Fast Encoder-Based 3D from Casual Videos via Point Track Processing

Yoni Kasten[1], Wuyue Lu[2], and Haggai Maron[1,3]

[1] NVIDIA Research
[2] Simon Fraser University
[3] Technion

Abstract. This paper addresses the long-standing challenge of reconstructing 3D structures from videos with dynamic content. Current approaches to this problem were not designed to operate on casual videos recorded by standard cameras or require a long optimization time. Aiming to significantly improve the efficiency of previous approaches, we present TracksTo4D, a learning-based approach that enables inferring 3D structure and camera positions from dynamic content originating from casual videos using a single efficient feed-forward pass. To achieve this, we propose operating directly over 2D point tracks as input and designing an architecture tailored for processing 2D point tracks. Our proposed architecture is designed with two key principles in mind: (1) it takes into account the inherent symmetries present in the input point tracks data, and (2) it assumes that the movement patterns can be effectively represented using a low-rank approximation. TracksTo4D is trained in an unsupervised way on a dataset of casual videos utilizing only the 2D point tracks extracted from the videos, without any 3D supervision. Our experiments show that TracksTo4D can reconstruct a temporal point cloud and camera positions of the underlying video with accuracy comparable to state-of-the-art methods, while drastically reducing runtime by up to 95%. We further show that TracksTo4D generalizes well to unseen videos of unseen semantic categories at inference time.

Keywords: Structure From Motion · Dynamic Videos · 3D Reconstruction · Equivariance · Symmetries

1 Introduction

Predicting 3D geometry in dynamic scenes is a challenging problem. In this problem setup, we are given access to multiple images of a scene taken sequentially, e.g., from a monocular video camera, where *both* the content in the scene and the camera are moving. Our task is to reconstruct the dynamic 3D positions of the points seen in the images and the camera poses. This fundamental problem has gained significant interest from the research community over the years [5, 22, 32, 57], mainly due to its important applications in many fields such

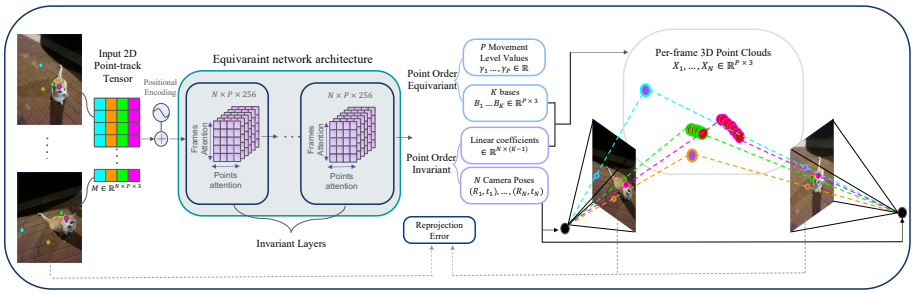

Fig. 1: We present TracksTo4D, a method for mapping a set of 2D point tracks extracted from casual dynamic videos into their corresponding 3D locations and camera motion. At inference time, our network predicts the dynamic structure and camera motion in a single feed-forward pass. Our network takes as input a set of 2D point tracks (left) and uses several multi-head attention layers while alternating between the time dimension and the track dimension (middle). The network predicts cameras, per-frame 3D points, and per-world point movement value (right). The 3D point internal colors illustrate the predicted 3D movement level values, such that points with high/low 3D motion are presented in red/purple colors respectively. These outputs are used to reproject the predicted points into the frames for calculating the reprojection error losses. See details in the text. The reader is encouraged to watch the supplementary video visualizations.

as robot navigation, autonomous driving and 3D reconstruction of general environments [16]. Importantly, in contrast to static scenes where the epipolar geometry constraints hold between the corresponding points of different views [14], determining the depth of a moving point from monocular views is an ill-posed problem [2]. This causes standard Structure from Motion techniques [28, 36, 49] to be inadequate in this setup [21].

Previous work and limitations. Many existing approaches for the above problem make simplifying assumptions that limit their applicability to real-world scenarios. Methods based on orthographic camera models and low-rank assumptions use matrix factorization techniques [5, 22], but the orthographic camera assumption might not be realistic and may cause reconstruction errors. Techniques that incorporate depth priors often require lengthy optimization processes in order to make the depth estimates across frames consistent [21, 57]. Other physics-based approaches make assumptions about rigid bones [52, 54] or isometric deformable surfaces [32] and typically involve complex, slow optimization per video. In addition, they may require foreground-background segmentation of the moving content, which is not always easily obtained. Alternatively, some methods are specifically tailored to certain object classes like humans [46], restricting their domain to those limited cases. Consequently, these prior methods are either not directly applicable to casual videos, or require long optimization time per video.

Our approach. We propose TracksTo4D,[4] a novel approach for fast reconstruction of sparse dynamic 3D point clouds and camera poses from casual videos. Our main idea is to train a neural network on multiple videos to learn the mapping from the input image sequence to a sequence of the scene's 3D point clouds and camera poses. After training, the trained network can be efficiently applied to new image sequences using a single feed-forward pass, avoiding costly optimization.

To enhance the method's ability to generalize across different types of videos and scenes, we made a crucial design choice: our approach processes point track tensors as input, rather than operating directly on the image sequence. Specifically, each entry (n, p) in these tensors represents the 2D position of a tracked point p in a specific video frame n [5]. Our main insight is that point track tensors may exhibit more common motion patterns across casual video domains compared to image pixels. In other words, we argue that processing the raw point track data rather than scene-specific pixels or features may enable learning class and scene-agnostic internal feature representations for improved generalization. Importantly, recent advances in point tracking [9, 17] enable efficiently inferring these point tracks from casual videos using pre-trained models. These two properties make point track matrices an attractive input for our learning method.

Following this design choice, we design our architecture according to two principles: (1) process point track tensors, which have a unique structure, and (2) encode meaningful prior knowledge about the reconstruction problem, as the problem is ill-posed in general. In the following, we address these desired properties.

First, we design a network architecture that can effectively and efficiently handle point track inputs. To do that, we propose a novel layer design that takes into account the symmetries of the problem: the mapping we aim to learn, from point track matrices to 3D point clouds and camera poses, preserves two natural symmetries: (i) the points being tracked can be arbitrarily permuted without affecting the problem; (ii) the frames containing these points exhibit temporal structure, adhering to an approximate time-translation symmetry. Following the Geometric Deep Learning paradigm [6], we build upon recent theoretical advances in equivariant learning [26] and integrate these two symmetries into our network architecture using dedicated attention and positional encoding mechanisms.

Second, a key challenge in predicting 3D dynamic motion and camera poses from 2D point tracks is that this problem is inherently ill-posed without additional constraints [2]. To address this, we integrate a low-rank movement assumption into our architecture, following the seminal work of [5] which constrained output point clouds to be linear combinations of basis elements. Specifically, given an input point track tensor, our architecture equivariantly predicts a small set of input-specific basis elements. The output point clouds at each time frame are then defined as a linear combination of these basis elements, with the coefficients also predicted by the network. Notably, the first basis is assumed to

[4] 4D since we have three Euclidean coordinates with an additional time coordinate

fully represent the 3D static points in the video, while the remaining basis elements capture the 3D dynamic deviations. This structure effectively restricts the predicted point clouds to have a more specific form, making the problem more constrained.

Our approach is trained on a dataset of extracted point track matrices [17] from raw videos without any 3D supervision by simply minimizing the reprojection errors, aiming to predict output point clouds that, after undergoing a perspective projection, will return the original 2D point tracks. In our experiments, TracksTo4D is trained on the Common Pets dataset [38]. We evaluate our method on test data with GT camera poses and GT depth information for point tracks, and demonstrate that it produces comparable results to state-of-the-art methods, while having a significantly shorter inference time by up to 95%. In addition, we show the method's ability to generalize to out-of-domain videos.

Contributions. In summary, our contributions are (1) A novel modeling of the dynamic reconstruction problem via learning on point tracks without 3D supervision; (2) A novel deep learning architecture incorporating two key principles: accounting for the symmetry of the data and encoding low-rank structure in the predicted point clouds (3) Experiments demonstrating extremely fast inference time compared to baselines, accurate results, and strong generalization across other categories.

2 Method

Problem formulation. Given a video of N frames, let $M \in \mathbb{R}^{N\times P\times 3}$ be a pre-extracted $2D$ point tracks tensor (Fig. 1, left side). This tensor represents the two-dimensional information about a set of P world points that are tracked throughout the video. Each element in the tensor, $M_{i,j,:}$, stores three values: (x, y, o) where $x, y \in \mathbb{R}$ are respectively the observed horizontal and vertical locations of point j in frame i, and $o \in \{0, 1\}$ indicates whether point j is observed in frame i or not. Our goal is to train a deep neural network to map the input point tracks tensor M into a set of per-frame camera poses $\{R_i(M), \mathbf{t}_i(M)\}_{i=1}^N$ and per-frame 3D points $\{X_i(M)\}_{i=1}^N$, where $R_i(M) \in \mathbb{SO}(3), \mathbf{t}_i(M) \in \mathbb{R}^3, X_i(M) \in \mathbb{R}^{P\times 3}$ (Fig. 1, right side).

Overview of our approach. Our method receives $M \in \mathbb{R}^{N\times P\times 3}$ as input. This tensor is being processed by a neural architecture composed of multi-head attention layers where the attention is applied in an alternating fashion on the P and the N dimensions in each layer. These layers are defined in Sec. 2.1. After a composition of several such layers, the network uses the resulting features in $\mathbb{R}^{N\times P\times d}$ to predict N camera poses in $\mathbb{SO}^3\times\mathbb{R}^3$ and N point clouds in $\mathbb{R}^{N\times P\times 3}$. These N point clouds are parameterized as a linear combination of K input specific point cloud bases $B_1(M), \dots B_K(M) \in \mathbb{R}^{P\times 3}$. This is discussed in detail in Sec. 2.2. Our network is trained in an unsupervised way on a dataset of videos by

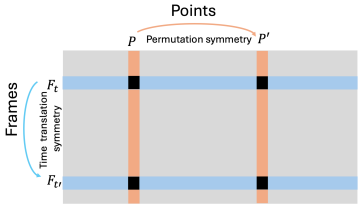

Fig. 2: The symmetry structure of our problem. Frames (vertical) have time translation symmetry while points (horizontal) have set permutation symmetry.

minimizing the reprojection error and other regularization losses (Sec. 2.3) that are used to update the model parameters. Our pipeline is illustrated in Fig. 1

2.1 Equivariant layers for point track tensors

Following the geometric deep learning paradigm, our goal is to design a neural architecture that respects the underlying symmetries and structure of the data.

Symmetry analysis. Our input is a tensor $M \in \mathbb{R}^{N\times P\times 3}$ representing a sequence of N frames each with P point coordinates. This structure gives rise to two key symmetries: First, the order of the P points within each frame does not matter - in other words, permuting this axis results in an equivalent problem [26]. Formally, this axis has a permutation symmetry S_P where S_P is the symmetric group on P elements. Second, along the temporal N axis, we have an approximate translation symmetry arising from the ordered video sequence. This means that shifting the time frames is required to result in the same shift in our output. We model this with a cyclic group C_N of order N. Both symmetries are illustrated in Fig. 2. We note that while the cyclic group assumption may not be entirely accurate, we still find it useful as it helps us to derive appropriate parametric layers for our data, similar to how the convolutional layer is derived for data with translational symmetries such as images. Taken together, the full symmetry group of the input space is the direct product $\mathcal{G} = C_N \times S_P$ combining these time and point permutation symmetries, acting on $\mathbb{R}^{N\times P\times 3}$ by $((t,\sigma)\cdot M)_{n,p,j} = M_{t^{-1}(n),\sigma^{-1}(p),j}$ for $(t,\sigma) \in \mathcal{G}$[5]. Next, we will design an architecture equivariant to $\mathcal{G}$, to ensure that the model takes into account the symmetries above.

Linear equivariant layers. Point track tensors can be viewed as a collection of N individual point tracks, each of which exhibits translational symmetry. The scenario where an object comprises a set of elements with their own symmetry group, such as a set of images or graphs, was previously explored in [26]. In that work, the authors characterized the general linear equivariant layer structure in

[5] This is different from the symmetry group studied in [27], where the temporal structure was not exploited.

such cases, termed the Deep Sets for Symmetric Elements (DSS) layer. Building on the DSS approach, our basic linear equivariant layer for the point track tensors M would take the form:

$$F(M)_{:,j} = L_1(M_{:,j}) + \sum_{j'=1}^{P} L_2(M_{:,j'}) \tag{1}$$

where L_i are linear translation equivariant function (i.e. convolutions), $M_{:,j} \in \mathbb{R}^{N\times d}$ are the columns of M representing all the inputs for a specific tracked point, $F(M)_{:,j} \in \mathbb{R}^{N\times d'}$ is the output column and d, d' are the input and output feature channels respectively. To construct a neural network, these layers can be interleaved with pointwise nonlinearities, similar to basic convolutional neural networks.

Implementation via transformers and positional encoding. While the linear layer design is reasonable, it may not be the optimal choice. To enhance the model, we design a new layer whose structure follows Equation (1), but incorporates nonlinear layers in the form of transformers [45]. Specifically, our layer F is formulated similarly to Equation (1), but instead of convolutions (L_i) and summations, it utilizes two self-attention mechanisms and suitable temporal positional encoding across the N dimension. Formally, our basic layer $F : \mathbb{R}^{N\times P\times d} \to \mathbb{R}^{N\times P\times d'}$ is computed via four steps, which are described below:

$$\begin{aligned} \bar{\mathbf{q}}_{ij} = \bar{W}^Q M_{ij},\ \bar{\mathbf{k}}_{ij} = \bar{W}^K M_{ij},\ \bar{\mathbf{v}}_{ij} = \bar{W}^V M_{ij} \\ \bar{M}_{ij} = \sum_{i'=1}^{N} \frac{\exp(\bar{\mathbf{q}}_{ij} \cdot \bar{\mathbf{k}}_{i'j})}{\sum_{l=1}^{N} \exp(\bar{\mathbf{q}}_{ij} \cdot \bar{\mathbf{k}}_{lj})} \bar{\mathbf{v}}_{i'j} \end{aligned} \tag{2}$$

$$\begin{aligned} \mathbf{q}_{ij} = W^Q \bar{M}_{ij},\ \mathbf{k}_{ij} = W^K \bar{M}_{ij},\ \mathbf{v}_{ij} = W^V \bar{M}_{ij} \\ F(M)_{ij} = \sum_{j'=1}^{P} \frac{\exp(\mathbf{q}_{ij} \cdot \mathbf{k}_{ij'})}{\sum_{l=1}^{P} \exp(\mathbf{q}_{ij} \cdot \mathbf{k}_{il})} \mathbf{v}_{ij'} \end{aligned} \tag{3}$$

Here, $M_{i,j} \in \mathbb{R}^d$ are the features associated with the j-th point in the i-th frame. The attention mechanism defined in the first equation above (2) is augmented with standard temporal positional encoding in the first layer and replaces the translation equivariant function L_i applied to the columns of M (Eq.(1)). The attention in the second equation (3) implements the set aggregation (summation) (also in Eq.(1)) applied to the rows of M. As commonly done, we use transformers with 16 attention heads [45].

2.2 Constraining 3D motion and camera poses via low-rank assumption

Given our 2D tracks, we aim to characterize the motion of the points by decomposing them into the global camera motion and the 3D motion of objects

in the scene. The 2D motion of static scene points provides useful constraints for estimating the camera motion. However, as previously mentioned, predicting camera and dynamic 3D motion solely from 2D motion is an ill-posed problem without additional constraints [2]. We tackle this challenge by adding two mechanisms to our architecture: (1) low-rank movement assumption; and (2) specific modeling of the static scene for camera estimation.

Low-rank movement assumption. First, motivated by classical orthographic Non-Rigid Structure from Motion [5], we constrain the output points to be formulated by a linear combination of input-specific basis elements. Specifically, given the input 2D point tracks, $M \in \mathbb{R}^{N\times P\times 3}$, our network predicts K point clouds: $B_1(M), \ldots, B_K(M) \in \mathbb{R}^{P\times 3}$ and $N(K-1)$ linear coefficients, $\{c_{1k}(M)\}_{k=2}^{K}, \ldots \{c_{Nk}(M)\}_{k=2}^{K}$, such that

$$X_i(M) = B_1(M) + \sum_{k=2}^{K} c_{ik}(M) B_k(M) \tag{4}$$

where $X_i(M) \in \mathbb{R}^{P\times 3}$ is the 3D point cloud at frame i. The point clouds and coefficients are computed by taking the output of the last equivariant layer as defined in the previous section and applying invariant aggregations on the respective dimension resulting in equivariant and invariant outputs. See more details in the supplementary material. We note that we deliberately chose the coefficient of $B_1(M)$ to be the constant 1, the reason is explained in the next paragraph.

Specific modeling of the static scene for camera estimation. Frequently, casual video data of dynamic scenes contains many static regions, which can be used to determine camera poses [59]. We leverage this observation by treating the first basis element $B_1(M) \in \mathbb{R}^{P\times 3}$ as a static approximation for all scene points and encourage $B_1(M)$ as well as the output camera poses to explain the 2D observations according to this approximation using a "static" reprojection loss ($\mathcal{L}_{\text{Static}}$, defined in the next section). We note, however, that a static point cloud is not likely to produce low reprojection errors for the non-static components, thus the reprojection error necessitates robustness to substantial errors from the non-static elements. To address this, our network predicts (equivariantly) P motion level values $\gamma_1(M), \ldots, \gamma_P(M) \in \mathbb{R}_+$, one for each point in our dynamic point cloud, which we use to weight the reprojection errors from $B_1(M)$. The main idea is to give less weight to non-static points so that the static projection loss can disregard them. Specifically, inspired by [57], each $\gamma_i(M)$ defines a Cauchy distribution that models the reprojection errors for its associated world point, such that a world point with higher γ is expected to produce a wider error distribution. Empirically, as noted by [57], the Cauchy distribution tends to be more robust for modeling reprojection error uncertainties compared to Gaussian noise modeling [18]. Then, $\mathcal{L}_{\text{Static}}$, minimizes the negative log-likelihood under this assumption. See details in Sec. 2.3.

2.3 Training and losses

Model outputs. Given the input 2D point tracks $M \in \mathbb{R}^{N\times P\times 3}$, our network produces outputs as a function of M: linear bases and coefficients $B_1(M),\dots,B_K(M) \in \mathbb{R}^{P\times 3}, \{c_{1k}(M)\}_{k=2}^K,\dots,\{c_{Nk}(M)\}_{k=2}^K \in \mathbb{R}$ which define a dynamic point cloud $X_1(M),\cdots,X_N(M) \in \mathbb{R}^{P\times 3}$,$\gamma_1(M),\dots,\gamma_P(M) \in \mathbb{R}_+$ movement level values, and $(R_1(M),\mathbf{t}_1(M)),\dots,(R_N(M),\mathbf{t}_N(M)) \in SO(3)\times\mathbb{R}^3$ camera poses.

We use these network outputs to define a self-supervised loss function with respect to the current network weights and M which is defined by:

$$\mathcal{L} = \lambda_{\text{Reproject}}\mathcal{L}_{\text{Reproject}} + \lambda_{\text{Static}}\mathcal{L}_{\text{Static}} + \lambda_{\text{Negative}}\mathcal{L}_{\text{Negative}} + \lambda_{\text{Sparse}}\mathcal{L}_{\text{Sparse}} \quad (5)$$

Reprojection loss. The reprojection loss encourages the consistency between the output 3D point clouds and camera poses, to the input 2D observations:

$$\mathcal{L}_{\text{Reproject}} = \frac{1}{\sum_{i=1}^N\sum_{j=1}^P M_{ij}^o}\sum_{i=1}^N\sum_{j=1}^P M_{ij}^o\mathcal{R}(X_{ij},R_i,\mathbf{t}_i,M_{ij}^{xy}) \quad (6)$$

where $\mathcal{R}(X_{ij},R_i,\mathbf{t}_i,M_{ij}^{xy})$ is the reprojection error when projecting the point X_{ij} with the camera pose $(R_i,\mathbf{t}_i)$ with respect to the measured point M_{ij}^{xy}:

$$\mathcal{R}(X_{ij},R_i,\mathbf{t}_i,M_{ij}^{xy}) = \left\| \frac{(R_i^T(\mathbf{X}_{ij}-\mathbf{t}_i))_{1,2}}{(R_i^T(\mathbf{X}_{ij}-\mathbf{t}_i))_3} - M_{ij}^{xy} \right\| \quad (7)$$

Static loss. As discussed in Sec. 2.2, to better constrain the camera poses, the first predicted basis element $B_1(M) \in \mathbb{R}^{P\times 3}$ defines a static (fixed in time) point cloud approximation. Our network also predicts a movement coefficient $\gamma_j(M)$ for each world point that defines a zero-mean Cauchy distribution. Given γ_j and the reprojection error $r_{ij} = \mathcal{R}(B_{1j},R_i,\mathbf{t}_i,M_{ij}^{xy})$[6] of the j^{th} point of B_1 that is projected by the i^{th} camera, the negative log-likelihood of r_{ij} distributed according to the γ_j-zero-mean Cauchy distribution is proportional to:

$$\mathcal{C}(r_{ij},\gamma_j) = \log\left(\gamma_j + \frac{r_{ij}^2}{\gamma_j}\right) \quad (8)$$

Note, that this loss reduces the contribution of the reprojection errors for points with high γ, but also encourages γ to be small, i.e. encouraging the static point cloud to represent the dynamic scene when possible. Our static loss is the mean negative log-likelihood over all observed points in all frames:

$$\mathcal{L}_{\text{Static}} = \frac{1}{\sum_{i=1}^N\sum_{j=1}^P M_{ij}^o}\sum_{i=1}^N\sum_{j=1}^P M_{ij}^o\mathcal{C}(\mathcal{R}(B_{1j},R_i,\mathbf{t}_i,M_{ij}^{xy}),\gamma_j) \quad (9)$$

[6] We denote the j^{th} 3D point of $B_k \in \mathbb{R}^{P\times 3}$ by $B_{kj} \in \mathbb{R}^3$. The 3 elements of this point are denoted by $B_{kj1},B_{kj2},B_{kj3} \in \mathbb{R}$ (see (11)).

Table 1: Pet evaluation. **Top**: Baseline method results for structure or camera estimation (or both). **Bottom**: Our results with several configurations. (C),(D), or (CD) respectively indicate the object categories in the training set: cats, dogs, or both. BA and FT respectively indicate a post-processing of Bundle Adjustment or fine-tuning.

	Abs Rel ↓		$\delta < 1.25$ ↑		$\delta < 1.25^2$ ↑		$\delta < 1.25^3$ ↑		ATE ↓	RPE Trans ↓	RPE Rot ↓	Time
	Dyn.	All	Dyn.	All	Dyn.	All	Dyn.	All	(mm)	(mm)	(deg)	(min)
D-SLAM [42]	-	-	-	-	-	-	-	-	5.08	3.60	0.20	0.16
ParticleSFM [59]	-	-	-	-	-	-	-	-	12.79	6.95	0.51	11.00
RCVD [21]	0.40	3.6E+07	0.43	0.72	0.75	0.90	0.92	0.96	43.95	25.77	2.31	20.00
CasualSAM [57]	**0.09**	**0.06**	**0.93**	**0.97**	0.99	**0.99**	**1.00**	**1.00**	6.90	3.95	0.22	1.3E+02
MiDaS [3]	0.16	6.2E+04	0.78	0.71	0.97	0.88	**1.00**	0.93	-	-	-	**0.15**
Ours (C)	0.11	0.08	0.88	0.92	0.99	0.98	**1.00**	**1.00**	8.96	3.79	0.23	**0.15**
Ours (C)+BA	0.11	0.08	0.88	0.92	0.99	0.98	**1.00**	**1.00**	4.22	2.86	0.17	**0.15**
Ours (C)+FT	**0.09**	**0.06**	0.90	0.96	**1.00**	**0.99**	**1.00**	**1.00**	4.00	**2.74**	**0.16**	4.86
Ours (D)	0.12	0.08	0.85	0.91	0.99	**0.99**	**1.00**	**1.00**	8.03	3.54	0.23	**0.15**
Ours (D)+BA	0.12	0.08	0.85	0.91	0.99	**0.99**	**1.00**	**1.00**	4.19	2.83	0.17	**0.15**
Ours (D)+FT	**0.09**	**0.06**	0.88	0.96	**1.00**	**0.99**	**1.00**	**1.00**	**3.98**	**2.74**	**0.16**	4.86
Ours (CD)	0.12	0.08	0.85	0.91	0.98	0.98	**1.00**	**1.00**	8.11	3.68	0.24	**0.15**
Ours (CD)+BA	0.12	0.08	0.85	0.91	0.98	0.98	**1.00**	**1.00**	4.21	2.86	0.17	**0.15**
Ours (CD)+FT	**0.09**	**0.06**	0.90	0.96	**1.00**	**0.99**	**1.00**	**1.00**	**3.98**	**2.74**	**0.16**	4.86

Table 2: Out-of-training-domain evaluation . Evaluation metrics on monocular videos from [56]. The table has the same structure as Tab. 1.

	Abs Rel ↓		$\delta < 1.25$ ↑		$\delta < 1.25^2$ ↑		$\delta < 1.25^3$ ↑		ATE ↓	RPE Trans ↓	RPE Rot ↓	Time
	Dyn.	All	Dyn.	All	Dyn.	All	Dyn.	All	(mm)	(mm)	(deg)	(min)
D-SLAM [42]	-	-	-	-	-	-	-	-	7.96	10.91	0.07	0.18
ParticleSFM [59]	-	-	-	-	-	-	-	-	26.66	23.83	0.20	2.13
RCVD [21]	0.19	2.6E+05	0.69	0.75	0.95	0.95	0.96	0.98	1.6E+02	3.2E+02	3.43	7.00
CasualSAM [57]	**0.05**	**0.03**	0.95	**0.99**	**0.99**	**1.00**	**1.00**	**1.00**	**7.81**	**10.09**	**0.06**	22.00
MiDaS [3]	2.8E+04	2.7E+05	0.59	0.58	0.73	0.72	0.83	0.80	-	-	-	**0.02**
Ours (C)	0.08	0.06	0.89	0.95	**0.99**	0.99	0.99	**1.00**	32.06	47.99	0.45	0.04
Ours (C)+BA	0.08	0.06	0.89	0.95	**0.99**	0.99	0.99	**1.00**	8.67	12.36	0.08	0.04
Ours (C)+FT	0.07	**0.03**	0.94	0.98	**0.99**	**1.00**	**1.00**	**1.00**	7.98	11.64	0.08	0.59
Ours (D)	0.08	0.07	0.92	0.93	**0.99**	0.98	0.99	**1.00**	33.77	51.64	0.61	0.04
Ours (D)+BA	0.08	0.07	0.92	0.93	**0.99**	0.98	0.99	**1.00**	8.40	12.06	0.08	0.04
Ours (D)+FT	**0.05**	**0.03**	**0.97**	**0.99**	**0.99**	**1.00**	0.99	**1.00**	8.15	11.88	0.09	0.59
Ours (CD)	0.10	0.08	0.93	0.94	**0.99**	0.99	**1.00**	**1.00**	36.17	53.94	0.67	0.04
Ours (CD)+BA	0.10	0.08	0.93	0.94	**0.99**	0.99	**1.00**	**1.00**	8.62	12.49	0.08	0.04
Ours (CD)+FT	0.06	**0.03**	**0.97**	**0.99**	**0.99**	**1.00**	0.99	**1.00**	8.04	11.84	0.09	0.59

Regularization losses. As in [27] we regularize the observed points to be in front of the camera by:

$$\mathcal{L}_{\text{Negative}} = -\sum_{i=1}^{N}\sum_{j=1}^{P} M_{ij}^{o}\ \text{Min}(R_i^T(\mathbf{X}_{ij} - \mathbf{t}_i))_3, 0) \tag{10}$$

We further find it beneficial to regularize the deviation from the static approximation B_1 to be sparse for static points, i.e. points with low γ values:

$$\mathcal{L}_{\text{Sparse}} = \frac{1}{P(K-1)}\sum_{k=2}^{K}\sum_{j=1}^{P}\frac{1}{3\gamma_j}\left(|B_{kj1}| + |B_{kj2}| + |B_{kj3}|\right) \tag{11}$$

where γ is detached from the gradient calculation for this loss.

3 Experiments

In this section, we conduct experiments to verify our proposed network's performance on real-world casual videos. We began by training the network on specific

Table 3: Ablation study. The contribution of different parts from our method. See details in the text.

	Abs Rel ↓		$\delta < 1.25$ ↑		$\delta < 1.25^2$ ↑		$\delta < 1.25^3$ ↑		Rep.(pix.) ↓		ATE ↓	RPE Trans ↓	RPE Rot ↓
	Dyn.	All	Dyn.	All	Dyn.	All	Dyn.	All	Dyn.	All	(mm)	(mm)	(deg)
Set of Sets	0.27	0.15	0.60	0.77	0.87	0.94	0.97	0.99	9.86	5.33	16.87	5.53	0.39
No $\mathcal{L}_{\text{Static}}$	0.77	0.36	0.25	0.46	0.48	0.70	0.68	0.82	**1.00**	**0.86**	96.20	29.86	0.99
No γ	0.22	0.16	0.66	0.73	0.93	0.91	0.99	0.97	4.54	2.41	13.91	4.86	0.29
K=30	0.14	0.09	0.81	0.90	0.97	**0.98**	0.99	0.99	4.88	2.78	9.39	**3.68**	**0.23**
K=2	**0.11**	**0.08**	**0.88**	0.91	0.98	**0.98**	**1.00**	**1.00**	8.58	3.56	9.31	3.86	0.25
DSS	1.65	0.58	0.19	0.35	0.34	0.60	0.47	0.74	63.75	70.60	34.90	22.63	1.64
No $\mathcal{L}_{\text{Sparse}}$	0.17	0.13	0.79	0.80	0.95	0.94	**1.00**	0.99	4.57	2.73	11.79	7.99	0.55
Full	**0.11**	**0.08**	**0.88**	**0.92**	**0.99**	**0.98**	**1.00**	**1.00**	3.98	1.97	**8.96**	3.79	**0.23**

domains and then evaluated its accuracy and running time on unseen videos from both, training and unseen domains.

Training procedure. We trained our network on the cat and dog partitions from the COP3D dataset [38], which contains a diverse set of casual real-world videos of pets. Our networks were trained from scratch three times to test our generalization capability between semantic categories: once on the cat partition, once on the dog partition, and once on both partitions combined. In total, we used 733 cat videos and 753 dog videos for training. We trained our networks for 7000 and 3500 epochs for the single-class and multi-class setups respectively. Training our method lasts about one week on a single Tesla V100 GPU with 32GB memory. We used the Adam optimizer [19] with a learning rate of 10^{-4}. Our method assumes known camera internal parameters which are provided by the dataset and used to normalize the point tracks as a preprocessing step. More technical details are provided in the supplementary material. We use $K = 12$ bases in all our experiments (Sec. 2.2).

Evaluation data. To assess our framework's performance on pet videos, we curated a new dataset[7] consisting of 21 casual videos of dogs and cats, each video containing 50 frames. These videos were captured using an RGBD (RGB-Depth) sensor. The depth maps were used as ground truth for evaluating the reconstructed structure. We extracted the cameras by running COLMAP on the images while masking out the pet areas with dilatated masks provided by [62]. The cameras were scaled to millimeter units using the provided GT depth. Note that our network did not see this test data during training and it was not used to tune our hyperparameters.

Additionally, to evaluate our method on out-of-domain evaluation data, we used the Nvidia Dynamic Scenes Dataset [56]. Specifically, while our network was trained on pet videos, this dataset contains other dynamic object types, e.g. human, balloon, truck, and umbrella, with a different camera motion type.

[7] While the COP3D dataset provides cameras that were extracted by COLMAP [36], we note that this evaluation data is insufficient in our case. This is because the dynamic structure was captured as well in part of their reconstruction which indicates that its reconstruction might not be accurate. Furthermore, the coordinates system units of these reconstructions are unknown. Finally, this dataset does not have any depth map information for evaluating the dynamic structure.

The dataset contains 8 dynamic scenes which are captured by 12 synchronized cameras, enabling accurate depth estimation which is treated as GT for evaluating monocular depth estimation. The ground truth camera poses were calculated by [36] with the synchronized multiview camera rig and the ground truth dynamic masks. Similarly to [23] we simulated 8 monocular dynamic video sequences using the camera rig, each with 24 frames, and used them for evaluation.

Evaluation results. Qualitative visualizations are presented in Fig. 3. [8] We also show a visualization of the movement level values, γ in Fig. 4. For comparisons, we chose state-of-the-art methods that as our method, can be applied to raw casual videos that were captured by standard pinhole camera models and do not need any static or dynamic segmentation. We evaluated both, the camera poses and the structure accuracies. Comparison results for the pet-test-set and out-of-domain dataset are presented in Tables 1 and 2 respectively. The camera poses are evaluated compared to the GT, using the Absolute Translation Error(ATE), the Relative Translation Error(RTE), and the Relative Rotation Error(RRE) metrics after coordinates system alignment. We compare three training configurations of our method of training only on cats, only on dogs, and on both. As can be seen in the tables, the results are similar in all 3 cases. Our output camera poses as inferred by the network are already accurate and outperform some of the prior methods. We further show the results of our method after a single and short round of Bundle Adjustment, which makes our method better than all baselines on the pet sequences, and comparable on the out-of-domain cases.

Importantly, Tables 1 and 2 also compare the method's runtime. It can be seen that our method, even with bundle adjustment, is the fastest camera prediction method. Tables 1 and 2 also show structure evaluation with the depth evaluation metrics [10] on the sampled point tracks. They demonstrate that our inferred structure is almost comparable to the state-of-the-art [57] while taking significantly shorter running times (a few seconds for our method versus more than two hours for [57] on pet videos). Short (0.6-5 minutes), per-sequence fine-tuning makes our method's accuracy comparable to [57]. In terms of running time, our method is a bit slower than MiDaS [3], which only provides depth maps without cameras, but achieves much better results. We note that in contrast to the other methods that predict the dynamic depth, ours does not use any depth-from-single-image prior. Note that our method running times include the point tracking time that is performed by [17] as a pre-process.

Ablation study To evaluate the contribution of our different method parts we run an ablation study which is presented in Tab. 3. In this study, the training was always done on the cat partition from COP3D and evaluated on our test data which contains dogs and cats. First, we performed an ablation study on our transformer architecture by taking the architecture suggested by [27] ("Set

[8] The reader is encouraged to watch the supplementary videos for better 4D perception.

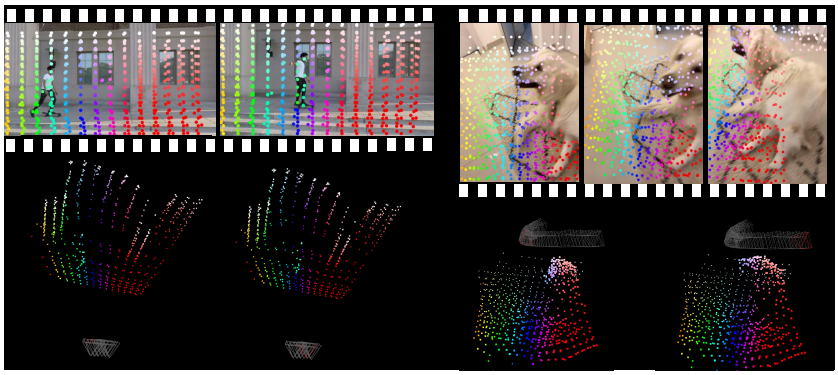

Fig. 3: Qualitative Results. **Top**. Frames from 2 different test video sequences with point tracks marked with corresponding colors. **Bottom**. A 3D visualization of our method's outputs, from two time stamps. The camera trajectory is present as gray frustums, whereas the current camera is marked in red. The reconstructed 3D scene points are presented in corresponding colors to the input tracks on the top. The scene is observed from the same viewpoint, enabling the visualization of the dynamic reconstructed structure.

Fig. 4: γ **Visualization.** We show a visualization of the γ outputs of our network that are described in Sec. 2.2. In each video sequence, we show the input tracks, where each color visualizes its movement level value, γ. Purple marks static points with low γ whereas red marks dynamic points with high γ. Note, that our network did not get any direct supervision for these values, but only the raw point tracks predictions from [17]. The γ visualizations for cats were produced by the model that was only trained on dogs and vice versa. We note that our model generalizes well to out-of-domain (non-pet) cases as well.

of Sets") or the DSS architecture that uses only linear layers [26] ("DSS"). As the table shows our architecture ("Full") achieved significantly better results. To test the losses in our framework, we also evaluated the following: (1) ignoring the γ outputs and using regular reprojection error on B_1 for all points ("No γ"); (2) removing our sparsity loss ("No $\mathcal{L}_{\text{Sparse}}$"); and (3) removing the static loss ("No $\mathcal{L}_{\text{Static}}$"). In all cases, the error increased whereas in the later one, the results became much worse. We further ablate the choice of $K = 12$ as the number of linear bases, by trying 2 extreme numbers of $K = 30, K = 2$ (we saw no significant differences when we used nearby choices such as $K = 11$). As can

be seen in the table, when $K = 30$ the output is not regularized enough and produces a higher depth error for the dynamic part. For $K = 2$ the depth is regularized but the reprojection error ("Rep. (pix.)") gets higher due to over-regularization. Overall, this study justifies our design choices ("Full").

4 Related Work

Simultaneous Localization and Mapping (SLAM) and Structure from Motion (SfM) SfM pipelines seek to recover static 3D structure and camera poses from unordered images. [1, 36, 39, 43, 49]. Learning-free pipelines [36] are effective but require repeated applications of Bundle Adjustment (BA) [44]. [7, 27] presented a method for learning prior from a dataset of multiview image sets, to accelerate SfM pipelines by using equivariant deep networks. Monocular Simultaneous Localization and Mapping (SLAM) methods [4, 11, 28, 29, 40, 48, 55, 60, 61] extract camera poses from video sequences while defining a scene map with keyframes. These methods assume static scenes, fail to produce the cameras in scenes with large portions of dynamic motion, and cannot reproduce dynamic parts of the scene. DROID-SLAM [42] used synthetic data with ground truth 3D supervision for learning to predict camera poses via deep-based BA on keyframes while excluding dynamic objects. ParticleSfM [59] filters out 2D dynamic content for reproducing the cameras in dynamic scenes, using its pre-trained motion prediction network. Both, [42, 59] do not infer the dynamic 3D structure.

Orthographic Non-Rigid SfM (NRSfM) Bregler et al. [5] introduced a factorization method for computing a non-rigid structure and rotation matrices from a point track matrix, by assuming a low dimensional basis model. While follow-up papers improved accuracies with different regularizations [8, 15, 22, 31] or neural representations [20, 30, 37], the orthographic camera model assumption is in general not valid for casual videos. Furthermore, these methods often assume background subtraction as a preprocessing. Even though a follow-up work proposed factorization solutions for pinhole cameras [13], its sensitivity to noise [16], makes it impractical for casual videos.

Test-time optimization for dynamic scenes Recent methods [21, 25, 57, 58] fine-tuned the monocular depth estimation from a pre-trained model [34,35] using optical flow constraints [41], for obtaining consistent dense depth maps for a monocular video. [57] further optimized motion maps for handling scenes with large dynamic motion. [12, 50] use depth from single image estimations, to improve novel view synthesis in dynamic scenes. [24] further optimizes for the unknown camera poses together with the dynamic radiance field optimization. [32, 33] model a single deformable surface from a monocular video by applying isometric constraints. LASR [52], ViSER [53] and BANMo [54] optimize for a dynamic surface by assuming rigid bones and linear blend skinning weights. However, all the above-mentioned methods require per-scene optimization, resulting in slow inference. Recently, [38] presented the Common Pets in 3D (COP3D) dataset that contains casual, in-the-wild videos of pets, and used it to learn priors for novel view synthesis in dynamic scenes.

Point tracking There has been a recent advance in 2D point tracking by learning [9,17], or optimization [47] techniques. Concurrently, [51] presented a method for jointly performing 2D tracking and 3D lifting, by learning to track with depth supervision while applying an as-rigid-as-possible loss. However, their method cannot predict camera poses or identify static parts directly.

5 Conclusions and limitations

We presented TracksTo4D, a novel deep-learning framework that directly maps 2D motion tracks from casual videos into their corresponding dynamic structure and camera motion. Our approach features a deep learning architecture that considers the symmetries in the problem with designed intrinsic constraints to handle the ill-posed nature of this problem. Notably, our network was trained using only raw supervision of 2D point tracks extracted by an off-the-shelf method [17] without any 3D supervision. Yet, it implicitly learned to predict camera poses and 3D structures while identifying the dynamic parts. During inference, our method demonstrates significantly faster processing times compared to previous methods while achieving comparable accuracy. Furthermore, our network exhibits strong generalization capabilities, performing well even on semantic categories that were not present in the training data.

Limitations and future work. While our experiments demonstrated that our network is efficient, accurate, and capable of generalizing to unseen video categories, there are several limitations and future work directions that we would like to address. First, our method cannot handle videos with too rapid motion, and in general, is limited by the accuracy of the tracking method [17]. We note that any future improvements with point tracking in terms of accuracy and inference time will directly improve our method as well. Our method assumes enough motion parallax to constrain the depth values and fails to generate accurate camera poses without it. A future and interesting work would be to try combining depth-from-single-image prior as additional inputs to our network for handling cases with minimal motion parallax and improving accuracies. While we found $K = 12$ basis elements to be effective for our evaluation set, balancing complexity reduction and motion representation, we acknowledge this fixed number may not capture all possible scene dynamics. Future work could explore automatically inferring the optimal number of bases per scene. Lastly, our network can handle up to about 1000 point tracks in 50 frames in one inference step when running on a single GPU. A possible extension to handle denser point clouds could involve querying point tracks iteratively while maintaining a shared state, but this approach remains to be explored.

Acknowledgments HM is the Robert J. Shillman Fellow, and is supported by the Israel Science Foundation through a personal grant (ISF 264/23) and an equipment grant (ISF 532/23).

References

1. Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S.M., Szeliski, R.: Building rome in a day. Communications of the ACM **54**(10), 105–112 (2011)
2. Akhter, I., Sheikh, Y., Khan, S., Kanade, T.: Nonrigid structure from motion in trajectory space. Advances in neural information processing systems **21** (2008)
3. Birkl, R., Wofk, D., Müller, M.: Midas v3.1 – a model zoo for robust monocular relative depth estimation. arXiv preprint arXiv:2307.14460 (2023)
4. Bloesch, M., Czarnowski, J., Clark, R., Leutenegger, S., Davison, A.J.: Codeslam—learning a compact, optimisable representation for dense visual slam. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2560–2568 (2018)
5. Bregler, C., Hertzmann, A., Biermann, H.: Recovering non-rigid 3d shape from image streams. In: Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662). vol. 2, pp. 690–696. IEEE (2000)
6. Bronstein, M.M., Bruna, J., Cohen, T., Veličković, P.: Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. arXiv preprint arXiv:2104.13478 (2021)
7. Brynte, L., Iglesias, J.P., Olsson, C., Kahl, F.: Learning structure-from-motion with graph attention networks. arXiv preprint arXiv:2308.15984 (2023)
8. Dai, Y., Li, H., He, M.: A simple prior-free method for non-rigid structure-from-motion factorization. International Journal of Computer Vision **107**, 101–122 (2014)
9. Doersch, C., Yang, Y., Vecerik, M., Gokay, D., Gupta, A., Aytar, Y., Carreira, J., Zisserman, A.: Tapir: Tracking any point with per-frame initialization and temporal refinement. arXiv preprint arXiv:2306.08637 (2023)
10. Eigen, D., Puhrsch, C., Fergus, R.: Depth map prediction from a single image using a multi-scale deep network. Advances in neural information processing systems **27** (2014)
11. Engel, J., Schöps, T., Cremers, D.: Lsd-slam: Large-scale direct monocular slam. In: European conference on computer vision. pp. 834–849. Springer (2014)
12. Gao, C., Saraf, A., Kopf, J., Huang, J.B.: Dynamic view synthesis from dynamic monocular video. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5712–5721 (2021)
13. Hartley, R., Vidal, R.: Perspective nonrigid shape and motion recovery. In: Computer Vision–ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part I 10. pp. 276–289. Springer (2008)
14. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge university press (2003)
15. Iglesias, J.P., Olsson, C., Valtonen Örnhag, M.: Accurate optimization of weighted nuclear norm for non-rigid structure from motion. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16. pp. 21–37. Springer (2020)
16. Jensen, S.H.N., Doest, M.E.B., Aanæs, H., Del Bue, A.: A benchmark and evaluation of non-rigid structure from motion. International Journal of Computer Vision **129**(4), 882–899 (2021)
17. Karaev, N., Rocco, I., Graham, B., Neverova, N., Vedaldi, A., Rupprecht, C.: Cotracker: It is better to track together. arXiv preprint arXiv:2307.07635 (2023)

18. Kendall, A., Gal, Y.: What uncertainties do we need in bayesian deep learning for computer vision? Advances in neural information processing systems **30** (2017)
19. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
20. Kong, C., Lucey, S.: Deep non-rigid structure from motion. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1558–1567 (2019)
21. Kopf, J., Rong, X., Huang, J.B.: Robust consistent video depth estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1611–1621 (2021)
22. Kumar, S., Van Gool, L.: Organic priors in non-rigid structure from motion. In: European Conference on Computer Vision. pp. 71–88. Springer (2022)
23. Li, Z., Niklaus, S., Snavely, N., Wang, O.: Neural scene flow fields for space-time view synthesis of dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6498–6508 (2021)
24. Liu, Y.L., Gao, C., Meuleman, A., Tseng, H.Y., Saraf, A., Kim, C., Chuang, Y.Y., Kopf, J., Huang, J.B.: Robust dynamic radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13–23 (2023)
25. Luo, X., Huang, J.B., Szeliski, R., Matzen, K., Kopf, J.: Consistent video depth estimation. ACM Transactions on Graphics (ToG) **39**(4), 71–1 (2020)
26. Maron, H., Litany, O., Chechik, G., Fetaya, E.: On learning sets of symmetric elements. In: International conference on machine learning. pp. 6734–6744. PMLR (2020)
27. Moran, D., Koslowsky, H., Kasten, Y., Maron, H., Galun, M., Basri, R.: Deep permutation equivariant structure from motion. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5976–5986 (2021)
28. Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: Orb-slam: a versatile and accurate monocular slam system. IEEE transactions on robotics **31**(5), 1147–1163 (2015)
29. Newcombe, R.A., Lovegrove, S.J., Davison, A.J.: Dtam: Dense tracking and mapping in real-time. In: 2011 international conference on computer vision. pp. 2320–2327. IEEE (2011)
30. Novotny, D., Ravi, N., Graham, B., Neverova, N., Vedaldi, A.: C3dpo: Canonical 3d pose networks for non-rigid structure from motion. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7688–7697 (2019)
31. Oskarsson, M., Batstone, K., Astrom, K.: Trust no one: Low rank matrix factorization using hierarchical ransac. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5820–5829 (2016)
32. Parashar, S., Pizarro, D., Bartoli, A.: Isometric non-rigid shape-from-motion with riemannian geometry solved in linear time. IEEE transactions on pattern analysis and machine intelligence **40**(10), 2442–2454 (2017)
33. Parashar, S., Pizarro, D., Bartoli, A.: Robust isometric non-rigid structure-from-motion. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(10), 6409–6423 (2021)
34. Ranftl, R., Bochkovskiy, A., Koltun, V.: Vision transformers for dense prediction. ICCV (2021)
35. Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., Koltun, V.: Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(3) (2022)
36. Schonberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4104–4113 (2016)

37. Sidhu, V., Tretschk, E., Golyanik, V., Agudo, A., Theobalt, C.: Neural dense non-rigid structure from motion with latent space constraints. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16. pp. 204–222. Springer (2020)
38. Sinha, S., Shapovalov, R., Reizenstein, J., Rocco, I., Neverova, N., Vedaldi, A., Novotny, D.: Common pets in 3d: Dynamic new-view synthesis of real-life deformable categories. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4881–4891 (2023)
39. Sturm, P., Triggs, B.: A factorization based algorithm for multi-image projective structure and motion. In: Computer Vision—ECCV'96: 4th European Conference on Computer Vision Cambridge, UK, April 15–18, 1996 Proceedings Volume II 4. pp. 709–720. Springer (1996)
40. Teed, Z., Deng, J.: Deepv2d: Video to depth with differentiable structure from motion. arXiv preprint arXiv:1812.04605 (2018)
41. Teed, Z., Deng, J.: 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. pp. 402–419. Springer (2020)
42. Teed, Z., Deng, J.: Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. Advances in neural information processing systems **34**, 16558–16569 (2021)
43. Tomasi, C., Kanade, T.: Shape and motion from image streams under orthography: a factorization method. International journal of computer vision **9**, 137–154 (1992)
44. Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W.: Bundle adjustment—a modern synthesis. In: Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms Corfu, Greece, September 21–22, 1999 Proceedings. pp. 298–372. Springer (2000)
45. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
46. Wan, Z., Li, Z., Tian, M., Liu, J., Yi, S., Li, H.: Encoder-decoder with multi-level attention for 3d human shape and pose estimation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 13033–13042 (2021)
47. Wang, Q., Chang, Y.Y., Cai, R., Li, Z., Hariharan, B., Holynski, A., Snavely, N.: Tracking everything everywhere all at once. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 19795–19806 (2023)
48. Wang, W., Hu, Y., Scherer, S.: Tartanvo: A generalizable learning-based vo. In: Conference on Robot Learning. pp. 1761–1772. PMLR (2021)
49. Wu, C.: Towards linear-time incremental structure from motion. In: 2013 International Conference on 3D Vision-3DV 2013. pp. 127–134. IEEE (2013)
50. Xian, W., Huang, J.B., Kopf, J., Kim, C.: Space-time neural irradiance fields for free-viewpoint video. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9421–9431 (2021)
51. Xiao, Y., Wang, Q., Zhang, S., Xue, N., Peng, S., Shen, Y., Zhou, X.: Spatialtracker: Tracking any 2d pixels in 3d space. arXiv preprint arXiv:2404.04319 (2024)
52. Yang, G., Sun, D., Jampani, V., Vlasic, D., Cole, F., Chang, H., Ramanan, D., Freeman, W.T., Liu, C.: Lasr: Learning articulated shape reconstruction from a monocular video. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15980–15989 (2021)
53. Yang, G., Sun, D., Jampani, V., Vlasic, D., Cole, F., Liu, C., Ramanan, D.: Viser: Video-specific surface embeddings for articulated 3d shape reconstruction. Advances in Neural Information Processing Systems **34**, 19326–19338 (2021)

54. Yang, G., Vo, M., Neverova, N., Ramanan, D., Vedaldi, A., Joo, H.: Banmo: Building animatable 3d neural models from many casual videos. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2863–2873 (2022)
55. Yang, N., Stumberg, L.v., Wang, R., Cremers, D.: D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 1281–1292 (2020)
56. Yoon, J.S., Kim, K., Gallo, O., Park, H.S., Kautz, J.: Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5336–5345 (2020)
57. Zhang, Z., Cole, F., Li, Z., Rubinstein, M., Snavely, N., Freeman, W.T.: Structure and motion from casual videos. In: European Conference on Computer Vision. pp. 20–37. Springer (2022)
58. Zhang, Z., Cole, F., Tucker, R., Freeman, W.T., Dekel, T.: Consistent depth of moving objects in video. ACM Transactions on Graphics (TOG) **40**(4), 1–12 (2021)
59. Zhao, W., Liu, S., Guo, H., Wang, W., Liu, Y.J.: Particlesfm: Exploiting dense point trajectories for localizing moving cameras in the wild. In: European Conference on Computer Vision. pp. 523–542. Springer (2022)
60. Zhao, W., Liu, S., Shu, Y., Liu, Y.J.: Towards better generalization: Joint depth-pose learning without posenet. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9151–9161 (2020)
61. Zhou, H., Ummenhofer, B., Brox, T.: Deeptam: Deep tracking and mapping. In: Proceedings of the European conference on computer vision (ECCV). pp. 822–838 (2018)
62. Zou, X., Yang, J., Zhang, H., Li, F., Li, L., Wang, J., Wang, L., Gao, J., Lee, Y.J.: Segment everything everywhere all at once. Advances in Neural Information Processing Systems **36** (2024)