BootsTAP: Bootstrapped Training for Tracking-Any-Point

Carl Doersch[1,3], 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
[3] Corresponding author, [lastname]@google.com

Abstract. To endow models with greater understanding of physics and motion, it is useful to enable them to perceive how solid surfaces move and deform in real scenes. This can be formalized as Tracking-Any-Point (TAP), which requires the algorithm to track any point on solid surfaces in a video, potentially densely in space and time. Large-scale ground-truth training data for TAP is only available in simulation, which currently has a limited variety of objects and motion. In this work, we demonstrate how large-scale, unlabeled, uncurated real-world data can improve a TAP model with minimal architectural changes, using a self-supervised student-teacher setup. We demonstrate state-of-the-art performance on the TAP-Vid benchmark surpassing previous results by a wide margin: for example, TAP-Vid-DAVIS performance improves from 61.3% to 67.4%, and TAP-Vid-Kinetics from 57.2% to 62.5%. For visualizations, see our project webpage at `https://bootstap.github.io/`

Keywords: Tracking-Any-Point · Self-Supervised · Semi-Supervised

1 Introduction

Despite impressive achievements in the vision and language capability of generalist AI systems, physical and spatial reasoning remain notable weaknesses of state-of-the art vision models [46, 59]. This limits their application in many domains like robotics, video generation, and 3D asset creation – all of which require an understanding of the complex motions and physical interactions in a scene. Tracking-Any-Point (TAP) [12] is a promising approach to represent precise motions in videos, and recent work has demonstrated compelling usage of TAP in robotics [2, 62, 69, 73], 3D reconstruction [64], video generation [13], video editing [71], zoology [48], and medicine [53]. In TAP, algorithms are fed a video and a set of query points—potentially densely across the video—and must output the tracked location of these query points in the video's other frames. If the point is not visible in a frame, the point is marked as occluded in that frame. This approach has many advantages: it is a highly general task, as correspondences for surface points are well-defined for opaque, solid surfaces, and

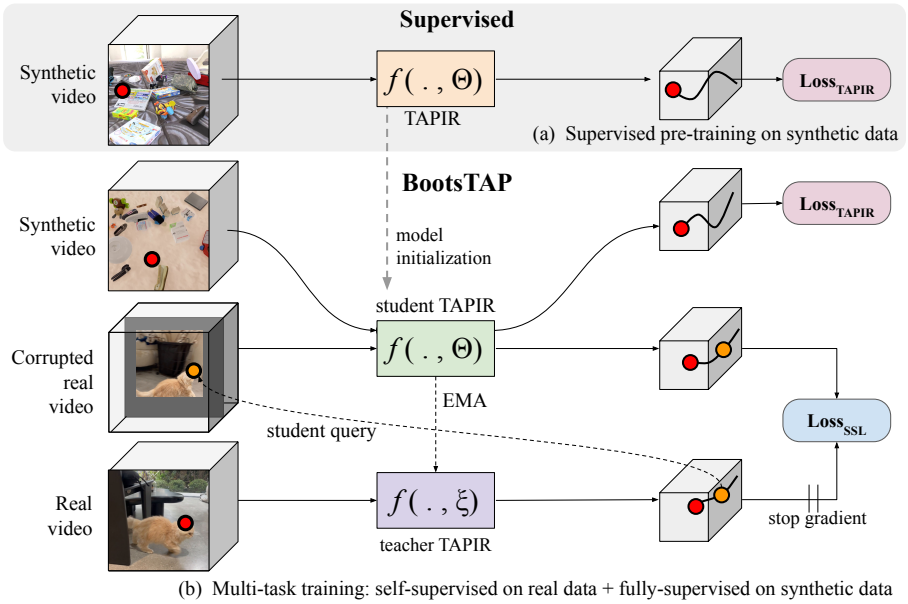

Fig. 1: Bootstrapped training for tracking-any-point. After initializing a TAPIR model with standard supervised training, we bootstrap the model on real data by adding an additional self-supervised loss. We apply a teacher model (a simple EMA of the student model) to get pseudo-ground-truth labels for a video. We then apply spatial transformations and corruptions to the video to make the task harder for the student, and train the student to reproduce the teacher's predictions from any query point along the teacher's trajectory.

it provides rich information about the deformation and motion of objects across long time periods.

The main challenge for building TAP models, however, is the lack of training data: in the real world, we must rely on manual labeling, which is arduous and imprecise [12], or on 3D sensing [1], which is only available in limited scenarios and quantity. Thus, state-of-the-art methods have relied on synthetic data [19, 75]. In this work, however, we overcome this limitation and demonstrate that unlabeled real-world videos can be used to improve point tracking, using self-consistency as a supervisory signal. In particular, we know that when tracks are correct for a given video, then 1) spatial transformations of the video should result in an equivalent spatial transformation of the trajectories, 2) that different query points along the same trajectory should produce the same track, and 3) that non-spatial data augmentation (e.g. image compression) should not affect results. Deviations from this can be treated as an error signal for learning.

Our architecture is outlined in Figure 1. We begin with a strong "teacher" model pre-trained using supervised learning on synthetic data (in our case, a TAPIR [13] model) which serves as initialization for both a "teacher" and a

"student" model. Given an unlabeled input video, we make a prediction using the teacher model, which serves as pseudo-ground-truth for the student. We then generate a second "view" of the video by applying affine transformations that vary smoothly in time, re-sampling frames to a lower resolution, and adding JPEG corruption, and padding back to the original size. We input the second view to the "student" network and use a query point sampled from the teacher's prediction (transformed consistently with the transformation applied to the video). The student's prediction is then transformed back into the original coordinate space. We then use a self-supervised loss (SSL) to update the student's weights: that is, we apply TAPIR's original loss function to the student predictions, using the teacher's predictions as pseudo-ground-truth. The teacher's weights are updated by using an exponential moving average (EMA) of the student's weights. We take steps to ensure that the teacher's predictions used for training are more likely to be accurate than the student's: (i) the corruptions that degrade and downsample the video are only applied to the student's inputs, (ii) we use an EMA of the student's weights as the teacher's weights, a common trick for stabilizing student-teacher learning [20,58]. Co-training using this formulation on real-world videos, in addition to training on synthetic data, provides a substantial boost over prior state-of-the-art across the entire TAP-Vid benchmark.
In summary, our contributions are as follows:

1. We demonstrate the first large-scale pipeline for improving video point tracking using a large dataset of unannotated videos, based on straightforward properties of real trajectories: (i) predictions should vary consistently with spatial transformations of the video, and (ii) predictions should be invariant to the choice of query point along a given trajectory.
2. We analyze the importance of varying model components, and show that a surprisingly simple formulation is sufficient to achieve good results.
3. We show that the resulting formulation achieves new SOTA results on point tracking benchmarks, while requiring minimal architectural changes.
4. We have released a model and checkpoint on GitHub, including model implementations in both JAX and PyTorch for the community to use.

2 Related Work

Tracking-Any-Point. The ability to track densely-sampled points over long video sequences is a generic visual capability [51, 52]. Because this visual task provides a rich output that is well-defined independent of semantic or linguisitic categories (unlike classification, detection, and semantic segmentation), it is more generically useful and can support other visual capabilities like video editing [71], 3D estimation [65], object segmentation [45, 49], camera tracking [8] and even robotics [62,69]. Point tracking has recently experienced a flurry of recent works including new datasets [1, 12, 75] and algorithms [3, 13, 22, 29, 42, 43, 65]. Current state-of-the-art works mainly train in a supervised manner, relying heavily on synthetic data [19, 75] which has a large domain gap with the real world.

Self-supervised correspondence via photometric loss. Tracking has long been a target of self-supervised learning due to the lack of reliable supervised data, especially at the point level. A wide variety of proxy supervisory signals have been proposed, all with their own limitations. Photometric losses use reconstruction, and are particularly popular in optical flow, but occlusions, lighting changes, and repeated (or constant) textures, typically result in multiple or false appearance matches. To compensate for this, these methods typically rely on complicated priors such as multi-frame estimation [26], explicit occlusion handling [56,68], improved data augmentation [35], additional loss terms [36,37,41], and robust loss functions to avoid degenerate solutions [39,50,72]. Methods combining feature learning with appearance reconstruction, such as [31,32,63], have demonstrated long-range tracking. Matches based on local appearance are more likely to correspond to motion in high resolution videos because they are able to resolve detailed textures [27]; we make use of this observation in our work.

Temporal continuity and cycle-consistency. Other works use images or videos to perform more general feature learning, with the aim that features in correspondence should be more similar than those which are not. Temporal continuity in videos has long been used to obtain such correspondences [15,16,25, 66,70], resulting in features which have proven to be effective for object tracking [10, 17]. Temporal cycle-consistency [4, 67] can also result in features useful for tracking; however this learning method fails to provide useful supervision in challenging situations such as occlusions.

Semi-supervised correspondence. A final self-supervised approach is to create pseudo-ground-truth correspondences for semi-supervised training [23, 54]. Such approaches have a long history in optical flow [24,36,37,44], although with mixed results, typically requiring complex training setups such as GANs [30] or connecting the student to the teacher [38] to prevent trivial solutions. They have only been applied to longer-term point tracking more recently [57,65]. OmniMotion computes initial point tracks using RAFT [60] or TAP-Net [12] and infers a full pseudo-3D interpretation of the scene in the form of a neural network. Although this method improves point tracks compared to their initialization, it never retrains a general TAP model on the self-labeled data. Li et al. [33], proposes a self-supervised loss based on reconstruction, in addition to supervised point tracking loss and an adversarial domain adaptation loss. The final algorithm is complex, and performs far below our work (59.8 on TAP-Vid-DAVIS $< \delta^x_{avg}$, versus 78.1 for our work), with the self-supervised providing a relatively small boost. Perhaps most related is work performed concurrently with ours [57], which saves a dataset of point tracks and retrains the underlying TAP model on them, using data augmentations similar to ours. We discuss the differences in detail in the following section, after presenting our approach.

3 Method

When developing a self-supervised training method for TAP, it is worth noting that TAP has a precise answer for almost every query point. This is different from

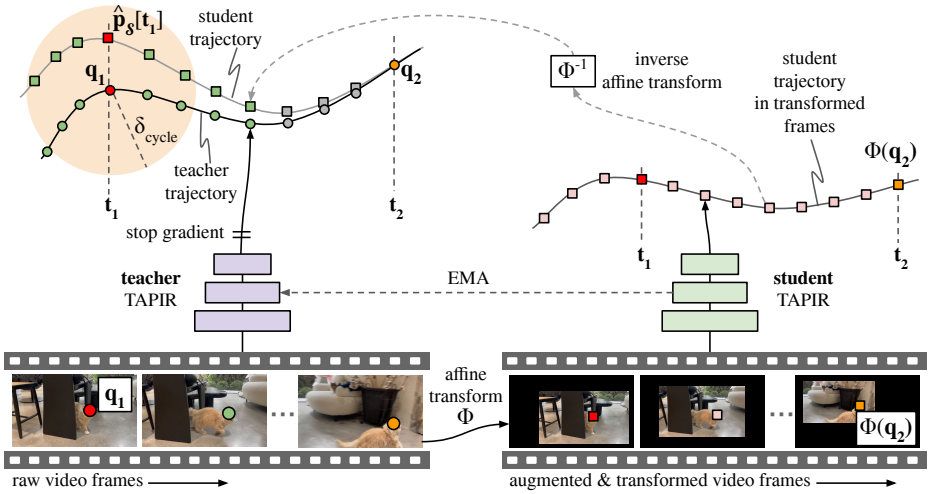

Fig. 2: Bootstrapped training for Tracking-Any-Point. The teacher TAPIR produces a pseudo-label trajectory from query point q_1 at time t_1. Video frames undergo affine transformations Φ that vary smoothly in time and are augmented with JPEG artifacts, then fed to the student TAPIR, which predicts a trajectory from query point $\Phi(q_2)$ at time t_2. The student trajectory is transformed back, and loss is computed against the teacher's trajectory. To maximize the chances that we train on accurate trajectories, we remove trajectories where the student's prediction at time t_1 is too far from the teacher query point q_1 (i.e. not cycle-consistent; light-orange disk).

typical visual self-supervised learning, where the representation can be arbitrary, as long as semantically similar images have similar representations. Supervised learning on synthetic data provides a strong initial guess in many situations, but care must be taken to ensure that the self-supervised algorithm does not find "trivial shortcuts" [11] that become self-reinforcing and harm the initialization.

Our formulation relies on two facts about point tracks that are true for points on any solid, opaque surface. First, spatial transformations (e.g. affine transformations) which are applied to the video will result in equivalent spatial transformations of the point tracks (i.e. the tracks are "equivariant" under spatial transformation), while the tracks are invariant to many other factors of variation that do not move the image content (e.g. color changes, noise). Second, the algorithm should output the same track regardless of which point along the track is used as a query; mathematically, this means that each trajectory forms an equivalence class. To capture these intuitions, we propose a student-teacher pipeline, loosely based on BYOL [20] and Mean Teachers [58], where the student must reproduce the teacher's prediction under different data augmentations and different query points. Figure 2 shows the overall pipeline.

Architecture. Our model closely follows TAPIR [13]. This model applies a ConvNet backbone to the video, extracts a feature for each query point with bilinear interpolation, and then compares this feature to features on all other frames (dot products) to get a per-frame heatmap, followed by a soft argmax to get an initial point for each frame. From there, the model applies multiple refinement steps, where each refinement step takes local comparisons between the query point and a neighborhood of features around the current track estimate (dot products), and passes these to a ConvNet operating across time, which produces an update. We start with a pretrained TAPIR model [13]. After pre-training, we add extra capacity to the model to absorb the extra training data: 5 layers of 2D conv-residual layers to the backbone with a channel multiplier of 4, which roughly doubles the number of parameters in the backbone (see supplementary for details). These are initialized to the identity following "zero init" [18]. Otherwise the architecture is unchanged.

Loss functions. Let $\hat{y} = \{\hat{p}, \hat{o}, \hat{u}\}$ be the predictions: $\hat{p} \in \mathbb{R}^{T\times 2}$ is the position, $\hat{o} \in \mathbb{R}^T$ is an occlusion logit, and $\hat{u} \in \mathbb{R}^T$ is an uncertainty logit, where T is the number of frames. Calling $p[t]$ and $o[t]$ the ground truth targets for frame t, recall that the standard TAPIR loss for a single trajectory is defined as:

$$\begin{aligned}\mathcal{L}_{tapir}(\hat{p}[t], \hat{o}[t], \hat{u}[t]) = &\ \text{Huber}(\hat{p}[t], p[t])(1 - o[t]) && \text{Position loss}\\ &+ \text{BCE}(\hat{o}[t], o[t]) && \text{Occlusion loss}\\ &+ \text{BCE}(\hat{u}[t], u[t])(1 - o[t]) && \text{Uncertainty loss}\end{aligned} \tag{1}$$

where Huber is the Huber loss and BCE is the sigmoid binary cross-entropy. The target for the uncertainty logit is defined as $u[t] = \mathbb{1}(d(p[t], \hat{p}[t]) > \delta)$, where d the L_2 distance and δ is a threshold on the distance, set to 6 pixels, and $\mathbb{1}$ is an indicator function. That is, the uncertainty loss trains the model to predict whether its own prediction is likely to be within a threshold of the ground truth.

Let $\hat{y}_\mathcal{S} = \{\hat{p}_\mathcal{S}, \hat{o}_\mathcal{S}, \hat{u}_\mathcal{S}\}$ now refer to the student predictions. We derive pseudo-labels $y_\mathcal{T} = \{p_\mathcal{T}, o_\mathcal{T}, u_\mathcal{T}\}$ from the teacher's predictions $\hat{y}_\mathcal{T} = \{\hat{p}_\mathcal{T}, \hat{o}_\mathcal{T}, \hat{u}_\mathcal{T}\}$ as follows:

$$p_\mathcal{T}[t] = \hat{p}_\mathcal{T}[t] \quad ; \quad o_\mathcal{T}[t] = \mathbb{1}(\hat{o}_\mathcal{T}[t] > 0); \quad u_\mathcal{T}[t] = \mathbb{1}(d(\hat{p}_\mathcal{T}[t], \hat{p}_\mathcal{S}[t]) > \delta) \tag{2}$$

where t indexes time. The loss $\ell_{ssl}(\hat{p}_\mathcal{S}[t], \hat{o}_\mathcal{S}[t], \hat{u}_\mathcal{S}[t])$ for a given video frame t is derived from the TAPIR loss, treating the pseudo-labels as ground-truth, and defined as:

$$\begin{aligned}\ell_{ssl}(\hat{p}_\mathcal{S}[t], \hat{o}_\mathcal{S}[t], \hat{u}_\mathcal{S}[t]) = &\ \text{Huber}(\hat{p}_\mathcal{S}[t], p_\mathcal{T}[t])(1 - o_\mathcal{T}[t])\\ &+ \text{BCE}(\hat{o}_\mathcal{S}[t], o_\mathcal{T}[t])\\ &+ \text{BCE}(\hat{u}_\mathcal{S}[t], u_\mathcal{T}[t])(1 - o_\mathcal{T}[t])\end{aligned} \tag{3}$$

Note that TAPIR's loss uses multiple refinement iterations, but we always use the teacher's final prediction to derive pseudo-ground-truth; therefore, refined predictions serve as supervision for unrefined ones, encouraging stronger features that enable faster convergence.

Video degradations. While the above formulation is well-defined, if the student and teacher both receive the same video and query point, we expect the loss to be trivially close to zero; therefore, we apply transformations and corruptions to the student's view of the video. Given an input video, we create a second view by resizing each frame to a smaller resolution r and superimposing it onto a black background at a random position (h, w) within this background. r varies linearly over time, meaning that the frames gradually become larger or smaller within the fixed-size black background. Overall, the decreased resolution degrades the student view, and this increases task difficulty for the student. The location of these frames also move with time, and (h, w) follows a linear trajectory within the black background. Formally, this is a frame-wise axis-aligned affine transformation Φ on coordinates, applied to the pixels. We also apply Φ to the student query coordinates. We further degrade this view by applying a random JPEG degradation to make the task more difficult, before pasting it onto the black background. Both operations lose texture information; therefore, the network must learn higher-level—and possibly semantic—cues (e.g. the tip of the top left ear of the cat), rather than lower-level texture matching in order to track points correctly. We apply the inverse affine transformation Φ^{-1} to map the student's predictions back to the original input coordinate space before feeding these to the loss. We describe these transformations in more detail in supplementary.

Choosing the sample point. We enforce that each trajectory forms an equivalence class by training the model to produce the same track regardless of which point is used as a query. While we do not have access to the ground-truth trajectories to sample different query points from, we can use the teacher model's predictions to form pairs of query points. First, we sample a query point $Q_1 = (q_1, t_1)$, where q_1 is an (x, y) coordinate, and t_1 is a frame index, both sampled uniformly. Then the student's query is sampled randomly from the teacher's trajectory, i.e. $Q_2 = (q_2, t_2) \in \{(p_{\mathcal{T}}[t], t); t \ s.t. \ o_{\mathcal{T}}[t] = 0\}$.

Note, however, that if the teacher has not tracked the point correctly, the student's query might be a different real-world point than the teacher's, leading to an erroneous training signal. To prevent this, we use cycle-consistency of the student and teacher trajectories, and ignore the loss for trajectories that don't form a valid cycle, as depicted by the orange circle in Figure 2. Formally, we implement this as a mask defined as:

$$m_{cycle} = \mathbb{1}\left(d(\hat{p}_{\mathcal{S}}[t_1], q_1) < \delta_{cycle}\right) \quad * \quad \mathbb{1}\left(\hat{o}_{\mathcal{S}}[t_1] \leq 0\right) \tag{4}$$

Here, δ_{cycle} is a distance threshold hyperparameter, which we set to 4 pixels.

Note that there is a special case when the student and teacher have the same query point: there is no longer any uncertainty regarding whether the point is on

the same trajectory. These points are reliable while also being less challenging. We compromise between extremes, and sample $Q_1 = Q_2$ with probability 0.5, and sample with equal probabilities from the remaining visible points in the teacher prediction. The final self-supervised loss for a single trajectory is then:

$$\mathcal{L}_{SSL} = \sum_t m^t_{cycle} * \ell^t_{ssl} \tag{5}$$

In practice, we sample 128 query points per input video and average the loss for all of them. We provide pseudocode for the algorithm in supplementary.

To avoid catastrophic forgetting, we continue training on the pretraining dataset with the regular supervised TAPIR loss. Our training setup follows prior work on multi-task self-supervised learning [14]: we maintain separate Adam optimizer parameters to compute separate updates for both tasks, and then apply the gradients with their own learning rates. As the self-supervised task is more expensive due to the extra forward pass, we use half the batch size for self-supervised updates, and therefore we halve the learning rate for these updates. See supplementary for more details.

Differences between our approach and PIPs+Refine [57]. PIPs+Refine, performed concurrently with ours, operates on a similar intuition, of retraining a base model using its own predictions after applying data augmentation, although this method performs non-trivially worse. They train on affine-transformed versions of the model's original predictions use cycle-consistency as a method of filtering. However, there are a few key differences. First, rather than a student-teacher setup, they compute trajectories only once and freeze the training data, meaning that the model is permanently trained to reproduce errors in the original labeling. They do not attempt to retrain on the results of the refined network; without any updates, the paper finds that the model rapidly overfits to the original predictions and performance degrades. Our work resolves this limitation by a student-teacher formulation, and overall we find that we can train for very long training schedules without degradation. Furthermore, PIPS+Refine fine-tunes on the target dataset, meaning that transfer to a new domain may require a large training set in that domain on which to fine-tune; in contrast, our work demonstrates that it's possible to train on a single large dataset that covers many domains, meaning that we can achieve SOTA results out-of-the-box. Overall, their performance is far worse (59.8 on TAP-Vid-DAVIS $< \delta^x_{avg}$, versus 78.1 for our work), both due to these limitations and also due to building on a lower-performing baseline algorithm.

4 Experiments

We train our model on over 15 million 24-frame clips from publicly-available online videos, in conjunction with standard training on Kubric. The resulting model is essentially a drop-in replacement for TAPIR (albeit with slightly larger computational requirements due to the extra layers). We evaluate on the TAP-Vid benchmark using the standard protocol.

4.1 Training datasets

Our models are pretrained on the Kubric dataset following [13]. For the bootstrapping stage we propose, we collected a video dataset from publicly accessible videos selected from categories that typically contain high-quality and realistic motion (such as lifestyle and one-shot videos). Conversely, we omitted videos from categories with low visual complexity or unrealistic motions, such as tutorial videos, lyrics videos, and animations. To maintain consistency, we exclusively obtained videos shot at 60 fps. Additionally, we applied a quality metric by only considering videos with over 200 views. We removed the first and last 2 seconds of each video, as these often contain intros and outros with text or other overlays. From each video, we randomly sampled five clips, excluding those with overlay/watermarked frames, which were identified by checking the horizontal and vertical gradients and computing the pixel-wise median (similar to [9]). Furthermore, we expect the teacher signal will be more reliable on continuous shots due to temporal continuity; therefore, clips with shot boundary changes are detected and removed based on [5, 40, 61, 74] with additional accuracy improvements based on full-frame geometric alignment. In total, we generated 15 million clips for training.

4.2 Evaluation datasets

We rely on the TAP-Vid [12] and RoboTAP [62] benchmarks for quantitative evaluation; in all cases, we evaluate zero-shot on the entire benchmark, resizing to 256×256 before evaluating according to the standard procedure [12]. This consists of five datasets: **TAP-Vid-Kinetics** contains online videos of human actions and may include cuts [7]; **TAP-Vid-DAVIS** is based on the DAVIS object tracking benchmark [47]; **TAP-Vid-RGB-Stacking** contains synthetic tracks for videos of robotic manipulation which have little texture; and **RoboTAP** contains real-world robotic manipulation videos [62], all of which include ground truth. Evaluation is performed by measuring occlusion accuracy (OA), $< \delta^{\mathbf{x}}_{\mathbf{avg}}$ which measures the fraction of point estimates within a specified distance to the ground truth location, averaged across 5 thresholds, and Average Jaccard (AJ) which measures a combination of these two. There are two dataset querying "modes": *query first* (*q_first*) uses the first visible point on each trajectory as a query, while *strided* uses every fifth point along the trajectory as a separate query. We also include qualitative evaluations on two robotics datasets without ground truth: **RoboCAT-NIST**, a subset of the data collected for RoboCat [6], and **Libero** [34], a dataset where point tracking has already proven useful for robotic manipulation [69]. See supplementary for details on these datasets and metrics.

4.3 Results

Our results are shown in Table 1. Note that all of our numbers come from a single checkpoint, which has not seen the relevant datasets. Relative to our base

Table 1: Comparison of performance on the TAP-Vid datasets. AJ (Average Jaccard; higher is better) measures both occlusion and position accuracy. $< \delta^x_{avg}$ (higher is better) measures only localization performance, ignoring occlusion accuracy. OA (Occlusion Accuracy; higher is better) measures only accuracy in predicting occlusion. Best method is written **bold** and the second best method is underlined.

	Kinetics			DAVIS			RGB-Stacking		
Method	AJ ↑	$< \delta^x_{avg}$ ↑	OA ↑	AJ ↑	$< \delta^x_{avg}$ ↑	OA ↑	AJ ↑	$< \delta^x_{avg}$ ↑	OA ↑
COTR [28]	19.0	38.8	57.4	35.4	51.3	80.2	6.8	13.5	79.1
Kubric-VFS-Like [19]	40.5	59.0	80.0	33.1	48.5	79.4	57.9	72.6	**91.9**
RAFT [60]	34.5	52.5	79.7	30.0	46.3	79.6	44.0	58.6	90.4
TAP-Net [12]	46.6	60.9	85.0	38.4	53.1	82.3	59.9	72.8	90.4
TAP-Net (tuned)	50.7	64.9	85.7	43.9	59.2	83.9	66.5	77.9	90.2
PIPs [22]	35.3	54.8	77.4	42.0	59.4	82.1	37.3	51.0	91.6
PIPs+Refinement [57]	-	-	-	42.5	60.0	-	-	-	-
TAPIR [13]	57.2	70.1	87.8	61.3	73.6	88.8	62.7	74.6	91.6
CoTracker [29]	57.3	70.6	87.5	64.8	**79.1**	88.7	65.9	80.6	85.0
BootsTAP-Net	51.7	65.3	85.1	44.3	59.0	82.7	64.9	76.4	89.1
BootsTAPIR	**61.4**	**74.2**	**89.7**	**66.2**	78.1	**91.0**	**72.4**	**83.1**	91.2

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

	Kinetics			DAVIS			RoboTAP		
Method	AJ ↑	$< \delta^x_{avg}$ ↑	OA ↑	AJ ↑	$< \delta^x_{avg}$ ↑	OA ↑	AJ ↑	$< \delta^x_{avg}$ ↑	OA ↑
TAP-Net [12]	38.5	54.4	80.6	33.0	48.6	78.8	45.1	62.1	82.9
TAP-Net (tuned)	42.0	57.2	80.5	38.4	53.8	80.5	59.2	72.2	**87.8**
TAPIR [13]	49.6	64.2	85.0	56.2	70.0	86.5	59.6	73.4	87.0
CoTracker [29]	48.7	64.3	**86.5**	60.6	**75.4**	**89.3**	54.0	65.5	78.8
BootsTAP-Net	42.6	57.5	79.4	39.1	53.9	79.8	60.7	73.3	86.9
BootsTAPIR	**54.6**	**68.4**	**86.5**	**61.4**	73.6	88.7	**64.9**	**80.1**	86.3

architecture, our bootstrapping approach provides a substantial gain across all metrics. We also outperform CoTracker on DAVIS, though this is due more to improvements in occlusion accuracy than position accuracy. This is despite TAPIR having a simpler architecture than CoTracker, which requires cross attention to other points which must be chosen with a hand-tuned distribution, whereas TAPIR tracks points independently. CoTracker results are also obtained by upsampling videos to 384×512, which further increases compute time, whereas ours are computed directly on 256×256 videos.

Table 2 shows performance under *q_first* mode. Here, we see that bootstrapping outperforms prior works by a wide margin on Kinetics; this is likely because TAPIR's global search is more robust to large occlusions and cuts, which are more prominent in Kinetics. This search might harm performance in datasets like DAVIS with a stronger temporal continuity bias. Perhaps most impressive is the strong improvement in RoboTAP–over 5% absolute performance–despite RoboTAP looking very different from typical online videos. We see similar re-

Fig. 3: Comparison between TAPIR (■), Cotracker (◆) and BootsTAPIR (●), and the ground-truth points (+) on TAP-Vid-DAVIS and RoboTAP benchmarks. We show the initial query frame, and a closeup of four later frames.

sults for RGB-Stacking in Table 1. These two datasets have large textureless regions; such regions are challenging to track without object-aware priors, which are difficult to obtain from synthetic datasets. We further improve performance by 1-4% extra percentage points by training on high resolution and longer clips; see our project webpage for these results. To demonstrate the generality of our approach, we also applied an identical training regimen to TAP-Net. Again we find that a longer training schedule matching ours improves performance (TAP-Net (tuned)), but applying BootsTAP on top shows an improvement in 5 of 6 settings, although not as large as for TAPIR, perhaps due to the lower performance ceiling due to the coarser features. The degradation on RGB-Stacking may be due to the fact that it is synthetic, may real-world data may not improve feature matching unless the algorithm can better use temporal context, as in TAPIR.

Figure 3 shows qualitative examples of some cases where BootsTAPIR improves performance. We see improvements on examples where texture cues are ambiguous (e.g. the dark jacket and trousers) where prior knowledge of common object shape can improve performance, as well as points near object boundaries (e.g. the dog's ears) where a model trained on synthetic data with different appearance may struggle to estimate the correct segmentation. We also note that BootsTAPIR improves on many cases where TAPIR marks a point as occluded even when it is still visible, such as the person's arm. On RoboTAP, the model

improves on occlusion estimation for the textureless gripper. It also deals well with shiny objects, both of which are less common in Kubric. Our supplementary and webpage have more qualitative results, including results on robotic gear insertion, and also results in video format, which make the improvements more obvious.

We also compare the inference time between TAPIR and BootsTAPIR. Despite the significant performance improvements achieved by BootsTAPIR, the running time remains nearly unchanged. For instance, on an A100 GPU, processing a 90-frame, 256 × 256 video with 100 points takes 0.25 seconds with TAPIR and 0.28 seconds with BootsTAPIR. Similarly, for 1000 points, TAPIR completes the task in 2.42 seconds, while BootsTAPIR requires 2.48 seconds.

4.4 Ablations

We focus on four main areas of ablation: **data transformations**, **pseudo-label filtering approaches**, **training setup**, and **training data**. To arrive at our final model, we performed ablations on a smaller-scale BASE setting with our best guesses at the optimal hyperparameter settings. This setting includes two components that we found could be removed without harming performance: It uses an additional mask on the occlusion loss, inspired by FixMatch [55], where any occlusion estimate that the teacher is uncertain about $\max(\sigma(\hat{o}_\mathcal{T}[t]), 1 - \sigma(\hat{o}_\mathcal{T}[t])) < 0.6$ is ignored in the loss. It uses a 3D-ConvNet backbone, which we find provides a slight improvement on DAVIS while harming performance on Kinetics (see supplementary), so we remove it for future compatibility with causal TAPIR models. Finally, BASE also halves the batch sizes (and proportionally halves the learning rate), and also halves the number of training steps. We report Average Jaccard on DAVIS using the *strided* mode and on Kinetics using the *q_first* mode.

Data transformations. We first investigate the effect of the transformations we apply on inputs and outputs in this setting. We respectively ablate: the use of random JPEG augmentations to enforce invariance to various factors of variation (denoted by BASE-no-augm); the use of framewise affine transformations on inputs and outputs to enforce equivariance with spatial transformations (denoted by BASE-no-affine). We also investigate sampling the student queries: recall that in our typical setup, we sample the student query from a distribution which places probability 0.5 on the original teacher query point, and 0.5 on a uniform distribution across visible points. In BASE-same-queries, we always use the teacher's query for the student, and in BASE-uniform, we sample from a purely uniform distribution. We report the results for each ablation in Table 3 (a). We observe that removing JPEG somewhat harms metrics, especially on Kinetics. In contrast, when ablating affine transformations, we find that performance drops massively across metrics, suggesting overfitting. Finally, we find that using different query points improves performance compared with BASE-same-queries, leading to more accurate position predictions in particular ($< \delta^{\mathbf{x}}_{\mathbf{avg}}$ increases from 77.5 to 77.9 on DAVIS *strided* and from 66.8 to 67.7 on Kinetics *q_first*). Note,

Table 3: Ablations of model hyperparameters, including (a) ablations of the data transformations and query point strategies, (b) comparisons of the pseudo-label filtering approaches, (c) ablations of the training setup, including the stop gradient, and (d) ablations of the dataset. We report Average Jaccard (AJ) across all experiments.

(a) Data transformations.

Method	DAVIS *strided*	Kinetics *q_first*
BASE	65.8	54.4
BASE-no-augm	65.7	53.5
BASE-no-affine	54.4	44.7
BASE-same-queries	65.6	53.2
BASE-uniform	65.6	54.3

(b) Pseudo-labels filtering. BASE filters occlusion loss terms based on teacher confidence.

Method	DAVIS *strided*	Kinetics *q_first*
BASE	65.8	54.4
BASE-no-filtering	65.9	54.1
BASE+cycle	66.1	54.3

(c) Training setup.

Method	DAVIS *strided*	Kinetics *q_first*
Full model	66.2	54.6
FULL-kubric-only	65.0	52.7
BASE	65.8	54.4
SIAMESE	49.8	29.6

(d) Training Data.

Method	DAVIS *strided*	Kinetics *q_first*
BASE	65.8	54.4
2-frame clips	64.3	50.5
6-frame clips	63.7	50.9
1% of real data	66.2	54.0

however, on DAVIS in particular, this improvement depends on sampling the original teacher query point more often than the others.

Pseudo-label filtering. We next consider the effectiveness of filtering possibly incorrect teacher tracks and points, with results in Table 3 (b). BASE-no-filtering removes the filtering that BASE uses on the occlusion confidence score, which makes little difference in performance on DAVIS, but degrades performance on Kinetics. BASE+cycle uses the cycle-consistency criterion from our full model instead and performs slightly better on DAVIS. These results suggest that correctly removing bad teacher tracks remains an open problem.

Training setup. Table 3 (c) shows ablations of the overall training setup. In particular, training for longer with a higher capacity model can improve results, so FULL-kubric-only uses an identical training setup to our full model, but removes self-supervised training, instead simply training on Kubric for longer. We see competitive performance, although self-supervised training still improves by 1.2% on DAVIS and almost 2% on Kinetics. One could imagine enforcing the desired equivariance and invariance properties using a simple Siamese-network formulation [21], where a single network is trained to output consistent predictions on two different 'views' of the data (i.e., augmented and transformed versions of the video and tracks). SIAMESE shows the effect of removing the EMA and stop-gradient and instead backpropping to both student and teacher models

(in this case, using the BASE setting): we note that performance on real-world datasets collapses as the model finds trivial shortcuts.

Training data. We ablate two questions regarding the dataset. Prior work has argued that simple semi-supervised learning for optical flow performs poorly [30, 38]; we hypothesize that more temporal context may be the key ingredient to change this story. To validate this, we reran our algorithm using 2- and 6-frame clips from our full dataset. In Table 3 (d), we see that this indeed performs poorly, possibly because the extra frames allow the teacher model to correct more errors. Interestingly, we also tried training on a 1% subset of the data, and found that this harms performance on Kinetics but actually improves it on DAVIS. It's possible that the algorithm begins overfitting to the data, but this may be useful for clean data like DAVIS. Regardless, it suggests this algorithm can be effective even in situations where less data is available.

5 Conclusion

In this work, we presented a simple and effective method for leveraging large-scale, unlabeled data for improving TAP performance. We demonstrated that a straightforward application of consistency principles, i.e. invariance to query points and non-spatial corruptions, and equivariance to affine transformations, enable the model to continue to improve on unlabeled data. Our formulation avoids more complex priors such as spatial smoothness of motion or temporal smoothness of tracks that are used in many prior works. In fact, our formulation bears similarities to baselines for two-frame, self-supervised optical flow that are considered too "unstable" to be effective (c.f. Fig. 2(a) in "Flow Supervisor" [24]). Yet in our multi-frame approach, we ultimately surpass the state-of-the-art performance by a large margin. We find little evidence of model 'overfitting' to its own biases in ways that cause performance to degrade with long training like in other work [57]. Instead, we find that performance continues to improve for as long as we train the model. Our work does have some limitations: training remains computationally expensive. Furthermore, our estimated correspondence is a single point estimate throughout the entire video, which means we cannot elegantly handle duplicated or rotationally symmetric objects where the actual correspondence is ambiguous. Nevertheless, our approach demonstrates that it is possible to better bridge the sim-to-real gap using self-supervised learning.

Acknowledgements: We thank Jon Scholz, Stannis Zhou, Mel Vecerik, Yusuf Aytar, Viorica Patraucean, Mehdi Sajjadi, Daniel Zoran, and Nando de Freitas for valuable discussions and support, and David Bridson, Lucas Smaira, and Michael King for help on datasets.

References

1. Balasingam, A., Chandler, J., Li, C., Zhang, Z., Balakrishnan, H.: Drivetrack: A benchmark for long-range point tracking in real-world videos. arXiv preprint arXiv:2312.09523 (2023)
2. Bharadhwaj, H., Mottaghi, R., Gupta, A., Tulsiani, S.: Track2Act: Predicting point tracks from internet videos enables diverse zero-shot robot manipulation. arXiv preprint arXiv:2405.01527 (2024)
3. Bian, W., Huang, Z., Shi, X., Dong, Y., Li, Y., Li, H.: Context-pips: Persistent independent particles demands context features. NeurIPS (2024)
4. Bian, Z., Jabri, A., Efros, A.A., Owens, A.: Learning pixel trajectories with multiscale contrastive random walks. In: Proc. CVPR (2022)
5. Boreczky, J.S., Rowe, L.A.: Comparison of video shot boundary detection techniques. Journal of Electronic Imaging **5**(2), 122–128 (1996)
6. 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)
7. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: Proc. CVPR. pp. 6299–6308 (2017)
8. Chen, W., Chen, L., Wang, R., Pollefeys, M.: Leap-vo: Long-term effective any point tracking for visual odometry. arXiv preprint arXiv:2401.01887 (2024)
9. Dekel, T., Rubinstein, M., Liu, C., Freeman, W.T.: On the effectiveness of visible watermarks. In: Proc. CVPR (2017)
10. Denil, M., Bazzani, L., Larochelle, H., de Freitas, N.: Learning where to attend with deep architectures for image tracking. Neural computation **24**(8), 2151–2184 (2012)
11. Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: Proc. ICCV (2015)
12. 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)
13. Doersch, C., Yang, Y., Vecerik, M., Gokay, D., Gupta, A., Aytar, Y., Carreira, J., Zisserman, A.: TAPIR: Tracking any point with per-frame initialization and temporal refinement. arXiv preprint arXiv:2306.08637 (2023)
14. Doersch, C., Zisserman, A.: Multi-task self-supervised visual learning. In: Proc. ICCV (2017)
15. Földiák, P.: Learning invariance from transformation sequences. Neural computation **3**(2), 194–200 (1991)
16. Goroshin, R., Bruna, J., Tompson, J., Eigen, D., LeCun, Y.: Unsupervised learning of spatiotemporally coherent metrics. In: Proc. ICCV (2015)
17. Goroshin, R., Mathieu, M.F., LeCun, Y.: Learning to linearize under uncertainty. NeurIPS (2015)
18. Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K.: Accurate, large minibatch SGD: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677 (2017)
19. 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)
20. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al.: Bootstrap your own latent - a new approach to self-supervised learning. In: NeurIPS (2020)

21. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In: Proc. CVPR (2006)
22. Harley, A.W., Fang, Z., Fragkiadaki, K.: Particle video revisited: Tracking through occlusions using point trajectories. In: Proc. ECCV (2022)
23. Huang, H.P., Herrmann, C., Hur, J., Lu, E., Sargent, K., Stone, A., Yang, M.H., Sun, D.: Self-supervised autoflow. In: Proc. CVPR (2023)
24. Im, W., Lee, S., Yoon, S.E.: Semi-supervised learning of optical flow by flow supervisor. In: Proc. ECCV (2022)
25. Jabri, A., Owens, A., Efros, A.: Space-time correspondence as a contrastive random walk. NeurIPS **33**, 19545–19560 (2020)
26. Janai, J., Guney, F., Ranjan, A., Black, M., Geiger, A.: Unsupervised learning of multi-frame optical flow with occlusions. In: Proc. ECCV (2018)
27. Janai, J., Guney, F., Wulff, J., Black, M.J., Geiger, A.: Slow flow: Exploiting high-speed cameras for accurate and diverse optical flow reference data. In: Proc. CVPR (2017)
28. Jiang, W., Trulls, E., Hosang, J., Tagliasacchi, A., Yi, K.M.: COTR: Correspondence transformer for matching across images. In: Proc. ICCV (2021)
29. 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)
30. Lai, W.S., Huang, J.B., Yang, M.H.: Semi-supervised learning for optical flow with generative adversarial networks (2017)
31. Lai, Z., Lu, E., Xie, W.: MAST: A memory-augmented self-supervised tracker. In: Proc. CVPR (2020)
32. Lai, Z., Xie, W.: Self-supervised learning for video correspondence flow. arXiv preprint arXiv:1905.00875 (2019)
33. Li, R., Zhou, S., Liu, D.: Learning fine-grained features for pixel-wise video correspondences. In: Proc. ICCV (2023)
34. 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)
35. Liu, L., Zhang, J., He, R., Liu, Y., Wang, Y., Tai, Y., Luo, D., Wang, C., Li, J., Huang, F.: Learning by analogy: Reliable supervision from transformations for unsupervised optical flow estimation. In: Proc. CVPR (2020)
36. Liu, P., King, I., Lyu, M.R., Xu, J.: Ddflow: Learning optical flow with unlabeled data distillation. In: Proceedings of the AAAI conference on artificial intelligence. vol. 33, pp. 8770–8777 (2019)
37. Liu, P., Lyu, M., King, I., Xu, J.: Selflow: Self-supervised learning of optical flow. In: Proc. CVPR (2019)
38. Liu, P., Lyu, M.R., King, I., Xu, J.: Learning by distillation: a self-supervised learning framework for optical flow estimation. IEEE PAMI **44**(9), 5026–5041 (2021)
39. Marsal, R., Chabot, F., Loesch, A., Sahbi, H.: Brightflow: Brightness-change-aware unsupervised learning of optical flow. In: Proc. WACV (2023)
40. Mas, J., Fernandez, G.: Video shot boundary detection based on color histogram. In: TRECVID (2003)
41. Meister, S., Hur, J., Roth, S.: Unflow: Unsupervised learning of optical flow with a bidirectional census loss. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32 (2018)
42. Moing, G.L., Ponce, J., Schmid, C.: Dense optical tracking: Connecting the dots. In: Proc. CVPR (2024)
43. Neoral, M., Šerỳch, J., Matas, J.: MFT: Long-term tracking of every pixel. In: Proc. WACV (2024)

44. Novák, T., Šochman, J., Matas, J.: A new semi-supervised method improving optical flow on distant domains. In: Computer Vision Winter Workshop. vol. 3 (2020)
45. Ochs, P., Malik, J., Brox, T.: Segmentation of moving objects by long term video analysis. IEEE transactions on pattern analysis and machine intelligence **36**(6), 1187–1200 (2013)
46. OpenAI: GPT-4V(ision) system card (September 25, 2023)
47. 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)
48. Polajnar, J., Kvinikadze, E., Harley, A.W., Malenovskỳ, I.: Wing buzzing as a mechanism for generating vibrational signals in psyllids. Insect Science (2024)
49. Rajič, F., Ke, L., Tai, Y.W., Tang, C.K., Danelljan, M., Yu, F.: Segment anything meets point tracking. arXiv preprint arXiv:2307.01197 (2023)
50. Ren, Z., Yan, J., Ni, B., Liu, B., Yang, X., Zha, H.: Unsupervised deep learning for optical flow estimation. In: Proceedings of the AAAI conference on artificial intelligence. vol. 31 (2017)
51. Rubinstein, M., Liu, C., Freeman, W.T.: Towards longer long-range motion trajectories. In: Proc. BMVC (2012)
52. Sand, P., Teller, S.: Particle video: Long-range motion estimation using point trajectories. Proc. ICCV (2008)
53. Schmidt, A., Mohareri, O., DiMaio, S., Salcudean, S.E.: Surgical tattoos in infrared: A dataset for quantifying tissue tracking and mapping. IEEE Transactions on Medical Imaging (2024)
54. Shen, Y., Hui, L., Xie, J., Yang, J.: Self-supervised 3d scene flow estimation guided by superpoints. In: Proc. CVPR (2023)
55. Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C.A., Cubuk, E.D., Kurakin, A., Li, C.L.: Fixmatch: Simplifying semi-supervised learning with consistency and confidence (2020)
56. Stone, A., Maurer, D., Ayvaci, A., Angelova, A., Jonschkowski, R.: Smurf: Self-teaching multi-frame unsupervised raft with full-image warping. In: Proc. CVPR (2021)
57. Sun, X., Harley, A.W., Guibas, L.J.: Refining pre-trained motion models. Proc. Intl. Conf. on Robotics and Automation (2024)
58. Tarvainen, A., Valpola, H.: Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In: NeurIPS (2017)
59. Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A.M., Hauth, A., et al.: Gemini: a family of highly capable multimodal models. arXiv preprint arXiv:2312.11805 (2023)
60. Teed, Z., Deng, J.: RAFT: Recurrent all-pairs field transforms for optical flow. In: Proc. ECCV (2020)
61. Truong, B.T., Dorai, C., Venkatesh, S.: New enhancements to cut, fade, and dissolve detection processes in video segmentation. In: Proceedings of the eighth ACM international conference on Multimedia. pp. 219–227 (2000)
62. 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)
63. Vondrick, C., Shrivastava, A., Fathi, A., Guadarrama, S., Murphy, K.: Tracking emerges by colorizing videos. In: Proc. ECCV (2018)
64. Wang, J., Karaev, N., Rupprecht, C., Novotny, D.: Visual geometry grounded deep structure from motion. Proc. CVPR (2024)

65. Wang, Q., Chang, Y.Y., Cai, R., Li, Z., Hariharan, B., Holynski, A., Snavely, N.: Tracking everything everywhere all at once. In: Proc. ICCV (2023)
66. Wang, X., Gupta, A.: Unsupervised learning of visual representations using videos. In: Proc. ICCV (2015)
67. Wang, X., Jabri, A., Efros, A.A.: Learning correspondence from the cycle-consistency of time. In: Proc. CVPR (2019)
68. Wang, Y., Yang, Y., Yang, Z., Zhao, L., Wang, P., Xu, W.: Occlusion aware unsupervised learning of optical flow. In: Proc. CVPR (2018)
69. 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)
70. Wiskott, L., Sejnowski, T.J.: Slow feature analysis: Unsupervised learning of invariances. Neural computation **14**(4), 715–770 (2002)
71. Yu, E., Blackburn-Matzen, K., Nguyen, C., Wang, O., Habib Kazi, R., Bousseau, A.: VideoDoodles: Hand-drawn animations on videos with scene-aware canvases. ACM Transactions on Graphics **42**(4), 1–12 (2023)
72. Yu, J.J., Harley, A.W., Derpanis, K.G.: Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In: ECCV 2016 Workshops (2016)
73. Yuan, C., Wen, C., Zhang, T., Gao, Y.: General flow as foundation affordance for scalable robot learning. arXiv preprint arXiv:2401.11439 (2024)
74. Yusoff, Y., Christmas, W.J., Kittler, J.: Video shot cut detection using adaptive thresholding. In: Proc. BMVC (2000)
75. Zheng, Y., Harley, A.W., Shen, B., Wetzstein, G., Guibas, L.J.: PointOdyssey: A large-scale synthetic dataset for long-term point tracking. In: Proc. CVPR (2023)