVIS and Kinetics, is small.

Synthetic benchmarks. Most of the data used to train and evaluate long-range motion estimation has been synthetic. Popular benchmarks include FlyingChairs [6], FlyingThings3D [17], and AutoFlow [20] providing short-range (*i.e.* 2 frames) annotations, and Kubric [9], PointOdyssey [29], and FlyingThings++ [10] providing long-range labels. These datasets contain different variants of generated objects moving in random directions on random backgrounds. Of these benchmarks, PointOdyssey has annotations with the greatest volume and fidelity, and uses rendering tools and real motion traces to synthesize photo-realistic scenes. However, it fails to capture visual artifacts characteristic of real scenes that hamper the performance of keypoint trackers (§3).

Keypoint tracking. Datasets with dense and accurate point tracks are critical to developing and evaluating keypoint tracking methods that follow the TAP formulation. TAPIR [5], TAP-Net [4], PIPs [10], and PIPs++ [29] are four methods that have pushed the state-of-the-art over the last two years. Other approaches to keypoint tracking rely on optical flow [22] or on structure-from-motion [28]. The focus of this paper is on DriveTrack and the new use cases it enables: fine-tuning trackers on real-world scenes and an analysis of keypoint sensitivity.

3. Motivation

Keypoint tracking on videos is a decades-old problem in computer vision, but it is also evolving rapidly. Over the last two years, four keypoint trackers [4, 5, 10, 29] have pushed state-of-the-art results. All recent methods train on large-scale

Figure 2. State-of-the-art keypoint trackers struggle on real-world scenes with complex lighting and texture attributes. Shown here are point tracks predicted by TAPIR [5] relative to their ground-truth locations for 4 scenes from the DAVIS dataset [19].

synthetic benchmarks [6, 9, 17, 29]. In this section, we make the case for a similar point tracking benchmark for real-world scenes by highlighting current limitations in the performance of state-of-the-art keypoint trackers.

Keypoint trackers suffer on real videos. Fig. 2 shows the tracking error achieved by TAPIR [5] on four scenes from the DAVIS dataset [19]. The markers indicate the locations predicted by TAPIR, and the line segments lead to their respective ground-truth locations.

Notice that the tracking quality varies significantly. For instance, TAPIR tracks the car turning the roundabout (top left) accurately around well-defined edges and corners, like the door handles and wheel spokes. However, the predicted tracks drift from ground truth on reflective surfaces, like the windows. In the paragliding video (bottom left), TAPIR struggles on patches of the parachute with lighting variations, and on the suspension ropes, which blend with the saturated background.

Limitations of synthetic datasets. While the synthetic benchmarks on which these models are trained offer annotations at a large scale and fidelity, they do not exhibit visual artifacts that are all too common in real-world scenes. Synthetic benchmarks, such as Kubric [9], FlyingThings++ [17], and PointOdyssey [29], only capture rudimentary lighting conditions like shadows and selects from a corpus of simplistic rendered objects. Textures are simple and lighting patterns are monotonic. Given these limitations, it is unrealistic to expect these keypoint trackers to excel on real-world scenes.

Annotating real-world datasets. The problem is that there do not exist any benchmarks on real-world scenes that offer the fidelity and scale of annotations available for synthetic benchmarks. Existing benchmarks on real-world scenes, like DAVIS and Kinetics, curate only a handful of human-labeled keypoint tracks per scene. Moreover, the annotated keypoints are biased toward locations that are naturally easier for a human to track.

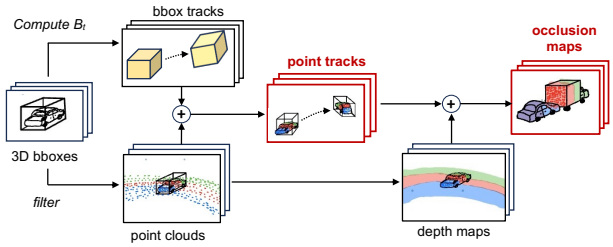

Figure 3. DriveTrack transforms each timestamped point cloud according to the vehicle's camera poses and a target object's bounding boxes to automatically and accurately annotate point tracks for that object.

Notation	Meaning
N	number of points tracked
F	duration of RGB video
R	camera extrinsic matrix
K	camera intrinsic matrix
W_t	transforms point from vehicle frame to world frame at time t
B_t	transforms bounding box from origin to location in world frame at time t
X_t	target object's LiDAR point cloud at time t
$\hat{X}_\tau^{(t)}$	projection of X_t at time τ
T_V	set of 3D point tracks in vehicle frame
$\hat{T}_I$	set of 2D point tracks in image space
$\hat{D}$	depth map
$\hat{O}$	occlusion map

Table 2. Notation for DriveTrack's point tracking workflow.

4. DriveTrack Overview

DriveTrack leverages autonomous driving datasets [2, 7, 21] to generate dense point tracks on real-world videos. Fig. 3 illustrates DriveTrack's data generation workflow, which combines LiDAR sweeps, 3D bounding box annotations, sensor calibrations, and camera poses/orientations to derive keypoint annotations with high fidelity.

DriveTrack computes point tracks by shifting recorded point clouds by the poses of the driving (ego) vehicle. To create the benchmark, we aggregate and filter the timestamped data (§4.1), transform and track the point clouds (§4.2), and estimate occlusions (§4.3) by computing depth maps (§4.4). We release annotations on the Waymo Open Dataset [21], but our data generation workflow is compatible with other autonomous driving datasets (§4.5). Table 2 introduces notation that we use in the rest of this section to formalize the point tracking procedure. DriveTrack computes a set of 2D point tracks $\hat{T}_I \in \mathbb{R}^{N\times F\times 2}$, as well as an occlusion map $\hat{O} \in \mathbb{R}^{N\times F}$ that indicates when points are not visible from the perspective of the camera.

4.1. Dataset preparation and requirements

Autonomous driving datasets split data by different modalities. DriveTrack pre-processes this data by joining tables by timestamp, and bundles the following for each unique object k:

- We extract a length-F sequence of RGB frames containing the object, along with camera calibration matrices R and K.

We consider each of the cameras on the ego vehicle separately.

- We convert the bounding box annotations for each unique object k into a sequence of timestamped transformations $B_{k,t}$ that describe how to transform the object from the world origin to its position in the global frame.
- We extract the LiDAR point clouds $X_{k,t}$ by filtering the scene's point cloud at time t by object k's bounding box.
- We export transformations W_t that map points in the ego vehicle's reference frame to the world frame.

For simplicity, we omit the object index k in the rest of this paper.

4.2. Point tracking

DriveTrack computes correspondences in 3D and then projects those points to the image space. We compute point tracks separately for each target object that is annotated in the dataset.

Tracking with point clouds. Unlike synthetic benchmarks, DriveTrack cannot render the scene in a simulator and generate points to track. It instead is confined to a finite set of points sampled by the ego vehicle's LiDAR sensor. However, since LiDAR samples different points in each sweep, the point clouds in adjacent timesteps X_t and X_{t+1} will not have a 1:1 correspondence.

DriveTrack instead takes each point cloud X_t and "shifts" it across time according to the bounding annotations, constructing a sequence of point clouds $\{\hat{X}_\tau^{(t)} \forall \tau \in [1,F]\}$. To project X_t, we need to account for the motion of the target object and the motion of the ego vehicle. The transformation B_t captures the location of the object at each time t in the global frame, and W_t specifies how to transform a point from the ego vehicle's reference frame to the global reference frame at each time t. Thus, to project the point cloud, we (i) project to the global frame, (ii) transform to bounding box coordinate system, (iii) apply the bounding box transformation in the next timestep, and (iv) project back to the vehicle's reference frame. Formally, the projected point cloud at τ is:

$$\hat{X}_\tau^{(t)} = W_\tau^{-1} B_\tau B_t^{-1} W_t X_t \tag{1}$$

This yields N length-F tracks, where N is the number points in X_t.[1] We can repeat this procedure for *each* $t \in [1,F]$. In total, this gives $N \times F$ point tracks of length F for this target object.

Point tracks. We can define the matrix $\hat{T}_V \in \mathbb{R}^{N \times F \times 3}$ to hold the point tracks projected using the procedure described above. Formally, if $\hat{T}_V^{(i)} = \left[\hat{X}_1^{(i)} ... \hat{X}_t^{(i)} ... \hat{X}_F^{(i)}\right]$, then $\hat{T}_V = \left[\hat{T}_V^{(1)\intercal} ... \hat{T}_V^{(i)\intercal} ... \hat{T}_V^{(F)\intercal}\right]$. We can then project these points to image space using the camera matrices as follows: $\hat{T}_I = K R^{-1} \hat{T}_V$.

4.3. Occlusion estimation

DriveTrack flags a point as occluded if its 3D position is further from the camera than the nearest physical point in the same direction [9, 29]. If $\hat{x}_t \in \mathbb{R}^3$ is the 3D projection of a point $p \in \mathbb{R}^2$ relative to the ego vehicle's reference frame, then $\hat{d}_t = \|\hat{x}_t\|$ gives the distance of that point from the camera. To find the nearest physical points from the camera, we compute depth maps $\hat{D}_t \in \mathbb{R}^{W \times H}$.[2] Then, a point $p = (p_x, p_y)$ is occluded if $\hat{d}_t > \hat{D}_t[p_x, p_y]$. We repeat this for all $p \in \hat{T}_I$ to compute $\hat{O}$.

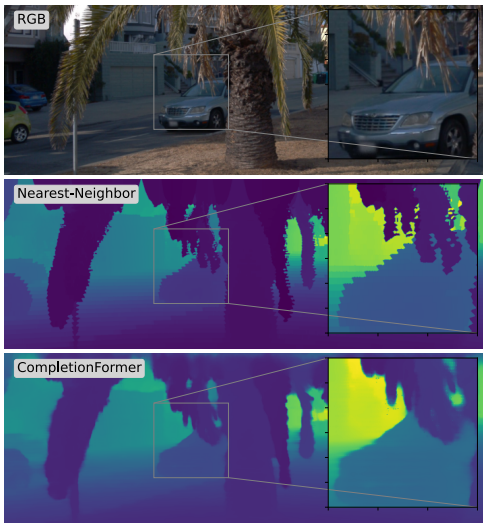

Figure 4. Depth maps computed by Nearest Neighbor and CompletionFormer [27] for a scene in the Waymo dataset [21].

4.4. Depth completion

Synthetic benchmarks export dense depth maps. However, the autonomous driving datasets that DriveTrack uses for its point tracking annotations only export *sparse* LiDAR point clouds, *i.e.* a collection of 3D points that the sensor samples through several sweeps. To create dense depth maps, DriveTrack first projects the 3D point cloud to 2D, creating a sparse depth map. Then, it leverages depth completion methods that learn a function f to interpolate a set of 2D points to a dense 3D map. We denote $X_{I,t}$ as the 2D projection of the ground-truth point cloud X_t at time t. f maps $X_{I,t}$ to a dense depth map $\hat{D}_t$, which we then use to estimate occlusions.

We consider two depth completion models. The first interpolates nearest neighbors in the image plane, by assigning each pixel in the image the same depth value as the closest ground-truth sparse depth point in $X_{I,t}$ by 2D Euclidean distance. The second method is CompletionFormer [27], a deep depth completion model that achieves state-of-the-art performance on the Kitti Depth Completion (DC) benchmark [23]. CompletionFormer uses both the sparse depth map and the RGB source image to produce a dense depth map. It uses non-local spatial propagation networks [18] to share affinity information from depth areas in the image about which the model has high confidence to areas with lower confidence. For both models, we interpolate depths at floating point pixel values [9, 29], by computing a max pooling over the neighborhood of the four corner pixels around which we interpolate. Max pooling overestimates depth

[1] This assumes (for simplicity) that each point cloud X_t has N points.

[2] §4.4 describes how we compute depth maps from the point clouds.

values compared to an alternative like bilinear interpolation, but we find that it yields more accurate occlusion maps.

Fig. 4 visualizes depth maps computed by Nearest Neighbor (middle) and CompletionFormer (bottom) for a scene. Empirically, we find that Nearest Neighbor produces accurate depth and occlusion maps on our benchmark. However, it produces jagged artifacts around the edges of objects, particularly in the foreground of the scene. Consequently, occlusion estimates tend to be inaccurate on the boundaries of vehicles or on other sharp gradients in depth. By contrast, CompletionFormer handles object edges more gracefully by leveraging non-local spatial propagation. However, it suffers on thin background objects like tree leaves or signposts. §6 reports results from fine-tuning state-of-the-art trackers on versions of DriveTrack built using both Nearest Neighbor and CompletionFormer [27].

4.5. Robustness

Filters. We find that noise from the hand-labeled bounding box annotations and interpolation errors from depth completion methods can degrade track quality. We implement several filters to ensure that the computed point tracks are accurate:

- Object must lie in the camera's field-of-view (either visible or occluded) for at least 24 frames. The exact duration does not matter; we choose 24 frames because it is the video length used to train TAPIR [5] and TAP-Net [4]. This ensures that videos are not too short for model training and are sufficient length to learn robust point tracks.
- Object must be within 20m of the camera at some point in the video sequence. We find that the quality of depth completion is inferior and highly variable for more distant objects, impacting DriveTrack's ability to determine occlusions accurately.
- Object must lie in the field-of-view continuously for at least 24 frames. The object also cannot fully leave the frame and return for simplicity, although we could allow this in the future.

Additionally, we only use the front, front-left, and front-right cameras. The side cameras often have short object tracks.

Other autonomous driving datasets. We implement the workflow described above for the Waymo Open Dataset [21]. However, DriveTrack's data generation workflow is compatible with the heavily-curated domain of autonomous driving videos [2, 7, 25]. Porting DriveTrack to other datasets might require more careful attention to interpolation, which we leave to future work. For instance, the nuScenes dataset [2] has annotations at 2 Hz, and requires interpolating bounding boxes to match the 10 fps video frame rate. Moreover, nuScenes's dataset has sparser point clouds, which would yield fewer total point tracks and poorer depth maps. We leave additional refinements to future work.

Implementation. App. A discusses optimizations that we added to DriveTrack to improve annotation speed.

5. DriveTrack Analysis

Point track quality. Synthetic benchmarks [9, 29] generate exact point tracks by using rendering tools to simulate motion. However, as we note in §4.5, DriveTrack is susceptible to annotation error. Fig. 5 shows, for a representative scene in the Waymo dataset [21], point correspondences computed by DriveTrack. We denote visible points with • and occluded points with ×. Note that the point tracks are extremely accurate. Our website[3] includes several videos that illustrate the quality of DriveTrack's annotations. App. B includes more annotated examples, including an edge case where DriveTrack fails.

Depth Completion	Speed Error (m/s)				
	p25	p50	p75	p95	p99
Nearest Neighbor	0.0651	0.1260	0.2471	0.5497	0.8687
CompletionFormer [27]	0.0702	0.1273	0.2663	0.4874	0.6615

Table 3. The speed of DriveTrack's point tracks is consistent (*i.e.* low error) with the speed of Waymo's ground-truth bounding boxes [21].

Point track consistency. Since we do not have reference ground truth tracks, we instead compare the velocity of the computed point tracks against the labeled velocities for object annotations, as a proxy for annotation error. Since DriveTrack tracks rigid bodies, we expect the velocity for each point track for an object to match the annotated velocity for that object. To quantify consistency, we compute the average velocity of a point track and compare the distribution (*i.e.* 25-99pct) of those point velocities to the average annotated velocity for that object. Table 3 shows the median value of different percentiles across all objects in the dataset. We report results for both depth completion methods (§4.4). The median error in speed is 0.13 m/s, indicating that DriveTrack is faithful to Waymo's ground-truth annotations.

Statistics. The DriveTrack benchmark on the Waymo dataset [21] includes 1000 scenes across 3 different cameras, totaling about 10,000 1280×1920 videos at 10 fps. Each video is for a single object with 100,000 trajectories on average per video. We further split our dataset into 800 scenes for training, 100 scenes for validation, and 100 scenes for test, which translates to about 8,000, 1,000, and 1,000 videos respectively, with an average video length of 84 frames. Table 1 compares DriveTrack with other standard datasets for point tracking.

6. Fine-tuning keypoint trackers

DriveTrack is large enough that, for the first time, we can train keypoint trackers on real-world scenes. In this section, we show results from fine-tuning three state-of-the-art keypoint trackers [4, 5, 29] on DriveTrack.

6.1. Setup and metrics

Dataset. We work with a subset of annotations from DriveTrack, comprising of 300 target objects split across 100 different scenes. We split each annotation into 24-frame subsets, resulting in 900 distinct training examples, 22,000 training frames, and 168 million point trajectories. We use the Nearest Neighbor depth completion model for most experiments.

Experimental setup. We use the official code for PIPs++ [29],

[3] drivetrack.csail.mit.edu

Figure 5. Visible point correspondences (●) and occlusions (×) computed by DriveTrack over eight samples from a 30-frame sequence.

Tracker	Training	Kubric [9]			DAVIS [19]			DriveTrack		
		AJ	$< \delta^x_{avg}$	OA	AJ	$< \delta^x_{avg}$	OA	AJ	$< \delta^x_{avg}$	OA
TAP-Net [4]	Kubric [9]	**65.4**	**77.7**	**93.0**	38.4	53.1	**82.3**	63.6	73.8	92.4
	+ DriveTrack	37.0	54.0	83.5	**39.2**	**54.7**	78.6	**70.3**	**80.4**	**93.2**
TAPIR [5]	Panning Kubric	**84.7**	**92.1**	**95.8**	62.8	74.7	**89.5**	78.8	87.1	94.4
	+ DriveTrack	80.8	89.6	93.8	**64.0**	**76.1**	88.0	**84.1**	**90.9**	**95.1**
PIPs++ [29]	PointOdyssey [29]	–	**25.8**	–	–	70.4	–	–	81.5	–
	+ DriveTrack	–	25.6	–	–	**71.2**	–	–	**85.3**	–

Table 4. Tracking performance of different models on different datasets before/after fine-tuning on DriveTrack. δ^x_{avg} measures positional tracking accuracy, OA measures binary occlusion accuracy, and AJ considers both position and occlusions. We report all metrics as percentages. Higher is better.

Depth Completion	Average Jaccard (AJ)	
	DAVIS [19]	DriveTrack
Nearest Neighbor	**39.29**	**70.3**
CompletionFormer [27]	38.60	69.5

Table 5. Comparing the Nearest Neighbor and CompletionFormer depth completion strategies for DriveTrack on fine-tuning TAP-Net.

TAP-Net [4], and TAPIR [5], and the pre-trained weights released with each paper. For TAPIR, we use the panning Kubric model checkpoint as described in their paper, as opposed to the one trained on the original MOVi-E dataset. We train all models for 5000 steps on 8 V100 GPUs with 500 warmup steps. For TAP-Net and TAPIR, we use a learning rate of 1×10^{-5} and an AdamW optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.95$ and weight decay 1×10^{-2}. For PIPs++, we use a learning rate of 1×10^{-5} and a AdamW optimizer with the default β parameters and a weight decay of 1×10^{-6}. For all models, we halt fine-tuning when performance on a given task has reached a peak.

Metrics. We evaluate these models on three standard TAP-Vid metrics [4]. $< \delta^x$ evaluates the positional accuracy for visible points, by measuring the fraction of points that are within a threshold of their ground-truth locations. We report $< \delta^x_{avg}$, which averages across 5 thresholds: 1, 2, 4, 8, and 16 pixels. Occlusion Accuracy (OA) reports the classification accuracy for the occlusion status predicted for each point. Average Jaccard (AJ) measures both position and occlusion accuracy. Jaccard is the fraction of "true positives" (*i.e.* points within the threshold of visible ground-truth points) divided by "true positives" plus "false positives" (*i.e.* points predicted visible, when the ground-truth reports occluded or farther than threshold) plus "false negatives" (*i.e.* ground-truth visible points that are predicted as occluded or farther than the threshold). AJ averages Jaccard across the same thresholds as $< \delta^x_{avg}$. Note that we do not report AJ and OA for PIPs++ since it does not export occlusions [29].

6.2. Quantitative results

Table 4 summarizes our results from fine-tuning. For each tracker, we report results on its reference models trained on a synthetic dataset and on our model fine-tuned with DriveTrack. We evaluate each model on Kubric, DAVIS, and DriveTrack.

Improvements on DriveTrack. Fine-tuning improves AJ for TAP-Net by 7%, for TAPIR by 5%, and for PIPs++ by 4%. The AJ of 84.1% for a TAPIR model fine-tuned on DriveTrack is on par (in terms of accuracy) with the equivalent for a synthetic benchmark, *e.g.* TAPIR fine-trained on Kubric yields an AJ of 84.7%. The largest improvement is in positional accuracy. As we noted in §3, synthetic benchmarks do not model high frequency image imperfections, and fine-tuning on DriveTrack's annotations exposes the models to these artifacts. The improvement in OA is smaller, but fine-tuning helps nevertheless.

Transferability. Fine-tuning on DriveTrack also transfers to DAVIS, which consists of real-world scenes drawn from a different distribution than the Waymo dataset [21] on which DriveTrack is built. In particular, for TAP-Net, TAPIR, and

Figure 6. Positional tracking error (in pixels) and occlusion estimation error (binary) for TAPIR [5] before and after fine-tuning on DriveTrack. Brighter colors indicate lower tracking error, and green dots on the occlusion map signify points that TAPIR predicted correctly. Fine-tuning improves tracking accuracy especially around parts of the cars that blend with the background. Occlusion estimation also improves, albeit to a smaller degree.

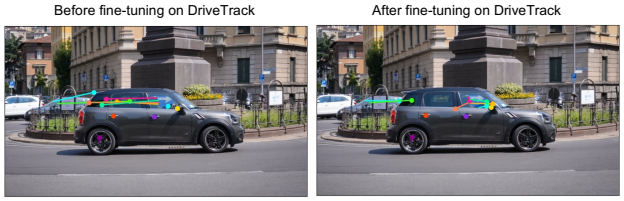

Figure 7. TAPIR [5] fine-tuned on DriveTrack transfers well to scenes in DAVIS [19]. Tracking improves, especially on the window.

PIPs++, AJ improves by 1-2%. This shows that, even though DriveTrack is confined to datasets like autonomous driving videos due to its constraints on bounding box annotations and LiDAR point clouds, trackers are capable of generalizing to real-world scenes more broadly. As expected, fine-tuning TAPIR and TAP-Net on DriveTrack degrade performance on the original training dataset, Kubric, which is synthetic.

Depth completion. Table 5 compares AJ after fine-tuning TAP-Net using the Nearest Neighbor and CompletionFormer [27] depth completion methods that we explored in §4.4. While both models yield similar performance, we find that Nearest Neighbor outperforms CompletionFormer [27] slightly on both DAVIS and DriveTrack. Although it can produce artifacts on the depth map, the max pooling that we apply on top of the Nearest Neighbor interpolation smooths out these artifacts. CompletionFormer ends up with more prediction errors that the max pooling feature cannot correct.

6.3. Qualitative results

DriveTrack. Fig. 6 compares TAPIR's predictions before and after fine-tuning. We show points sampled on a query frame, as well as the positional error (in pixels) and occlusion estimation error (as a binary map) for each model. We show three representative scenes from DriveTrack. In all examples, the positional error generally decreases (*i.e.* brighter colors) after fine-tuning, with the greatest improvement on points in the centers of the vehicles. Notice, for example, with the middle scene, that fine-tuning enables TAPIR to track accurately despite lighting variations caused by rain. App. C includes more annotated examples that illustrate improvements due to fine-tuning.

DAVIS. Fig. 7 shows TAPIR before and after fine-tuning, on a scene from DAVIS [19]. Tracking on the window still exhibits significant error, but notice that fine-tuning yields a tangible improvement in positional accuracy. See App. C for more results.

7. Sensitivity of trackers to keypoints

Despite the improvements from fine-tuning that we demonstrate in §6, there is still an appreciable gap between the performance of these trackers on real-world scenes and the performance on synthetic datasets. §3 illustrates how trackers struggle particularly in scenes with complex lighting conditions and textures. While fine-tuning on DriveTrack can help make models more robust to these imperfections, we believe that precise tracking is fundamentally difficult in these settings. In this section, we use the scale of annotations in DriveTrack to quantify the sensitivity of keypoint trackers to particular keypoints.

7.1. Quantifying sensitivity with DriveTrack

How is tracking error distributed over the DriveTrack benchmark? For each scene in the Waymo dataset [21], we randomly sample 50 keypoints from a query frame and use TAPIR [5] to track them over the entire video. We compute average positional tracking error for each keypoint against DriveTrack's ground-truth annotations, and consider the distribution of errors across all keypoints across all scenes. Fig. 8 shows these

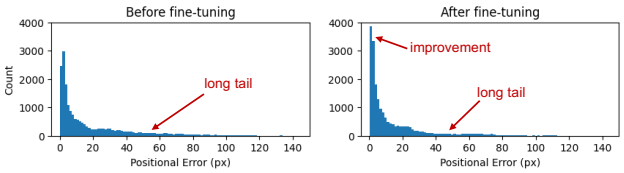

Figure 8. The distribution of tracking error across DriveTrack benchmark is heavy-tailed, even after fine-tuning. Certain kinds of keypoints are more susceptible to greater error.

Figure 9. Existing keypoint selectors either offer a sparse set of options (Harris) or fail to avoid regions with visual imperfections (SuperPoint).

histograms of TAPIR's error before and after fine-tuning.

First, notice that fine-tuning reduces positional error noticeably on average: the fraction of points with tracking error less than 10 pixels is higher after fine-tuning. However, both distributions have long tails: the worst-case performance of DriveTrack improves marginally after fine-tuning. The 90th percentile error in both cases is over 60 pixels. This indicates that no amount of additional data will alleviate tail performance.

One might wonder why the tail performance of keypoint trackers matter: if we track tens of thousands of keypoints won't the effects at the tail be negligible? Invoking trackers like TAPIR and PIPs++ is computationally-expensive, and is not practical for tracking thousands of pixels. Moreover, these trackers cater to the TAP formulation, where users want accurate tracks for very specific points. Those desired points might correspond to ones that yield error at the tail, and users currently have no way of knowing that.

7.2. Characterizing keypoint sensitivity

The annotations in Fig. 6 and Fig. 7 indicate that points with high tracking error continue to correlate with visual imperfections, despite improvements from fine-tuning. For instance, notice from Fig. 6 that tracking performance is poorest on edges of the vehicle that are reflective (*e.g.* top row, Prius windshield) and when the shadows partially cover the vehicle (*e.g.* bottom row, minivan). In both examples, the lighting conditions vary between the query frame and the evaluation frame, and tracking error is exacerbated on reflective surfaces. Similarly, on the DAVIS dataset (Fig. 7), we find that TAPIR continues to struggle on the car's windows. By contrast, TAPIR tracks the door handle and wheel spokes perfectly.

7.3. Towards track-aware keypoint selectors

Given that positional tracking error correlates with visual imperfections and given the heavy-tailed error distributions, we argue that *keypoint selection* is as vital as keypoint tracking. We believe that users of any TAP-type tracker should also use a keypoint selector to strategically choose query points that will yield superior tracking accuracy with a high probability.

Keypoint selection from images has been an area of research in computer vision. The Harris Corner detector [11] was one of the earliest methods to select trackable keypoints by looking at changes in image intensity to detect corners. SIFT [15] is a featurization scheme that is popular for object recognition tasks. SuperPoint [3] is a more recent method that predicts interest points by self-supervising on sets of images and various homographic transformations. While these techniques capture some elements of tracking tasks, none are well-suited for long-range tracking through occlusions and in cluttered scenes, which has become a popular domain for keypoint trackers [4, 5, 10, 29] over the last few years.

Fig. 9 compares the positional accuracy achieved by TAPIR for keypoints selected by Harris and by SuperPoint for an example in DriveTrack. Harris anchors to corners and selects points that TAPIR tracks accurately; however, it finds only 4 points to track in this scene. SuperPoint, by contrast, anchors to points on the window, which TAPIR tracks poorly. Neither method is suitable for finding a large and robust set of trackable keypoints.

Effective keypoint selection will make trackers robust to visual imperfections. We need a keypoint selector that is aware that the downstream task is tracking. DriveTrack offers a rich dataset to train a feature representation to select robust keypoints in real-world scenes.

8. Conclusion

We developed DriveTrack, the first benchmark to automatically annotate long-range point tracks in real-world videos. We release the largest point tracking dataset on real-world scenes to date, consisting of 1 billion point tracks and 84 billion annotated points in total. DriveTrack's annotation workflow works with autonomous driving datasets consisting of point clouds, bounding box annotations, and camera poses. We develop a new way to robustly track a sequence of sparse point clouds that do not have 1:1 correspondence, and implement several refinements to be robust to labeling noise. Finally, we show that fine-tuning keypoint trackers with DriveTrack improves accuracy on real-world scenes, and we conduct a sensitivity analysis to motivate using keypoint selectors alongside trackers.

We believe that DriveTrack and the data generation workflow that we developed has several use cases outside of keypoint tracking, such as optical flow and structure-from-motion. Moreover, the ideas behind DriveTrack could be applied to other domains in the real-world with semi-annotated videos. For instance, we could consider building a similar workflow for annotate scenes imaged by iPhones and comparable smartphones, which can sense depth through LiDAR and can record camera pose and orientation.

References

[1] Simon Baker, Stefan Roth, Daniel Scharstein, Michael J. Black, J.P. Lewis, and Richard Szeliski. A database and evaluation methodology for optical flow. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, 2007. 2

[2] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. 1, 3, 5

[3] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. *CoRR*, abs/1712.07629, 2017. 8

[4] Carl Doersch, Ankush Gupta, Larisa Markeeva, Adria Recasens Continente, Kucas Smaira, Yusuf Aytar, Joao Carreira, Andrew Zisserman, and Yi Yang. Tap-vid: A benchmark for tracking any point in a video. In *NeurIPS Datasets Track*, 2022. 1, 2, 5, 6, 8

[5] Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew Zisserman. Tapir: Tracking any point with per-frame initialization and temporal refinement. *ICCV*, 2023. 1, 2, 3, 5, 6, 7, 8

[6] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *CoRR*, abs/1504.06852, 2015. 2, 3

[7] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 1, 2, 3, 5

[8] Rohit Girdhar, Georgia Gkioxari, Lorenzo Torresani, Manohar Paluri, and Du Tran. Detect-and-track: Efficient pose estimation in videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 350–359, 2018. 1

[9] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam Laradji, Hsueh-Ti (Derek) Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: a scalable dataset generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2, 3, 4, 5, 6

[10] Adam W Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. Particle video revisited: Tracking through occlusions using point trajectories. In *ECCV*, 2022. 1, 2, 8

[11] Christopher G. Harris and Mike Stephens. A combined corner and edge detector. In *Proceedings of the Alvey Vision Conference, AVC 1988, Manchester, UK, September, 1988*, pages 1–6. Alvey Vision Club, 1988. 8

[12] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset, 2017. 1, 2

[13] Alex X. Lee, Coline Devin, Yuxiang Zhou, Thomas Lampe, Konstantinos Bousmalis, Jost Tobias Springenberg, Arunkumar Byravan, Abbas Abdolmaleki, Nimrod Gileadi, David Khosid, Claudio Fantacci, Jose Enrique Chen, Akhil Raju, Rae Jeong, Michael Neunert, Antoine Laurens, Stefano Saliceti, Federico Casarini, Martin Riedmiller, Raia Hadsell, and Francesco Nori. Beyond pick-and-place: Tackling robotic stacking of diverse shapes. In *Conference on Robot Learning (CoRL)*, 2021. 1

[14] Peiliang Li, Xiaozhi Chen, and Shaojie Shen. Stereo r-cnn based 3d object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7644–7652, 2019. 1

[15] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, pages 1150–1157. Ieee, 1999. 8

[16] Dominic Maggio, Marcus Abate, Jingnan Shi, Courtney Mario, and Luca Carlone. Loc-nerf: Monte carlo localization using neural radiance fields. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4018–4025. IEEE, 2023. 1

[17] Nikolaus Mayer, Eddy Ilg, Philip Häusser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. *CoRR*, abs/1512.02134, 2015. 2, 3

[18] Jinsun Park, Kyungdon Joo, Zhe Hu, Chi-Kuei Liu, and In So Kweon. Non-local spatial propagation network for depth completion. In *Proc. of European Conference on Computer Vision (ECCV)*, 2020. 4

[19] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017. 1, 2, 3, 6, 7

[20] Deqing Sun, Daniel Vlasic, Charles Herrmann, Varun Jampani, Michael Krainin, Huiwen Chang, Ramin Zabih, William T. Freeman, and Ce Liu. Autoflow: Learning a better training set for optical flow. *CoRR*, abs/2104.14544, 2021. 2

[21] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2, 3, 4, 5, 6, 7, 11

[22] Zachary Teed and Jia Deng. RAFT: recurrent all-pairs field transforms for optical flow. *CoRR*, abs/2003.12039, 2020. 2

[23] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. In *International Conference on 3D Vision (3DV)*, 2017. 4

[24] Mel Vecerik, Carl Doersch, Yi Yang, Todor Davchev, Yusuf Aytar, Guangyao Zhou, Raia Hadsell, Lourdes Agapito, and Jon Scholz. Robotap: Tracking arbitrary points for few-shot visual imitation. *arXiv preprint arXiv:2308.15975*, 2023. 1

[25] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. BDD100K: A diverse driving video database with scalable annotation tooling. *CoRR*, abs/1805.04687, 2018. 1, 5

[26] Hanchao Yu, Xiao Chen, Humphrey Shi, Terrence Chen, Thomas S Huang, and Shanhui Sun. Motion pyramid networks for accurate and efficient cardiac motion estimation. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part VI 23*, pages 436–446. Springer, 2020. 1

[27] Youmin Zhang, Xianda Guo, Matteo Poggi, Zheng Zhu, Guan Huang, and Stefano Mattoccia. Completionformer: Depth completion with convolutions and vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18527–18536, 2023. 4, 5, 6, 7, 11

[28] Wang Zhao, Shaohui Liu, Hengkai Guo, Wenping Wang, and Yong-Jin Liu. Particlesfm: Exploiting dense point trajectories for localizing moving cameras in the wild, 2022. 2

[29] Yang Zheng, Adam W. Harley, Bokui Shen, Gordon Wetzstein, and Leonidas J. Guibas. Pointodyssey: A large-scale synthetic dataset for long-term point tracking. In *ICCV*, 2023. 1, 2, 3, 4, 5, 6, 8