Supplementary material for BootsTAP: Bootstrapped Training for Tracking-Any-Point

Carl Doersch[1], Pauline Luc[1], Yi Yang[1], Dilara Gokay[1], Skanda Koppula[1], Ankush Gupta[1], Joseph Heyward[1], Ignacio Rocco[1], Ross Goroshin[1], João Carreira[1], and Andrew Zisserman[1,2]

[1] Google DeepMind
[2] VGG, Department of Engineering Science, University of Oxford

1 Contents

The supplementary material consists of this document as well as an html file, which has been published at https://bootstap.github.io/. The document contains:

- Overall summary of the approach in Sec. 2.
- Qualitative visualizations of results on NIST Gears in Sec. 3.
- Higher resolution and some further data improvements, including bugfixes for the Kubric data loader and longer clips in Sec. 4, resulting in a model which outperforms the version in our paper and will be used for the public release. This version is too expensive to ablate properly and contains some minor engineering fixes, so in order to ensure all of the main-paper results were comparable, we did not use it as the basis for our experiments there, even though it achieves slightly better performance.
- Results with bootstrapping the causal version of TAPIR in Sec. 5.
- Results on the Perception Test point tracking challenge in Sec. 6.
- Implementation details, including details of transformations in Sec. 7.1, training details in Sec. 7.2, and details on experiments which fine-tune on Libero in Sec. 7.3.
- Comparison of a 3D ConvNet backbone versus a 2D ConvNet backbone in Sec. 8.
- More detailed description of the evaluation datasets in Sec. 9.

Our webpage includes video visualizations, including:

- Video visualizations of densely tracked points on NIST gears, comparing TAPIR and BootsTAPIR.
- Video visualizations of densely tracked points on Libero, comparing TAPIR, BootsTAPIR, and BootsTAPIR finetuned on Libero.
- Video visualizations of tracked points on TAP-Vid DAVIS and RoboTAP, comparing TAPIR, BootsTAPIR, and ground truth.

To ensure reproducibility, we release an open-source implementation of BootsTAPIR along with a pretrained model at https://github.com/google-deepmind/tapnet.

2 Summary of the approach

We summarize notation and computation of our self-supervised loss in Algorithm 1.

Algorithm 1 BootsTAP self-supervised loss. Notation:
$\mathcal{U}(D)$ refers to the uniform distribution over domain D;
we denote queries as $Q = (q, t)$ where q is x/y coordinates and t is a frame index.
In a slight abuse of notation, we call Φ_t the transformation and the mapping that transforms coordinates and leaves other model outputs unchanged.

Require:
X – video of shape $T \times H \times W \times C$
f – model
Θ, ξ – student parameters, teacher parameters
$\mathcal{A}, \mathcal{D}_\Phi$ – distribution over augmentations, distribution over transformations
$\mathcal{V} \mapsto \mathcal{D}_\mathcal{V}$ – mapping that maps a set of points $\mathcal{V}$ to a distribution $\mathcal{D}_\mathcal{V}$ over $\mathcal{V}$
δ, δ_{cycle} – threshold values for uncertainty target definition and cycle-consistency filtering criterion
$d(\cdot, \cdot)$ – distance function

Uniformly sample teacher query points $Q_1 \sim \mathcal{U}([0, H) \times [0, W) \times [\![0, T-1]\!])$.
Sample augmentation $a \sim \mathcal{A}$ and a frame-wise affine transformation $\Phi = \{\Phi_t\}_t \sim \mathcal{D}_\Phi$.
Augment and transform each frame to form X': $\forall t, X'_t \leftarrow \text{resampling}(a(X_t), \Phi_t)$.
For each query point Q_1:
Predict tracks and occlusions with teacher model: $\{\hat{p}_\mathcal{T}[t], \hat{o}_\mathcal{T}[t]\}_t \leftarrow f(X, Q_1; \xi)$.
Derive pseudo-labels from teacher predictions with:

$$p_\mathcal{T}[t] = \hat{p}_\mathcal{T}[t] \quad ; \quad o_\mathcal{T}[t] = \mathbb{1}(\hat{o}_\mathcal{T}[t] > 0); \quad u_\mathcal{T}[t] = \mathbb{1}(d(p_\mathcal{T}[t], \hat{p}_\mathcal{S}[t]) > \delta)$$

Calling $\mathcal{V}$ the set of visible points along the teacher trajectory,
sample $Q_2 = (q_2, t_2) \sim \mathcal{D}_\mathcal{V}$.
Transform query points: $Q'_2 \leftarrow (\Phi_{t_2}(q_2), t_2)$.
Predict tracks with the student model and transform predicted coordinates with the inverse of Φ_t: $\{\hat{p}_\mathcal{S}[t], \hat{o}_\mathcal{S}[t], \hat{u}_\mathcal{S}[t]\}_t \leftarrow \Phi_t^{-1}(f(X', Q'_2; \Theta))$.
Compute masks used to filter out loss terms (when t_1 and t_2 differ):

$$m_{cycle} = \mathbb{1}\left(d(\hat{p}_\mathcal{S}[t_1], q_1) < \delta_{cycle}\right) \quad * \quad \mathbb{1}\left(\hat{o}_\mathcal{S}[t_1] \leq 0\right)$$

Compute the loss:

$$\mathcal{L}_{SSL} = m_{cycle} * \frac{1}{T} \sum_t \ell^t_{ssl}$$

where ℓ^t_{ssl} is the self-supervised TAPIR loss term for t.

3 Results on NIST Gears

Figure 1 further illustrates improvements on the RoboCAT-NIST. Due to the lack of ground truth, sample a grid of points on the red pixels for RoboCAT-

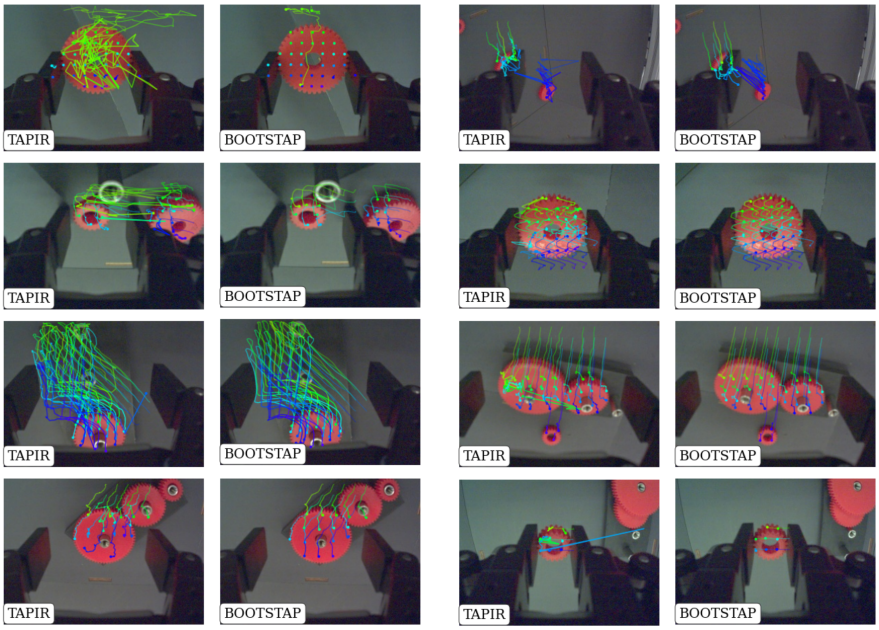

Fig. 1: Comparison: TAPIR vs. BootsTAPIR on the real RoboCAT-NIST dataset. Since we lack ground truth, we show the TAPIR and the BootsTAPIR predictions in Rainbow tail style side-by-side. On NIST, BootsTAPIR works more consistently on location prediction. Note how points that were originally predicted as occluded are now visible.

NIST. We display a few examples comparing the predicted tracks between the two models. As these are rigid objects, we expect the points to move consistently within each gear; deviations from this are errors. Due to the lack of texture on the gears and the nontrivial domain gap, the original TAPIR trained on Kubric works poorly here, with many jittery tracks and severe tracking failures. This is particularly bad for points that are close to occlusion or out of image boundary. The bootstrapped model fixes many of these failures: the tracks are much smoother and occlusion predictions become much more accurate. Results are comparable on Libero, although the motions there are more complicated and unsuitable for a static figure; see webpage for video visualizations.

4 Higher Resolution and Data Improvements

The original TAPIR data loader, which we used as the basis for our experiments, recently fixed a minor bug that leads to slight performance improvements. Specifically, recall that the data augmentations used for the Kubric dataset include a random axis-aligned crop. The image cropping mechanism was not pixel-aligned with the transforms used for points, leading to an almost imperceptible error in

Table 1: Comparison of performance on the TAP-Vid datasets for the released version of BootsTAPIR. Fix refers to the bugfix to coordinates. Snap refers to the snap-to-occluder bias in the training data. Data refers to extra training data which has longer clips and higher resolution.

Method	TAP-Vid-Kinetics			TAP-Vid-DAVIS			TAP-Vid-RGB-Stacking		
	AJ ↑	$< \delta^x_{avg}$ ↑	OA ↑	AJ ↑	$< \delta^x_{avg}$ ↑	OA ↑	AJ ↑	$< \delta^x_{avg}$ ↑	OA ↑
CoTracker [5]	57.3	70.6	87.5	64.8	79.1	88.7	65.9	80.6	85.0
Tuned TAPIR+fix	60.4	72.5	88.6	64.7	76.6	89.6	70.0	82.1	89.9
Tuned TAPIR+fix+snap	59.6	71.9	88.8	63.8	75.9	89.9	67.1	80.3	87.8
BootsTAPIR	61.4	74.2	89.7	66.2	78.1	91.0	72.4	83.1	91.2
BootsTAPIR+fix	61.4	74.5	89.4	66.5	78.5	90.8	75.9	85.7	93.8
BootsTAPIR+fix+snap	61.3	74.7	89.1	67.1	78.9	91.2	75.7	85.7	93.4
BootsTAPIR+fix+snap+data	62.5	74.8	89.5	67.4	79.0	91.3	77.4	86.7	93.2

the track locations. Fixing this bug leads to an improvement in performance for the original TAPIR model, but surprisingly has relatively little effect on BootsTAP performance. However, we find that the reason isn't because BootsTAP compensates for the bug, but rather, because the bug creates a bias toward tracking foreground objects (the tracks tend to be slightly expanded relative to the underlying objects). We find we can replicate this bias by altering query points that are very near occlusion edges (1 pixel away) to track the foreground object rather than the background, which we call the "snap to occluder" technique. See Section 7.4 for details.

To further tune performance, we also trained on higher-resolution clips, and also longer clips, as we find these improve generalization for real-world applications with longer or higher-resolution videos. To implement this, we add more 'tasks' with different data shapes, using the same multi-task framework (i.e., separate optimizers) as described above. Specifically, one extra task uses 512×512 Kubric clips (24 frames), trained using the same losses. We use the hierarchical refinement approach described in the original TAPIR paper, wherein the initialization and one refinement pass is performed at 256×256, and then a further refinement pass is performed at 512×512. We also use an analogous high-resolution self-supervised task, which also uses 24-frame, 512×512 videos from the same real-world dataset. Finally, we add 150-frame, 256×256 videos, this time at 30 frames per second.

Tables 1 and 2 show our results. Note that CoTracker implemented its own data augmentation algorithms and is not affected by the same bug. We see that "snap to occluder" harms TAPIR performance, but improves BootsTAP performance. One possible interpretation is that the snapping is compensating for a particular bias in the bootstrapping toward tracking background. This may be because background is easier to track, especially relative to thin objects. In a bootstrapping framework, the model's reliable predictions that follow background become self-reinforcing, whereas unreliable predictions for thin foreground objects, are not. Therefore, they tend to get lost over time. Finding more principled solutions to this issue is an interesting area for future work.

Table 2: Comparison of performance under query-first metrics for Kinetics, TAP-Vid DAVIS, and RoboTAP (standard for this dataset).

Method	TAP-Vid-Kinetics			TAP-Vid-DAVIS			RoboTAP		
	AJ ↑	$< \delta^x_{avg}$ ↑	OA ↑	AJ ↑	$< \delta^x_{avg}$ ↑	OA ↑	AJ ↑	$< \delta^x_{avg}$ ↑	OA ↑
CoTracker [5]	48.7	64.3	86.5	60.6	75.4	89.3	54.0	65.5	78.8
Tuned TAPIR+fix	53.3	66.0	85.1	58.9	71.6	86.4	67.3	78.4	90.0
Tuned TAPIR+fix+snap	52.5	65.3	85.5	58.3	71.1	87.7	66.4	77.3	90.5
BootsTAPIR	54.6	68.4	86.5	61.4	73.6	88.7	64.9	80.1	86.3
BootsTAPIR+fix	54.7	68.5	86.3	61.6	74.1	89.0	65.7	80.5	87.2
BootsTAPIR+fix+snap	54.5	68.8	86.3	61.8	74.3	89.1	63.5	81.1	84.2
BootsTAPIR+fix+snap+data	55.8	68.8	86.6	62.4	74.6	89.6	69.2	81.3	89.5

Table 3: Comparison of performance for high-resolution setting.

Method	TAP-Vid-Kinetics			TAP-Vid-DAVIS		
	AJ ↑	$< \delta^x_{avg}$ ↑	OA ↑	AJ ↑	$< \delta^x_{avg}$ ↑	OA ↑
BootsTAPIR	62.5	74.8	89.5	67.4	79.0	91.3
BootsTAPIR+hires	63.7	76.0	88.4	70.2	81.2	91.2

Regardless, the extra training data leads to a non-trivial boost in performance. Note that, for tables 1 and 2, all evaluation videos are still at 256×256, and unlike many prior methods we do not upsample them before creating the feature representation. To assess the impact of increased *evaluation* resolution, we also performed evaluation on 512×512 videos. Table 3 shows results. We see that performance improves by 1.2% on Kinetics and 2.8% on DAVIS, the best reported performance on this dataset by a wide margin. Surprisingly, we found that further resolution at test-time did not improve results, suggesting another interesting area for future work.

5 Causal model

RoboTAP [10] pointed out that point tracking can be very useful in an online setting, e.g. when used as a signal to control agents in real time. It remains straightforward to extend BootsTAPIR to the online setting: the only temporal dependency of the model is in the 1D convolutions in the iterative refinements, so these can be directly converted into causal convolutions to create a causal model. We trained this model using the full training setup for the release model, including the extra high-resolution, long-clip data. Table 4 shows results. We see an overall 4.6% improvement on Kinetics and a 3.0% improvement on DAVIS, in both cases using the query-first evaluation procedure.

6 Perception test

We also perform experiments on the point tracks in the Perception Test [8] validation set, a challenging dataset of point tracks annotated on videos of unusual

Table 4: Causal model performance.

Method	TAP-Vid-Kinetics *q_first* AJ ↑	$< \delta^x_{avg}$ ↑	OA ↑	TAP-Vid-DAVIS *q_first* AJ ↑	$< \delta^x_{avg}$ ↑	OA ↑
Causal TAPIR	51.5	64.4	85.2	56.7	70.2	85.7
Causal BootsTAPIR	55.1	67.5	86.3	59.7	72.3	86.9

Table 5: Performance on Perception Test relative to TAPIR.

Method	Overall	Static	Dynamic
TAPIR	55.7	57.4	46.3
BootsTAPIR	59.6	61.3	49.7

situations filmed by participants. Results are shown in Table 5; we see a similar magnitude gap over prior results.

As a final note, we performed informal benchmarking of our model using an A100 and the latest JAX compiler. We found that after compilation, BootsTAPIR can perform inference of $10,000$ points on a 256×256, 50-frame video in 5.6 seconds. Furthermore, the causal model can track 400 points on a 256×256 video at 30.1 frames per second.

7 Implementation details

7.1 Distribution over transformations

We design transformations of the inputs to enforce equivariance of the predictions with *realistic* spatial transformations. At a high level, we intend for the transformations to mimic the effects of additional, simple and plausible camera motion and zooming on the video. Hence, our transformations should vary smoothly in time; they should cover a reasonable ratio of the original video content; and aspect ratio should be roughly preserved.

We define a family of frame-wise affine transformations that has these properties, and a procedure to sample these randomly. Essentially, we sample top-left crop coordinates and crop dimensions for each frame in the video; where coordinates and dimensions are computed as interpolations between values sampled for the start and end frames from a distribution that achieves the desired coverage and aspect ratios.

More formally, we first sample a pair of spatial dimensions (H_0, W_0) for the start frame as follows. We sample an area A uniformly over $[0.6, 1.0]$. Next we sample values $a^1, a^2 \sim \mathcal{U}([A, 1])$ and derive random height value by averaging them $h = \frac{a^1 + a^2}{2}$ and width value $w = \frac{A}{h}$; and finally, we multiply these values by the input's original shape (H, W). This gives us a pair of spatial dimensions biased towards aspect ratios close to 1, and covering an area between 60% and

100% of the original input. We proceed the same way to sample a pair of spatial dimensions(H_{T-1}, W_{T-1}) for the end frame.

Next, we uniformly sample a pair of top-left corner coordinates (C_0^x, C_0^y) for the start frame, such that a crop of dimensions (H_0, W_0) can be extracted within the frame. We proceed the same way to sample a pair of spatial coordinates (C_{T-1}^x, C_{T-1}^y), given (H_{T-1}, W_{T-1}), for the end frame.

We then interpolate linearly on one hand between the start and end spatial dimensions; and on the other hand between the start and end top-left corner coordinates. Let $t \in \{0, ..., T-1\}$ be a frame index. Calling $\alpha_t = \frac{t}{T-1}$, we define:

$$h_t = (1 - \alpha_t) * H_0 + \alpha_t * H_{T-1} \tag{1}$$

$$w_t = (1 - \alpha_t) * W_0 + \alpha_t * W_{T-1} \tag{2}$$

$$c_t^x = (1 - \alpha_t) * C_0^x + \alpha_t * C_{T-1}^x \tag{3}$$

$$c_t^y = (1 - \alpha_t) * C_0^y + \alpha_t * C_{T-1}^y. \tag{4}$$

This gives us parameters of scaling parameters (h_t, w_t) and translation parameters (c_t^x, c_t^y) vary linearly over time. Finally, our frame-wise affine transformations $\Phi = \{\Phi_t\}_t$ are defined as follows:

$$\forall t, \Phi_t : (x, y) \mapsto \Big(\frac{w_t}{W} * x + c_t^x, \frac{h_t}{H} * y + c_t^y\Big), \tag{5}$$

We refer to the distribution resulting from our sampling procedure as $\mathcal{D}_\Phi$. Given a query point coordinates $Q = (q, t)$ and input frames $\{X_t\}_t$, the corresponding transformation is applied with:

$$Q' = (\Phi_t(q), t); \quad \forall t, X'_t = \text{resample}(X_t, \Phi_t), \tag{6}$$

where resample$(\cdot, \Phi_t)$ consists in scaling its input frame to resolution (h_t, w_t) using bilinear interpolation and placing it within a zero-valued array of shape (H, W) such that its top-left corner in the array is at coordinates (c_t^x, c_t^y). We note that in our approach, this transformation is performed after augmenting each frame, *i.e.* on $a(X_t)$.

7.2 Training details

We train for 200,000 iterations on 256 nVidia A100 GPUs, with a batch size of 4 Kubric videos and 2 real videos per device. The extra layers consist of 5 residual blocks on top of the backbone (which has stride 8, 256 channels), each of which consists of 2 sequential 3×3 convolutions with a channel expansion factor of 4, which is then added to the input. We use a cosine learning rate schedule with 1000 warmup steps and a peak learning rate of 2e-4. We found it improved stability to reduce the learning rate for the PIPs mixer steps relative to the backbone by a factor of 5. We keep all other hyperparameters the same as TAPIR.

7.3 Libero finetuning

We compare results on Libero, using the gripper view, which contains large and difficult motions. Qualitative results show that BootsTAP trained on internet videos as described improves results substantially. However, since there's a large domain gap between Libero data and internet videos, it's natural to ask whether performance can be improved by further self-supervised training on the Libero dataset.

We use the full set of demonstrated trajectories in the dataset for all tasks, again using only the gripper view. We begin with the model trained as described in the main paper, and then further train it for another 50K steps using three tasks jointly: Kubric, internet videos, and Libero videos, again using separate optimizers for each and summing updates across tasks. We use an update weight of 0.2 for both self-supervised tasks, and keep all other parameters the same between Libero and the internet video tasks. We see that this approach further improves results despite having no labels: the model can track with surprisingly high fidelity over large changes in scale and viewpoint. See the webpage for visualizations.

7.4 Snap-to-occluder

We aim to slightly modify the training objective to bias TAPIR to track foreground objects rather than background, to counteract the tendency of bootstrapped models to track background. The Kubric data loader works by sampling query pixels randomly (biased toward objects), and then computing the full track by back-projecting into the relevant object's local coordinate system. We first modify the procedure by preventing the model from sampling pixels on the 'back side' of an occlusion boundary: this is defined as any pixel with a neighboring pixel (within a 3x3 square) which is less than 95% of the pixel's depth. After tracking points, we identify query points that are on the 'front side' of an occlusion boundary: that is, any neighboring pixel which is more than 105% of the depth of the query point. If such pixels exist, with 50% probability we randomly choose one such pixel and replace the query point with it. Therefore, in a small fraction of cases, the model will receive a query point on the background but need to track the foreground object instead. We did not use "snap to occluder" in any experiments in the main paper, but only in the high-resolution release version, and don't claim it as a contribution of our paper.

8 Comparison with and without a 3D ConvNet Backbone

Recall that TAPIR extracts features using a ResNet, with a final feature map of dimension 256 at stride 8 (although it uses an earlier feature map as well at stride 4). The architecture is similar to a ResNet-18, and therefore has relatively little capacity to learn about the full diversity of objects in the world. Therefore, we add extra capacity: 5 more ResNet layers consisting of a LayerNorm, a 3×3

Table 6: Architectures. We compare a 2D backbone and a 3D backbone using Temporal Shift Modules (TSM) to aggregate information locally over time.

	TAP-Vid-DAVIS *strided*			TAP-Vid-Kinetics *query_first*		
	AJ ↑	$< \delta^x_{avg}$ ↑	OA ↑	AJ ↑	$< \delta^x_{avg}$ ↑	OA ↑
2D ConvNet Backbone	66.2	78.1	91.0	54.6	68.4	86.5
3D ConvNet Backbone	66.3	78.4	90.7	54.0	68.0	85.0

convolution, followed by a GeLU, followed by another 3×3 convolution which is added to the input of the layer. Our full model applies the feature extractor independently on every frame, meaning that the model cannot use temporal cues for feature extraction. Is this choice optimal? Intuitively, we might expect motion to provide segmentation cues that could enable better matching. Therefore, we develop an alternative model which adds a simple 3D ConvNet into the backbone: specifically, we convert the first convolution of each residual block layer into a $3 \times 3 \times 3$, giving the features a temporal receptive field of 21 frames.

We report results Table 6. We observe that this yields a slight performance increase on TAP-Vid-DAVIS (*strided* evaluation), and in particular, slightly improves the position accuracy, although it harms occlusion accuracy. However, it significantly degrades performance on Kinetics (*query_first*). It's possible that the model struggles more with the cuts or camera shake present in Kinetics. Hence, we keep a 2D backbone for the final model, although the optimal model may depend on the desired downstream application.

9 Evaluation Datasets

TAP-Vid-Kinetics contains videos collected from the Kinetics-700-2020 validation set [2] with original focus on video action recognition. This benchmark contains 1K YouTube videos of diverse action categories, approximately 10 seconds long, including many challenging elements such as shot boundaries, multiple moving objects, dynamic camera motion, cluttered background and dark lighting conditions. Each video contains ∼26 tracked points on average, obtained from careful human annotation.

TAP-Vid-DAVIS contains 30 real-world videos from DAVIS 2017 validation set [9], a standard benchmark for video object segmentation, which was extended to TAP. Each video contains ∼22 point tracks using the same human annotation process as TAP-Vid-Kinetics.

TAP-Vid-RGB-Stacking contains 50 synthetic videos generated with Kubric [4] which simulate a robotic stacking environment. Each video contains 30 annotated point tracks and has a duration of 250 frames.

RoboTAP contains 265 real world Robotics Manipulation videos with on average ∼272 frames and ∼44 annotated point tracks per video [10]. These videos are even longer, with textureless and symmetric objects that are far out-of-domain

for both Kubric and the online lifestyle videos that we use for self-supervised learning.

RoboCAT-NIST is a subset of the data collected for RoboCat [1]. Inspired by the NIST benchmark for robotic manipulation [6], it includes gears of varying sizes (small, medium, large) and a 3-peg base, introduced for a systematic study of insertion affordance. All videos are collected by human teleoperation. It includes robot arms operating and inserting gears, which are a particularly challenging case due to the rotational symmetry and lack of texture. In this work, we processed videos to 64 frames long with 222 $\times$ 296 resolution. This dataset is mainly for demonstration purpose, there are no human groundtruth point tracks.

Libero [7] is a dataset where point tracking has already proven useful for robotic manipulation [11]. It includes demos of a human-driven robot arm performing a wide variety of tasks in a synthetic environment, intended for use in imitation learning. Sequences are variable length at 128×128 resolution and has no ground truth tracks.

9.1 Evaluation metrics

We use three evaluation metrics same as proposed in [3]. (1) $< \delta^{\mathbf{x}}_{\mathbf{avg}}$ is the average position accuracy across 5 thresholds for δ: 1, 2, 4, 8, 16 pixels. For a given threshold δ, it computes the proportion of visible points (not occluded) that are closer to the ground truth than the respective threshold. (2) **Occlusion Accuracy (OA)** is the average binary classification accuracy for the point occlusion prediction at each frame. (3) **Average Jaccard (AJ)** combines the two above metrics and is typically considered the target for this benchmark. It is the average Jaccard score across the same thresholds as $< \delta^{x}_{avg}$. Jaccard at δ measures both occlusion and position accuracy. It is the fraction of 'true positives', i.e., points within the threshold of any visible ground truth points, divided by 'true positives' plus 'false positives' (points that are predicted visible, but the ground truth is either occluded or farther than the threshold) plus 'false negatives' (groundtruth visible points that are predicted as occluded, or where the prediction is farther than the threshold).

For TAP-Vid datasets, evaluation is split into *strided mode* and *query-first mode*. Strided mode samples query points every 5 frames on the groundtruth tracks when they are visible. Query points can be any time in the video hence it tests the model prediction power both forward and backward in time. Query-first mode samples query points only when they are first time visible and the evaluation only measures tracking accuracy in future frames.

References

1. Bousmalis, K., Vezzani, G., Rao, D., Devin, C., Lee, A.X., Bauza, M., Davchev, T., Zhou, Y., Gupta, A., Raju, A., et al.: Robocat: A self-improving foundation agent for robotic manipulation. arXiv preprint arXiv:2306.11706 (2023)

2. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: Proc. CVPR. pp. 6299–6308 (2017)
3. Doersch, C., Gupta, A., Markeeva, L., Recasens, A., Smaira, L., Aytar, Y., Carreira, J., Zisserman, A., Yang, Y.: TAP-Vid: A benchmark for tracking any point in a video. NeurIPS (2022)
4. Greff, K., Belletti, F., Beyer, L., Doersch, C., Du, Y., Duckworth, D., Fleet, D.J., Gnanapragasam, D., Golemo, F., Herrmann, C., et al.: Kubric: A scalable dataset generator. In: Proc. CVPR (2022)
5. Karaev, N., Rocco, I., Graham, B., Neverova, N., Vedaldi, A., Rupprecht, C.: CoTracker: It is better to track together. arXiv preprint arXiv:2307.07635 (2023)
6. Kimble, K., Van Wyk, K., Falco, J., Messina, E., Sun, Y., Shibata, M., Uemura, W., Yokokohji, Y.: Benchmarking protocols for evaluating small parts robotic assembly systems. Proc. Intl. Conf. on Robotics and Automation **5**(2), 883–889 (2020)
7. Liu, B., Zhu, Y., Gao, C., Feng, Y., Liu, Q., Zhu, Y., Stone, P.: Libero: Benchmarking knowledge transfer for lifelong robot learning. NeurIPS **36** (2024)
8. Patraucean, V., Smaira, L., Gupta, A., Recasens, A., Markeeva, L., Banarse, D., Koppula, S., Malinowski, M., Yang, Y., Doersch, C., Matejovicova, T., Sulsky, Y., Miech, A., Frechette, A., Klimczak, H., Koster, R., Zhang, J., Winkler, S., Aytar, Y., Osindero, S., Damen, D., Zisserman, A., Carreira, J.: Perception test: A diagnostic benchmark for multimodal video models. NeurIPS
9. Perazzi, F., Pont-Tuset, J., McWilliams, B., Van Gool, L., Gross, M., Sorkine-Hornung, A.: A benchmark dataset and evaluation methodology for video object segmentation. In: Proc. CVPR (2016)
10. Vecerik, M., Doersch, C., Yang, Y., Davchev, T., Aytar, Y., Zhou, G., Hadsell, R., Agapito, L., Scholz, J.: RoboTAP: Tracking arbitrary points for few-shot visual imitation. In: Proc. Intl. Conf. on Robotics and Automation (2024)
11. Wen, C., Lin, X., So, J., Chen, K., Dou, Q., Gao, Y., Abbeel, P.: Any-point trajectory modeling for policy learning. arXiv preprint arXiv:2401.00025 (2023)