PointOdyssey: A Large-Scale Synthetic Dataset for Long-Term Point Tracking

Yang Zheng Adam W. Harley† Bokui Shen Gordon Wetzstein Leonidas J. Guibas
Stanford University
{yzheng18,harleya,willshen,gordonwz,guibas}@stanford.edu

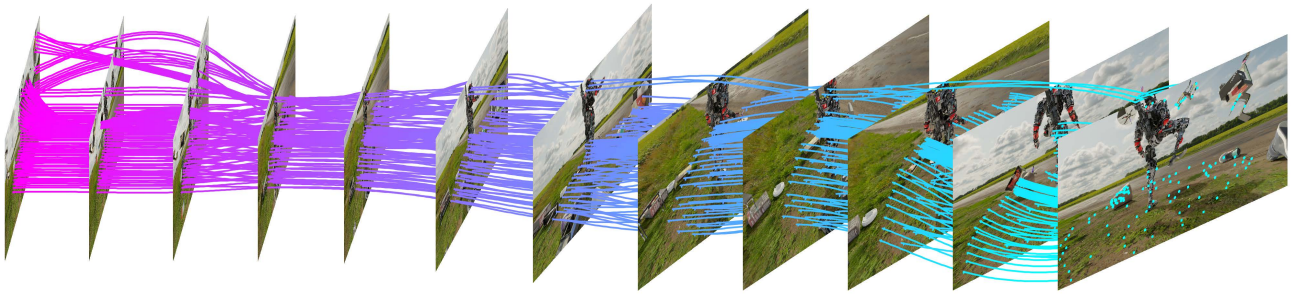


Figure 1: **PointOdyssey dataset.** We provide point correspondence annotations across long continuous videos. Here we visualize pixel-coordinate trajectories from frame 0 to frame 1600 in a sample video from our dataset.

Abstract

We introduce PointOdyssey, a large-scale synthetic dataset, and data generation framework, for the training and evaluation of long-term fine-grained tracking algorithms. Our goal is to advance the state-of-the-art by placing emphasis on long videos with naturalistic motion. Toward the goal of naturalism, we animate deformable characters using real-world motion capture data, we build 3D scenes to match the motion capture environments, and we render camera viewpoints using trajectories mined via structure-from-motion on real videos. We create combinatorial diversity by randomizing character appearance, motion profiles, materials, lighting, 3D assets, and atmospheric effects. Our dataset currently includes 104 videos, averaging 2,000 frames long, with orders of magnitude more correspondence annotations than prior work. We show that existing methods can be trained from scratch in our dataset and outperform the published variants. Finally, we introduce modifications to the PIPs point tracking method, greatly widening its temporal receptive field, which improves its performance on PointOdyssey as well as on two real-world benchmarks. Our data and code are publicly available at: https://pointodyssey.com

1. Introduction

In a variety of computer vision tasks, large-scale annotated datasets have provided a highway for the development of accurate models. In this paper, we aim to provide such a highway for the task of *fine-grained long-range tracking*. The goal of fine-grained long-range tracking is: given any pixel coordinate in any frame of a video, track the corresponding world surface point for as long as possible.

While there exist multiple generations of datasets targeting fine-grained short-range tracking (*i.e.*, optical flow) [6, 13, 40], and annually updated datasets targeting several forms of coarse-grained long-range tracking (*i.e.*, single-object tracking [20], multi-object tracking [32], video object segmentation [45]), there are only a handful of works at the intersection of fine-grained *and* long-range tracking.

Harley *et al*. [25] and Doersch *et al*. [16], train fine-grained trackers on unrealistic synthetic data (FlyingThings++ [40, 25] and Kubric-MOVi-E [24]), consisting of random objects moving in random directions on random backgrounds, and test on real-world videos with sparse human-provided annotations (BADJA [8] and TAP-Vid [16]). While it is interesting that generalization to real video emerges from these models, the use of such simplistic training data precludes the learning of long-range temporal context, and scene-level semantic awareness. We argue that long-range point tracking should not be treated as an extension of optical flow, where naturalism might indeed be discarded without ill effect [50]. Pixels in real video may move somewhat unpredictably, but they take a journey which reflects a variety of modellable factors, including camera shake, object-level motions and deformations, and multi-object relationships such as physical and social interactions. Realizing the grand scope of this problem, both in our data and in our methods, is critical for progress.

† Corresponding author

	MPI Sintel [13]	Flyingthings++ [40, 25]	Kubric [24]	TAP-Vid-Kinetics [16]	TAP-Vid-DAVIS [16]	PointOdyssey
Resolution	436×1024	540×960	256×256	$\geq 720 \times 1280$	1080×1920	540×960
Frame rate	24	8	8	25	25	30
Avg. trajectory count	436×1024	1,024	Flexible	26.3	21.7	18,700
Avg. span of trajectories	4%	100%	100%	30%	30%	100%
Avg. frames per video	50	8	24	250	67	2,035
Training frames	1064	21818	Flexible	-	-	166K
Validation frames	-	4248	-	-	-	24K
Test frames	564	2247	-	297K	1999	26K
Total point annotations	7×10^8	3×10^8	-	8×10^7	4×10^5	4.9×10^{10}
Depth & normals	✓	✓	✓	×	×	✓
Segmentation masks	✓	✓	✓	×	×	✓
Retargeted motion	×	×	×	×	×	✓
Scene randomization	×	×	✓	×	×	✓
Multiple views	×	×	×	×	×	✓
Continuous	✓	✓	✓	×	✓	✓
Object-object interaction	✓	×	✓	×	×	✓
Human-object interaction	✓	×	×	✓	✓	✓
Human-human interaction	✓	×	×	✓	✓	✓

Table 1: **Comparison of point tracking datasets.** PointOdyssey is larger, has longer videos, and includes trajectories which reflect interactions between the objects and the scene. Note that the TAP-Vid datasets are real-world, with sparse human annotations, and are typically reserved for testing [16], whereas most synthetic datasets provide train/test splits.

We propose *PointOdyssey*, a large-scale synthetic dataset for the training and evaluation of long-term fine-grained tracking. Our dataset aims to provide the complexity, diversity, and naturalism of real-world video, with pixel-perfect annotation only possible in simulation. Besides the length of our videos, the key aspects differentiating our work from prior synthetic datasets are (1) we use motions, scene layouts, and camera trajectories mined from real-world videos and motion captures (as opposed to being random or hand-designed), and (2) we use domain randomization on a wider range of scene attributes, including environment maps, lighting, human and animal bodies, camera trajectories, and materials (similar to Shen *et al*. [48]). Thanks to progress in the availability of high-quality assets and rendering tools, we are also able to deliver better photo-realism than possible in years past.

The motion profiles in our data come from large-scale motion-capture datasets of humans and animals [38, 34]. We use these captures to drive humanoids and animals in outdoor scenes, producing realistic long-range trajectories. In outdoor scenes, we pair these actors with 3D assets randomly scattered on the ground plane, which react to the actors according to physics (*e.g*., being kicked away as the feet collide with the objects). To produce realistic indoor scenes, we use motion captures of indoor scenes [67, 66], and *manually replicate* the capture environments in our simulator, allowing us to re-render the exact motions and interactions, and preserve their scene-aware nature. Finally, we import camera trajectories computed from real video [35], and attach additional cameras to the synthetic humans' heads, giving challenging multi-view data of the scenes. Our capture-driven approach is in contrast to the mostly random motion patterns used in Kubric [24] and FlyingThings [40]. We hope that our data will encourage the development of tracking methods which use scene-level cues to provide strong priors on tracking, pushing past the tradition of relying entirely on bottom-up cues such as feature-matching.

Our data's visual diversity stems from a large set of simulated assets: 42 humanoid shapes with artist-made textures, 7 animals, 1K+ object/background textures, 1K+ objects, 20 unique 3D scenes, and 50 environment maps. We randomize the scene lighting to achieve a wide range of dark and bright scenes. We also render dynamic fog and smoke effects into our scenes, introducing a form of partial occlusion entirely missing from FlyingThings and Kubric.

PointOdyssey unlocks a variety of new challenges, one of them being: how to use long-range temporal context. Since prior datasets have only included short videos for training (< 30 frames, see Table 1), existing models only exploit similarly short temporal context. For example, the current state-of-the-art method Persistent Independent Particles (PIPs) [25], uses an 8-frame temporal window when tracking. As a step toward leveraging *arbitrarily long* temporal context, we propose some modifications to PIPs [25], greatly widening its 8-frame temporal window, and incorporating a template-update mechanism. Experimental results show that our method achieves higher tracking accuracy than all existing methods, both on the PointOdyssey test set and on real-world benchmarks.

In summary, the main contribution of this paper is *PointOdyssey*, a large-scale synthetic dataset for long-term point tracking, which aims to reflect the challenges—and opportunities—of real-world fine-grained tracking. The dataset, and the code for the simulation engine, are available at: https://pointodyssey.com

2. Related Work

Motion Datasets. For many years, the Middlebury dataset [6] was the primary benchmark for stereo and motion estimation methods. This dataset contains a mix of synthetic and real data, with high-quality annotations, but is a very small dataset by today's standards (< 100 frames). The MPI Sintel dataset [13] provided a large step forward in visual and motion diversity, by extracting 1064 frames from a movie animated in Blender [9], including lighting variation, shadows, specular reflections, complex materials, and atmospheric effects. Our dataset is similar to Sintel, but is orders of magnitude larger, both in overall frame count and in the length of the video clips, and is also far more realistic, making use of rendering advancements in Blender.

The KITTI dataset [23] provides stereo and flow annotations for real-world driving scenes. Real-world annotation is difficult, and therefore approximated: the authors use LiDAR combined with egomotion information to estimate motion in the static parts of the scene, and then fit 3D models to the cars to estimate the motion of car pixels. We opt for synthetic data generation to avoid these approximations and to ensure perfect fine-grained ground truth.

A series of synthetic datasets have been introduced specifically for training neural nets for motion estimation: FlyingChairs [17], FlyingThings3D [40], FlyingThings++ [25], AutoFlow [50], and Kubric [24]. These datasets consist of random objects moving in random directions on random backgrounds, yielding unrealistic but extremely diverse data. Of these, only FlyingThings++ [25] and Kubric-MOVi-E [24] provide multi-frame trajectories (as opposed to 2-frame motion). Our dataset has similar motivations, in terms of enabling generalization via diversity, but is targeted toward longer-range tracking—across thousands of frames, instead of merely dozens. Our dataset also includes humans, which interact with each other and with the scene, which we hope will give advantage to methods that use high-level contextual cues (such as scene layout), in addition to the low-level motion and appearance signals.

The recently released TAP-Vid benchmark [16] aligns well with our work: it argues for the importance of fine-grained multi-frame tracking, and suggests a train/test pipeline where training happens in synthetic data (Kubric-MOVi-E [24] and RGB-stacking [33]), and testing happens in real data, which consists of manually annotated point tracks for videos in Kinetics [30] and DAVIS [46]. We show that by training in our new richer synthetic data, we improve performance on the TAP-Vid test set. The "test" split of our dataset also covers a gap in TAP-Vid by providing accurate annotations *during occlusions*, while TAP-Vid only provides annotations during visibility.

There is also a long line of work which trains directly on unlabelled data, using a variety of auxiliary objectives to encourage tracking to emerge [65, 60, 62, 29, 7]. An advantage of these works is that they need not worry about a sim-to-real gap, because they train directly on real video. On the other hand, current rendering tools deliver such high photo-realism that the risk of a sim-to-real gap may be much smaller than seen in years past, making synthetic supervision increasingly viable [18, 61].

Motion Understanding. Early motion estimation methods cast point tracking as an optimization problem defined on handcrafted features [27, 37, 57, 52, 12], and these techniques continue to drive structure-from-motion [11, 31, 43] and simultaneous localization and mapping systems [54]. Given the success of neural networks in other computer vision tasks, researchers now typically train deep neural nets to solve the task in a feedforward manner [17, 28], or mix feedforward and iterative inference [51, 55].

While most early work focuses on estimating optical flow (the motion field that links two consecutive frames), there has recently been a push to estimate fine-grained correspondences across multiple frames. PIPs [25] estimates 8-frame trajectories for pixels, using a learned iterative inference procedure that considers match costs and an implicit temporal prior, considering all 8 timesteps jointly with a powerful MLP-Mixer [56]. These 8-frame trajectories can be chained across time to produce longer-range tracks, but these longer tracks are more susceptible to drift, and slow to compute. TAP-Net [16] estimates correspondences for pixels by taking the argmax of frame-by-frame cost maps, which are computed efficiently using time-shifted convolutions [36]. Empirically, TAP-Net outperforms PIPs when there are long occlusions or hard cuts in the video, likely because the 8-frame temporal window in PIPs is incapable of resolving occlusions that exceed this window, and because hard cuts are inconsistent with the learned prior [16].

In this work, we extend PIPs by eliminating its hard 8-frame constraint, allowing it to take much wider temporal context into account. We achieve this by replacing the MLP-Mixer component (in which some parameters were tied to the size of the temporal window), with a deep 1D convolutional network (in which fixed-length kernels are applied to arbitrary temporal spans). We show that our model, trained from scratch in PointOdyssey, outperforms both PIPs and TAP-Net. Additionally, we retain a key advantage of PIPs over TAP-Net, which is the ability to produce reasonable estimates *during occlusions*, by tracking multiple timesteps jointly instead of frame-by-frame.

3. PointOdyssey Dataset

A sample from our dataset is shown in Fig. 1, and an overview of our data generation pipeline is shown in Fig. 2. To generate complex but realistic long-range motion, we use humanoids, robots, and animals, driven by motion capture data [38, 34, 67, 66]. This allows us to render long-term dynamic sequences that incorporate long-range inter-

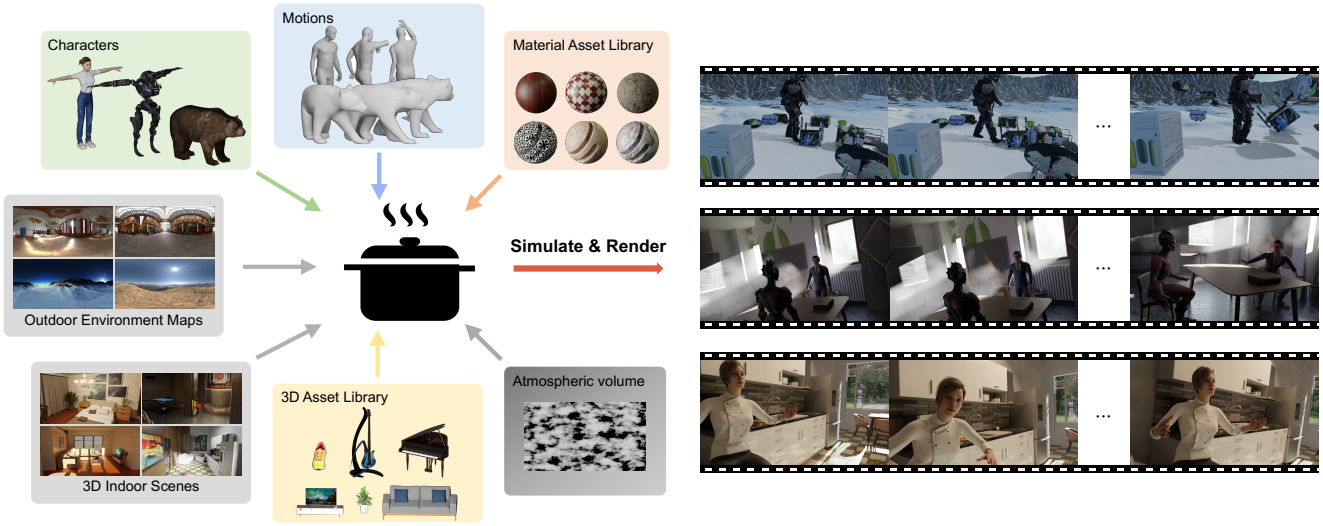


Figure 2: **Overview of our data generation pipeline.** We randomly generate physically realistic and semantically plausible scenes, by sampling human and animal subjects, motion trajectories for the subjects and the camera, 3D physical assets, materials, environment maps for outdoor scenes, manually created environments for indoor scenes, as well as lighting and atmospheric effects. From these scenes we render videos, paired with various ground truth.

actions between the deformable characters and the 3D environments. We maximize the diversity of the dataset by randomizing the scenes with various materials, textures, and lighting. To add further visual complexity, we introduce random noise to the scene volume density to create changing fog and smoke, which act as a natural occluder and have a significant impact on the appearance and visibility of the scene. This section summarizes our data collection process.

3.1. Long-Term Motion Data

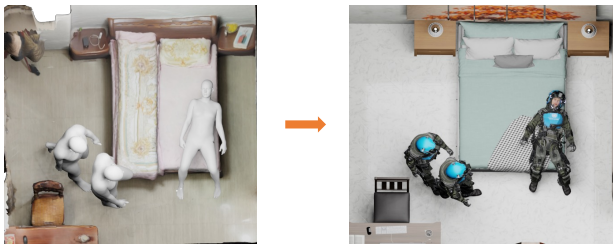
Deformable Characters. We collect 42 open-sourced artist-designed humans and robots from BlenderKit [1], Mixamo [2], and TurboSquid [4], along with 7 animals from DeformingThings4D [34]. These assets provide high-poly meshes, along with photorealistic materials and textures, and are rigged to enable animation.

Motion Retargeting. To animate the humanoid characters, we use real-world human motion data [38, 67, 66]. We retarget the source motions represented as SMPL-X [44] sequences to target characters, using the motion retargeting algorithm from the Rokoko Toolkit [49]. Defining S_{Rig} as the rig of the SMPL-X human model and T_{Rig} as the rig of the target character with z-axis up in the resting body pose, we equalize the scale between two rigs as:

$$s = \frac{Z_{max}(T_{Rig}) - Z_{min}(T_{Rig})}{Z_{max}(S_{Rig}) - Z_{min}(S_{Rig})}, \quad (1)$$

where s is then applied to the source rig as $S'_{Rig} = S_{Rig}/s$. Defining $B_i(p_i)$ as a bone in a rig parameterized by p_i, (*e.g.*, head and tail location and rotation) we align the source rig with the target rig, setting $p_i^{S^*} = p_j^T$, where $p_i^{S^*}$ is the parameter of the bone B_i in the source rig, and p_j^T denotes the parameter of the corresponding bone in the target rig, using the bone mapping between the two rigs. Using the aligned source rig S^*_{Rig}, we copy the animation to the target rig.

For animals, since the motions in DeformingThings4D [34] are already bound to the meshes, we do not need a retargeting process.



Human motion in a scanned scene — Animated character in the rebuilt scene

Figure 3: We use real-world motion capture data [67, 66] within 3D scenes manually re-built to match the motion capture environments.

3.2. 3D Environment Context

Our dataset contains outdoor scenes, which involve randomized but physically coherent agent-object and object-object interactions, and indoor scenes, which involve realistic agent-scene and agent-agent interactions.

Outdoor Scenes. Similar to Kubric [24], we populate outdoor scenes with random rigid objects from GSO [19] and PartNet [42]. We animate our deformable characters to move around in these scenes, and treat these characters as passive objects with infinite mass, causing the scattered rigid objects to react as though being kicked. We also apply random forces to the rigid objects at random timesteps, to

Figure 4: We randomize the textures, materials, atmospheric volume, and lighting, to maximize data diversity.

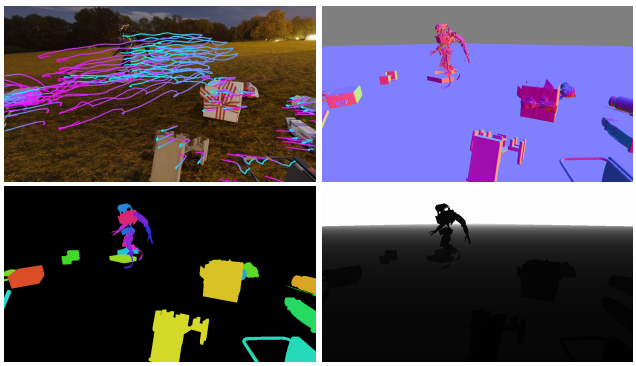


Figure 5: We export 2D and 3D point trajectories, instance masks, depth, normals, and camera calibration data.

create difficult near-random motion trajectories, with realistic physical collisions. We use HDR environment textures collected from PolyHaven[3] mapped into a dome-like region [41] to simulate natural backgrounds.

Indoor Scenes. We manually build twenty 3D indoor scenes to replicate specific 3D environments from our motion capture datasets [67, 66], matching the scene layouts and furniture as closely as possible, sourcing furniture assets from Blenderkit [1] and 3D-FRONT [21, 22]. We then use motion capture data from these same scenes to animate our characters in the environments, yielding collision-free and naturalistic motion, as shown in Fig. 3. We note that unlike the outdoor scenes and unlike prior work, these motions reflect true affordances of the 3D environments.

3.3. Camera Motion

For outdoor scenes, we drive the camera using trajectories extracted from YouTube videos via structure-from-motion [35]. For indoor scenes, we manually create cinematic camera trajectories consisting of orbits, swoops, and zooms, as well as render ego-centric videos by attaching cameras to the heads of the virtual subjects. Similar to real-world egocentric video [15], our synthetic ego-centric views yield particularly challenging motion trajectories.

3.4. Scene Randomization

We add diveristy by randomizing our synthetic scenes, in steps similar to iGibson [48]. For indoor scenes, we randomize the texture of floors, walls, and ceilings, by sampling from 80 high-quality materials from BlenderKit [1], and randomize the lighting. For outdoor scenes, we randomize the textures of objects by sampling from 1000 texture maps from GSO [19]; we randomize the appearance of the animals by sampling from 24 high-fidelity fur materials from Blenderkit; we randomize the background by sampling from 50 4K-resolution HDR images from PolyHaven [3]. We additionally generate fog and smoke by adding procedural atmospheric effects to the scene volume. As shown in Fig. 4, these scene randomization steps add diversity and difficulty to the data.

3.5. Annotation Generation

We generate point trajectories by exporting tracked 2D and 3D coordinates of random foreground and background vertices. We additionally compute visibility annotations, by comparing the depth of the tracked points to the rendered depth values at the projected coordinates. As shown in Fig. 5, we also export depth, normals, instance segmentation, camera extrinsics, and camera intrinsics. While our focus is on point tracking, we hope these extra annotations will support a wide set of applications.

3.6. Statistics

Our dataset consists of 43 outdoor scenes and 61 indoor scenes, totaling 216K 540×960 images at 30 FPS. The data was rendered in 2600 GPU hours using the Cycles engine in Blender. We divide the dataset into 166K frames for training, 24K frames for validation, and 26K frames for testing. Table 1 summarizes key statistics of our dataset compared to related works.

4. Long-Term Tracking with PIPs++

In this section, we propose a method that takes advantage of PointOdyssey's realistic long-range motion annotations, both to establish a reasonable benchmark on the dataset's "test" split, as well as to improve state-of-the-art on real-world performance. We base our approach on "Persistent Independent Particles" (PIPs) [25], a state-of-the-art method for fine-grained tracking. Its main advantage over prior work is that it inspects 8 frames at a time, whereas prior work typically used just 2. This gives the model some robustness to occlusions, since it can use frames before and after occlusions to estimate the missing parts of the trajectory. We highlight two key limitations, which we aim to address in the following subsections: (1) the temporal field of view is *only* 8 frames, meaning that the method cannot survive occlusions which are longer than this timespan, and (2) the model relies entirely on the first-frame appearance of the target, making correspondence difficult across appearance

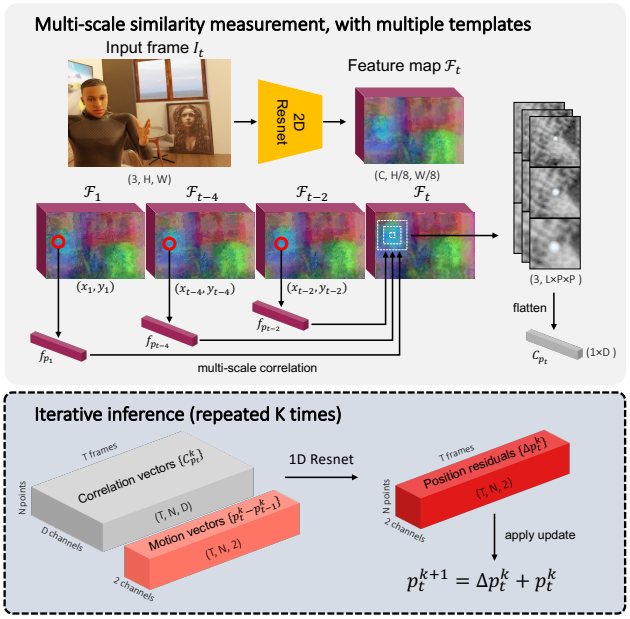


Figure 6: Overview of our method. Top: for any query point p_t, we first compute the similarity cost volume C_{p^t}. We propose to leverage informative features within the local context and incorporate global guidance to maintain consistent and robust tracking. Bottom: we iteratively update the trajectory of positions with a 1D Resnet.

changes. We begin by describing the PIPs architecture in detail, and then describe how we resolve these limitations.

4.1. Preliminaries (PIPs)

PIPs takes an 8-frame RGB video as input, along with a coordinate $p_1 = (x_1, y_1)$ indicating a target to track. It produces a 8×2 matrix as output, representing the trajectory of the target across the given frames. This process can be repeated across 8-frame segments, to produce long-range tracks. An arbitrary number of targets can be tracked in parallel, but there is no message-passing between the trajectories (hence persistent *independent* particles). Inference has two main stages: initialization, and an iterative loop.

Initialization. Before tracking begins, we compute a feature map $\mathcal{F}_t$ for each frame, with a 2D residual convnet [26]. We obtain a vector representing the appearance of the target, by bilinear sampling at the target's position on the first frame's feature map: $f_{p_1} = \texttt{sample}(\mathcal{F}_t, p_1)$. Using this first coordinate and feature vector, we initialize a list of positions and features, $\{(p_t, f_t)\} = \{(p_1, f_1)\}$ for all $t \in \{1, 2, \cdots, T\}$.

Iterative Updates. The main inference stage is an iterative update process, which primarily aims to improve the positions p_t, so that they track the target more closely. Denoting the current iteration's workspace on iteration k as $\{(p_t^k, f_t^k)\}$, we begin an iteration by measuring the similarity between the per-timestep feature vectors and the per-timestep feature maps, within local windows centered at the current estimates:

$$C_{p_t}^k = f_t^k \otimes \texttt{multicrop}(\mathcal{F}_t, p_t^k)/\sigma \tag{2}$$

where $\otimes$ denotes a dot product, $\texttt{multicrop}(\mathcal{F}_t, p_t^k)$ produces multi-scale crops from $\mathcal{F}_t$ centered at p_t^k, and σ is a temperature parameter. A 12-block MLP-Mixer [56] takes these correlations as input, along with the apparent point motions $p_t^k - p_1^k$, and the features f_t, and produces updates to the full sequence of positions and features: $\{\Delta p_t^k, \Delta f_t^k\}$. These updates are then applied additively, which leads to sampling new local correlations in the next iteration. The feature vectors are eventually fed to a linear layer, which produces per-timestep visibility estimates.

Limitations. PIPs is locked to the temporal field of view that it is trained with, due to the use of the MLP-Mixer in the iterative stage. While the tracker can be chained across time to produce long tracks, these are sensitive to drift, especially when the target becomes occluded beyond the range of the temporal window. We also note that the *visibility-aware* chaining proposed in PIPs cannot be easily parallelized, and so long-range multi-particle tracking is computationally very expensive. Additionally, we point out that the feature-update operator *cannot* perform a task resembling a template-update, because it does not have access to the input frames. The residual updates to the feature list likely only serve visibility estimation.

4.2. Expanding the temporal field of view (PIPs+)

Our first proposed modification to PIPs aims to widen its temporal field of view, and enable longer-range tracking. The key component here is the MLP-Mixer, which (by design) has a fixed-width temporal field of view, set to 8 in PIPs. We propose to replace the MLP-Mixer with an 8-block 1D Resnet [26], doing convolutions across time.[1] This means learning kernels that slide across the time axis. Each residual block consists of two convolution layers with kernel size 3, with instance normalization [58] and ReLU [5]. At the final block, the receptive field is 35 timesteps. Note however that since this module is *iterated* during inference, the effective receptive field is much larger.

We find that this convolutional variant of PIPs, which we name PIPs+, improves long-range tracking accuracy, and also speeds inference in long videos (from 4 FPS on average, to 55 FPS on average, at 720×1080 on an Nvidia V100 GPU). The convolutional design enables us to train and test with videos of different lengths, similar to how fully convolutional 2D networks can train and test with different image

[1] In the PIPs paper, Harley et al. [25] briefly mention an unsuccessful attempt at using temporal convolutions instead of the MLP-Mixer. It may be that their effort failed due to lack of long-sequence training data.

sizes, but in practice we find it is still important to train and test with roughly similar sequence lengths.

4.3. Extending to multiple templates (PIPs++)

Tracked targets are likely to undergo appearance changes across time, and it is important to keep up with these changes. In the original PIPs architecture, the first-frame feature f_1 (ignoring the negligible feature-update step already discussed) is used for cross-correlation on every frame in the temporal span. This is liable to produce weak matches after appearance changes, and erroneous matches during occlusions. Our second proposed modification to PIPs aims to tackle this "template update" problem [39].

Our main idea is simply to accommodate appearance changes by collecting "recent appearance" templates along the estimated trajectory, to complement the "initial appearance" template from the first frame [10]. Specifically, when computing local correlations for frame t, we use the estimated trajectory to extract new features at fixed temporal offsets from this timestep, such as $\{t-2, t-4\}$. This means using p_{t-k} to extract a temporary feature vector $f_{t-k} = \texttt{sample}(\mathcal{F}_{t-k}, p_{t-k})$. We use these features to compute additional correlations in the current frame's feature map $\mathcal{F}_t$, as done in Eq. 2. This process is illustrated in Fig. 6. The key idea is that if tracking was successful on one of these offset frames, then the extracted feature will reflect the updated appearance of the target, and will yield a more-useful correlation map than the one from f_1. These multiple correlation maps are simply concatenated, increasing the channels input to our 1D Resnet. Note that similar to current methods in object tracking [63], we always retain the initial template f_1, to help prevent "forgetting".

An alternative strategy here would be to generate templates exclusively from timesteps with high visibility confidence, as commonly done in object tracking [63]. While this is intuitively appealing, we note that our simpler strategy instead allows the model to (temporarily) capture the appearance of an *occluder*, which can sometimes be the appropriate entity to track (*e.g.*, during object self-occlusions).

Our multi-template strategy, combined with temporally-flexible computation, obviates the residual feature updates, so we simply omit this component. We also omit visibility estimation for simplicity. We name our full model PIPs++.

5. Experiments

In this section we explain our experimental setup and results. We recommend watching the supplementary video for better visualization of our dataset and results.

5.1. Experimental setup

Baselines. We benchmark point trackers, PIPs [25], TAP-Net [16], our proposed PIPs+ and PIPS++, an optical flow method RAFT [55] (estimating the flow between consecutive pairs of frames and chaining the flows to form trajectories), and a strong feature-matching method, DINO [14].

Implementation details. We use the official code of PIPs [25], RAFT [55], and DINO [55], and reimplement TAP-Net [16] in PyTorch. For RAFT and DINO, we use the pretrained weights for evaluation. We train and test PIPs and TAP-Net on our dataset, using 4-8 A5000 GPUs in parallel. In addition to evaluating on the PointOdyssey test set, we evaluate on TAP-Vid-DAVIS [16] and CroHD [53], which are real-video evaluation benchmarks. TAP-Vid-Davis mostly consists of videos of animals and humans, with sparse tracks annotated on foreground and background points; CroHD consists of surveillance-like recordings of crowds (*e.g.*, in train stations), with tracks annotated on all human heads. We leave out TAP-Vid-Kinetics [16], as it contains hard cuts, while our focus is on continuous video.

5.2. Evaluation

Evaluation metrics. We report the average position accuracy δ_{avg} as proposed in TAP-Vid [16]. This measures the percentage of tracks within a threshold distance to ground truth, averaged over thresholds $\{1, 2, 4, 8, 16\}$, defined in a normalized resolution of 256×256. We use Median Trajectory Error (MTE) to measure the distance between the estimated tracks and ground truth tracks. While Harley et al. [25] reported average trajectory error (ATE) using a *mean*, we find the median more informative, as it is less sensitive to outliers. We also measure a "Survival" rate, which we define as: the average number of frames until tracking failure, and report this as a ratio of video length. Failure is when L2 distance exceeds an error threshold, i.e., 50 pixels for long-term data and 16 pixels for short-term data in the normalized 256×256 resolution.

Quantitative results. We compile our results in Table 2. First, inspecting results across rows (*i.e.*, comparing methods), we can see that PIPs+ and PIPs++ achieve the best results among all methods, demonstrating the effectiveness of the wide temporal awareness. The narrow gap between PIPs+ and PIPs++ suggests that the multi-template strategy has only a modest effect, but is helpful on average. The results also demonstrate that prior methods perform better on real-world datasets when they are re-trained (from scratch) in our dataset. An exception here is TAP-Net [16], where model trained by the authors (on Kubric) performs best; this is likely due to our smaller compute budget. All of our models are trained on 4-8 GPUs (c.f. 64 TPU-v3 cores in the original TAP-Net). Inspecting the results across columns (*i.e.*, comparing datasets), we observe that PointOdyssey appears to be a more challenging benchmark than TAP-Vid-DAVIS [16] and CroHD [53]. We can also observe that the ranking of methods appears consistent among PointOdyssey and the two real-world datasets,

Method	Training	PointOdyssey			TAP-Vid-DAVIS [16]			CroHD [59]		
		MTE ↓	δ ↑	Survival$_{50}$ ↑	MTE ↓	δ ↑	Survival$_{16}$ ↑	MTE ↓	δ ↑	Survival$_{50}$ ↑
TAP-Net [16]	Pretrained	92.00	23.75%	17.01%	10.56	53.40%	60.14%	101.12	23.39%	34.28%
RAFT [55]	Pretrained	319.46	10.07%	32.61%	9.16	50.93%	70.68%	82.76	15.82%	62.22%
DINO [14]	Pretrained	118.38	8.61%	31.29%	20.14	34.35%	60.22%	116.80	8.46%	37.11%
PIPs [25]	Pretrained	147.45	16.53%	32.90%	4.75	64.01%	82.20%	19.23	40.23%	75.15%
TAP-Net [16]	Kubric	92.70	26.92%	9.59%	32.57	39.68%	59.41%	99.15	18.08%	28.43%
TAP-Net [16]	PointOdyssey	63.51	28.37%	18.27%	18.49	44.46%	62.54%	60.94	22.24%	35.00%
PIPs [25]	PointOdyssey	63.98	27.34%	42.33%	4.30	66.97%	86.01%	11.94	44.02%	74.93%
PIPs+	PointOdyssey	28.93	32.41%	49.88%	4.61	69.13%	88.11%	**11.20**	**45.51%**	75.07%
PIPs++	PointOdyssey	**26.95**	**33.64%**	**50.47%**	**4.16**	**69.68%**	**89.73%**	11.21	44.09%	**75.43%**

Table 2: Tracking performance on the PointOdyssey test set, TAP-Vid-DAVIS [16], and CroHD [59].

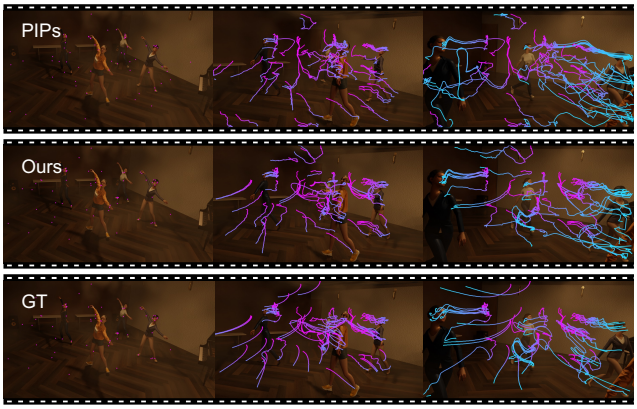


Figure 7: Qualitative results on our dataset. Left to right columns show the start, middle and end frame respectively.

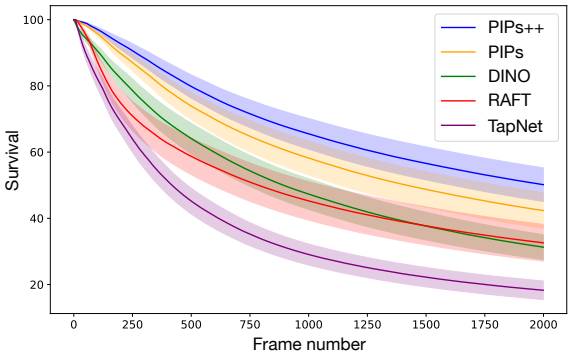


Figure 8: Survival rate over time in PointOdyssey. Higher is better. We show the mean and standard deviation for each method. Note that the endpoints correspond to the values reported in Table 2.

suggesting a correlation between progress on PointOdyssey and progress on videos in the wild. Fig. 8 plots survival rate over time in PointOdyssey, revealing that all methods struggle to keep tracks "alive" over long durations, but the PIPs models degrade more slowly than the rest.

Qualitative results. We show qualitative results in Fig. 7. Our method generates more-stable point trajectories compared to PIPs [25] and other baselines. Please see the supplementary for video visualizations. We find that all methods have difficulty with targets which are close to boundaries (*e.g.*, targets on thin objects are the hardest).

6. Limitations

PointOdyssey currently lacks large outdoor scenes where the camera travels a large distance, which is, for example, a frequent scenario in driving data [18]. It would be interesting to explore long-range agent-scene and agent-agent interactions in that context. We also note that our human and animal motion profiles are limited by our base datasets, and this constraint could be lifted with the help of recent generative models [47, 64]. While the focus in this paper is on point tracking, our dataset connects to a wide range of applications which we have not yet explored, such as 3D scene flow estimation, novel view synthesis in dynamic scenes (which would be especially challenging with PointOdyssey's atmospheric effects), human and animal pose estimation, and ego-centric vision. Finally, while PIPs++ takes a step toward modelling longer-range temporal priors, it is still a fairly low-level tracker, relying entirely on appearance-matching cues and a temporal prior. This leaves open the challenge of leveraging scene-level and semantic cues for tracking, where we expect PointOdyssey's training data will be especially valuable.

7. Conclusion

PointOdyssey is a large-scale synthetic dataset, and data generator, for long-term point tracking. The data is diverse and naturalistic, making it an ideal resource for training general-purpose fine-grained trackers. We demonstrate its usefulness through a new tracker called PIPs++, which leverages long-term temporal context and outperforms state-of-the-art. PointOdyssey also opens opportunities for developing trackers which utilize scene-level and semantic cues, though we have not explored this yet. We

hope our work will also be useful beyond point tracking, enabling work in 3D and 4D scene analysis, and higher-level video understanding.

Acknowledgments. The authors thank Andrew Zisserman for feedback on an early version of the title. This work was supported by the Toyota Research Institute under the University 2.0 program, ARL grant W911NF-21-2-0104, and a Vannevar Bush Faculty Fellowship.

References

[1] Blenderkit. www.www.blenderkit.com. 4, 5

[2] Mixamo. www.mixamo.com. 4

[3] Polyhaven. www.polyhaven.com/hdris. 5

[4] Turbosquid. www.turbosquid.com. 4

[5] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv:1803.08375*, 2018. 6

[6] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 92:1–31, 2011. 1, 3

[7] Zhangxing Bian, Allan Jabri, Alexei A. Efros, and Andrew Owens. Learning pixel trajectories with multiscale contrastive random walks. *CVPR*, 2022. 3

[8] Benjamin Biggs, Thomas Roddick, Andrew Fitzgibbon, and Roberto Cipolla. Creatures great and SMAL: Recovering the shape and motion of animals from video. In *ACCV*, pages 3–19, 2018. 1

[9] Blender Online Coummunity. Blender. Blender Foundation, https://www.blender.org/. 3

[10] Vasyl Borsuk, Roman Vei, Orest Kupyn, Tetiana Martyniuk, Igor Krashenyi, and Jiři Matas. Fear: Fast, efficient, accurate and robust visual tracker. In *ECCV*. Springer, 2022. 7

[11] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3D shape from image streams. In *CVPR*, 2000. 3

[12] Thomas Brox and Jitendra Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *TPAMI*, 33, 2011. 3

[13] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, pages 611–625, 2012. 1, 2, 3

[14] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021. 7, 8

[15] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. The EPIC-KITCHENS dataset: Collection, challenges and baselines. *TPAMI*, 43(11):4125–4141, 2020. 5

[16] 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. *NeurIPS Datasets and Benchmarks*, 2022. 1, 2, 3, 7, 8

[17] 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 *ICCV*, 2015. 3

[18] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *CORL*, 2017. 3, 8

[19] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *ICRA*, 2022. 4, 5

[20] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. LaSOT: A high-quality benchmark for large-scale single object tracking. In *CVPR*, 2019. 1

[21] Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3D-FRONT: 3D furnished rooms with layouts and semantics. In *ICCV*, 2021. 5

[22] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3D-FUTURE: 3D furniture shape with texture. *IJCV*, 2021. 5

[23] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *IJRR*, 2013. 3

[24] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *CVPR*, 2022. 1, 2, 3, 4

[25] Adam W Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. Particle video revisited: Tracking through occlusions using point trajectories. In *ECCV*, 2022. 1, 2, 3, 5, 6, 7, 8

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6

[27] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17, 1981. 3

[28] 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 *CVPR*, 2017. 3

[29] Allan Jabri, Andrew Owens, and Alexei A Efros. Space-time correspondence as a contrastive random walk. *NeurIPS*, 2020. 3

[30] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv:1705.06950*, 2017. 3

[31] Chen Kong and Simon Lucey. Deep non-rigid structure from motion with missing data. *TPAMI*, 2021. 3

[32] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler. MOTChallenge 2015: Towards a benchmark for multi-target tracking. *arXiv:1504.01942*, 2015. 1

[33] Alex X Lee, Coline Manon Devin, Yuxiang Zhou, Thomas Lampe, Konstantinos Bousmalis, Jost Tobias Springenberg,

Arunkumar Byravan, Abbas Abdolmaleki, Nimrod Gileadi, David Khosid, et al. Beyond pick-and-place: Tackling robotic stacking of diverse shapes. In *CoRL*, 2021. 3

[34] Yang Li, Hikari Takehara, Takafumi Taketomi, Bo Zheng, and Matthias Nießner. 4DComplete: Non-rigid motion estimation beyond the observable surface. In *ICCV*, 2021. 2, 3, 4

[35] 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 *CVPR*, 2019. 2, 5

[36] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *ICCV*, 2019. 3

[37] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. *IJCAI*, 2:674–679, 1981. 3

[38] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. AMASS: Archive of motion capture as surface shapes. In *ICCV*, 2019. 2, 3, 4

[39] Iain Matthews, Takahiro Ishikawa, and Simon Baker. The template update problem. *TPAMI*, 26(6), 2004. 7

[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 *CVPR*, 2016. 1, 2, 3

[41] Elie Michel. Lily surface scraper. GitHub repository. https://github.com/eliemichel/LilySurfaceScraper. 5

[42] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *CVPR*, 2019. 4

[43] David Novotny, Nikhila Ravi, Benjamin Graham, Natalia Neverova, and Andrea Vedaldi. C3dpo: Canonical 3d pose networks for non-rigid structure from motion. In *ICCV*, 2019. 3

[44] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *CVPR*, 2019. 4

[45] 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:1704.00675*, 2017. 1

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

[47] Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J. Guibas. Humor: 3d human motion model for robust pose estimation. In *ICCV*, 2021. 8

[48] Bokui Shen, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Claudia Pérez-D'Arpino, Shyamal Buch, Sanjana Srivastava, Lyne P. Tchapmi, Micael E. Tchapmi, Kent Vainio, Josiah Wong, Li Fei-Fei, and Silvio Savarese. iGibson 1.0: a simulation environment for interactive tasks in large realistic scenes. In *IROS*. IEEE, 2021. 2, 5

[49] Matias Sondergaard. Rokoko. GitHub repository. https://github.com/Rokoko/rokoko-studio-live-blender. 4

[50] 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 *CVPR*, 2021. 1, 3

[51] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018. 3

[52] Narayanan Sundaram, Thomas Brox, and Kurt Keutzer. Dense point trajectories by GPU-accelerated large displacement optical flow. In *ECCV*, 2010. 3

[53] Ramana Sundararaman, Cedric De Almeida Braga, Eric Marchand, and Julien Pettre. Tracking pedestrian heads in dense crowd. In *CVPR*, 2021. 7

[54] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual slam algorithms: A survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9(1):1–11, 2017. 3

[55] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020. 3, 7, 8

[56] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *NeurIPS*, 2021. 3, 6

[57] Carlo Tomasi and Takeo Kanade. Detection and tracking of point. *IJCV*, 1991. 3

[58] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv:1607.08022*, 2016. 6

[59] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. Mots: Multi-object tracking and segmentation. In *CVPR*, 2019. 8

[60] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *ECCV*, 2018. 3

[61] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. Tartanair: A dataset to push the limits of visual slam. In *IROS*. IEEE, 2020. 3

[62] Xiaolong Wang, Allan Jabri, and Alexei A Efros. Learning correspondence from the cycle-consistency of time. In *CVPR*, 2019. 3

[63] Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. Learning spatio-temporal transformer for visual tracking. In *ICCV*, 2021. 7

[64] Hongwei Yi, Chun-Hao P. Huang, Shashank Tripathi, Lea Hering, Justus Thies, and Michael J. Black. MIME: Human-aware 3D scene generation. In *CVPR*, June 2023. 8

[65] Jason J Yu, Adam W Harley, and Konstantinos G Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *ECCVW*, 2016. 3

[66] Siwei Zhang, Qianli Ma, Yan Zhang, Zhiyin Qian, Taein Kwon, Marc Pollefeys, Federica Bogo, and Siyu Tang. Ego-

body: Human body shape and motion of interacting people from head-mounted devices. In *ECCV*, 2022. 2, 3, 4, 5

[67] Yang Zheng, Yanchao Yang, Kaichun Mo, Jiaman Li, Tao Yu, Yebin Liu, C Karen Liu, and Leonidas J Guibas. Gimo: Gaze-informed human motion prediction in context. In *ECCV*, 2022. 2, 3, 4, 5