TAPIR: Tracking Any Point with per-frame Initialization and temporal Refinement

Carl Doersch[*] **Yi Yang**[*] **Mel Vecerik**[*†] **Dilara Gokay**[*] **Ankush Gupta**[*]
Yusuf Aytar[*] **Joao Carreira**[*] **Andrew Zisserman**[*‡]

[*]Google DeepMind [†] University College London
[‡]VGG, Department of Engineering Science, University of Oxford

Abstract

We present a novel model for Tracking Any Point (TAP) that effectively tracks any queried point on any physical surface throughout a video sequence. Our approach employs two stages: (1) a matching stage, which independently locates a suitable candidate point match for the query point on every other frame, and (2) a refinement stage, which updates both the trajectory and query features based on local correlations. The resulting model surpasses all baseline methods by a significant margin on the TAP-Vid benchmark, as demonstrated by an approximate 20% absolute average Jaccard (AJ) improvement on DAVIS. Our model facilitates fast inference on long and high-resolution video sequences. On a modern GPU, our implementation has the capacity to track points faster than real-time. Given the high-quality trajectories extracted from a large dataset, we demonstrate a proof-of-concept diffusion model which generates trajectories from static images, enabling plausible animations. Visualizations, source code, and pretrained models can be found at `https://deepmind-tapir.github.io`.

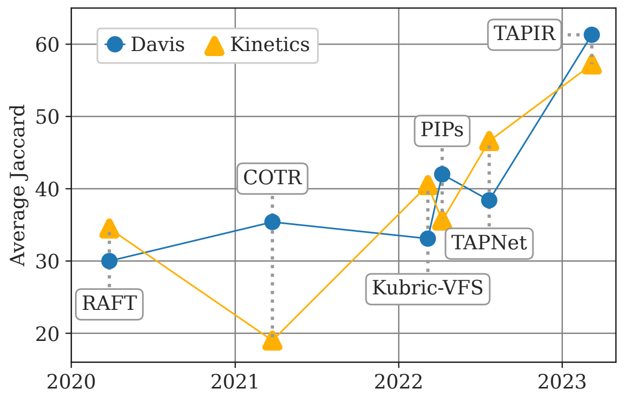

Figure 1. Retrospective evolution of point tracking performance over time on the recent TAP-Vid-Kinetics and TAP-Vid-DAVIS benchmarks, as measured by Average Jaccard (higher is better). In this paper we introduce TAPIR, which significantly improves performance over the state-of-the-art. This unlocks new capabilities, which we demonstrate on motion-based future prediction.

1. Introduction

The problem of point level correspondence —i.e., determining whether two pixels in two different images are projections of the same point on the same physical surface— has long been a fundamental challenge in computer vision, with enormous potential for providing insights about physical properties and 3D shape. We consider its formulation as "Tracking Any Point" (TAP) [12]: given a video and (potentially dense) query points on solid surfaces, an algorithm should reliably output the locations those points correspond to in every other frame where they are visible, and indicate frames where they are not – see Fig. 4 for illustration.

Our main contribution is a new model: TAP with per-frame Initialization and temporal Refinement (TAPIR), which greatly improves over the state-of-the-art on the recently-proposed TAP-Vid benchmark [12]. There are many challenges to TAP: we must robustly estimate occlusions and recover when points reappear (unlike optical flow and structure-from-motion keypoints), meaning that *search* must be incorporated; yet when points remain visible for many consecutive frames, it is important to integrate information about appearance and motion across many of those frames in order to optimally predict positions. Furthermore, little real-world ground truth is available to learn from, so supervised-learning algorithms need to learn from synthetic data without overfitting to the data distribution (i.e., sim2real).

There are three core design decisions that define TAPIR. The first is to use a coarse-to-fine approach. This approach has been used across many high-precision estimation problems in computer vision [7, 28, 32, 38, 41, 58, 69, 73, 78]. For our application, the initial 'coarse' tracking consists of an occlusion-robust matching performed separately on every frame, where tracks are hypothesized using low-resolution features, without enforcing temporal continuity. The 'fine' refinement iteratively uses local, spatio-temporal information at a higher resolution, wherein a neural network can trade-off smoothness of motion with local appearance cues to produce the most likely track. The second design decision is to be fully-convolutional in time: the layers of our neural network consist principally of feature comparisons, spatial convolutions, and temporal convolutions, resulting in a model which efficiently maps onto modern GPU and TPU hardware. The third design decision is that the model should estimate its own uncertainty with regard to its position estimate, in a self-supervised manner. This ensures that low-confidence predictions can be suppressed, which improves benchmark scores. We hypothesize that this may help downstream algorithms (e.g. structure-from-motion) that rely on precision, and can benefit when low-quality matches are removed.

We find that two existing architectures already have some of the pieces we need: TAP-Net [12] and Persistent Independent Particles (PIPs) [19]. Therefore, a key contribution of our work is to effectively combine them while achieving the benefits from both. TAP-Net performs a global search on every frame independently, providing a coarse track that is robust to occlusions. However, it does not make use of the continuity of videos, resulting in jittery, unrealistic tracks. PIPs, meanwhile, gives a recipe for refinement: given an initialization, it searches over a local neighborhood and smooths the track over time. However, PIPs processes videos sequentially in chunks, initializing each chunk with the output from the last. The procedure struggles with occlusion and is difficult to parallelize, resulting in slow processing (i.e., 1 month to evaluate TAP-Vid-Kinetics on 1 GPU). A key contribution of this work is observing the complementarity of these two methods.

As shown in Fig. 1, we find that TAPIR improves over prior works by a large margin, as measured by performance on the TAP-Vid benchmark [12]. On TAP-Vid-DAVIS, TAPIR outperforms TAP-Net by $\sim$20% while on the more challenging TAP-Vid-Kinetics, TAPIR outperforms PIPs by $\sim$20%, and substantially reduces its inference runtime. To demonstrate the quality of TAPIR trajectories, we showcase a proof-of-concept model trained to *generate* trajectories given individual images, and find that this model can generate plausible animations from single photographs.

In summary, our contributions are as follows: 1) We propose a new model for long term point tracking, bridging the gap between TAP-Net and PIPs. 2) We show that the model achieves state-of-the-art results on the challenging TAP-Vid benchmark, with a significant boost on performance. 3) We provide an extensive analysis of the architectural decisions that matter for high-performance point tracking. 4) We provide a proof-of-concept of video prediction enabled by TAPIR's high-quality trajectories. Finally, 5) after analyzing components, we separately perform careful tuning of hyperparameters across entire method, in order to develop the best-performing model, which we release at `https://www.github.com/deepmind/tapnet` for the benefit of the community.

2. Related Work

Physical point correspondence and tracking have been studied in a wide range of settings.

Optical Flow focuses on dense motion estimation between image pairs [22, 36]; methods can be divided into classical variational approaches [4, 5] and deep learning approaches [14, 23, 47, 60, 62, 71]. Optical flow operates only on subsequent frames, with no simple method to track across long videos. Groundtruth is hard to obtain [6], so it is typically benchmarked on synthetic scenes [6, 14, 40, 59], though limited real data exists through depth scanners [15].

Keypoint Correspondence methods aim at sparse keypoint matching given image pairs, from hand-defined discriminative feature descriptors [3, 34, 35, 54] to deep learning based methods [11, 27, 37, 43]. Though it operates on image pairs, it is arguably "long term" as the images may be taken at different times from wide baselines. The goal, however, is typically not to track *any* point, but to find easily-trackable "keypoints" sufficient for reconstruction. They are mainly used in structure-from-motion (SfM) settings and typically ignore occlusion, as SfM can use geometry to filter errors [20, 52, 65, 66]. This line of work is thus often restricted to rigid scenes [39, 75], and is benchmarked accordingly [1, 10, 29, 30, 53, 76].

Semantic Keypoint Tracking is often represented as keypoints or landmarks [26, 45, 56, 61, 64, 70]. Landmarks are often "joints", which can have vastly different appearance depending on pose (i.e. viewed from the front or back); thus most methods rely on large supervised datasets [2, 9, 48, 55, 72, 77], although some works track surface meshes [18, 67]. One interesting exception uses sim2real based on motion [13], motivating better surface tracking. Finally, some recent methods discover keypoints on moving objects [24, 25, 63, 74], though these typically require in-domain training on large datasets.

Long-term physical point tracking is addressed in a few early works [33, 49, 51, 57, 68], though such hand-engineered methods have fallen out of favor in the deep learning era. Our work instead builds on two recent works, TAP-Net [12] and Persistent Independent Particles (PIPs) [19], which aim to update these prior works to the deep learning era.

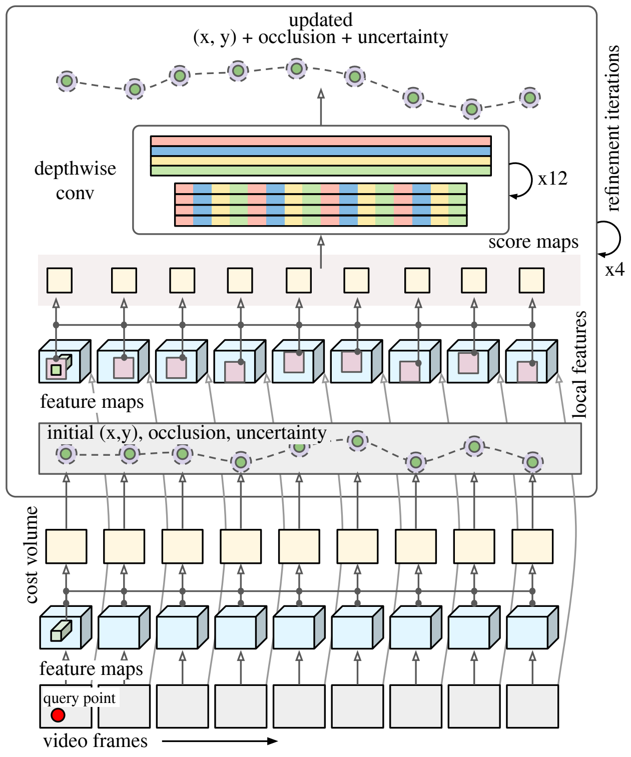

Figure 2. **TAPIR architecture summary.** Our model begins with a global comparison between the query point features and the features for every other frame to compute an initial track estimate, including an uncertainty estimate. Then, we extract features from a local neighborhood (shown in pink) around the initial estimate, and compare these to the query feature at a higher resolution, post-processing the similarities with a temporal depthwise-convolutional network to get an updated position estimate. This updated position is fed back into the next iteration of refinement, repeated for a fixed number of iterations. Note that for simplicity, multi-scale pyramids are not shown.

3. TAPIR Model

Given a video and a query point, our goal is to estimate the 2D location p_t that it corresponds to in every other frame t, as well as a 1D probability o_t that the point is occluded and a 1D probability u_t on the uncertainty of the estimated location. To be robust to occlusion, we first *match* candidate locations in other frames, by comparing the query features with all other features, and post-processing the similarities to arrive at an initial estimate. Then we refine the estimate, based on the *local* similarities between the query point and the target point. Note that both stages depend mostly on *similarities* (dot products) between the query point features and features elsewhere (i.e. not on the feature alone); this ensures that we don't overfit to specific features that might only appear in the (synthetic) training data.

Fig. 2 gives an overview of our model. We initialize an estimate of the track by finding the best match within each

	TAP-Net [12]	PIPs [19]	TAPIR
Per-frame Initialization	✓	✗	✓
Fully Convolutional In Time	✓	✗	✓
Temporal Refinement	✗	✓	✓
Multi Scale Feature	✗	✓	✓
Stride-4 Feature	✗	test-time only	✓
Parallel Inference	✓	✗	✓
Uncertainty Estimate	✗	✗	✓
Number of Parameters	2.8M	28.7M	29.3M

Table 1. Overview of models. TAPIR combines the merits from both TAP-Net and PIPs and further adds a handful of crucial components to improve performance.

frame independently, ignoring the temporal nature of the video. This involves a *cost volume*: for each frame t, we compute the dot product between the query feature F_q and all features in the t'th frame. We post-process the cost volume with a ConvNet, which produces a spatial heatmap, which is then summarized to a point estimate, in a manner broadly similar to TAP-Net. Given the initialization, we then compare the query feature to all features in a small region around an initial track. We feed the comparisons into a neural network, which updates both the query features and the track. We iteratively apply multiple such updates to arrive at a final trajectory estimate. TAPIR accomplishes this with a fully-convolutional architecture, operating on a pyramid of both high-dimensional, low-resolution features and low-dimensional, high-resolution features, in order to maximize efficiency on modern hardware. Finally, we add an estimate of uncertainty with respect to position throughout the architecture in order to suppress low-accuracy predictions. The rest of this section describes TAPIR in detail.

3.1. Track Initialization

The initial cost volume is computed using a relatively coarse feature map $F \in \mathbb{R}^{T\times H/8\times W/8\times C}$, where T is the number of frames, H and W are the image height and width, and C is the number of channels, computed with a standard TSM-ResNet-18 [31] backbone. Features for the query point F_q are extracted via bilinear interpolation at the query location, and we perform a dot product between this query feature and all other features in the video.

We compute an initial estimate of the position $p_t^0 = (x_t^0, y_t^0)$ and occlusion o_t^0 by applying a small ConvNet to the cost volume corresponding to frame t (which is of shape $H/8\times W/8\times 1$ per query). This ConvNet has two outputs: a heatmap of shape $H/8\times W/8\times 1$ for predicting the position, and a single scalar logit for the occlusion estimate, obtained via average pooling followed by projection. The heatmap is converted to a position estimate by a "spatial soft argmax", i.e., a softmax across space (to make the heatmap positive and sum to 1). Afterwards, all heatmap values which are too far from the argmax position in this heatmap are set to zero. Then, the positions for each cell of the heatmap are averaged spatially, weighted by the thresholded heatmap magnitudes. Thus, the output is typically a location close to the heatmap's maximum; the "soft argmax" is differentiable, but the thresholding suppresses spurious matches, and prevents the network from "averaging" between multiple matches. This can serve as a good initialization, although the model struggles to localize points to less than a few pixels' accuracy on the original resolution due to the 8-pixel stride of the heatmap.

Position Uncertainty Estimates. A drawback of predicting occlusion and position independently from the cost volume (a choice that is inspired by TAP-Net) is that if a point is visible in the ground truth, it's *worse* if the algorithm predicts a vastly incorrect location than if it simply incorrectly marks the point as occluded. After all, downstream applications may want to use the track to understand object motion; such downstream pipelines must be robust to occlusion, but may assume that the tracks are otherwise correct. The Average Jaccard metric reflects this: predictions in the wrong location are counted as *both* a "false positive" and a "false negative." This kind of error tends to happen when the algorithm is uncertain about the position, e.g., if there are many potential matches. Therefore, we find it's beneficial for the algorithm to also estimate its own certainty regarding position. We quantify this by making the occlusion pathway output a second logit, estimating the probability u_t^0 that the prediction is likely to be far enough from the ground truth that it isn't useful, even if the model predicts that it's visible. We define "useful" as whether the prediction is within a threshold δ of the ground truth.

Therefore, our loss for the initialization for a given video frame t is $\mathcal{L}(p_t^0, o_t^0, u_t^0)$, where $\mathcal{L}$ is defined as:

$$\begin{aligned} \mathcal{L}(p_t, o_t, u_t) = & \text{Huber}(\hat{p}_t, p_t) * (1 - \hat{o}_t) \\ & + \text{BCE}(\hat{o}_t, o_t) \\ & + \text{BCE}(\hat{u}_t, u_t) * (1 - \hat{o}_t) \qquad (1) \\ \text{where,} \quad \hat{u}_t = & \begin{cases} 1 & \text{if } d(\hat{p}_t, p_t) > \delta \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Here, $\hat{o}_t \in \{0, 1\}$ and $\hat{p} \in \mathbb{R}^2$ are the ground truth occlusion and point locations respectively, d is Euclidean distance, δ is the distance threshold, Huber is a Huber loss, and BCE is binary cross entropy (both o_t and u_t have sigmoids applied to convert them to probabilities). $\hat{u}_t$, the target for the uncertainty estimate u_t, is computed from the ground truth position and the network's prediction: it's 0 if the model's position prediction is close enough to the ground truth (within the threshold δ), and 1 otherwise. At test time, the algorithm should output that the point is visible if it's both predicted as unoccluded and if the model is confident in the prediction. We therefore do a soft combination of the two probabilities: the algorithm outputs that the point is visible if $(1 - u_t) * (1 - o_t) > 0.5$.

3.2. Iterative Refinement

Given an estimated position, occlusion, and uncertainty for each frame, the goal of each iteration i of our refinement procedure is to compute an update $(\Delta p_t^i, \Delta o_t^i, \Delta u_t^i)$, which adjusts the estimate to be closer to the ground truth, integrating information across time. The update is based on a set of "local score maps" which capture the query point similarity (i.e. dot products) to the features in the neighborhood of the trajectory. These are computed on a pyramid of different resolutions, so for a given trajectory, they have shape $(H' \times W' \times L)$, where $H' = W' = 7$, the size of the local neighborhood, and L is the number of levels of the spatial pyramid (different pyramid levels are computed by spatially pooling the feature volume F). As with the initialization, this set of similarities is post-processed with a network to predict the refined position, occlusion, and uncertainty estimate. Unlike the initialization, however, we include "local score maps" for many frames simultaneously as input to the post-processing. We include the current position estimate, the raw query features, and the (flattened) local score maps into a tensor of shape $T \times (C + K + 4)$, where C is the number of channels in the query feature, $K = H' \cdot W' \cdot L$ is the number of values in the flattened local score map, and 4 extra dimensions for position, occlusion, and uncertainty. The output of this network at the i'th iteration is a residual $(\Delta p_t^i, \Delta o_t^i, \Delta u_t^i, \Delta F_{q,t,i})$, which is added to the position, occlusion, uncertainty estimate, and the query feature, respectively. $\Delta F_{q,t,i}$ is of shape $T \times C$; thus, after the first iteration, slightly different "query features" are used on each frame when computing new local score maps.

These positions and score maps are fed into a 12-block convolutional network to compute an update $(\Delta p_t^i, \Delta o_t^i, \Delta u_t^i, \Delta F_{q,t,i})$, where each block consists of a 1×1 convolution block and a depthwise convolution block. This architecture is inspired by the MLP-Mixer used for refinement in PIPs: we directly translate the Mixer's cross-channel layers into 1×1 convolutions with the same number of channels, and the within-channel operations become similarly within-channel depthwise convolutions. Unlike PIPs, which breaks sequences into 8-frame chunks before running the MLP-Mixer, this convolutional architecture can be run on sequences of any length.

We found that the feature maps used for computing the "score maps" are important for achieving high precision. The pyramid levels $l = 1$ through $L - 1$ are computed by spatially average-pooling the raw TSM-ResNet features F at a stride of $8 \cdot 2^{l-1}$, and then computing all dot products between the query feature F_q and the pyramid features. For the t'th frame, we extract a 7×7 patch of dot products centered at p_t. At training time, PIPs leaves it here, but at test time alters the backbone to run at stride 4, introducing a train/test domain gap. We find it is effective to train on stride 4 as well, although this puts pressure on system memory. To alleviate this, we compute a 0-th score map on the 64-channel, stride-4 input convolution map of the TSM-ResNet, and extract a comparable feature for the query from this feature map via bilinear interpolation. Thus, the final set of local score maps has shape $(7 \cdot 7 \cdot L)$.

After $(\Delta p_t^i, \Delta o_t^i, \Delta u_t^i, \Delta F_{q,t,i})$ is obtained from the above architecture, we can iterate the refinement step as many times as desired, re-using the same network parameters. At each iteration, we use the same loss $\mathcal{L}$ on the output, weighting each iteration the same as the initialization.

Discussion As there is some overlap with prior work, Figure 3 depicts the relationship between TAPIR and other architectures. For the sake of ablations, we also develop a "simple combination" model, which is intended to be a simpler combination of a TAP-Net-style initialization with PIPs-style refinements, and follows prior architectural decisions regardless of whether they are optimal. To summarize, the major architectural decisions needed to create the "simple combination" of TAP-Net and PIPs (and therefore the contributions of this model) are as follows. First we exploit the complementarity between TAP-Net and PIPs. Second, we remove "chaining" from PIPs, replacing that initialization with TAP-Net's initialization. We directly apply PIPs' MLP-Mixer architecture by 'chunking' the input sequence. Note that we adopt TAP-Net's feature network for both initialization and refinement. Finally, we adapt the TAP-Net backbone to extract a multi-scale and high resolution pyramid (follow PIPs stride-4 test-time architecture, but matching TAP-Net's end-to-end training). We apply losses at all layers of the hierarchy (whereas TAP-Net has a single loss, PIPs has no loss on its initialization). The additional contributions of TAPIR are as follows: first, we entirely replace the PIPs MLP-Mixer (which must be applied to fixed-length sequences) with a novel depthwise-convolutional architecture, which has a similar number of parameters but works on sequences of any length. Second, we introduce uncertainty estimate, which are computed at every level of the hierarchy. These are 'self-supervised' in the sense that the targets depend on the model's own output.

3.3. Training Dataset

Although TAPIR is designed to be robust to the sim2real gap, we can improve performance by minimizing the gap itself. One subtle difference between the Kubric MOVi-E [16] dataset and the real world is the lack of *panning*: although the Kubric camera moves, it is always set to "look at" a single point at the center of the workspace. Thus, stationary parts of the scene rotate around the center point of the workspace on the 2D image plane. Models thus tend to fail on real-world videos with panning. Therefore, we modified the MOVi-E dataset so that the "look at" point moves along a random linear trajectory. The start and end points of this trajectory are enforced to be less than 1.5 units off the ground and lie within a radius of 4 units from the workspace center; the trajectory must also pass within 1 unit of the workspace center at some point, to ensure that the camera is usually looking toward the objects in the scene. See Appendix 5.2 for details.

4. Experiments

We evaluate on the TAP-Vid [12] benchmark, a recent large-scale benchmark evaluating the problem of interest. TAP-Vid consists of point track annotations on four different datasets, each with different challenges. DAVIS [44] is a benchmark designed for tracking, and includes complex motion and large changes in object scale. Kinetics [8] contains over 1000 labeled videos, and contains all the complexities of YouTube videos, including difficulties like cuts and camera shake. RGB Stacking is a synthetic dataset of robotics videos with precise point tracks on large textureless regions. Kubric MOVi-E, used for training, contains realistic renderings of objects falling onto a flat surface; it comes with a validation set that we use for comparisons. We conduct training exclusively on the Kubric dataset, and selected our best model primarily by observing DAVIS. We performed no automated evaluation for hyperparameter tuning or model selection on any dataset. We train and evaluate at a resolution of 256×256.

Implementation Details We train for 50,000 steps with a cosine learning rate schedule, with a peak rate of 1×10^{-3} and 1,000 warmup steps. We use 64 TPU-v3 cores, using 4 24-frame videos per core in each step. We use an Adam-W optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.95$ and weight decay 1×10^{-2}. For most experiments, we use cross-replica batch normalization within the (TSM-)ResNet backbone only, although for our public model we find instance normalization is actually more effective. For most experiments we use $L = 5$ to match PIPs, but in practice we find $L > 3$ has little impact (See Appendix 4.3). Therefore, to save computation, our public model uses $L = 3$. See Appendix 5.1 for more details.

4.1. Quantitative Comparison

Table 2 shows the performance of TAPIR relative to prior works. TAPIR improves by 10.6% absolute performance over the best model on Kinetics (TAP-Net) and by 19.3% on DAVIS (PIPs). These improvements are quite significant according to the TAP-Vid metrics: Average Jaccard consists of 5 thresholds, so under perfect occlusion estimation, a 20% improvement requires improving *all* points to the next distance threshold (half the distance). It's worth noting

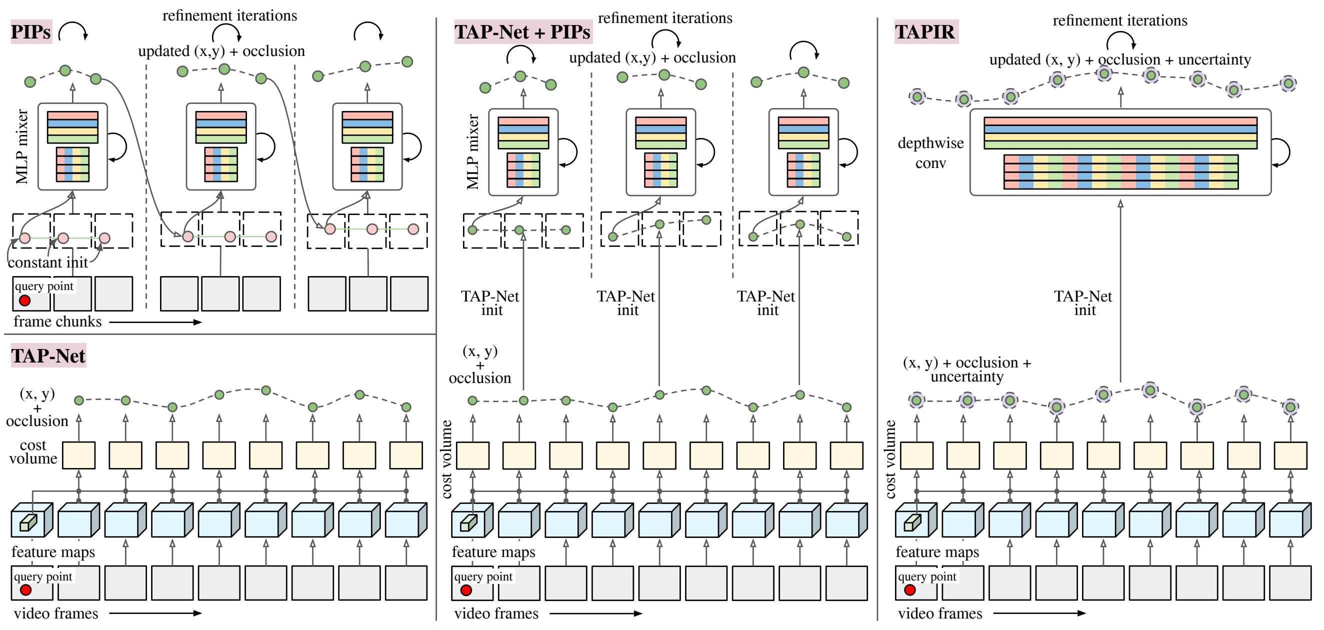

Figure 3. **Comparison of Architectures.** Left: the initial TAP-Net and PIPs models: TAP-Net has an independent estimate per frame, whereas PIPs breaks the video into fixed-size chunks and processes chunks sequentially. Middle: a "simple combination" of these two architectures, where TAP-Net is used to initialize PIPs-style chunked iterations. Right: Our model, TAPIR, which removes the chunking and adds uncertainty estimate. Note that for simplicity, multi-scale pyramids are not shown.

	Kinetics			Kubric			DAVIS			RGB-Stacking		
Method	AJ	$< \delta^x_{avg}$	OA	AJ	$< \delta^x_{avg}$	OA	AJ	$< \delta^x_{avg}$	OA	AJ	$< \delta^x_{avg}$	OA
COTR [27]	19.0	38.8	57.4	40.1	60.7	78.55	35.4	51.3	80.2	6.8	13.5	79.1
Kubric-VFS-Like [16]	40.5	59.0	80.0	51.9	69.8	84.6	33.1	48.5	79.4	57.9	72.6	91.9
RAFT [62]	34.5	52.5	79.7	41.2	58.2	86.4	30.0	46.3	79.6	44.0	58.6	90.4
PIPs [19]	35.3	54.8	77.4	59.1	74.8	88.6	42.0	59.4	82.1	37.3	51.0	91.6
TAP-Net [12]	46.6	60.9	85.0	65.4	77.7	93.0	38.4	53.1	82.3	59.9	72.8	90.4
TAPIR (MOVi-E)	57.1	70.0	87.6	84.3	91.8	95.8	59.8	72.3	87.6	**66.2**	**77.4**	**93.3**
TAPIR (Panning MOVi-E)	**57.2**	**70.1**	**87.8**	**84.7**	**92.1**	**95.8**	**61.3**	**73.6**	**88.8**	62.7	74.6	91.6

Table 2. **Comparison of TAPIR to prior results on TAP-Vid**. $< \delta^x_{avg}$ is the fraction of points unoccluded in the ground truth, where the prediction is within a threshold, ignoring the occlusion estimate; OA is the accuracy of predicting occlusion. AJ is Average Jaccard, which considers both position and occlusion accuracy.

that, like PIPs, TAPIR's performance on DAVIS is slightly higher than on Kinetics, whereas for TAP-Net, Kinetics has higher performance, suggesting that TAPIR's reliance on temporal continuity isn't as useful on Kinetics videos with cuts or camera shake. Note that the bulk of the improvement comes from the improved model rather than the improved dataset (TAPIR (MOVi-E) is trained on the same data as TAP-Net, while 'Panning' is the new dataset). The dataset's contribution is small, mostly improving DAVIS, but we found qualitatively that it makes the most substantial difference on backgrounds when cameras pan. This is not well represented in TAP-Vid's query points, but may matter for real-world applications like stabilization. RGB-stacking, however, is somewhat of an outlier: training on MOVi-E is still optimal (unsurprising as the cameras are static), and there the improvement is only 6.3%, possibly because TAPIR does less to help with textureless objects, indicating an area for future research. Fig. 4 shows some example predictions on DAVIS. TAPIR corrects many types of errors, including problems with temporal coherence from TAP-Net and problems with occlusion from PIPs. We include further qualitative results on our project webpage, which illustrate how noticeable a 20% improvement is.

TAP-Vid proposes another evaluation for videos at the resolution of 256 × 256 called 'query-first.' In the typical case (Table 2), the query points are sampled in a 'strided' fashion from the ground-truth tracks: the same trajectory is therefore queried with multiple query points, once every 5 frames. In the 'query first' evaluation, only the first point along each trajectory is used as a query. These points

	Kinetics			DAVIS			RGB-Stacking		
Method	AJ	$< \delta^x_{avg}$	OA	AJ	$< \delta^x_{avg}$	OA	AJ	$< \delta^x_{avg}$	OA
TAP-Net	38.5	54.4	80.6	33.0	48.6	78.8	53.5	68.1	86.3
TAP-Net (Panning MOVi-E)	39.5	56.2	81.4	36.0	52.9	80.1	50.1	65.7	88.3
TAPIR (MOVi-E)	49.6	64.2	**85.2**	55.3	69.4	84.4	**56.2**	**70.0**	**89.3**
TAPIR (Panning MOVi-E)	**49.6**	**64.2**	85.0	**56.2**	**70.0**	**86.5**	55.5	69.7	88.0

Table 3. **Comparison under query first metrics.** We see largely the same relative performance trend whether the model is queried in a ‘strided’ fashion or whether the model is queried with the first frame where the point appeared, although performance is overall lower than the ‘strided’ evaluation.

are slightly harder to track than those in the middle of the video, because there are more intervening frames wherein the point’s appearance may change. Table 3 shows our results on this approach. We see that the performance broadly follows the results from the ‘strided’ evaluation: we outperform TAP-Net by a wide margin on real data (10% absolute on Kinetics, 20% on DAVIS), and by a smaller margin on RGB-Stacking (3%). This suggests that TAPIR remains robust even if the output frames are far from the query point.

We also conduct a comparison of computational time for model inference on the DAVIS video *horsejump-high* with a resolution of 256x256. All models are evaluated on a single GPU using 5 runs on average. With Just In Time (JIT) Compilation with JAX, for 50 query points randomly sampled on the first frame, TAP-Net finishes within 0.1 seconds and TAPIR finishes within 0.3 seconds, i.e., roughly 150 frames per second. This is possible because TAPIR, like TAP-Net, maps efficiently to GPU parallel processing. In contrast, PIPs takes 34.5 seconds due to a linear increase with respect to the number of points and frames processed. See Appendix 1 for more details.

4.2. Ablation Studies

TAPIR vs Simple Combination We analyze the contributions of the main ideas. Our first contribution is to combine PIPs and TAP-Net (Fig. 3, center). Comparing this directly to PIPs is somewhat misleading: running the open-source ‘chaining’ code on Kinetics requires roughly 1 month on a V100 GPU (roughly 30 minutes to process a 10-second clip). But how crucial is chaining? After all, PIPs MLP-mixers have large spatial receptive fields due to their multi-resolution pyramid. Thus, we trained a naïve version of PIPs without chaining: it initializes the entire track to be the query point, and then processes each chunk independently. For a fair comparison, we train using the same dataset and feature extractor, and other hyperparameters as TAPIR, but retain the 8-frame MLP Mixer with the same number of iterations as TAPIR. Table 4 shows that this algorithm (Chain-free PIPs) is surprisingly effective, on par with TAP-Net. Next, we combine the Chain-free PIPs with TAP-Net, replacing the constant initialization with TAP-Net’s prediction. The “simple combination” (Fig. 3, center) alone improves over PIPs, Chain-free PIPs, and TAP-Net (i.e. 43.3% v.s. 50.0% on DAVIS). However, the other new

Average Jaccard (AJ)	Kinetics	DAVIS	RGB-Stacking	Kubric
PIPs [19]	35.3	42.0	37.3	59.1
TAP-Net [12]	48.3	41.1	59.2	65.4
Chain-Free PIPs	47.2	43.3	61.3	75.3
Simple Combination	52.9	50.0	62.7	78.1
TAPIR	**57.2**	**61.3**	**62.7**	**84.7**

Table 4. **PIPs, TAP-Net, and a simple combination vs TAPIR.** We compare TAPIR (Fig. 3 right) to TAP-Net, PIPs (Fig. 3 left), and the “simple combination” (Fig. 3 center). Chain-Free PIPs is our reimplementation of PIPs that removes the extremely slow chaining operation and uses the TAPIR backbone network. All models in this table are trained on Panning MOVi-E except PIPs, which uses its own dataset.

Average Jaccard (AJ)	Kinetics	DAVIS	RGB-Stacking	Kubric
Full Model	**57.2**	**61.3**	**62.7**	**84.7**
- No Depthwise Conv	54.9	53.8	61.9	79.7
- No Uncertainty Estimate	54.4	58.6	61.5	83.4
- No Higher Res Feature	54.0	54.0	59.5	80.4
- No TAP-Net Initialization	54.7	59.3	60.3	84.1
- No Iterative Refinement	48.1	41.6	60.3	64.9

Table 5. **Model ablation by removing one major component at a time.** -No Depthwise Conv uses a MLP-Mixer on a chunked video similar to PIPs. -No Uncertainty Estimate uses the losses directly as described in TAP-Net. -No Higher Res Feature uses only 4 pyramid levels in iterative refinement and has no features at stride 4. -No TAP-Net Initialization uses a constant initialization for iterative refinement. All the components contribute non-trivially to the performance of the model.

ideas in TAPIR also account for a substantial portion of the performance (i.e. 50.0% v.s. 61.3% on DAVIS).

Model Components Table 5 shows the effect of our architectural decisions. First, we consider two novel architectural elements: the depthwise convolution which eliminates chunking from the PIPs model, and the uncertainty estimate. We see for Kinetics, both methods contribute roughly equally to performance, while for DAVIS, the depthwise conv is more important. A possible interpretation is that DAVIS contains more temporal continuity than Kinetics, so chunking is more harmful. Higher-resolution features are also predictably important to performance, especially on DAVIS. Table 5 also considers a few other model components that are not unique to TAPIR. TAP-Net initialization is important, but the model still gives reasonable performance without it. Interestingly, the full model without refinement (i.e. TAP-Net trained on Panning MOVi-E with the uncertainty estimate) performs similarly to TAP-Net trained on Panning MOVi-E (Table 4, second line); one possible interpretation is that TAP-Net is overfitting to its training data, leading to biased uncertainty estimate.

Iterations of Refinement Finally, Table 6 shows how performance depends on the number of refinement iterations. We use the same number of iterations during training and

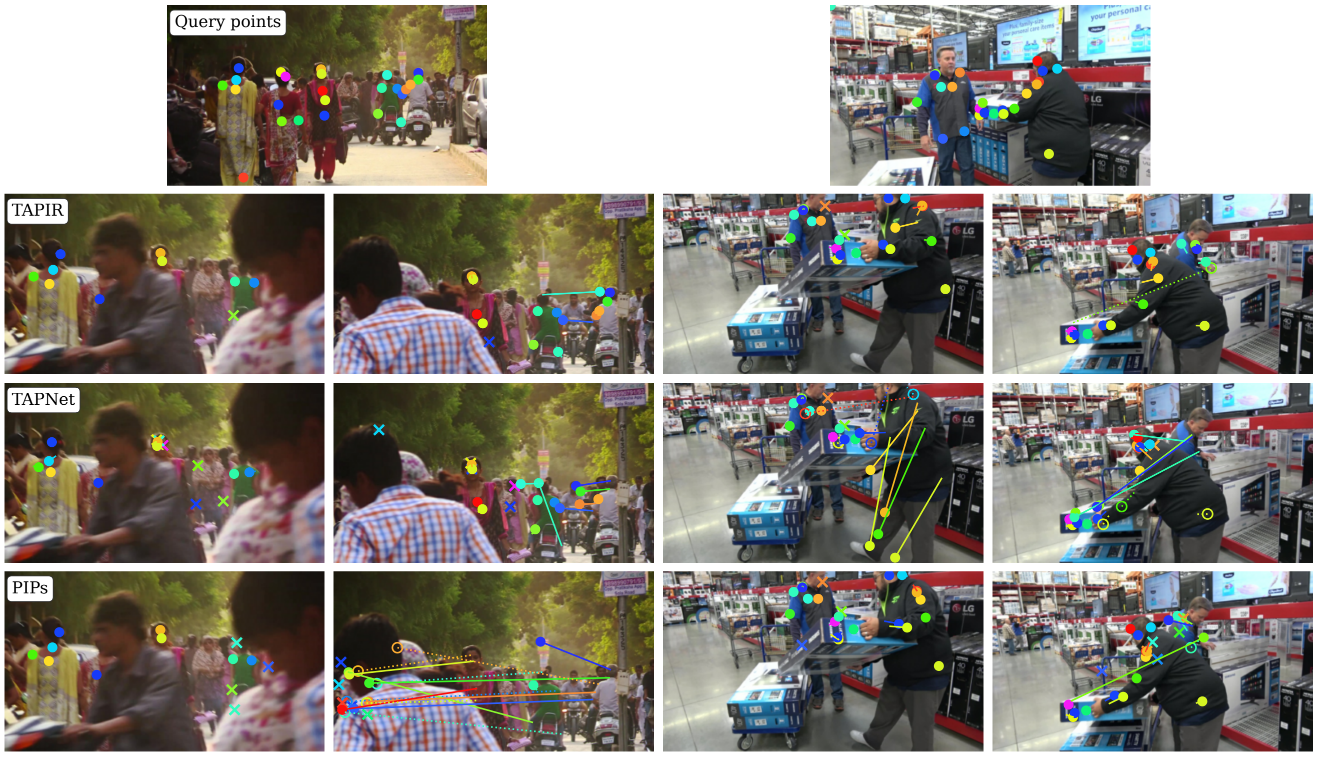

Figure 4. **TAPIR compared to TAP-Net and PIPs.** Top shows the query points on the video's first frame. Predictions for two later frames are below. We show predictions (filled circles) relative to ground truth (GT) (ends of the associated segments). x's indicate predictions where the GT is occluded, while empty circles are points visible in GT but predicted occluded (note: models still predict position regardless of occlusion). The majority of the street scene (left) gets occluded by a pedestrian; as a result, PIPs loses many points. TAP-Net fails on the right video, possibly because the textureless clothing is difficult to match without relying on temporal continuity. TAPIR, meanwhile, has far fewer failures.

Average Jaccard (AJ)	Kinetics	DAVIS	RGB-Stacking	Kubric
0 iterations	48.1	41.6	60.3	64.9
1 iteration	55.7	55.0	62.7	80.1
2 iterations	56.7	58.8	62.0	83.1
3 iterations	56.8	60.6	60.9	84.1
4 iterations	57.2	61.3	**62.7**	**84.7**
5 iterations	**57.8**	**61.6**	61.5	84.3
6 iterations	57.2	60.4	56.6	82.9

Table 6. **Evaluation performance against the number of iterative updates.** It can be observed that after 4 iterations, the performance no longer improves significantly. This could be attributed to the fact that TAP-Net already provides a good initialization.

testing for all models. Interestingly, we observed the best performance for TAPIR at 4-5 iterations, whereas PIPs used 6 iterations. Likely a stronger initialization (i.e., TAP-Net) requires fewer iterations to converge. The decrease in performance with more iterations implies that there may be some degenerate behavior, such as oversmoothing, which may be a topic for future research. In this paper, we use 4 iterations for all other experiments.

Further model ablations, including experiments with RNNs and dense convolutions as replacements for our depthwise architecture, the number of feature pyramid layers, and the importance of time-shifting in the features, can be found in Appendix 4. Depthwise convolutions work as well or better than the alternatives while being less expensive. We find little impact of adding more feature pyramid layers as PIPs does, so we remove them from TAPIR. Finally, we find the impact of time-shifting to be minor.

5. Animating still images with TAPIR

Generative modeling of images and videos has experienced an explosion in popularity due to recent striking text-conditioned generation results using diffusion models [17, 21, 42, 46, 50]. Animating still images remains extremely challenging, due to the high computational cost of video modeling and the difficulty of generating realistic motion. Is it possible to improve the generated videos via an explicit representation of surface motion? Here, we develop a proof-of-concept for a diffusion-based system that uses TAPIR tracks to do this.

We perform video prediction using a pair of diffusion models: one which generates *dense trajectories* given an input image (the "trajectory" model), and a second which takes the input image and the trajectories and generates pixels (the "pixel" model). At test time, the outputs from the trajectory model dictate the motion that should be present in the video, ideally ensuring physical plausibility. The pixel model uses the trajectories to attend to the appropriate location in the first frame, in order to produce appearance which is consistent across the video.

For the trajectory model, we encode the conditioning image with a small ConvNet, which produces a dense grid of features at stride 4. This is used as conditioning for a U-Net that performs diffusion denoising on the trajectories. We find that encoding the trajectories is important: for each frame, we include the raw x and y positions, both in absolute coordinates and coordinates relative to the starting position, and we also include a position encoding of the absolute locations with 32 Fourier features. The U-Net architecture includes self-attention layers, so the Fourier features allow the model to measure whether trajectories are spatially nearby late in an animation. For simplicity and memory efficiency, we stack these values for all frames along the channel axis: thus, the input is a 2D grid of shape $H//4 \times W//4 \times (T \cdot (1+2+2+32))$, (occlusion, relative position, absolute position, fourier encoding), which is processed with a 2D U-Net. The model predicts the clean data given the noisy data, which is the typical "clean data prediction" diffusion setup. The pixel model is a more traditional image generation model, which predicts the noise residual given noisy pixels. It takes three inputs, and outputs a single denoised frame. The first input is 3 noisy frames, allowing the model to infer some temporal context. The second is the first frame, warped according to the trajectories from the trajectory model. The third is features for the first frame, computed by a standard ConvNet before, being warped according to the same trajectories. We train on an internal dataset of 1M video clips, processed into 24-frame clips at 256×256 resolution. See supplementary for more details.

Fig. 5 shows the trajectories produced by our trajectory prediction model on selected images. Our model can make multiple different physically-plausible predictions for motion for each input image, suggesting that it can recognize people and do rudimentary human pose estimation without supervision. Note that this demonstrates two levels of transfer: TAPIR was never trained on humans, while the trajectory-prediction model was trained on video rather than still images like these. It is difficult to judge these motions from a static figure, so we encourage the reader to view videos generated with our pipeline in the supplementary.

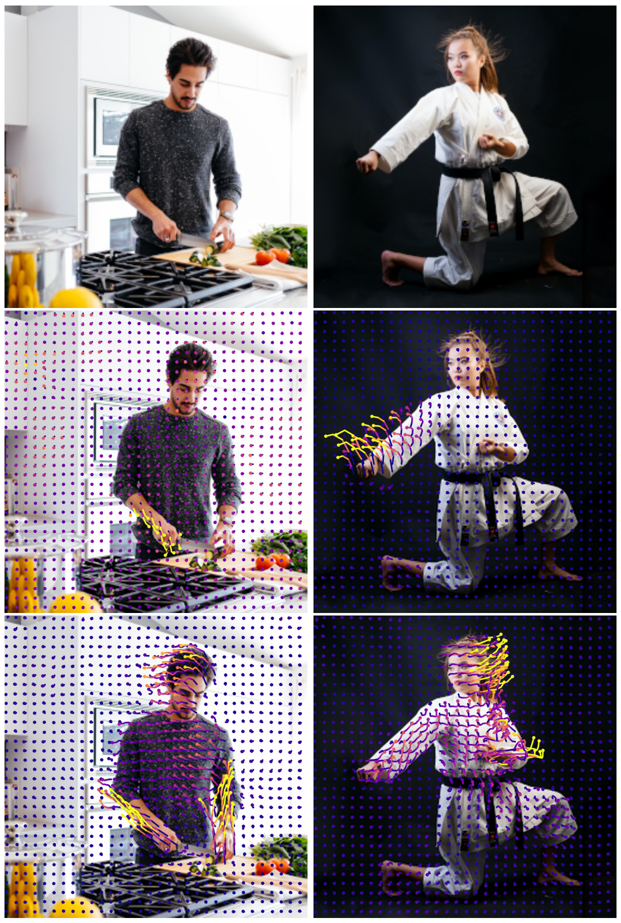
Figure 5. **Animating still frames.** Top row shows the input images. Each following row shows a visualization of a single sample from our trajectory model: the dark purple end is the starting point of the trajectory, while the colors get brighter and yellower with distance. In the left image, one sample shows the hand moving forward in a cutting motion, with a slight incline of the head; in the other, the entire person moves to the left. On the right, the first sample moves one arm in a circle, while the second shows the head turning and the other arm being raised.

6. Conclusion

In this paper, we introduce TAPIR, a novel model for Tracking Any Point (TAP) that adopts a two-stage approach, comprising a matching stage and a refinement stage. It demonstrates a substantial improvement over the state-of-the-art, providing stable predictions and occlusion robustness, and scales to high-resolution videos. We also show a proof-of-concept of animating still frames, which we hope can serve as a foundation for future graphics research. TAPIR still has some limitations, such as difficulty in precisely perceiving the boundary between background and foreground, and it may still struggle with large appearance changes. Nevertheless, our findings indicate a promising future for TAP in computer vision.

Acknowledgements We would like to thank Lucas Smaira and Larisa Markeeva for help with datasets and codebases, and Jon Scholz and Todor Davchev for advice on the needs of robotics. We also thank Adam Harley, Viorica Patraucean and Daniel Zoran for helpful discussions, and Mateusz Malinowski for paper comments. Finally, we thank Nando de Freitas for his support.

References

[1] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5173–5182, 2017.

[2] Marian Stewart Bartlett, Gwen Littlewort, Mark G Frank, Claudia Lainscsek, Ian R Fasel, Javier R Movellan, et al. Automatic recognition of facial actions in spontaneous expressions. *J. Multim.*, 1(6):22–35, 2006.

[3] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 404–417. Springer, 2006.

[4] Thomas Brox, Christoph Bregler, and Jitendra Malik. Large displacement optical flow. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 41–48. IEEE, 2009.

[5] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 25–36. Springer, 2004.

[6] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 611–625. Springer, 2012.

[7] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human pose estimation with iterative error feedback. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4733–4742, 2016.

[8] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6299–6308, 2017.

[9] Zezhou Cheng, Jong-Chyi Su, and Subhransu Maji. On equivariant and invariant learning of object landmark representations. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 9897–9906, 2021.

[10] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5828–5839, 2017.

[11] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 224–236, 2018.

[12] Carl Doersch, Ankush Gupta, Larisa Markeeva, Adrià Recasens, Lucas Smaira, Yusuf Aytar, João Carreira, Andrew Zisserman, and Yi Yang. Tap-vid: A benchmark for tracking any point in a video. *Proceedings of Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[13] Carl Doersch and Andrew Zisserman. Sim2real transfer learning for 3d human pose estimation: motion to the rescue. *Proceedings of Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[14] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766, 2015.

[15] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361. IEEE, 2012.

[16] 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 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[17] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10696–10706, 2022.

[18] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7297–7306, 2018.

[19] Adam W Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. Particle video revisited: Tracking through occlusions using point trajectories. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 59–75. Springer, 2022.

[20] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[21] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv:2204.03458*, 2022.

[22] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial Intelligence*, 17(1-3):185–203, 1981.

[23] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2462–2470, 2017.

[24] Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object landmarks through conditional image generation. *Proceedings of Neural Information Processing Systems (NeurIPS)*, 31, 2018.

[25] Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. Self-supervised learning of interpretable keypoints from unlabelled videos. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8787–8797, 2020.

[26] Hueihan Jhuang, Juergen Gall, Silvia Zuffi, Cordelia

Schmid, and Michael J Black. Towards understanding action recognition. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 3192–3199, 2013.

[27] Wei Jiang, Eduard Trulls, Jan Hosang, Andrea Tagliasacchi, and Kwang Moo Yi. Cotr: Correspondence transformer for matching across images. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 6207–6217, 2021.

[28] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 336–345, 2017.

[29] Zhengqi Li, Tali Dekel, Forrester Cole, Richard Tucker, Noah Snavely, Ce Liu, and William T Freeman. Learning the depths of moving people by watching frozen people. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4521–4530, 2019.

[30] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2041–2050, 2018.

[31] Ji Lin, Chuang Gan, Kuan Wang, and Song Han. Tsm: Temporal shift module for efficient and scalable video understanding on edge devices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[32] Lahav Lipson, Zachary Teed, Ankit Goyal, and Jia Deng. Coupled iterative refinement for 6d multi-object pose estimation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6728–6737, 2022.

[33] Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):978–994, 2010.

[34] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1150–1157. Ieee, 1999.

[35] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[36] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 674–679, 1981.

[37] Lucas Manuelli, Yunzhu Li, Pete Florence, and Russ Tedrake. Keypoints into the future: Self-supervised correspondence in model-based reinforcement learning. *arXiv preprint arXiv:2009.05085*, 2020.

[38] Francesco Marchetti, Federico Becattini, Lorenzo Seidenari, and Alberto Del Bimbo. Mantra: Memory augmented networks for multiple trajectory prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7143–7152, 2020.

[39] David Marr and Tomaso Poggio. A computational theory of human stereo vision. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 204(1156):301–328, 1979.

[40] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4040–4048, 2016.

[41] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*, pages 127–136. Ieee, 2011.

[42] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

[43] Yuki Ono, Eduard Trulls, Pascal Fua, and Kwang Moo Yi. Lf-net: Learning local features from images. *Proceedings of Neural Information Processing Systems (NeurIPS)*, 31, 2018.

[44] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017.

[45] Deva Ramanan, David A Forsyth, and Andrew Zisserman. Strike a pose: Tracking people by finding stylized poses. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 271–278. IEEE, 2005.

[46] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[47] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4161–4170, 2017.

[48] Yu Rong, Jingbo Wang, Ziwei Liu, and Chen Change Loy. Monocular 3d reconstruction of interacting hands via collision-aware factorized refinements. In *2021 International Conference on 3D Vision (3DV)*, pages 432–441. IEEE, 2021.

[49] Michael Rubinstein, Ce Liu, and William Freeman. Towards longer long-range motion trajectories. In *Proceedings of British Machine Vision Conference*, 2012.

[50] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

[51] Peter Sand and Seth Teller. Particle video: Long-range motion estimation using point trajectories. *International Journal of Computer Vision*, 80(1):72–91, 2008.

[52] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016.

[53] Thomas Schops, Johannes L Schonberger, Silvano Galliani,

Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3260–3269, 2017.

[54] Ishwar K Sethi and Ramesh Jain. Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (1):56–73, 1987.

[55] Jie Shen, Stefanos Zafeiriou, Grigoris G Chrysos, Jean Kossaifi, Georgios Tzimiropoulos, and Maja Pantic. The first facial landmark tracking in-the-wild challenge: Benchmark and results. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, pages 50–58, 2015.

[56] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2107–2116, 2017.

[57] Josef Sivic, Frederik Schaffalitzky, and Andrew Zisserman. Object level grouping for video shots. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2004.

[58] Yongzhi Su, Mahdi Saleh, Torben Fetzer, Jason Rambach, Nassir Navab, Benjamin Busam, Didier Stricker, and Federico Tombari. Zebrapose: Coarse to fine surface encoding for 6dof object pose estimation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6738–6748, 2022.

[59] 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. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10093–10102, 2021.

[60] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8934–8943, 2018.

[61] Yi Sun, Yuheng Chen, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation by joint identification-verification. *Proceedings of Neural Information Processing Systems (NeurIPS)*, 27, 2014.

[62] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 402–419. Springer, 2020.

[63] James Thewlis, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object landmarks by factorized spatial embeddings. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 5916–5925, 2017.

[64] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics (ToG)*, 33(5):1–10, 2014.

[65] Philip HS Torr and Andrew Zisserman. Feature based methods for structure and motion estimation. In *International workshop on vision algorithms*, pages 278–294. Springer, 1999.

[66] Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, and Katerina Fragkiadaki. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*, 2017.

[67] Timo von Marcard, Roberto Henschel, Michael J Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 601–617, 2018.

[68] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 3551–3558, 2013.

[69] Yue Wang and Justin M Solomon. Prnet: Self-supervised learning for partial-to-partial registration. *Proceedings of Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[70] Xuehan Xiong and Fernando De la Torre. Supervised descent method and its applications to face alignment. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 532–539, 2013.

[71] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, and Dacheng Tao. Gmflow: Learning optical flow via global matching. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8121–8130, 2022.

[72] Jiarui Xu and Xiaolong Wang. Rethinking self-supervised correspondence learning: A video frame-level similarity perspective. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 10075–10085, 2021.

[73] Zhichao Yin, Trevor Darrell, and Fisher Yu. Hierarchical discrete distribution decomposition for match density estimation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6044–6053, 2019.

[74] Yuting Zhang, Yijie Guo, Yixin Jin, Yijun Luo, Zhiyuan He, and Honglak Lee. Unsupervised discovery of object landmarks as structural representations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2694–2703, 2018.

[75] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[76] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

[77] Christian Zimmermann, Duygu Ceylan, Jimei Yang, Bryan Russell, Max Argus, and Thomas Brox. Freihand: A dataset for markerless capture of hand pose and shape from single rgb images. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2019.

[78] Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rehmann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, et al. Real-time non-rigid reconstruction using an rgb-d camera. *ACM Transactions on Graphics (ToG)*, 33(4):1–12, 2014.