SpatialTracker: Tracking Any 2D Pixels in 3D Space

Yuxi Xiao[1,3*] Qianqian Wang[2*] Shangzhan Zhang[1,3] Nan Xue[3]
Sida Peng[1] Yujun Shen[3] Xiaowei Zhou[1†]
[1]Zhejiang University [2]UC Berkeley [3]Ant Group

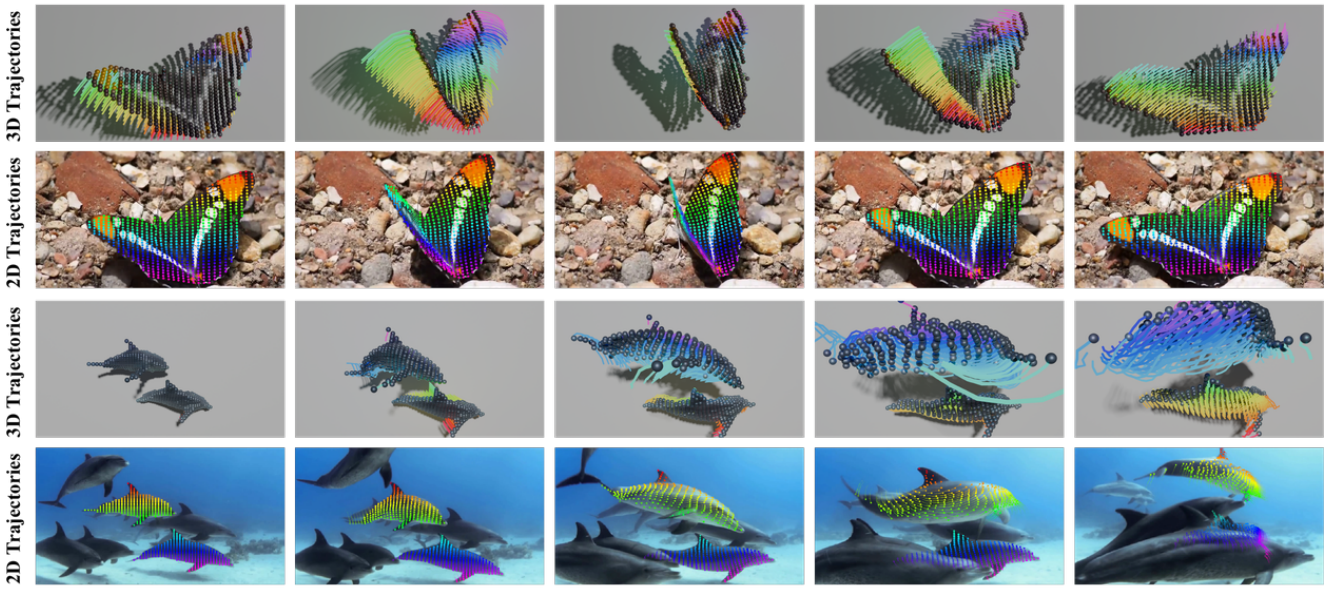

Figure 1. **Tracking 2D pixels in 3D space.** To estimate 2D motion under the occlusion and complex 3D motion, we lift 2D pixels into 3D and perform tracking in the 3D space. Two cases of the estimated 3D and 2D trajectories of a waving butterfly (top) and a group of swimming dolphins (bottom) are illustrated.

Abstract

Recovering dense and long-range pixel motion in videos is a challenging problem. Part of the difficulty arises from the 3D-to-2D projection process, leading to occlusions and discontinuities in the 2D motion domain. While 2D motion can be intricate, we posit that the underlying 3D motion can often be simple and low-dimensional. In this work, we propose to estimate point trajectories in 3D space to mitigate the issues caused by image projection. Our method, named SpatialTracker*, lifts 2D pixels to 3D using monocular depth estimators, represents the 3D content of each frame efficiently using a triplane representation, and performs iterative updates using a transformer to estimate 3D trajectories. Tracking in 3D allows us to leverage as-rigid-as-possible (ARAP) constraints while simultaneously learning a rigidity embedding that clusters pixels into different rigid parts. Extensive evaluation shows that our approach achieves state-of-the-art tracking performance both qualitatively and quantitatively, particularly in challenging scenarios such as out-of-plane rotation. And our project page is available at* https://henry123-boy.github.io/SpaTracker/.

1. Introduction

Motion estimation has historically been approached through two main paradigms: feature tracking [37, 38, 58, 64] and optical flow [1, 19, 62]. While each type of method enables numerous applications, neither of them fully captures the motion in a video: optical flow only produces motion for adjacent frames, whereas feature tracking only tracks sparse pixels.

An ideal solution would involve the ability to estimate

*The first two authors contributed equally. The authors from Zhejiang University are affiliated with the State Key Lab of CAD&CG. †Corresponding author: Xiaowei Zhou.

both *dense* and *long-range* pixel trajectories in a video sequence [55, 56]. Seminal work like Particle Video [56] has bridged the gap by representing video motion using a set of semi-dense and long-range particles. More recently, several efforts [11, 12, 17, 29] have revisited this problem, formulating it as *tracking any point* and addressing it through supervised learning frameworks. While trained solely on synthetic datasets [14, 81], these methods consistently demonstrate strong generalization abilities to real-world videos, pushing the boundaries of long-range pixel tracking through occlusions.

While great progress has been achieved, current solutions still struggle in challenging scenarios, particularly in cases of complex deformation accompanied by frequent self-occlusions. We argue that one potential cause for this difficulty stems from tracking only in the 2D image space, thereby disregarding the inherent 3D nature of motion. As motion takes place in 3D space, certain properties can only be adequately expressed through 3D representations. For example, rotation can be succinctly explained by three parameters in 3D, and occlusion can be simply expressed with z-buffering, but they are much more complicated to express within a 2D representation. In addition, the key component of these methods — using *2D* feature correlation to predict motion updates — can be insufficient. Image projection can bring spatially distant regions into proximity within the 2D space, which can cause the local 2D neighborhood for correlation to potentially contain irrelevant context (especially near occlusion boundaries), thereby leading to difficulties in reasoning.

To tackle these challenges, we propose to leverage geometric priors from state-of-the-art monocular depth estimators [2] to lift 2D pixels into 3D, and perform tracking in the 3D space. This involves conducting feature correlation in 3D, which provides more meaningful 3D context for tracking especially in cases of complex motion. Tracking in 3D also allows for enforcing 3D motion priors [52, 63] such as ARAP constraint. Encouraging the model to learn which points move rigidly together can help track ambiguous or occluded pixels, as their motion can then be inferred using neighboring unambiguous and visible regions within the same rigid group.

Specifically, we propose to represent the 3D scene of each frame with triplane feature maps [10], which are obtained by first lifting image features to 3D featured point clouds and then splatting them onto three orthogonal planes. The triplane representation is compact and regular, suitable for our learning framework. Moreover, it covers the 3D space densely, enabling us to extract the feature vectors of any 3D point for tracking. We then compute 3D trajectories for query pixels through iterative updates predicted by a transformer using features from our triplane representation. To regularize the estimated 3D trajectories with 3D motion prior, our model additionally predicts a rigidity embedding for each trajectory, which allows us to softly group pixels exhibiting the same rigid body motion and enforce an ARAP regularization for each rigid cluster. We demonstrate that the rigidity embedding can be learned self-supervisedly and produce reasonable segmentation of different rigid parts.

Our model achieves state-of-the-art performance on various public tracking benchmarks including TAP-Vid [11], BADJA [4] and PointOdyssey [81]. Qualitative results on challenging Internet videos also demonstrate the superior capability of our model to handle fast complex motion and extended occlusion.

2. Related Work

Optical flow. Optical flow is the task of estimating dense 2D pixel-level motion between a pair of frames. While traditional methods [1, 6, 8, 19, 74, 78] formulate it as an energy minimization problem, recent approaches [13, 23, 24, 61, 76] have demonstrated the ability to predict optical flow directly using deep neural networks. Notably, RAFT [62] employs a 4D correlation volume and estimates optical flow through iteratively updates with a recurrent operator. More recently, transformer-based flow estimators [21, 26, 59, 80] achieved superior performance, showing the strong capacity of the transformer architecture. However, pairwise optical flow methods are not suitable for long-term tracking, as they are not designed to handle long temporal contexts [8, 74]. Multi-frame optical flow methods [25, 28, 32, 54, 69] extend pairwise flow by incorporating multi-frame contexts (typically 3-5 frames), but this remains insufficient for tracking through long occlusions in videos spanning tens or hundreds of frames.

Tracking any point. Recognizing the limitations of optical flow, seminal work Particle Video [56] proposes to represent video motion as a set of long-range particles that move through time, which are optimized by enforcing long-range appearance consistency and motion coherence with variational techniques. However, Particle Video only generates semi-dense tracks and cannot recover from occlusion events [55]. Recently, PIPs [17] revisited this idea by introducing a feedforward network that takes RGB frames of a fixed temporal window (8 frames) as input and predicts the motion for any given query point through iterative updates. However, PIPs tracks points independently, neglecting spatial context information, and will lose the target if they stay occluded beyond the temporal window. Several recent advancements [3, 11, 12, 42, 71, 81] in point tracking have surfaced, addressing some of PIPs' limitations. TAPIR [12] relaxes the fixed-length window constraint by using a temporal depthwise convolutional network capable of accommodating variable lengths. Co-

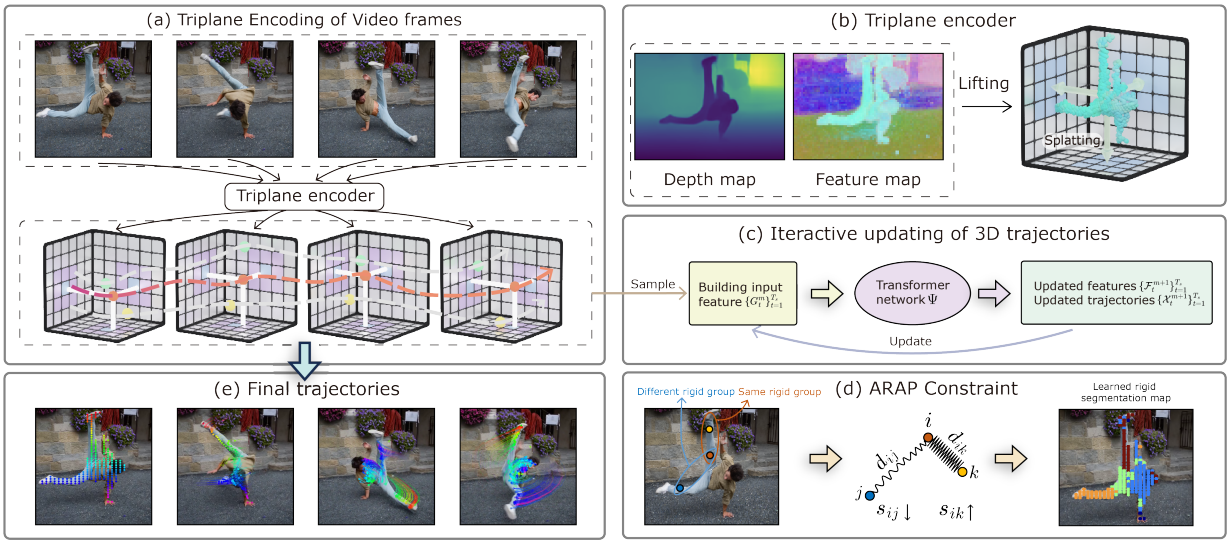

Figure 2. **Overview of Our Pipeline.** We first encode each frame into a triplane representation (**a**) using a triplane encoder (**b**). We then initialize and iteratively update point trajectories in the 3D space using a transformer with features extracted from these triplanes as input (**c**). The 3D trajectories are trained with ground truth annotations and are regularized by an as-rigid-as-possible (ARAP) constraint with learned rigidity embedding (**d**). The ARAP constraint enforces that 3D distances between points with similar rigidity embeddings remain constant over time. Here d_{ij} represents the distance between points i and j, while s_{ij} denotes the rigid similarity. Our method produces accurate long-range motion tracks even under fast movements and severe occlusion (**e**).

Tracker [29] proposed to jointly track multiple points and leverage spatial correlation between them, leading to state-of-the-art performance.

Though significant progress has been made, these works all compute feature correlation in the 2D image space, losing important information about the 3D scene where the motion actually takes place. In contrast, we lift 2D points into 3D and perform tracking in the 3D space. The more meaningful 3D contexts (as opposed to 2D), along with an as-rigid-as-possible regularization, facilitate improved handling of occlusions and enhance tracking accuracy. Previous studies have also explored computing 2D motion with a touch of 3D, e.g., through depth-separated layers [30, 57, 60, 77] or quasi-3D space [71]. However, distinct from their optimization-based pipelines, we perform long-range 3D tracking in a more efficient, feedforward manner.

Scene flow. Scene flow defines a dense 3D motion field of points in a scene. Early work estimates scene flow in multi-view stereo settings [49, 66, 79] through variational optimization [22]. The introduction of depth sensors enabled more effective scene flow estimation from pairs or sequences of RGB-D frames [16, 18, 20, 27, 52, 63, 72]. A considerable number of recent scene flow methods rely on stereo inputs [39, 41], but many of them are tailored specifically for self-driving scenes, lacking generalizability to diverse non-automotive contexts. Another line of research [7, 15, 34, 36, 45, 75] estimates 3D motion from a pair or a sequence of point clouds. An important prior that is often used for scene flow estimation is local rigidity [67, 68], where pixels are grouped into rigidly moving clusters (object or part-level), in either a soft or hard manner. For example, RAFT-3D [63] learns rigid-motion embeddings to softly group pixels into rigid objects. Scene flow estimation is also often solved as a sub-task in non-rigid reconstruction pipelines [31, 35, 43]. For example, DynamicFusion [44] takes depth maps as input and computes dense volumetric warp functions by interpolating a sparse set of transformations as bases. In contrast to prior works, we learn to predict long-range 3D trajectories through supervised learning, providing generalization capabilities for handling complex real-world motion.

3. Method

Given a monocular video as input, our method tracks any given query pixels across the entire video. Different from prior methods that establish correspondences solely in the 2D space, we lift pixels to 3D using an off-the-shelf monocular depth estimator and perform tracking in a 3D space with richer and more spatially meaningful 3D contextual information, thereby enhancing the overall tracking performance.

Fig. 2 presents the overview of our proposed pipeline. We first encode the appearance and geometry information of each frame into a triplane representation (Sec. 3.1). Then we perform iterative prediction of trajectories in the 3D space using these triplanes in a sliding window fashion (Sec. 3.2). We leverage the as-rigid-as-possible (ARAP) 3D motion prior during training to facilitate track-

ing especially in challenging scenarios of occlusion and large motion (Sec. 3.3). Finally, we describe our training strategy in Sec. 3.4.

3.1. Triplane Encoding of Input Video Frames

To perform tracking in the 3D space, we need to lift 2D pixels into 3D and construct a 3D representation that encodes the feature for each 3D location. To this end, we propose to use triplane features as the 3D scene representation for each frame detailed below.

To start with, for each frame, we obtain its monocular depth map using a pretrained monocular depth estimator, alongside multi-scale feature maps generated by a convolutional neural network (CNN). Subsequently, 2D pixels are unprojected into a set of 3D point clouds, where each 3D point is associated with a feature vector. This feature vector is a concatenation of the corresponding image feature and a positional embedding [65] of its 3D location.

While this featured point cloud captures both geometry and appearance information, it is incomplete and only covers visible regions (2.5D). Additionally, its irregular and unordered nature poses challenges for effective learning. One simple solution involves voxelizing the point cloud into a 3D feature volume and completing it with 3D convolutions. Yet, this approach is memory and computationally intensive. To obtain 3D features densely and efficiently, we propose to use triplane feature maps, which are obtained by orthographically projecting and average splatting [46] the featured point cloud onto three orthogonal 2D planes, as illustrated in Fig. 2(b). Finally, additional convolutional layers are applied to process and complete each feature map. This triplane feature encoding process is applied to each video frame. Since we do not assume access to camera poses, each triplane is defined within the camera coordinate frame of its respective frame.

This triplane representation is compact and enables us to efficiently represent the 3D feature for any given 3D point within the field of view. This process involves projecting the point onto three feature planes, extracting its corresponding feature vectors through bilinear interpolation, and fusing them via simple addition.

Note that while similar concepts of triplanes are explored in related fields [10, 48, 73], our focus here is distinct. Rather than learning a triplane to represent the 3D scene from scratch, we directly leverage monocular depth priors to obtain a triplane where the primary objective is to facilitate tracking in the 3D space, introducing a novel perspective to the field of pixel tracking.

3.2. Iterative Trajectory Prediction

Given a set of query pixels in the query frame, Sec. 3.1 allows us to obtain their 3D locations and their corresponding triplane features. We now describe the process to estimate their 3D trajectories across the entire video.

Following CoTracker [29], we partition the video into overlapping windows of length T_s. In each window, we iteratively estimate 3D trajectories for query points over M steps using a transformer. The final 3D trajectories are then propagated to the next window and updated, and this process continues until the end of the video.

Iterative prediction. We now focus on the iterative prediction of 3D trajectories within the first temporal window. Given the 3D location $\boldsymbol{X}_1 \in \mathbb{R}^3$ of a query pixel in the first frame, our goal is to predict its 3D corresponding locations (or in other words, its 3D trajectory) in subsequent frames $\{\boldsymbol{X}_t\}_{t=2}^{T_s}$, where t is the frame index.

Because we adopt an iterative updating strategy to estimate the 3D trajectories, we further denote the prediction at the m-th step as $\{\boldsymbol{X}_t^m\}_{t=2}^{T_s}$. To start with, we initialize $\{\boldsymbol{X}_t^0\}_{t=2}^{T_s}$ to be all equal to $\boldsymbol{X}_1$, and then we iteratively update the 3D trajectory using a transformer Ψ.

Specifically, for the point $\boldsymbol{X}_t^m$ at the m-th iteration, we define its input feature $\boldsymbol{G}_t^m$ to the transformer as:

$$\boldsymbol{G}_t^m = [\gamma(\boldsymbol{X}_t^m), \boldsymbol{F}_t^m, \boldsymbol{C}_t^m, \gamma(\boldsymbol{X}_t^m - \boldsymbol{X}_1)] \in \mathbb{R}^D, \quad (1)$$

where γ is the positional encoding function and $\boldsymbol{F}_t^m$ is the feature of point $\boldsymbol{X}_t^m$. At the first iteration, $\boldsymbol{F}_t^0$ is extracted from the triplane of frame t at $\boldsymbol{X}_t^0$. For later iterations, $\boldsymbol{F}_t^m$ is a direct output of the transformer from the previous iteration. $\boldsymbol{C}_t^m$ denotes correlation features, which are computed by comparing $\boldsymbol{F}_t^m$ and local triplane features around $\boldsymbol{X}_t^m$ at frame t. More details of correlation features can be found in the supplementary material.

For each update, the transformer takes as input the features for the trajectories of all query points across the entire window. We denote this set of features at the m-th iteration as $\mathcal{G}^m \in \mathbb{R}^{N \times T_s \times D} = \{\boldsymbol{G}_{i,t}^m \mid i = 1, ..., N; t = 1, ..., T_s\}$, where i is the query point index and N is the number of query points. Ψ then takes $\mathcal{G}^m$ as input and predicts the new set of point positions and features:

$$\mathcal{X}^{m+1}, \mathcal{F}^{m+1} = \Psi(\mathcal{G}^m), \quad (2)$$

where $\mathcal{X}^{m+1}$ denotes the set of updated point positions, and $\mathcal{F}^{m+1}$ denotes the set of updated point features. New $\mathcal{G}^{m+1}$ can then be defined according to Eq. 1, and the same process is repeated M times to obtain the final 3D trajectories for all query points $\mathcal{X}^M = \{\boldsymbol{X}_{i,t}^M\}$. The 2D correspondence predictions can be computed by simply projecting $\{\boldsymbol{X}_{i,t}^M\}$ back onto the 2D image plane.

As query pixels may not have corresponding pixels at some frames due to occlusions, we additionally predict the visibility for each point of the 3D trajectories at the final iteration M. Specifically, for each point $\boldsymbol{X}_{i,t}^M$, we employ an MLP network that takes the feature $\boldsymbol{F}_{i,t}^M$ as input and predicts a visibility score $v_{i,t}$.

Handling long videos. To track points across a long video, we utilize overlapping sliding windows where each pair of adjacent windows has half of their frames overlapped. Given the results from the previous window, we initialize trajectories of the first $\frac{T_s}{2}$ frames of the current window by copying the results of the last $\frac{T_s}{2}$ frames from the previous window. The trajectories of the last $\frac{T_s}{2}$ frames in the current window are initialized by copying the result of the frame $\frac{T_s}{2}$.

3.3. As Rigid As Possible Constraint

An advantage of tracking points in 3D is that we can enforce an as-rigid-as-possible (ARAP) constraint, which enhances spatial consistency and facilitates the prediction of motion especially during occlusions.

Enforcing proper ARAP constraints requires identifying if two points belong to the same rigid part. To this end, at each iteration m, we additionally compute a rigidity embedding $\boldsymbol{E}_i^m$ for each trajectory by aggregating its features $\{\boldsymbol{G}_{i,t}^m\}_{t=1}^{T_s}$ across time. Then, the rigidity affinity s_{ij}^m between any two trajectories i and j can be calculated as:

$$s_{ij}^m = \text{sim}(\boldsymbol{E}_i^m, \boldsymbol{E}_j^m), \tag{3}$$

where $\text{sim}(\cdot,\cdot)$ is the cosine similarity measure.

By the definition of rigidity, the distances between points that are rigidly moving together should be preserved over time. Therefore, we formulate our ARAP loss as follows, encouraging the distances between pairs of points exhibiting high rigidity to remain constant over time:

$$\mathcal{L}_{\text{arap}} = \sum_{m=1}^{M} \sum_{t=1}^{T_s} \sum_{\Omega_{ij}} w^m s_{ij}^m ||d(\boldsymbol{X}_{i,t}^m, \boldsymbol{X}_{j,t}^m) - d(\boldsymbol{X}_{i,1}, \boldsymbol{X}_{j,1})||_1, \tag{4}$$

where Ω_{ij} is the set of all pairwise indices and $d(\cdot,\cdot)$ is the Euclidean distance function, and $w^m = 0.8^{M-m}$ is the weight for the m-th step. This ARAP loss provides gradients for learning both the 3D trajectories and the rigidity embeddings.

Based on the affinity score between any two points, we can perform spectral clustering [47, 70] to obtain the segmentation of query pixels. Experiments in Sec. 4 show that our method can generate meaningful segmentation of rigid parts.

3.4. Training

In addition to the ARAP loss, we supervise the predicted trajectories using ground truth 3D trajectories at each iteration, which is defined as:

$$\mathcal{L}_{\text{traj}} = \sum_{m=1}^{M} \sum_{i=1}^{N} \sum_{t=1}^{T_s} w^m ||\boldsymbol{X}_{i,t}^m - \hat{\boldsymbol{X}}_{i,t}^m||_1, \tag{5}$$

where $\boldsymbol{X}_{i,t}^m$ and $\hat{\boldsymbol{X}}_{i,t}^m$ are the predicted and ground-truth 3D corresponding locations, respectively, and w^m is the weight for the m-th step, identical to that in Eq. 4.

The predicted visibilities are supervised using:

$$\mathcal{L}_{\text{vis}} = \sum_{i=1}^{N} \sum_{t=1}^{T_s} \text{CE}(v_{i,t}, \hat{v}_{i,t}), \tag{6}$$

where $v_{i,t}$ and $\hat{v}_{i,t}$ denote the predicted and ground-truth visibility, respectively. CE represents the cross entropy loss.

The total loss function for training is defined as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{traj}} + \alpha \mathcal{L}_{\text{vis}} + \beta \mathcal{L}_{\text{arap}}, \tag{7}$$

where α and β are weighting coefficients. In practice, they are set as 10 and 0.1, respectively.

3.5. Implementation Details

We train our model on the TAP-Vid-Kubric dataset [11, 14]. Our training data contains 11,000 24-frame RGBD sequences with full-length 3D trajectory annotations. During training, we use ground truth depth maps and camera intrinsics to unproject pixels into 3D space. In cases where the depth map and intrinsics are unavailable at inference, we use ZoeDepth [2] to predict the metric depth map for each video frame, and simply set the focal length to be the same as the image width. To generate triplane feature maps, we discretize the depth values into $d = 256$ bins. The resolutions of the triplane feature maps are $h \times w$, $w \times d$, $h \times d$ for XY, XZ, and YZ planes, respectively, where h, w are the image height and width. The number of channels of the triplane features is 128.

We train our model with eight 80GB A100 GPUs for 200k iterations. The total training time is around 6 days. The iteration steps M and sliding window length T_s are set to 6 and 8 respectively. In each training batch, we sample $N = 256$ query points. The transformer Ψ consists of six blocks, each comprising both spatial and temporal attention layers. For more details, please refer to the supplementary material.

4. Experiments

At inference, our method can operate in two different modalities. The first modality (and the primary focus of this paper) is long-range 2D pixel tracking. In this modality, the input is an RGB video without known depth or camera intrinsics, and we rely on ZoeDepth [2] to estimate the depth maps. Due to the lack of precise depth and intrinsics information, we only evaluate the 2D projection of the 3D trajectories onto the image plane, i.e., 2D pixel trajectories. When RGBD videos and camera intrinsics are available, our method can be used in the second modality to predict long-range 3D trajectories. We evaluate our method for both 2D and 3D tracking performance in Sec 4.1 and Sec 4.2, respectively, and then conduct ablation studies in Sec. 4.3.

Methods	Kinetics [9]			DAVIS [50]			RGB-Stacking [33]			Average		
	AJ ↑	$< \delta_{\text{avg}}$ ↑	OA ↑	AJ ↑	$< \delta_{\text{avg}}$ ↑	OA ↑	AJ ↑	$< \delta_{\text{avg}}$ ↑	OA ↑	AJ ↑	$< \delta_{\text{avg}}$ ↑	OA ↑
TAP-Net [11]	38.5	54.4	80.6	33.0	48.6	78.8	54.6	68.3	87.7	42.0	57.1	82.4
PIPs [17]	31.7	53.7	72.9	42.2	64.8	77.7	15.7	28.4	77.1	29.9	50.0	75.9
OmniMotion [71]		-	-	46.4	62.7	85.3	**69.5**	**82.5**	**90.3**	-	-	-
TAPIR [12]	49.6	64.2	85.0	56.2	70.0	86.5	54.2	69.8	84.4	53.3	68.0	85.3
CoTracker [29]	48.7	64.3	86.5	60.6	75.4	89.3	63.1	77.0	87.8	57.4	72.2	87.8
Ours	**50.1**	**65.9**	**86.9**	**61.1**	**76.3**	**89.5**	63.5	77.6	88.2	**58.2**	**73.3**	**88.2**

Table 1. **2D Tracking Results on the TAP-Vid Benchmark.** We report the average jaccard (AJ), average position accuracy ($< \delta^x_{\text{avg}}$), and occlusion accuracy (OA) on Kinetics [9], DAVIS [50] and RGB-Stacking [33] datasets.

4.1. 2D Tracking Evaluation

We conduct our evaluation on three long-range 2D tracking benchmarks: TAP-Vid [11], BADJA [4] and PointOdyssey [81]. Our method is compared with baseline 2D tracking methods, namely TAP-Net [11], PIPs [17], OmniMotion [71], TAPIR [12] and CoTracker [29]. The evaluation protocols and comparison results on each of the benchmarks are represented below.

TAP-Vid Benchmark [11] contains a few datasets: TAP-Vid-DAVIS [50] (30 real videos of about 34-104 frames), TAP-Vid-Kinetics [9] (1144 real videos of 250 frames) and RGB-Stacking [33] (50 synthetic videos of 250 frames). Each video in the benchmark is annotated with ground truth 2D trajectories and occlusions spanning the entire video duration for well-distributed points. We evaluate performance using the same metrics as the TAP-Vid benchmark [11]: average position accuracy ($< \delta^x_{\text{avg}}$), Average Jaccard (AJ), and Occlusion Accuracy (OA). Please refer to the supplement for more details. We follow the "queried first" evaluation protocol in CoTracker [29]. Specifically, we use the first frame as the query frame and predict the 2D locations of query pixels from this frame in all subsequent frames. The quantitative comparisons are reported in Tab. 1, which shows our method consistently outperforms all baselines except Omnimotion across all three datasets, demonstrating the benefits of tracking in the 3D space. Omnimotion also performs tracking in 3D and obtains the best results on RGB-Stacking by optimizing all frames at once, but it requires very costly test-time optimization.

BADJA [4] is a benchmark containing seven videos of moving animals with annotated keypoints. The metrics used in this benchmark include segment-based accuracy (segA) and 3px accuracy ($\delta^{3\text{px}}$). The predicted keypoint positions are deemed accurate when its distance from the ground truth keypoint is less than $0.2\sqrt{A}$, where A is the summation of the area of the segmentation mask. $\delta^{3\text{px}}$ depicts the percentage of the correct keypoints whose distances from their ground truth are within three pixels. As shown in Tab. 2, our method demonstrates competitive performance in terms of $\delta^{3\text{px}}$ and surpasses all baseline methods by a large margin in segment-based accuracy.

Methods	segA ↓	$\delta^{3\text{px}}$ ↑
TAP-Net [11]	54.4	6.3
PIPs [17]	61.9	13.5
TAPIR [12]	66.9	15.2
OmniMotion [71]	57.2	13.2
CoTracker [29]	63.6	**18.0**
Ours	**69.2**	17.1

Table 2. **2D Tracking Results on the BADJA Dataset [4].** The segment-based accuracy (segA) and 3px accuracy ($\delta^{3\text{px}}$) are reported.

Methods	MTE↓	$< \delta^{\mathbf{x}}_{\text{avg}}$ ↑	Survival↑
TAP-Net [11]	37.8	29.2	52.8
PIPs [81]	41.0	30.4	67.0
CoTracker [29]	30.5	56.2	76.1
Ours *w/* ZoeDepth [2]	28.3	58.4	**78.6**
Ours *w/* GT depth	**26.6**	**64.1**	78.0

Table 3. **2D Tracking Results on the PointOdyssey Dataset [81].** The Median Trajectory Error (MTE), average position accuracy ($< \delta^x_{\text{avg}}$), and survival rate (Survival) are reported.

PointOdyssey [81] is a large-scale synthetic dataset featuring diverse animated characters ranging from humans to animals, placed within diverse 3D environments. We evaluate our method on PointOdyssey's test set which contains 12 videos with complex motion, each spanning approximately 2000 frames. We adopt the evaluation metrics proposed in PointOdyssey [81] which are designed for evaluating very long trajectories. These metrics include the Median Trajectory Error (MTE), $< \delta^x_{\text{avg}}$ (consistent with TAP-Vid), and the survival rate. The survival rate is defined as the average number of frames until tracking failure over the video length. Tracking failure is identified when the L2 error exceeds 50 pixels at a resolution of 256×256. In Tab. 3, we report results for baseline methods as well as our method using depths from ZoeDepth [2] (default) and GT depth annotations. Our method consistently outperforms

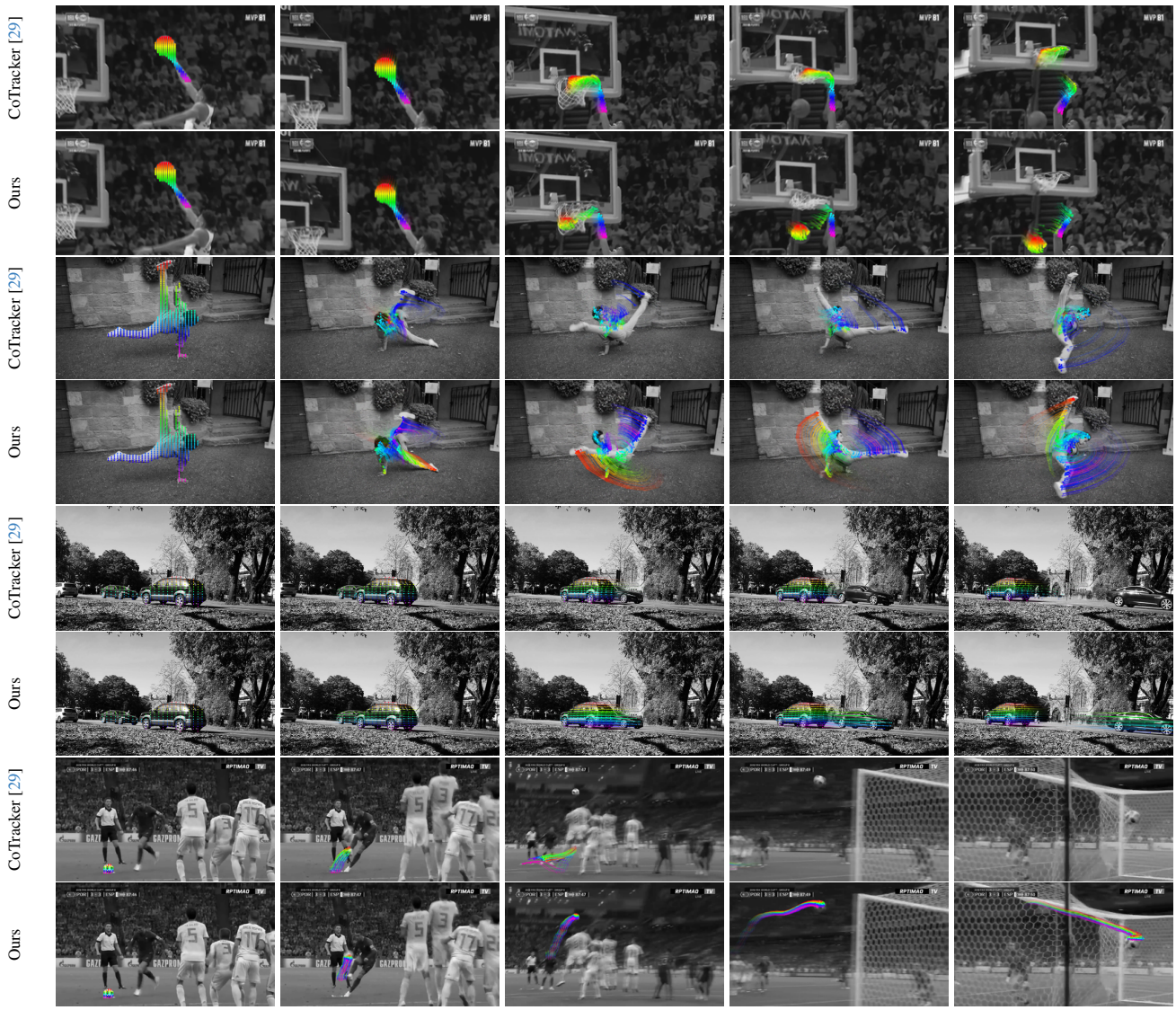

Figure 3. **Qualitative Comparison.** For each sequence we show tracking results of CoTracker [29] and our method SpatialTracker.

the baselines across all metrics by a noticeable margin. In particular, we demonstrate that with access to more accurate ground truth depth, our performance can be further enhanced. This suggests the potential of our method to continue improving alongside advancements in monocular depth estimation.

Qualitative Results. We present qualitative comparisons with CoTracker [29] on challenging videos from DAVIS [50] and Internet footage in Fig. 3. Our method outperforms CoTracker in handling complex human motion with self-occlusions, achieves a better understanding of rigid groups, and can effectively track small, rapidly moving objects even in the presence of occlusions. Please refer to the supplementary video for more results and better visualizations.

4.2. 3D Tracking Evaluation

Given an RGBD video (with known depth and intrinsics) as input, our method can estimate true 3D trajectories. Since no baseline method can directly be used for long-range 3D tracking, we construct our baselines by composing existing methods. Our first baseline is chained RAFT-3D [63]. RAFT-3D is designed for pairwise scene flow estimation, so to obtain long-range scene flow, we chain its scene flow prediction of consecutive frames. Our second baseline is to directly lift the predicted 2D trajectories from CoTracker [29] using the input depth maps.

We evaluate our method and baselines on the PointOdyssey [81] dataset. We create 231 testing sequences from the test set, each consisting of 24 frames and with a reduced frame rate set at one-fifth of the original. We

Methods	ATE_{3D} ↓	$\delta_{0.1}$ ↑	$\delta_{0.2}$ ↑
Chained RAFT3D [63]	0.70	0.12	0.25
Lifted CoTracker [29]	0.77	0.51	0.64
Ours	**0.22**	**0.59**	**0.76**

Table 4. **3D Tracking Results on the PointOdyssey Dataset.** ATE_{3D} is the average trajectory error in 3D space and δ_t measures the percentage of points whose distances are within t (in meter) from the ground truth.

Methods	AJ↑	$< \delta^x_{avg}$↑	OA↑
Ours *w/o* ARAP	55.1	71.6	87.4
Ours *w/* DPT [53]	51.4	70.7	83.3
Ours *w/* MiDaS [5]	56.3	73.9	86.6
Ours *w/* ZoeDepth [2] (default)	**61.1**	**76.3**	**89.5**

Table 5. **Ablation Study on the DAVIS Dataset.** The Average Jaccard (AJ), average position accuracy ($< \delta^x_{avg}$), and Occlusion Accuracy (OA) are reported. We evaluate the effectiveness of the ARAP constraint and the influence of different monocular depth estimators (ZoeDepth [2], MiDaS [5] and DPT [53]). "Ours *w/* ZoeDepth" is the default model we use in our experiments.

use three evaluation metrics, namely ATE_{3D}, $\delta_{0.1}$, and $\delta_{0.2}$. ATE_{3D} is the average trajectory error in 3D space. $\delta_{0.1}$ and $\delta_{0.2}$ measure the percentage of points whose distances are within 0.1m and 0.2m from the ground truth, respectively.

The results are shown in Tab. 4. Our method outperforms both "Chained RAFT-3D" and "Lifted CoTracker" consistently on all three metrics by a large margin. We found that RAFT-3D, trained on FlyingThings [40], generalizes poorly on PointOdyssey, possibly due to the fact that its dense-SE3 module is sensitive to domain gaps. In contrast, also trained on a different dataset (Kubric), our method exhibits strong generalization to PointOdyssey, affirming the efficacy of our design for 3D trajectory prediction. In addition, both baselines cannot handle occlusion and will lose track of points once they become occluded, hurting the performance significantly.

4.3. Ablation and Analysis

Effectiveness of ARAP loss and rigidity embedding. We ablate the ARAP loss and report the result "Ours *w/o* ARAP" on the TAP-Vid-DAVIS [11, 51] dataset in Tab. 5. Without the ARAP loss, the performance drops substantially, verifying the effectiveness of the ARAP constraint. We additionally showcase qualitative results of the rigid part segmentation, utilizing our learned rigidity embeddings in Fig. 4, demonstrating their effectiveness.

Analysis on monocular depth estimators. To study the influence of different monocular depth estimation methods on our model, we evaluate our method with three monocular depth models: ZoeDepth [2] (default), MiDaS [5], and DPT [53]. We report the results on the TAP-Vid-DAVIS [11, 51] dataset in Tab. 5. "Ours *w/* ZoeDepth" achieves the best results, probably due to the fact that ZoeDepth [2] is a metric depth estimator and exhibits less temporal inconsistency across frames compared to relative depth estimators MiDaS [5] and DPT [53]. Furthermore, it is noteworthy that the efficacy of our model has a positive correlation with the advancements in the underlying monocular depth models. Please refer to the supplementary material for additional analysis and ablations.

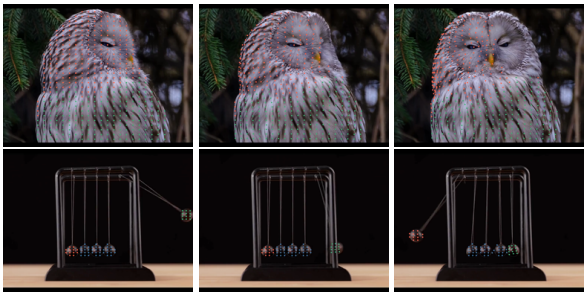

Figure 4. **Rigid Part Segmentation.** We utilize spectral clustering on the rigidity embedding to determine rigid groups. Each color represents a distinct rigid group.

5. Conclusion and Discussion

In this work, we show that a properly designed 3D representation is crucial for solving the long-standing challenge of dense and long-range motion estimation in videos. Motion naturally occurs in 3D and tracking motion in 3D allows us to better leverage its regularity in 3D, e.g., the ARAP constraint. We proposed a novel framework that estimates 3D trajectories using triplane representations with a learnable ARAP constraint that identifies the rigid groups in the scene and enforces rigidity within each group. Experiments demonstrated the superior performance of our method compared to existing baselines and its applicability to challenging real-world scenarios.

Our current model relies on off-the-shelf monocular depth estimators whose accuracy may affect the final tracking performance as shown in Tab. 5. However, we anticipate that advancements in monocular reconstruction will enhance the performance of motion estimation. We expect a closer interplay between these two problems, benefiting each other in the near future.

Acknowledgement

This work was partially supported by National Key Research and Development Program of China (No. 2020AAA0108900), Ant Group, and Information Technology Center and State Key Lab of CAD&CG, Zhejiang University.

References

[1] Steven S. Beauchemin and John L. Barron. The computation of optical flow. *ACM computing surveys (CSUR)*, 27(3):433–466, 1995. 1, 2

[2] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth. *CoRR*, abs/2302.12288, 2023. 2, 5, 6, 8

[3] Weikang Bian, Zhaoyang Huang, Xiaoyu Shi, Yitong Dong, Yijin Li, and Hongsheng Li. Context-tap: Tracking any point demands spatial context features. *arXiv preprint arXiv:2306.02000*, 2023. 2

[4] Benjamin Biggs, Thomas Roddick, Andrew W. Fitzgibbon, and Roberto Cipolla. Creatures great and SMAL: recovering the shape and motion of animals from video. In *Asian Conf. Comput. Vis.*, pages 3–19. Springer, 2018. 2, 6

[5] Reiner Birkl, Diana Wofk, and Matthias Müller. Midas v3. 1–a model zoo for robust monocular relative depth estimation. *arXiv preprint arXiv:2307.14460*, 2023. 8

[6] Michael J Black and Padmanabhan Anandan. A framework for the robust estimation of optical flow. In *Int. Conf. Comput. Vis.*, pages 231–236. IEEE, 1993. 2

[7] Aljaž Božič, Pablo Palafox, Michael Zollhöfer, Justus Thies, Angela Dai, and Matthias Nießner. Neural deformation graphs for globally-consistent non-rigid reconstruction. *CVPR*, 2021. 3

[8] Thomas Brox, Christoph Bregler, and Jitendra Malik. Large displacement optical flow. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 41–48, 2009. 2

[9] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 6

[10] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 16123–16133, 2022. 2, 4

[11] 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. *Adv. Neural Inform. Process. Syst.*, 35: 13610–13626, 2022. 2, 5, 6, 8

[12] Carl Doersch, Yi Yang, Mel Vecerík, Dilara Gokay, Ankush Gupta, Yusuf Aytar, João Carreira, and Andrew Zisserman. TAPIR: tracking any point with per-frame initialization and temporal refinement. *CoRR*, abs/2306.08637, 2023. 2, 6

[13] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick Van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*, 2015. 2

[14] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J. Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam H. Laradji, Hsueh-Ti Derek Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, A. Cengiz Öztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: A scalable dataset generator. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3739–3751. IEEE, 2022. 2, 5

[15] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3254–3263, 2019. 3

[16] Simon Hadfield and Richard Bowden. Kinecting the dots: Particle based scene flow from depth sensors. In *2011 International Conference on Computer Vision*, pages 2290–2295. IEEE, 2011. 3

[17] Adam W. Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. Particle video revisited: Tracking through occlusions using point trajectories. In *Eur. Conf. Comput. Vis.*, pages 59–75. Springer, 2022. 2, 6

[18] Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d flow: Dense 3-d motion estimation using color and depth. In *2013 IEEE international conference on robotics and automation*, pages 2276–2282. IEEE, 2013. 3

[19] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981. 1, 2

[20] Michael Hornacek, Andrew Fitzgibbon, and Carsten Rother. Sphereflow: 6 dof scene flow from rgb-d pairs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3526–3533, 2014. 3

[21] Zhaoyang Huang, Xiaoyu Shi, Chao Zhang, Qiang Wang, Ka Chun Cheung, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer: A transformer architecture for optical flow. In *European Conference on Computer Vision*, pages 668–685. Springer, 2022. 2

[22] Frédéric Huguet and Frédéric Devernay. A variational method for scene flow estimation from stereo sequences. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–7. IEEE, 2007. 3

[23] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8981–8989, 2018. 2

[24] 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 *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2462–2470, 2017. 2

[25] Michal Irani. Multi-frame optical flow estimation using subspace constraints. In *Int. Conf. Comput. Vis.*, pages 626–633, 1999. 2

[26] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al.

Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021. 2

[27] Mariano Jaimez, Mohamed Souiai, Javier Gonzalez-Jimenez, and Daniel Cremers. A primal-dual framework for real-time dense rgb-d scene flow. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 98–104. IEEE, 2015. 3

[28] Joel Janai, Fatma Guney, Anurag Ranjan, Michael Black, and Andreas Geiger. Unsupervised learning of multi-frame optical flow with occlusions. In *Eur. Conf. Comput. Vis.*, pages 690–706, 2018. 2

[29] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. *CoRR*, abs/2307.07635, 2023. 2, 3, 4, 6, 7, 8

[30] Yoni Kasten, Dolev Ofri, Oliver Wang, and Tali Dekel. Layered neural atlases for consistent video editing. In *SIGGRAPH Asia*, 2021. 3

[31] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *2013 International Conference on 3D Vision-3DV 2013*, pages 1–8. IEEE, 2013. 3

[32] Ryan Kennedy and Camillo J Taylor. Optical flow with geometric occlusion estimation and fusion of multiple frames. In *Energy Minimization Methods in Computer Vision and Pattern Recognition: 10th International Conference, EMMCVPR 2015, Hong Kong, China, January 13-16, 2015. Proceedings 10*, pages 364–377. Springer, 2015. 2

[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 *5th Annual Conference on Robot Learning*, 2021. 6

[34] Jiahui Lei and Kostas Daniilidis. Cadex: Learning canonical deformation coordinate space for dynamic surface representation via neural homeomorphism. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6624–6634, 2022. 3

[35] Wenbin Lin, Chengwei Zheng, Jun-Hai Yong, and Feng Xu. Occlusionfusion: Occlusion-aware motion estimation for real-time dynamic 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1736–1745, 2022. 3

[36] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 529–537, 2019. 3

[37] David G Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.*, 60:91–110, 2004. 1

[38] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981. 1

[39] Wei-Chiu Ma, Shenlong Wang, Rui Hu, Yuwen Xiong, and Raquel Urtasun. Deep rigid instance scene flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3614–3622, 2019. 3

[40] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. arXiv:1512.02134. 8

[41] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3061–3070, 2015. 3

[42] Michal Neoral, Jonáš Šerỳch, and Jiří Matas. Mft: Long-term tracking of every pixel. *arXiv preprint arXiv:2305.12998*, 2023. 2

[43] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*, pages 127–136. Ieee, 2011. 3

[44] Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 343–352. IEEE Computer Society, 2015. 3

[45] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5379–5389, 2019. 3

[46] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 4

[47] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 5

[48] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 523–540. Springer, 2020. 4

[49] Jean-Philippe Pons, Renaud Keriven, and Olivier Faugeras. Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *International Journal of Computer Vision*, 72:179–193, 2007. 3

[50] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017. 6, 7

[51] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 DAVIS challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017. 8

[52] Julian Quiroga, Thomas Brox, Frédéric Devernay, and James Crowley. Dense semi-rigid scene flow estimation from rgbd images. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VII 13*, pages 567–582. Springer, 2014. 2, 3

[53] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12179–12188, 2021. 8

[54] Zhile Ren, Orazio Gallo, Deqing Sun, Ming-Hsuan Yang, Erik B Sudderth, and Jan Kautz. A fusion approach for multi-frame optical flow estimation. In *IEEE Winter Conf. Appl. Comput. Vis.*, pages 2077–2086, 2019. 2

[55] Michael Rubinstein, Ce Liu, and William T Freeman. Towards longer long-range motion trajectories. In *Brit. Mach. Vis. Conf.*, 2012. 2

[56] Peter Sand and Seth J. Teller. Particle video: Long-range motion estimation using point trajectories. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2195–2202. IEEE Computer Society, 2006. 2

[57] Laura Sevilla-Lara, Deqing Sun, Varun Jampani, and Michael J Black. Optical flow with semantic segmentation and localized layers. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3889–3898, 2016. 3

[58] Jianbo Shi et al. Good features to track. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 593–600. IEEE, 1994. 1

[59] Xiaoyu Shi, Zhaoyang Huang, Dasong Li, Manyuan Zhang, Ka Chun Cheung, Simon See, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer++: Masked cost volume autoencoding for pretraining optical flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1599–1610, 2023. 2

[60] Deqing Sun, Erik Sudderth, and Michael Black. Layered image motion with explicit occlusions, temporal consistency, and depth ordering. In *Adv. Neural Inform. Process. Syst.*, 2010. 3

[61] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8934–8943, 2018. 2

[62] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer, 2020. 1, 2

[63] Zachary Teed and Jia Deng. Raft-3d: Scene flow using rigid-motion embeddings. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8375–8384, 2021. 2, 3, 7, 8

[64] Carlo Tomasi and Takeo Kanade. Detection and tracking of point. *Int. J. Comput. Vis.*, 9:137–154, 1991. 1

[65] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 4

[66] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 722–729. IEEE, 1999. 3

[67] Christoph Vogel, Konrad Schindler, and Stefan Roth. 3d scene flow estimation with a rigid motion prior. In *2011 International Conference on Computer Vision*, pages 1291–1298. IEEE, 2011. 3

[68] Christoph Vogel, Konrad Schindler, and Stefan Roth. Piecewise rigid scene flow. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1377–1384, 2013. 3

[69] Sebastian Volz, Andres Bruhn, Levi Valgaerts, and Henning Zimmer. Modeling temporal coherence for optical flow. In *Int. Conf. Comput. Vis.*, pages 1116–1123. IEEE, 2011. 2

[70] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17:395–416, 2007. 5

[71] Qianqian Wang, Yen-Yu Chang, Ruojin Cai, Zhengqi Li, Bharath Hariharan, Aleksander Holynski, and Noah Snavely. Tracking everything everywhere all at once. *arXiv preprint arXiv:2306.05422*, 2023. 2, 3, 6

[72] Zirui Wang, Shuda Li, Henry Howard-Jenkins, Victor Prisacariu, and Min Chen. Flownet3d++: Geometric losses for deep scene flow estimation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 91–98, 2020. 3

[73] Zhen Wang, Shijie Zhou, Jeong Joon Park, Despoina Paschalidou, Suya You, Gordon Wetzstein, Leonidas Guibas, and Achuta Kadambi. Alto: Alternating latent topologies for implicit 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 259–270, 2023. 4

[74] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. DeepFlow: Large displacement optical flow with deep matching. In *Int. Conf. Comput. Vis.*, pages 1385–1392, 2013. 2

[75] Wenxuan Wu, Zhi Yuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. Pointpwc-net: Cost volume on point clouds for (self-) supervised scene flow estimation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pages 88–107. Springer, 2020. 3

[76] Jia Xu, René Ranftl, and Vladlen Koltun. Accurate optical flow via direct cost volume processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1289–1297, 2017. 2

[77] Vickie Ye, Zhengqi Li, Richard Tucker, Angjoo Kanazawa, and Noah Snavely. Deformable sprites for unsupervised video decomposition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2657–2666, 2022. 3

[78] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Pattern Recognition: 29th DAGM Symposium, Heidelberg, Germany, September 12-14, 2007. Proceedings 29*, pages 214–223. Springer, 2007. 2

[79] Ye Zhang and Chandra Kambhamettu. On 3d scene flow and structure estimation. In *Proceedings of the 2001*

IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, pages II–II. IEEE, 2001. 3

[80] Shiyu Zhao, Long Zhao, Zhixing Zhang, Enyu Zhou, and Dimitris Metaxas. Global matching with overlapping attention for optical flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17592–17601, 2022. 2

[81] Yang Zheng, Adam W Harley, Bokui Shen, Gordon Wetzstein, and Leonidas J Guibas. Pointodyssey: A large-scale synthetic dataset for long-term point tracking. In *Int. Conf. Comput. Vis.*, pages 19855–19865, 2023. 2, 6, 7