Supplementary Material for TAP-Vid: A Benchmark for Tracking Any Point in a Video

Carl Doersch[*] **Ankush Gupta**[*] **Larisa Markeeva**[*] **Adrià Recasens**[*]
Lucas Smaira[*] **Yusuf Aytar**[*] **João Carreira**[*] **Andrew Zisserman**[*†] **Yi Yang**[*]

[*]DeepMind [†]VGG, Department of Engineering Science, University of Oxford

1 Dataset Examples

To provide more context and information to the reader, we expand Figure 2 in the paper providing many examples per dataset. We provide examples of TAP-Vid-Kinetics in Figures 1 and 2, TAP-Vid-Davis in Figure 3 and TAP-Vid-Kubric and TAP-Vid-RGB-Stacking in Figure 4. These figures illustrate the diversity of the tracked points across all the different datasets. Figures 1, 2 and 3 show how we are tracking points beyond the most salient objects in the video; the vast majority of tracked points do not correspond to the semantic keypoints of existing datasets. The website contains even more examples in video format, under the 'Visualized Ground Truth Examples' heading.

2 Supplementary Videos, Code, and Website

The website for this project `https://storage.googleapis.com/dm-tapnet/index.html` contains the following:

1. **Ground Truth Point Tracking Annotations.** Video examples of groundtruth point annotations in our datasets, as summarized in Section 1 above.
2. **Dataset Download Links and Processing.** Download links for the TAP-Vid-{Kinetics, DAVIS, RGB-Stacking} datasets, and instructions for processing and aligning raw Kinetics videos to the annotations and for running the visualization scripts. It also includes licensing information.
3. **Improving Human Point Annotation with Flow-Based Tracker.** Comparison of human point annotation quality on the DAVIS dataset with and without optical flow track assistance.

The `index.html` included with this supplementary file contains some supplementary information which is not included in the TAP-Vid webpage as we do not consider it a part of the dataset:

1. **DAVIS/RGB-Stacking Point Tracks Prediction.** TAP-Net point tracking predictions on videos from the DAVIS point tracking benchmark and from the RGB-Stacking robotics benchmark.
2. **Videos used for Quantitative Evaluation of the Flow-Based Tracker.** For reference, we include the exact 10 videos that annotators labeled before their annotations were compared with Kubric ground truth.

3 Annotation Instructions and Workflow

Figure 5 summarizes the guidelines given to the annotators to encourage acquisition of high-quality point tracks. Note, the guidelines do not limit the scope of the kind or type of points the annotators

36th Conference on Neural Information Processing Systems (NeurIPS 2022) Track on Datasets and Benchmarks.

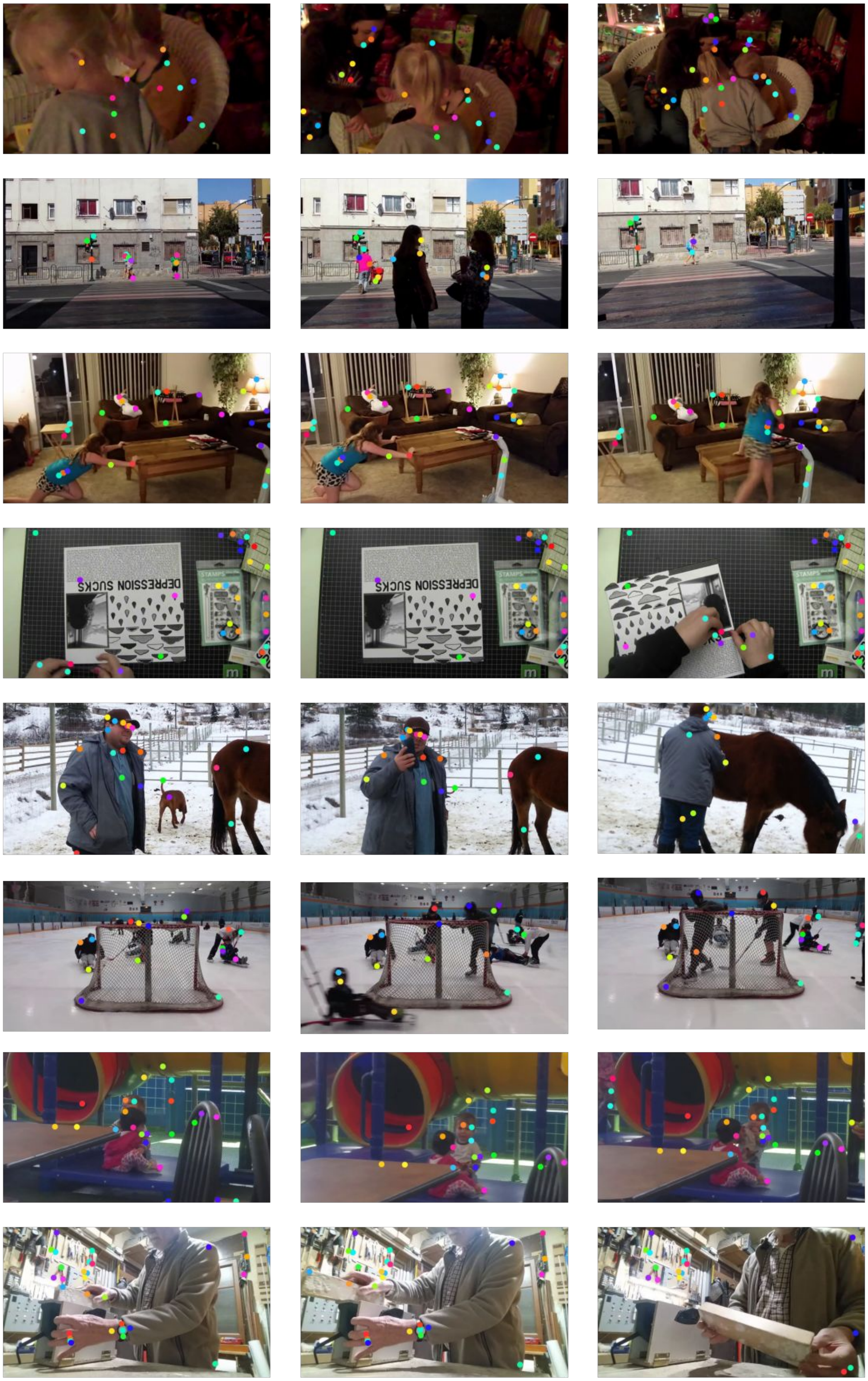


Figure 1: **Examples from TAP-Vid-Kinetics**: In this figure we show a few examples of TAP-Vid-Kinetics annotations with 3 frames per example.

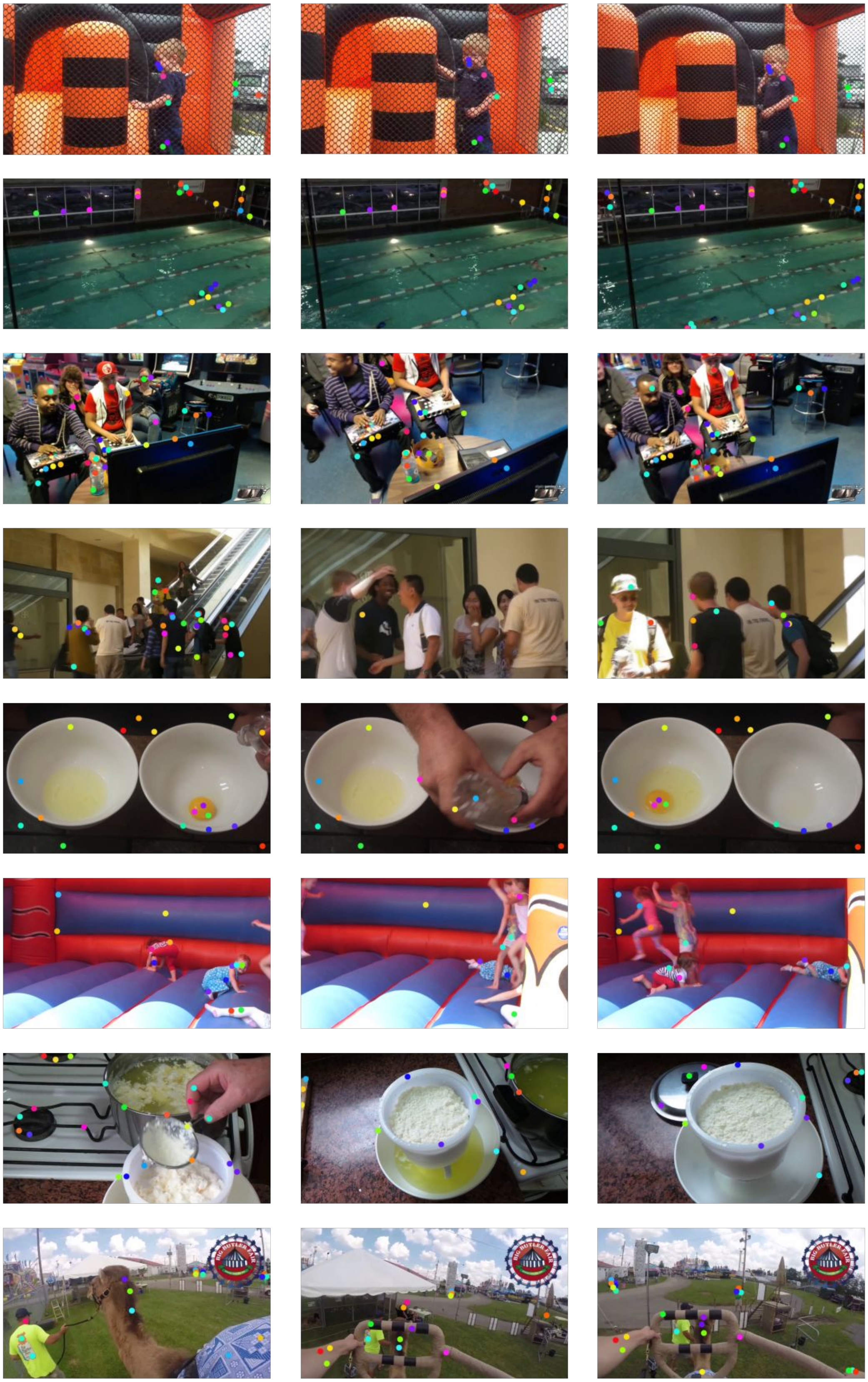


Figure 2: **Examples from TAP-Vid-Kinetics**: In this figure we show a few examples of TAP-Vid-Kinetics annotations with 3 frames per example.

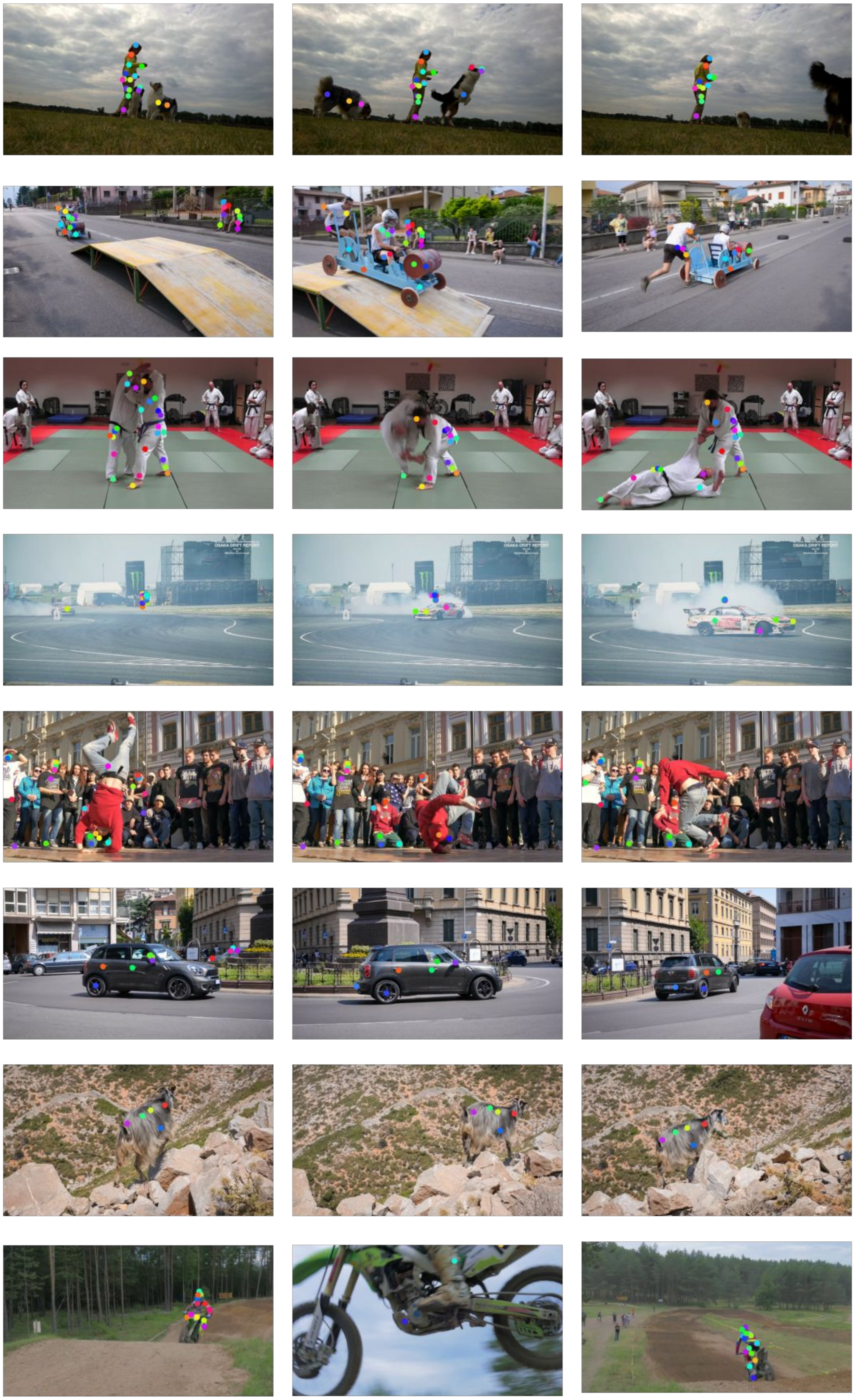


Figure 3: **Examples from TAP-Vid-Davis**: In this figure we show examples of TAP-Vid-Davis annotations with 3 frames per example.

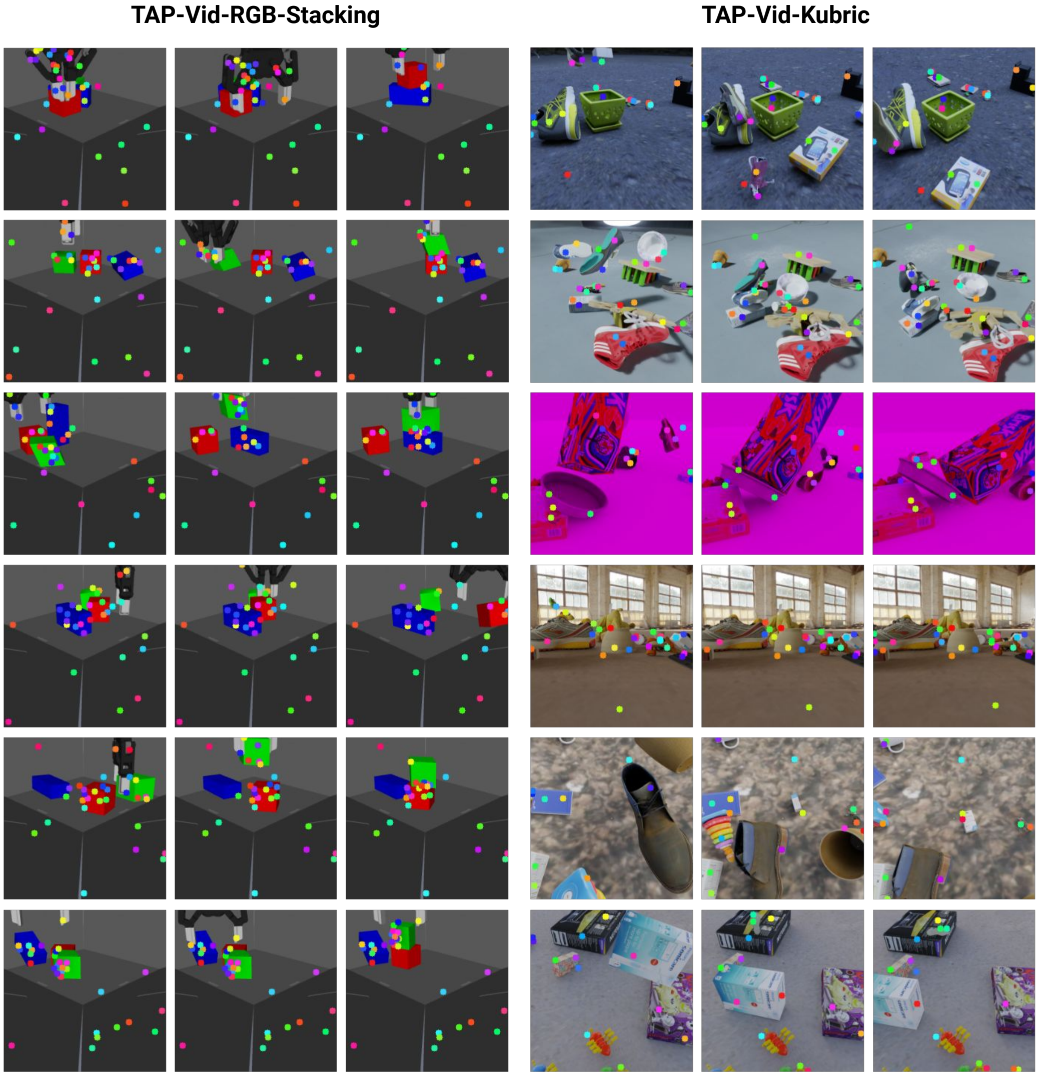
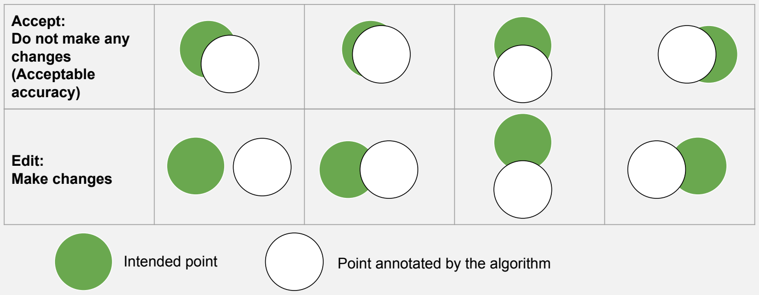


Figure 4: **Examples from TAP-Vid-Kubric and TAP-Vid-RGB-Stacking**: In this figure we show examples of TAP-Vid-Kubric and TAP-Vid-RGB-Stacking annotations with 3 frames per example.

> 1. View the video once. Choose a point that appears on the object (specified by the given bounding-box) for the very first time. Prefer locations such that the point marker plotted at the chosen location should have its center and periphery both completely inside the object region. Try to spread out the points on different parts and label them with part tags if possible (i.e., human hand, human head, car wheel etc.).
> 2. Find the last frame after which the point gets occluded (e.g., because the object moves outside of the frame) and end the track. Then if it becomes visible again, and start a new track to follow it under the same tag, repeating this process until the end of the video.
> 3. Wait for the optical flow assist algorithmic annotations to adjust based on the manual input. Once the algorithmic annotations are available, add manual annotations until the automated annotations are consistent with the schematic below:
>
> 
>

Accept:
> Do not make any changes (Acceptable accuracy)
> Edit:
> Make changes
> Intended point
> Point annotated by the algorithm
>
> 4. Go through the video again to identify any interpolated points in the un-annotated frames with obvious position errors. Correct the error by manually adding a point. Wait a few seconds for the assist algorithm to update the point tracks. Continue the process of reducing errors in the remaining un-annotated frames.
> 5. Once satisfied with the point track in all frames where it is visible, choose another point and repeat the process from the beginning.

Figure 5: **Representative annotator guidelines.** (Minor variations were used as we saw results and gave feedback).

can select (e.g., a specific object category etc.); hence, *any* point which the annotators can track reliably is admissible, resulting in a diverse set of point trajectories in the dataset. The annotators are encouraged to leverage the optical flow track assist algorithm to help improve their accuracy and productivity, although they have the option to turn it off if it's not helpful, resulting in a fallback to linear interpolation. A visual representation of acceptable discrepancy between manual and automated point selections is provided to help the annotators decide when to intervene by adding another point.

4 Simulated Dataset Generation

For synthetic data, obtaining point tracks is conceptually straightforward, as the simulator typically knows the shape and position of all objects. However, most modern simulators don't directly expose point tracks, so we must compute them. Since both Kubric and RGB-Stacking contain only rigid objects, we therefore obtain point tracks in the following manner. We first choose query points in pixel space, which ensures that the query is visible. Then we find the location of that point in the object's local coordinate frame, using either coordinate maps (available for Kubric objects) or by back-projecting using the depth map and applying the inverse object transformation. Next, we compute the 3D location in world coordinates for every other frame using the known transformation of each object. Finally we project the point to the image plane. To compute occlusion, we compare the computed depth at the target frame with the depth map rendered from the simulator. Specifically, if the point is behind the depth map (the max depth of the 4 nearest pixels to the reprojected point) by a sufficient margin (1%), we mark it as occluded.

For Kubric, we sample exactly 256 query points from each video, and we try to have the same number of points for every object, including the background. However, this can lead to far too many points for very small objects, so we limit sampling to 0.16% of the total pixels on each object. There may then not be enough pixels on the smallest objects to sample the same number for each object. Therefore, we sort objects by the number of pixels, and sample iteratively starting with the smallest: if there are K objects, and the smallest object has only P pixels, then we sample N pixels from this smallest object, where $N = \lfloor \min(P * .0016, 256/K) \rfloor$. Then we reduce 256 by N, reduce K by 1, and repeat until we've sampled points from all objects. This ensures the most balanced set of points possible without having too many points per object.

For TAP-Vid-RGB-Stacking, we sample exactly 600 query points uniformly from the first frame for each video and tracked them via the described method above throughout the video. Then we sample 30 points per video such that 20 of them are from moving objects (i.e. tracks with significant motion) and 10 points are from static objects/background (i.e. tracks with minimal or no motion).

5 Dataset Statistics: Agglomerative Clustering of Trajectories

To assess diversity of point motion in our annotations, we cluster the point trajectories in each labelled video (approx. 30 trajectories per video). Agglomerative clustering is performed starting with assigning each trajectory to an independent cluster and recursively merging the clusters until all the inter-cluster distance are greater than a threshold (set to 2 pixels). The cluster distance between two given clusters is the minimum distance between any two trajectories in the two clusters. The distance between trajectories is computed as the mean-centered Euclidean distance between non-occluded trajectory points; this distance is assumed to be undefined (i.e. not counted when comparing clusters) if there are less than 10 frames of overlap. Listing 1 gives a Python-pseudocode implementation of the above.

6 Quantitative Validation of the Human Annotation Procedure

6.1 Validation on Simulated Videos

While it is difficult to validate our annotation procedure on real data, we can easily do so on synthetic data where the ground truth is known. We annotate videos with length 2-seconds (24-frame) from the Kubric dataset [1], where we have known ground truth point tracks. Annotators are instructed to label 3 points on any object up to a maximum of 10 objects (up to 30 points per video). We instruct annotators to choose points which are visible in the first frame, and treat those as queries for computing ground truth tracks.

Because it is a graphics package, Kubric contains all the information required to obtain perfect ground-truth tracks for any point the annotators choose. We can compare the annotated tracks to the ground truth using the same metrics proposed in the original point tracking task.

To make the videos more representative of realistic annotation challenges, we simulate camera jitter (not present in MOVi-E), which is a common phenomenon in many real-world videos. We also increase the framerate from 12 FPS to 25, to match our real videos. All generated videos are included in our supplementary html files. Annotators are instructed to label 3 points on any object up to a maximum of 10 objects (up to 30 points per video). We can compare the annotated tracks to the ground truth. In both cases, annotation was performed at 1024×1024 resolution, although the evaluation is still done at 256×256 pixels to be consistent with our evaluation procedure. The most intuitive method is the fraction of points with error less than a given threshold, although for completeness we include all metrics proposed for evaluation.

Table 1 gives our full results, demonstrating the accuracy of our annotators given the optical flow track assist algorithm. Table 1 also compares results with and without the proposed optical flow track assist algorithm. We see non-trivial improvements in accuracy with the proposed tool, especially on the metrics emphasizing higher levels of precision: for instance, the error rate for pixels $< \delta^1$ (2 pixels away) has been cut by almost half, and for $< \delta^0$ (1 pixel away) it has increased by over 25% in absolute terms. This is true even though annotation was done much faster: time was reduced from 50 minutes per video to 36 minutes per video by using the OF track assist.

```
import numpy as np

def track_dist(t1, t2):
  vis = np.logical_and(t1.raw_vis, t2.raw_vis)
  if np.sum(vis.astype(np.float32)) <= 10:
    return float('inf')
  xy1 = t1.t_xy[vis]
  xy2 = t2.t_xy[vis]
  offset = np.mean(xy1 - xy2, axis=0, keepdims=True)
  xy2 = xy2 + offset
  dist = np.sqrt(np.sum(np.square(xy1 - xy2), axis=-1))
  return np.mean(dist)

def cluster_dist(c1, c2):
  min_dist = float('inf')
  for t1 in c1.children:
    for t2 in c2.children:
      d = track_dist(t1, t2)
      min_dist = min(d, min_dist)
  return min_dist

def all_dists(clusters):
  n = len(clusters)
  dists = np.zeros((n, n)) + float('inf')
  for i in range(n-1):
    c_i = clusters[i]
    for j in range(i+1, n):
      c_j = clusters[j]
      if c_i and c_j are not merged:
        dists[i, j] = cluster_dist(clusters[i], clusters[j])
  return dists

def greedy_cluster(xy, occ, dist_thresh=2/256):
  """
  Agglomerative trajectory clustering.

  xy: [N, T, 2] shaped batch of N trajectory coordinates.
  occ: [N, T] shaped batch of N trajectory occlusions.
  dist_thresh: float, distance threshold for merging clusters.
  """
  # initialize with one cluster per trajectory
  clusters = []
  for i in range(xy.shape[0]):
    clusters.append((xy[i], occ[i]))
  # compute all NxN pairwise trajectory distances
  dists = all_dists(clusters)
  while np.min(dists) < dist_thresh:
    min_ij = np.argmin(dists)
    i, j = np.unravel_index(min_ij, dists.shape)
    # merge trajectories i and j (impl omitted)
    ...
    # re-compute trajectory distances:
    dists = all_dists(clusters)
  return clusters
```

Listing 1: **Agglomerative clustering of trajectories.**

method	AJ	$< \delta^x_{avg}$	OA	Jac. δ^0	Jac. δ^1	Jac. δ^2	Jac. δ^3	Jac. δ^4	$< \delta^0$	$< \delta^1$	$< \delta^2$	$< \delta^3$	$< \delta^4$
No OF	74.7	83.1	97.4	25.8	64.2	90.5	96.2	96.9	41.2	78.8	96.3	99.5	100.0
With OF	81.4	90.0	96.9	49.5	76.1	90.0	95.2	96.0	66.6	87.5	96.2	99.5	100.0

Table 1: Annotation accuracy with and without the optical flow track assist algorithm. Jac. δ^x is the Jaccard metric measuring both occlusion estimation and point accuracy, with a threshold of δ^x; AJ is the Average Jaccard across x between 0 and 4. $< \delta^x$ is the fraction of points not occluded in the ground truth for which the prediction is less than δ^x, and $< \delta^x_{avg}$ is the average across x between 0 and 4. Occlusion Accuracy is denoted with OA. We set $\delta = 2$.

OA	$< \delta^0$	$< \delta^1$	$< \delta^2$	$< \delta^3$	$< \delta^4$
95.5	51.8	78.3	92.5	98.7	100.0

Table 2: Inter-rater agreement on the same set of point tracks for DAVIS. $< \delta^x$ is the fraction of points not occluded in the ground truth for which the prediction is less than δ^x, x between 0 and 4. We set $\delta = 2$. Occlusion Accuracy is denoted with OA.

6.2 Validation on Inter-Rater Agreement

We further conduct inter-human agreement studies on the DAVIS point track videos. For the total of 650 points in 30 videos, we select the first-frame points from the first round of annotation, and ask 2 human raters annotate the following frames for each point. We then compare the similarity between the 2 human annotations using our established metrics. Table 2 shows the human agreement results. On average, human on occlusion with 95.5% accuracy, and 92.5% on location with a 4 pixel threshold. Note that the metrics are evaluated under 256x256 resolution.

7 TAPNet Implementation and Analysis

We propose the first end-to-end deep learning algorithm we are aware of for tracking any point. The network takes a video and a set of query points as inputs and outputs full tracks. The critical component of our system is the *cost volume*, whereby features for a query point are compared to features at every other location in the video. We then apply a simple network on top of the cost volume at each frame to predict the position and occlusion.

7.1 Cost Volume

Any tracking problem involves comparing the query point to other locations in the video in order to find a match. Our approach is inspired by cost volumes [3, 13, 18], which have proven successful for optical flow. We first compute a dense feature grid for the video, and then compare the features for the query point with the features everywhere else in the video. Then we treat the set of comparisons as if it were a feature itself, and perform more neural network computation on top. Cost volumes can detect repeated patterns even when those patterns are unlike those seen in training. Unlike prior work, however, we compute an *multi-headed* cost volume, i.e., we compute multiple feature vectors for every spatial position, and concatenate the cost volumes for each. This gives us a richer feature for further processing.

Figure 6 shows the procedure. Given a video, we first compute a feature grid F, where F_{ijt} is a d-dimensional feature representing the image content at spatial location i, j and time t. For this, we use a TSM-ResNet-18 [12] with time shifting in only the first two layers, as we find that single-frame feature maps, and feature maps with a larger temporal footprint, tended to perform worse; see supplementary for details. This feature grid is broken into n *heads* along the channel axis, each of dimension d/n. Given a query point at position x_q, y_q and time t_q, we extract a feature to represent it from the feature grid F_t via bilinear interpolation on the grid, at position i_q, j_q, t_q. We call the extracted feature F_q. We then compute the cost volume as a matrix product: i.e., if each feature in the

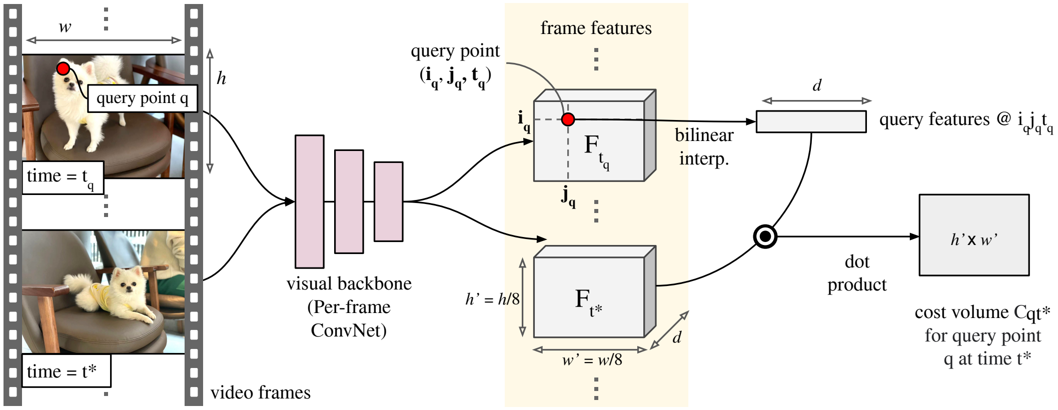


Figure 6: **Cost volume computation for a single query.** Features for all video frames are extracted using a per-frame ConvNet. Features for the given query point location (i, j) in the t_q–th frame are then obtained through 2D bilinear interpolation. The features are dotted with the spatial features from a different time t^* to obtain the corresponding cost volume.

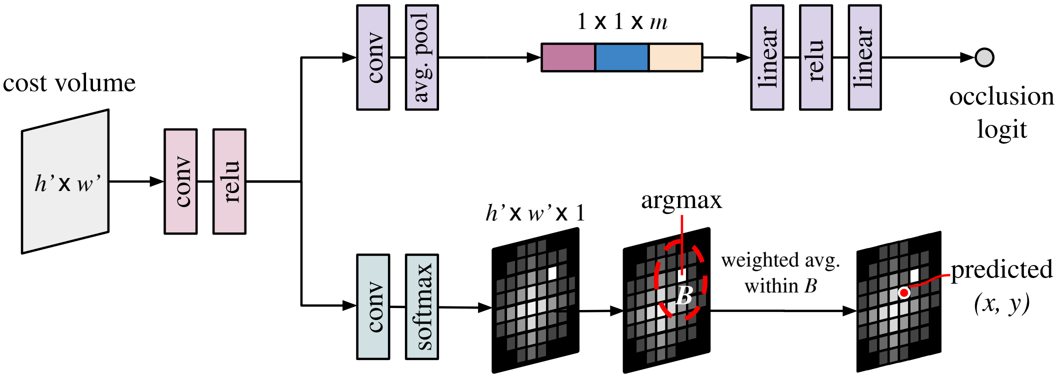


Figure 7: **Inferring position and occlusion from a cost volume.** The cost volume slice (for frame t^* and query q) is fed into two branches: occlusion and coordinate regression. The occlusion branch collapses the spatial features through pooling before regressing a scalar occlusion logit. The point-regression branch collapses the channels through a Conv layer, applies a spatial softmax, and then applies a soft argmax (weighted average of grid-coordinates within a Euclidean ball $\mathcal{B}$).

feature map F is shape $n \times (d/n)$, then the output cost volume $C_{qijt} = F_q^\top F_{ijt}$. Thus, C_q can be seen as a 4-D tensor with n channels.

7.2 Track Prediction

The next step is to post-process the cost volume associated with the query point, which is done independently for each frame. The architecture for one frame is shown in Figure 7. After a first Conv+ReLU, we split into two branches: one for occlusion inference, and the other for position inference. In practice, we find that standard average pooling for occlusion estimation results in unstable training. Therefore, our occlusion branch uses a Conv (32 units), followed by spatial average pooling. This is followed by a linear layer (16 units), a ReLU, and another linear layer which produces a single logit.

For position inference, we apply a Conv layer with a single output, followed by a spatial softmax. This is followed by a soft argmax [9], which means computing the argmax of the heatmap, followed by a spatial average position of the activations within a radius around that argmax location.

Mathematically, let us assume that $S_{qijt} \in \mathbb{R}$ is the softmax activation at time t and spatial position i, j for query q. Let G be a spatial grid, i.e., $G_{ij} \in \mathbb{R}^2$ is the spatial position of S_{ij} in image coordinates. Finally, let $(\hat{i}_{qt}, \hat{j}_{qt})$ be the argmax location of S_{qt}. Then we compute the output position as:

$$p_{qt} = \frac{\sum_{ij} \mathbb{1}\left(||(\hat{i}_{qt}, \hat{j}_{qt}) - (i,j)||_2 < \tau\right) S_{qijt} G_{ij}}{\sum_{ij} \mathbb{1}\left(||(\hat{i}_{qt}, \hat{j}_{qt}) - (i,j)||_2 < \tau\right) S_{qijt}} \tag{1}$$

τ here is a constant, typically equal to 5 grid cells. Note that some parts of this expression are not differentiable: notably the thresholding and the argmax. However, the gradients will still point in the right direction, as errors will typically cause the network to shift the overall mass of the softmax toward the ground truth location [9].

7.3 Loss Definition

Our loss for each query point then has the following form:

$$L(\hat{p}, \hat{o}, p^{gt}, o^{gt}) = \sum_t (1 - o_t^{gt}) L_H(\hat{p}_t, p_t^{gt}) - \lambda \left[\log(\hat{o}) o^{gt} + \log(1 - \hat{o})(1 - o^{gt})\right] \tag{2}$$

Here, L_H is the Huber loss [4]. p^{gt} is the ground-truth position, and o^{gt} is a (binary) ground truth occlusion. That is, we treat position as a simple regression problem for frames where the point is visible. We use a Huber loss because we expect occasional large errors, and we don't want to penalize these excessively. For occlusions, the loss is a standard cross entropy. λ is the trade-off parameter.

7.4 Other Architectural Details

After computing the cost volume, we feed the output to a Conv layer with stride 1, a 3 × 3 receptive field, and 16 hidden units, followed by a ReLU.

The occlusion branch takes this output and applies a convolution with stride 2, 3 × 3 receptive field, and 32 hidden units. This is followed by average pooling, a linear layer with 16 hidden units, ReLU, and a final linear layer for output.

The prediction branch applies a Conv layer with stride 1, 3 × 3 receptive field, and 1 hidden unit, which produces the input for the softmax, which is then used in the soft argmax above.

When computing the Huber loss, we first rescale both the prediction and ground truth coordinates to be in the range [-1,1]. For each coordinate, we compute the Euclidean distance $||(\hat{x}_{qt}, \hat{y}_{qt}) - (x_{qt}^{gt}, y_{qt}^{gt})||_2 = \sqrt{(x_{qt} - \hat{x}_{qt}^{gt})^2 + (y_{qt} - \hat{y}_{qt}^{gt})^2)}$. Then we apply a Huber loss with a threshold $1/32$ (which corresponds to 4 pixels error in the 256 × 256 images we train on). The Huber loss is scaled much smaller than the sigmoid cross entropy, so we set $\lambda = 100$.

Our TSM-ResNet-18 follows ResNet-18: i.e., it has no 'bottleneck' layers, so each residual unit is 2 convolutional layers with a 3 × 3 spatial receptive field. Each 'layer' contains 2 units. It uses no temporal downsampling, meaning that each output has a temporal receptive field of 9 frames. The final output has 256 channels.

7.5 Training

We use a cosine learning rate schedule with a peak rate of $2e^{-3}$ and 5,000 warmup steps. We train on 64 TPU-v3 cores with a batch size of 4 MOVi-E 24-frame videos per core. Each model trains for 50,000 steps. We use an Adam optimizer with $\beta_1 = .9$ and $\beta_2 = .95$. We also use weight decay of $1e^{-2}$, applied after the Adam rescaling (i.e. Adam-W). We use cross-replica batch norm in the ResNet/TSM-ResNet backbone, but don't use batch norm anywhere in or after the cost volume computation.

We performed minimal hyperparameter tuning by observing transfer performance primarily on DAVIS, tuning learning rate, model size, and optimizer parameters in separate sweeps. Each experiment requires roughly 24 hours. Roughly 25 full-scale experiments were required to arrive at the final model.

Method	AJ	$< \delta^x_{avg}$	OA	Jac. δ^0	Jac. δ^1	Jac. δ^2	Jac. δ^3	Jac. δ^4	$< \delta^0$	$< \delta^1$	$< \delta^2$	$< \delta^3$	$< \delta^4$
Full Model	46.6	60.9	85.0	11.3	31.6	53.3	65.8	71.2	19.8	46.4	69.3	81.7	87.4
ResNet-18	46.0	60.7	85.8	9.6	29.5	52.7	66.2	71.9	17.6	45.0	69.8	82.8	88.6
Full TSM	45.4	60.1	83.7	10.8	31.0	52.1	64.0	69.1	19.2	45.9	68.5	80.6	86.4
No Soft Argmax	45.8	59.0	84.6	10.4	30.8	52.6	64.8	70.2	18.2	44.8	67.4	79.1	85.3

Table 3: Ablation of the architectural choices for our model on TAP-Vid-Kinetics. ResNet-18 has no temporal receptive field in the backbone, whereas Full TSM uses time-shifting at every layer, giving a large temporal receptive field. No Soft Argmax replaces the soft argmax with a simple MLP.

Query method	Kinetics			DAVIS			RGB-Stacking		
Strided	46.6	60.9	85.0	38.4	53.1	82.3	59.9	72.8	90.4
First	38.5	54.4	80.6	33.0	48.6	78.8	53.5	68.1	86.3

Table 4: Comparison of performance for the same output tracks, but with different query points. ‘First’ refers to using only the first visible point in each trajectory as a query, whereas ‘Strided’ means sampling multiple query points per trajectory at a stride of 5 frames.

7.6 Model Ablations

We next ablate our design decisions. For this, we use TAP-Vid-Kinetics, the largest and most realistic dataset available to us. We follow the same training setup as TAP-Net for all experiments.

We first ablate our backbone network. Recall that we use a TSM-ResNet-18, with shifting only in the first two layers. Thus we try TSM-ResNet-18, which uses time shifting at every layer, and ResNet-18, without time shifting. We also tried replacing the soft argmax operation with a 2-layer MLP regressor (128 hidden units, ReLU activations) that operates directly on the output heatmap.

We see that all of these changes harm the performance to some extent. Surprisingly, TSM-ResNet performs worse despite allowing the network to integrate across more frames; one possible interpretation is that it gives the network an opportunity to memorize motions in the Kubric dataset. On the other hand, giving no temporal information is also somewhat worse, particularly for the Average Jaccard metric (though not for occlusion estimation), suggesting that this method makes errors on occlusion estimation even when the location is correct and vice-versa. One possible interpretation is that with a small amount of temporal information, the network can do a better job of segmenting the object of interest, which allows the network to track and estimate occlusion for the body as a whole. Allowing this kind of reasoning while preventing memorization of motion patterns presents an interesting challenge for future research. The soft argmax is also important, possibly because it enforces that the network performs matching rather than memorizing motion patterns.

8 Query sampling strategies

Recall that our prediction algorithms take as input *query points*, which specify which point needs to be tracked. For most of this work, we assume that all points are treated equally: any point along a track may be sampled as a query. In practice, we sample queries in a *strided* fashion: every 5 frames starting at frame 0, we sample a query for every visible point. This means we may have multiple queries with the same target output trajectory. An alternative approach which is potentially relevant in an online setting is to start with the *first* frame where the point is visible, and track only into the future (during evaluation, we ignore the predictions for frames earlier than the query frame, as the algorithm can easily assume that these points are occluded, and we don’t want to encourage researchers to hardcode this). In practice, this setting is harder, because on average, the query frame will be farther from each output frame. However, we include both in order to facilitate comparisons with online methods. Table 4 shows the comparison for the three relevant datasets. Kubric is not included, as it uses a different, randomized sampling strategy detailed in the original paper [1]. Note that query sampling is a potential area of improvement for this dataset: some queries may be ambiguous if they are, e.g., behind translucent objects or very close to occlusion boundaries. However, we leave the problem of query sampling to future work.

	Kinetics			Kubric			DAVIS			RGB-Stacking			JHMDB PCK	
Method	AJ	$< \delta^x_{avg}$	OA	AJ	$< \delta^x_{avg}$	OA	AJ	$< \delta^x_{avg}$	OA	AJ	$< \delta^x_{avg}$	OA	@0.1	@0.2
Kubric only	46.6	60.9	85.0	**65.4**	**77.7**	**93.0**	38.4	53.1	82.3	59.9	72.8	**90.4**	62.3	79.8
Finetune Kinetics	**51.6**	**64.9**	**90.4**	59.9	73.1	90.8	**41.5**	**56.0**	**82.4**	**63.2**	**74.7**	90.2	63.4	80.1
Finetune JHMDB	36.4	58.6	71.3	51.4	69.9	83.9	31.6	49.0	77.0	53.0	69.1	90.9	**71.3**	**87.7**

Table 5: **TAP-Net performance after fine-tuning**. We see that fine-tuning on JHMDB harms performance on all datasets except JHMDB by a large margin, but finetuning on Kinetics improves performance on JHMDB, demonstrating the domain transfer properties of the TAP task when the dataset contains arbitrary points tracked on a diverse YouTube dataset. The highlight numbers show the gap specifically.

9 Implementation of the Comparison to JHMDB

Numerous prior methods which aim to benchmark class-agnostic point tracking do so using a dataset called JHMDB. These algorithms typically begin by training self-supervised point tracking algorithms in a class-agnostic manner, and then applying them directly to JHMDB in a semi-supervised manner: joint locations are given on the first frame, and the algorithm must track them over time. The popularity of JHMDB used in this way [5, 7, 8, 10, 15–17] underscores the demand for point tracking evaluations in the literature.

However, JHMDB provides ground truth locations exclusively for human joint locations, which are not points on the surface of objects, but rather *inside* them. This means that tracking based on appearance alone is poorly defined (algorithms must somehow infer the depth of the points beneath the surface). Furthermore, it gives advantages to algorithms that are biased toward humans, ignoring common cases in robotics where the surfaces of inanimate objects must be tracked. In this section, we aim to show the limitations of using JHMDB for class-agnostic point tracking by showing that the dataset isn't as general as TAP-Vid. Specifically, we will train a straightforward model (TAPNet) on TAP-Vid and demonstrate that it can improve class-agnostic tracking on JHMDB (resulting in SOTA performance on this dataset). However, training on JHMDB actually *harms* tracking on TAP-Vid, as it ignores many important cases of point tracking.

We take the full model pretrained on TAP-Vid-Kubric and fine-tune using the same settings on both Kinetics and JHMDB. For simplicity, we use no data augmentation on either dataset; both datasets are resized to $256x256$. We sample 128 queries from each video uniformly at random from all possible queries (unoccluded tracked points), and feed the corresponding tracks as training examples. We train with 1 video per TPU core, and otherwise use the same training setup as for training on Kubric (64 TPU-v3 cores, Adam optimizer), but we train for only 5000 steps (100 warmup steps followed by a cosine schedule) with a much smaller learning rate of $1e-5$ (otherwise the network overfits badly to the relatively small number of points). For JHMDB, we fine-tune on the full train+val dataset, but we evaluate (after Kinetics fine-tuning) on only the JHMDB val-set in order to be comparable with prior results in the literature.

Table 5 shows the performance across all datasets with and without finetuning. Not that we're particularly interested in transfer (salmon-colored rows); i.e., it's not surprising that performance on Kinetics is best for the model finetuned there, and same for JHMDB, as we train on the evaluation images for both. However, what's interesting is that our full model obtains 62.3 PCK@0.1 and 79.8 PCK@0.2 on JHMDB when trained only on Kubric, but 63.4 PCK@0.1 and 80.1 PCK@0.2 after finetuning on Kinetics. However, the model finetuned on JHMDB obtains $58.6 < \delta^x_{avg}$ on Kinetics, and 36.4 Average Jaccard, versus 46.6 and 60.9 respectively training purely on synthetic data. Note the loss of performance on occlusion accuracy and Average Jaccard are perhaps not so surprising, as JHMDB contains no information about occlusion (the evaluation assumes that points which are unoccluded in the first frame are unoccluded in all frames, and we train on all such points). However, $< \delta^x_{avg}$ ignores occlusion, and TAP-Net is still substantially worse with JHMDB finetuning than without it. We see a consistent downtrend in $< \delta^x_{avg}$ (and average Jaccard) throughout training for all datasets, suggesting that this is not merely an issue of overfitting, but rather a fundamental problem with the JHMDB dataset.

10 Baselines

Kubric-VFS-Like We first evaluate a point tracking algorithm inspired by VFS [17] released as part of Kubric [1], and the only algorithm which can be applied directly to our setup. This algorithm uses a contrastive loss to learn features, contrasting points along the trajectory with points off of the trajectory. At test time, this algorithm computes dot products between the query features and the features at every other frame, before applying a spatial soft argmax similar to the one described in equation 1 for each frame. It finally estimates occlusion via cycle consistency: i.e., by computing correspondence backward from the target frame to the query frame, and seeing if the correspondence is the same. Note that our metrics don't exactly match those from the original paper [1]; the codebase is the same, but we used a ResNet-18 backbone to match this paper, and unlike the original work, we include the query points in the evaluation. We ran each evaluation on an internal V100 GPU; the full set took approximately 1 hour to run.

RAFT [14] This algorithm performs near state-of-the-art for optical flow estimation, i.e., tracking points between individual frames. We extend this to multiple frames by integrating the flow from the query point: i.e., we use bilinear interpolation of the flow estimate to update the query point, and then move to the next frame and repeat. If the point is outside the frame, we use the flow value at the nearest visible pixel. While RAFT is extremely accurate over short time scales, this approach has some obvious problems with long time scales. First, it provides no simple way to deal with occlusions (we simply mark points as occluded if they're outside the frame). Furthermore, it has no way to recover from errors, so slight inaccuracies on each frame tend to accumulate over time even when the point remains visible. Experiments were on an internal V100 GPU and took approximately 24 hours.

COTR [6] Like RAFT, COTR operates on pairs of images; however, it was designed to handle substantially more motion between images. The underlying architecture uses a transformer to find global scene alignment, followed by more local refinement steps to ensure high accuracy. It is trained on MegaDepth [11], which contains real-world scenes and uses reconstruction to get ground-truth correspondence. Due to this robustness, we use a different strategy from RAFT to apply COTR to videos: we compare the query frame to every other frame in the video, and find correspondences directly. COTR has no simple mechanism for detecting occlusions, so we apply the cycle-consistency strategy proposed in Kubric-VFS-Like: given correspondences between the query and target frame, we take the points in the target frame and find their correspondence in the query image. If the distance to the original query is greater than 48 pixels, we mark the point as occluded. Note that it's very expensive to run the model in this way: if there's queries on every frame, then we need to run COTR on every frame pair. Each pair takes 1 second with 30 points. Thus, for evaluation, we simplify our benchmark and take only queries from a single frame; thus these numbers are not precise, but good enough to get an idea of COTR's performance on our benchmarks. Experiments were on a GCP machine with 4 A100 GPUs and required approximately 20 hours.

PIPs [2] We ran Persistent Independent Particles (PIPs) following the "chaining" workflow, as the bulk of the algorithm is intended to operate on 8-frame segments. PIPs extracts features for both the query point and the rest of the video using a ConvNet, similar to TAP-Net. With the chaining workflow, tracks on each segment are estimated one-by-one, with the final point of each segment used to initialize the subsequent segment. Given an initial position estimate for a segment, it refines both the query features and the output trajectory using an MLP-Mixer applied iteratively. Similar to TAP-Net, the input for this refinement network includes the dot product between the query features and the features in the other frames, although in the case of PIPs, the dot products come from a local neighborhood around the spatial position estimate. The initial spatial position for each 8-frame chunk is the last spatial position that the point was estimated to be visible. As PIPs operates only forward in time, we run it twice for each query point: once forward, and once backward in time. Otherwise, we run the algorithm without modification, directly using its visibility estimates with a threshold of 0.5 to estimate occlusion. We found that the chaining algorithm requires more than 30 minutes to process a 250-frame video at 256×256 resolution, meaning that we only had time to run on 167 random Kinetics videos before the deadline. Unlike for COTR, however, these results use all points available in each video. Experiments were run on a GCP machine with 1 V100 GPU and required approximately 100 hours.

11 Semantic analysis

Although our dataset is not intended for classification, it is useful to consider the distribution of labeled objects to understand performance. While DAVIS and RGB-stacking are small enough for the object distribution to be observed qualitatively, for Kinetics, we had annotators explicitly label the boxes before adding points. This process was informal, and we don't control for duplicates or typos; nevertheless, we show the full list of labels as well as their frequencies in Tables 6–9. We see a large variety of objects, although unsurprisingly for Kinetics, they tend to be people, clothing, and moving objects that are relevant for human actions, across indoor and outdoor, as well as several types of animals. We manually broke these into broader categories, and found 26.1% are on humans, 23.4% on clothing (on or off the body), and 50.4% are on other types of objects. This is consistent with the instructions (to target moving/foreground objects), and we believe it makes the dataset relevant for agents interacting with humans and performing tasks in human environments.

person:7012
shoe:842
pant:833
tshirt:796
object:614
shirt:608
hand:542
car:447
short:413
t shirt:412
cap:377
chair:372
box:370
table:347
bottle:310
jacket:247
helmet:235
ring:223
light:215
watch:214
bowl:209
cloth:207
glove:186
machine:186
bag:176
pole:167
toy:163
plate:158
spects:150
ball:147
frame:147
wood:140
door:139
dress:138
paper:135
glass:132
vehicle:132
stick:119
sofa:117
coat:115
window:114
board:108
rod:105
rope:102
stand:97
tool:93
knife:92
flag:92
pipe:91
bucket:91
band:86
mat:84
right hand:81
brush:79
curtain:76
thread:75
hat:75
book:75
left hand:74
chain:74
dog:72
mirror:72
horse:72
belt:70
wire:70
hand band:63
socket:63
cover:62
button:61
cycle:60
goggles:58
bike:58
handle:56
photo frame:55
speaker:55
spectacles:55
flexi:9
mini car:9
plant pot:9
cat:9
weight plate:9
musical instrument:9
clock:9
comb:9
paper roll:9
ac:9
roller:9
lamp light:9
screw tighter:9
garland:9
tower:9
black coat:9
plastic item:9
barrier:9
tree pot:9
crown:9
chess piece:9
bedsheet:9
wrist band:9
pencil:9
glouse:9
yacht:9
wall poster:9
headphones:9
red button:9
blue tshirt:9
lantern:9
black short:9
wood board:9
black top:9
bat:9
photographer:9
jug:9
wood block:9
wind cart:9
electric pole:9
instrument:9
light stand:9
flying disc:9
vacuum cleaner:9
oil bottle:9
mouse:9
weight lift:9
watermelon:9
zip:9
fish:9
cleaner:8
card shelf:8
cushion:8
bandage:8
red box:8
black t shirt:8
red ball:8
wooden board:8
carpet:8
mixer:7
spatula:7
fire extinguisher:7
googles:7
tile:7
bottle cap:7
carry bag:7
wall clock:7
straightener:6
key:6
elephant statue:6
specs:6
bridge:6
yoga mat:6
mattress:6
plastic bowl:6
black pant:6
hacksaw:3
statue table:3
dried palm leaves:3
blue blanket:3
blue mat:3
steel tap:3
silver foil:3
toy train tracker:3
gas:3
plastic plate:3
sledge:3
mini trampoline:3
pendant:3
snake:3
face cream:3
headphone:3
violin stick:3
circular frame:3
skiers:3
system:3
paint spray machine:3
iron grill:3
marker pen:3
yarn:3
plastic glass:3
glass bowl:3
p:3
pouch:3
mic stand:3
car cover:3
staff:3
chips:3
handband:3
snow bike:3
bed cot:3
yellow belt:3
shark:3
plastic box:3
luggage bag:3
double tape:3
decoration light:3
baw:3
refrigerator:3
teeth braces:3
side mirror:3
tool box:3
machinery equipment:3
cable:3
putty plate:3
coupling pipe:3
stairs:3
neck support:3
glass object:3
gear shifter:3
drilling machine:3
hinge:3
door engis:3
goal post:3
fedlight:3
yellow bucket:3
wood bench:3
paino:3
western top:3
black item:3
coconut:3
green mat:3
art sheet:3
blue egg:3
pink basket:3
yellow egg:3
dog sofa:3
left gloves:3
green button:3
suitcase:3
light lamp:3
car engine:3
lady:3
black curtain:3
electric pencil sharpner:3
white jacket:3
flowers:3
window sheet:3
floor mat:3
decor:3
frame stand:3
wash basin:3
bathtub:3
showcase plant:3
spinning top:3
hook:3
wall hanger:3
mining helmet:3
color box:3
sleeve dress:3
crocodile:3
christmas tree:3
white tray:3
helmate:3
red shirt:3
righthand:3
streetlight:3
trumpet:3
whiper:3
shaving razor:3
golfstick:3
banner stand:3
flex:3
bean bag:3
egg:3
holdingbelt:3
air pump machine:3
blue barrel:3
dumbell:3
broom stick:3
right gloves:3
plunger:3
western toilet seat:3
cutting machine:3
pallet:3
black dress:3
blade:3
meat pieces:3
groove machine:3
white short:3
measuring tape:3
baby boy:3
wooden spoon:3
white pad:3
black cup:3
red block:3
shield:3
electric stove:3
streeing:3
metal chrome polish bottle:3
exhaust pipe:3
wooden staircase:3
ladder stand:3
bubble bottle:3
scrubber:3
water sprinkler:3
chandelier:3
cot:3
pumpkin piece:3
backet:3
exercise suit:3
watermelon piece:3
torch light:3
leather hand band:3
hoop hula:3
rings:3
white hoody:3
soap bubble sticks:3
cleaning brush:3
sleeveless:3
strap:3
aquarium:3
green grill:3
tumbler:3
arrow holder:3
roll:3
elastic band:3
welding machine:3
cotton box:3
steel rods:3
white scooter:3
propane fueling station:3
blue scooter:3
hanging handle:3
wall decor:3
thermocol:3
green bag:3
machine part:3
wood object:3
screwdriver:3
concrete fire bowl:3
side pin:3
whistle:3
icebreaker:3
sushi food:3
plaster roller:3
gift wrap:3
plaster:3
blue cover:3
spects frame:3
snow scooter:3
switch box:3
man hole:3
embroidery hoop ring:3
staple puller:3
cutting plier:3
clothes tub:3
black bucket:3
clothes bag:3
silver stick:3
wooden roller:3
support board:3
metal water pipe:3
paper sheet:3
automatic door operator:3
chef cap:3
green apple:3
petrol holder:3
car petrol cap:3
petrol pipe:3
orange light:3
white light:3
red light:3
car bumper:3
dumbbell plate:3
flex banner:3
podium:3
baby mattress:3
wood tray:3
sharpner:3
tissue box:3
brown sheet:3
cup board:3
fruits:3
snow lifter:3
wood cutting machine:3
wood box:3
snow toy:3
vaccum pipe:3
carry baby bag:3
straw:3
pink jacket:3
mitten:3
pepper crusher:3

Table 6: **Full list of object categories named by annotators in TAP-Vid-Kinetics.** Number of points per category is listed next to the category.

basket:54
poster:54
mike:53
pot:53
girl:52
mobile:51
boy:51
switch board:51
packet:51
ear ring:48
top:47
right shoe:46
spoon:46
bed:46
cup:45
tyre:45
card:45
pan:44
equipment:43
apple piece:42
boat:42
hair band:42
jar:42
hoodie:41
net:40
tray:40
guitar:40
left shoe:40
stool:39
kid:39
wheel:39
bracelet:39
parachute:38
mic:37
scissor:36
stone:36
socks:35
pillow:35
television:35
rock:35
slipper:35
camera:34
apple:34
pen:33
monitor:33
child:33
house:33
bolt:33
tub:33
towel:33
black object:33
jeans:32
cupboard:32
tie:30
statue:30
baby:30
bangle:30
red object:29
lamp:29
white box:28
sheet:28
hammer:28
dustbin:27
bicycle:27
laptop:27
holder:27
bench:27
candle:27
needle:27
green rope:6
flower:6
wooden pole:6
basketball board:6
cutting player:6
wall:6
paper rocket:6
rolling mat:6
doughnut:6
washing machine:6
clothes:6
motorcycle:6
scooter:6
left glove:6
jack:6
blue jacket:6
red bag:6
rim:6
chimney:6
sandle:6
speedometer:6
wind gong:6
nut:6
number plate:6
blue joystick:6
swim suit:6
stage:6
tire:6
digging tool:6
ribbon:6
steel table:6
animal:6
binding rod:6
steel rod:6
necklace:6
rightshoe:6
hipbelt:6
display:6
inner wear:6
log:6
metal sheet:6
utensil:6
cucumber:6
wall frame:6
goal:6
diving shoe:6
duster:6
time adjuster:6
white bin:6
bus:6
wooden cupboard:6
play card:6
hairclip:6
charger:6
swim cap:6
trolley:6
clay:6
wooden bar:6
blue jeans:6
signboard:6
bread piece:6
brown object:6
white cap:6
dumble:6
wooden chair:6
potted plant:6
head cap:6
red thread:6
measuring tool:6
injector pipe:3
tube:3
desktop:3
support rod:3
machine tool:3
woolen thread:3
weaving needle:3
trolly:3
brown bricks:3
train toy:3
chisel:3
orange pencil:3
white marble:3
paper punch:3
papers:3
backpack:3
knife box:3
phone mount:3
ornament:3
motor switch:3
seat cover:3
sign:3
milk box:3
mixer grinder:3
steel net:3
white bricks:3
blue button:3
white machine:3
glass bottle:3
carrot piece:3
wing screw:3
microphone stand:3
sand tray:3
brown pant:3
extinguisher:3
coke bottle:3
purse:3
tripod:3
ridge gourd:3
costume:3
ridge gourd piece:3
pink shirt:3
cable roller:3
airtrack:3
nose ring:3
steel tray:3
igloo house:3
wooden table:3
vechile:3
blue short:3
yellow tshirt:3
green short:3
woodframe:3
cock:3
mannequin:3
soft boat:3
fire stand:3
yellow handle:3
stroller:3
back wheel:3
shell:3
beanie:3
yellow rope:3
iron handler:3
mushroom:3
strings:3
wick:3
black board:3
black half sleeve shirt:3
white item:3
pagdi:3
wooden mud pot:3
tyrecap:3
black paper:3
right ring:3
left bracelet:3
guitar box:3
baby chair:3
refridgerator:3
string:3
food packet:3
swim glasses:3
traning fin:3
scuba diving fins:3
hookah pipe:3
lorry:3
chopping board:3
cake piece:3
steel plate:3
decorative:3
painting board:3
key hole:3
glass pipe:3
frock:3
music instrument plate:3
crash cymbal:3
card door lock:3
selfie stick:3
banana:3
basketball hoop:3
filmy equipment:3
wooden pad:3
hand towel:3
water jug:3
plates:3
mud remover:3
leaves:3
swim fin:3
paint spray:3
chain locket:3
ceiling ac vents:3
pink object:3
mike stand:3
baby crib:3
wooden stool:3
metal wall design:3
brake:3
rubber:3
pad stand:3
round table:3
fruit basket:3
mike transmitter:3
silver nut:3
water pot:3
yellow bowl:3
snow shovel:3
book rack:3
white pipe:3
black stand:3
fire wood:3
small girl:3
chess board:3
pink box:3
pink book:3
meat:3
helicopter:3
statue head:3
finger cap:3
oil container:3
mug jar:3
wafer:3
orange rope:3
paint bottle:3
table cover:3
dolpin:3
railing bridge:3
ice box:3
cold jar:3
black rod:3
net holder:3
exercise ball:3
big basket:3
ash shirt:3
wooden base:3
hair straightener:3
blue board:3
cement brick:3
pompoms:3
gift box:3
exam pad:3
wooden floor:3
stretcher:3
powered parachute:3
ring box:3
stopper cone:3
gas stove:3
wheel barrow:3
wax strip:3
router box:3
small tin:3
pipe slide:3
ear stud:3
shorts:3
barbell rod:3
score card:3
baseball bat:3
id card:3
black grill:3
music instrument:3
pad plates:3
pool bridge:3
air meter:3
swimming item:3
eyebrow pencil:3
eye lash shaper:3
curling machine:3
kids play slide:3
plastic ring:3
pink tshirt:3
water skater:3
sand machine:3
extension socket:3
paint tube:3
cotton cake box:3
weight:3
round comb:3
creddle:3
glass plate:3
water safety:3
cake plate:3
white stool:3
hair roller:3
dish wash bottle:3
solar board:3
screw holder:3
nail polish:3
paper packet:3

Table 7: **Full list of object categories named by annotators in TAP-Vid-Kinetics, continued.**

ladder:26
phone:25
sign board:25
tree:25
dumbbell:25
rack:25
bow:24
photo:24
mascara:24
doll:24
fan:24
tap:24
gloves:24
cutter:24
earring:24
sweater:23
cake:21
blue cloth:21
spray bottle:21
dolphin:21
sticker:21
balloon:21
fence:21
drum:21
steering:21
banner:21
fork:21
block:21
screw:21
tin:21
black box:21
yellow object:20
berry:20
fridge:20
white board:20
blue shirt:20
stove:20
remote:20
oven:19
napkin:19
hanger:18
potato:18
steel:18
white object:18
vest:18
tissue:18
cow:18
skate board:18
lighter:18
flute:18
hand glove:18
black bag:18
shovel:18
screen:18
apron:18
headband:17
plant:17
seat:17
sock:16
red cloth:16
knob:16
coin:15
ship:15
umbrella:15
blanket:15
blue box:15
pin:15
tv:15
mask:15
drill machine:15
brick:15
cabinet:15
tape:15
cylinder:15
camel:15
windmill:15
red cap:15
leg:15
gate:15
bird:15
cardboard:15
shoe rack:6
white block:6
glass jar:6
food:6
painting:6
baby bed:6
orange:6
orange slice:6
giraffe:6
sheep:6
metal candle cup:6
glass table:6
tomato:6
steel object:6
stainer:6
projector:6
steel box:6
white strip:6
sponge:6
door handle:6
blue top:6
head scarf:6
cradle:6
light pole:6
head band:6
boxing bag:6
cd box:6
wardrobe:6
door lock:6
woman:6
rubber band:6
dish:6
piano:6
train:6
scale:6
gear rod:6
slide:6
water bottle:6
mixer jar:6
yellow thread:6
machine rod:6
wool:6
battery box:6
microphone:6
iron sheet:6
thread bundle:6
black button:6
car seat:6
white t shirt:6
crane:6
weight equipment:6
spray:6
red carpet:6
steamer:6
wooden bench:6
commode:6
wristband:6
wiper:6
sword:6
leftshoe:6
scissors:6
books:6
green box:6
extension box:6
soap:6
right leg:6
traffic cone:6
walkie talkie:6
black stick:6
ski:6
trunk belt:6
boot:6
tab:6
weight lifting equipment:6
blue pant:6
watering can:6
left leg:6
disk:6
weight lifting:6
cigar pipe:6
lefthand:6
oxygen tank:3
toy track:3
purple jacket:3
fuel can:3
match stick:3
pink top:3
footwear:3
stove stand:3
white flower:3
tape dispenser:3
balloon toy:3
ballon:3
supporter:3
aeroplane:3
pineapple:3
pink bin:3
cards:3
right indicator:3
left indicator:3
match box:3
broom:3
roti maker:3
cleaning machine:3
cabin:3
card board:3
food box:3
pricetag:3
stamps:3
gum tube:3
number sheet:3
roughfile:3
label:3
softdrink tin:3
arrow:3
foot:3
yellow short:3
dressing table:3
cristmas tree:3
jumping castle:3
machine top:3
bike mirror:3
front shield:3
swim board:3
torch:3
bra pad:3
bra tag:3
sewing machine:3
blue object:3
cotton:3
cleaning mop:3
decoration item:3
white ball:3
white boat:3
legs:3
wallet:3
globe:3
trimble scanning:3
avacado:3
handles:3
rubiks cube:3
right mirror:3
metal wire:3
chopstick:3
nail cutter:3
hatchet:3
black file:3
hand catcher:3
speed meter:3
inhaler:3
horn switch:3
letter m:3
t shirt roll:3
menu book:3
green jacket:3
ski helmet:3
traffic light:3
steel bowl:3
water melon piece:3
toy brick:3
flip flop:3
left socks:3
steel item:3
girl dress:3
short pant:3
surfboard:3
drum plate:3
door closer:3
wood plank:3
gear:3
plastic knob:3
wooden rack:3
snowman:3
wind instrument:3
safety cap:3
window glass:3
dash board:3
electric engraving machine:3
wood stand:3
orangestool:3
calender:3
blinds:3
vaccum machine:3
game equipment:3
game board:3
coolent box:3
funnel:3
infuser:3
bike delivery box:3
carrot:3
wooden pot:3
showpiece:3
stainless steel water jug:3
cigarette pipe:3
bluetooth:3
watering pipe:3
radio:3
decor light:3
disco light:3
music box:3
drum stick:3
music plate:3
bell plate:3
windgong:3
air pump:3
spading stick:3
pool lane rope:3
bee box:3
steel glass:3
tea pot:3
sleeve shirt:3
fencing mesh:3
cooling glasses:3
sticky notes:3
album:3
stick paper:3
file handle:3
hand kerchief:3
id tag:3
snow stick:3
metal tool:3
treadmill:3
spin bike:3
left slipper:3
tongs:3
wrist belt:3
handwash:3
honey tray:3
shutter gear box:3
gear box:3
shop:3
football net:3
sink:3
breather:3
handle knob:3
red machine:3
show case:3
yellow stone:3
chappal:3
white background:3
wrapper:3
land line:3
ventilator:3
peeler:3
oil tub:3
locker:3
water can:3
support stand:3
white band:3
dust bin:3
pine apple:3
white blind:2
right sock:2
sketch:2
pink thread:2
filter:2
saddle:2
stethoscope:2
foam:2
white cardboard:2
hinges:2
saddle rope:2
news paper:2
doormate:2
bull ride:2
pineapple slice:2
box cap:2
grinder:2
bracelette:2
wind manipulator:2
underwear:2
white napkin:2
robot toy:2
mobile holder:2
oxygen pumping machine:2
gym bench:2
tennis bat:2
red rod:2
green pant:2
orange shirt:2
violet dress:2
left handle:2
flower balloons:2
yellow button:2
frontwheel:2
black band:2
watertin:2
pertson:2
vehcile:2
hanging light:2
fishing stick:2
plane:2
fishing rod:2
plastic disk:2
iron stand:2
table fan:2
shed:2
goat:2
head cover:2
fencing rod:2
dartboard:2
blue banner:2
textperson:2
shampoo bottle:2
tperson:1
boxcap:1
parchute:1
bracelettle:1
letter e:1
dishwasher:1
frige:1
shie:1
wodd:1
coffee maker:1
blue suit:1
pine apple slice:1
bandage clip:1
new paper:1
pany:1
newspaper:1
water melon peice:1
show peice:1
dart board:1
white cupboard:1

Table 8: **Full list of object categories named by annotators in TAP-Vid-Kinetics, continued.**

hand bag:15
van:15
wooden box:15
iron rod:15
pad:15
axe:15
berries:15
wooden object:15
gun:15
lock:15
tent:15
idol:15
can:14
hockey stick:14
scarf:14
cone:14
black shirt:14
white cloth:14
clip:13
white sheet:13
flower vase:13
bin:12
grill:12
bed sheet:12
black jacket:12
gym equipment:12
file:12
duck:12
headset:12
pumpkin:12
black cloth:12
sandal:12
sprayer:12
puzzle:12
elephant:12
chess coin:12
tissue roll:12
skirt:12
wooden block:12
tong:12
wood piece:12
wooden plank:12
diaper:12
wrist watch:12
racket:12
lid:12
flower pot:12
glasses:12
hair clip:12
carton box:12
truck:12
joystick:12
keyboard:12
bulb:12
screw driver:12
stud:12
plastic cover:12
right glove:11
fruit:11
switch:11
green pipe:11
text:11
switchboard:10
kangaroo:10
plastic bag:10
paint brush:9
egg yolk:9
head light:9
black tshirt:9
trimmer:6
mortar board holder:6
metal:6
wrench:6
spanner:6
basin:6
couch:6
bud:6
white tshirt:6
chips packet:6
burner:6
vase:6
eyelash curler:6
marble:6
ribbon roll:5
blue sheet:5
container:5
pig:5
chip:5
guage:5
tube light:5
left mirror:5
kite:5
playing machine:5
dish washer:5
ice axe:5
photoframe:5
show piece:5
wooden log:5
skipping rope:5
white shirt:5
hoddie:5
cpu:5
dining table:5
point:5
track:5
hair dryer:4
gauge:4
building:4
black thread:4
motor:4
iron box:4
power box:4
wooden stick:4
blue funnel:3
handler:3
plucker tool:3
skate boarding:3
iron lid:3
greentub:3
cement lid:3
router:3
yellow carpet:3
tractor:3
baby bottle:3
kids walker:3
store:3
dumbbell set:3
left ear stud:3
needle bag:3
parachute rope:3
controller:3
violin:3
cellphone:3
kids slide:3
red toy:3
electrical box cover:3
big spoon:3
left sock:3
knife holder:3
megaphone speaker:3
barricating floor stand:3
porch swing:3
watertank:3
tiger:3
black suit:3
wooden shelf:3
red clip:3
hairpin:3
mobile case:3
wood blocks:3
black pipe:3
blue block:3
finger tool:3
green shirt:3
glue gun tool:3
iron frame:3
umberlla stand:3
bathing basin:3
plastic curtain:3
fan hanger:3
rod piece:3
orange tool:3
maroon cloth:3
yellow cloth:3
telephone:3
car number plate:3
blue thread:3
toy stick:3
olive oil bottle:3
detecting machine:3
mixing bowl:3
steel tool:3
paint filler:3
piano keyboard:3
tv stand:3
toy car:3
cutting board:3
sound box:3
tambourine hand percussion:3
pool lane divider:3
desk:3
wooden tray:3
bread:3
red jacket:3
work piece:3
red blanket:3
cloths:3
pebble:3
trowel:3
red suit:3
rope ball:3
woolen roll:3
man:3
talcum powder:3
drink bottle:3
turban:3
ash tshirt:3
round disk:3
safety jacket:3
plank:3
teddy:3
chart:3
cake pan:3
orange object:3
tank:3
room cleaner:3
wheel cap:3
suit:3
sleeveless tshirt:3
detector:3
sniper box:3
thermocol sheet:3
track pant:3
white bucket:3
green truck:3
circle balloon:3
toy block:3
traffic light toy:3
spectacle:3
baby dress:3
baby apron:3
sausage:3
blue chalk:3
soap stand:3
gamepad:3
yellow toy:3
door handler:3
purple bottle:3
c c camera:3
hairband:3
hut:3
yellow paper:3
syringe:3
metal piece:3
drawer:3
toy house:3
marker:3
barricade:3
hand break:3
wooden piece:3
tambourine:3
guitar bag:3
white bottle:3
glass door:3
stop indicator board:3
leaning bench:3
bar lifting:3
threadmill:3
mosquito killer:3
shelf:3
dvd player:3
paper crusher:3
auto:3
woollen:3
blue color object:3
green object:3
person hand:3
candle cap:3
cube:3
brown box:3
paper glass:3
wood machine:3
light box:3
laptop table:3
rack cupboard:3
tyer:3
orange tshirt:3
rod stand:3
red curtain:3
dryer:3
railing:3
topwear:3
strainer:3
brown bag:3
mascara stick:3
pink sheet:3
head:1
oxygen pumpimg machine:1
wall poster top:1
cushoin:1
galss:1
frame top:1
peerson:1
pizza oven:1
gyn bench:1
branch:1
preson:1
sipper:1
street light:1
headlight:1
cardshelf:1
persoon:1
greenpipe:1
glasse middle:1
glass right:1
glass left:1
laddder:1
vihicle:1
paper shredder:1
frame middle:1
persson:1
form:1
tbowl:1
boardboard:1
plastic desk:1
whiteshirt:1
palne:1
bowling equipment:1
hooddie:1
res cloth:1
red scarf:1
yelllow object:1
tabel:1
plastic object:1
mug:1
balack band:1
front wheel:1
spects left:1
white rolling button:1
person back shoulder:1
person forehead:1
cupboard top:1
cupboard middle:1
cupboard bottom:1
personn:1
rind:1
door mate:1
jeans jacket:1
white card board:1
ttshirt:1
table right:1
table middle:1
table left:1
parachute skydiving:1
nuts packet bottom:1
perason:1
blue stone:1
swimming goggles:1
hesd cover:1
photofrane:1
balll:1
nuts packet top:1
nuts packet middle:1
frame bottom:1

Table 9: **Full list of object categories named by annotators in TAP-Vid-Kinetics, continued.**

References

[1] Greff, K., Belletti, F., Beyer, L., Doersch, C., Du, Y., Duckworth, D., Fleet, D.J., Gnanapragasam, D., Golemo, F., Herrmann, C., Kipf, T., Kundu, A., Lagun, D., Laradji, I., Liu, H.T.D., Meyer, H., Miao, Y., Nowrouzezahrai, D., Oztireli, C., Pot, E., Radwan, N., Rebain, D., Sabour, S., Sajjadi, M.S.M., Sela, M., Sitzmann, V., Stone, A., Sun, D., Vora, S., Wang, Z., Wu, T., Yi, K.M., Zhong, F., Tagliasacchi, A.: Kubric: a scalable dataset generator. In: Proceedings of the IEEE conference on computer vision and pattern recognition (2022)

[2] Harley, A.W., Fang, Z., Fragkiadaki, K.: Particle videos revisited: Tracking through occlusions using point trajectories. In: European Conference on Computer Vision (2022)

[3] Hosni, A., Rhemann, C., Bleyer, M., Rother, C., Gelautz, M.: Fast cost-volume filtering for visual correspondence and beyond. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**(2), 504–511 (2012)

[4] Huber, P.J.: Robust estimation of a location parameter. In: Breakthroughs in statistics, pp. 492–518. Springer (1992)

[5] Jabri, A., Owens, A., Efros, A.: Space-time correspondence as a contrastive random walk. Advances in neural information processing systems **33**, 19545–19560 (2020)

[6] Jiang, W., Trulls, E., Hosang, J., Tagliasacchi, A., Yi, K.M.: Cotr: Correspondence transformer for matching across images. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6207–6217 (2021)

[7] Lai, Z., Lu, E., Xie, W.: Mast: A memory-augmented self-supervised tracker. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6479–6488 (2020)

[8] Lai, Z., Xie, W.: Self-supervised learning for video correspondence flow. arXiv preprint arXiv:1905.00875 (2019)

[9] Levine, S., Finn, C., Darrell, T., Abbeel, P.: End-to-end training of deep visuomotor policies. The Journal of Machine Learning Research **17**(1), 1334–1373 (2016)

[10] Li, X., Liu, S., De Mello, S., Wang, X., Kautz, J., Yang, M.H.: Joint-task self-supervised learning for temporal correspondence. Advances in Neural Information Processing Systems **32** (2019)

[11] Li, Z., Snavely, N.: Megadepth: Learning single-view depth prediction from internet photos. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2041–2050 (2018)

[12] Lin, J., Gan, C., Wang, K., Han, S.: Tsm: Temporal shift module for efficient and scalable video understanding on edge devices. IEEE transactions on pattern analysis and machine intelligence (2020)

[13] Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. International journal of computer vision **47**(1), 7–42 (2002)

[14] Teed, Z., Deng, J.: Raft: Recurrent all-pairs field transforms for optical flow. In: European conference on computer vision. pp. 402–419. Springer (2020)

[15] Vondrick, C., Shrivastava, A., Fathi, A., Guadarrama, S., Murphy, K.: Tracking emerges by colorizing videos. In: Proceedings of the European conference on computer vision (ECCV). pp. 391–408 (2018)

[16] Wang, X., Jabri, A., Efros, A.A.: Learning correspondence from the cycle-consistency of time. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2566–2576 (2019)

[17] Xu, J., Wang, X.: Rethinking self-supervised correspondence learning: A video frame-level similarity perspective. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10075–10085 (2021)

[18] Zbontar, J., LeCun, Y., et al.: Stereo matching by training a convolutional neural network to compare image patches. J. Mach. Learn. Res. **17**(1), 2287–2318 (2016)