AnyCam: Learning to Recover Camera Poses and Intrinsics from Casual Videos

Felix Wimbauer[1,2,3] Weirong Chen[1,2,3] Dominik Muhle[1,2] Christian Rupprecht[3]
Daniel Cremers[1,2]
[1]Technical University of Munich [2] MCML [3] University of Oxford
felix.wimbauer@tum.de cremers@tum.de

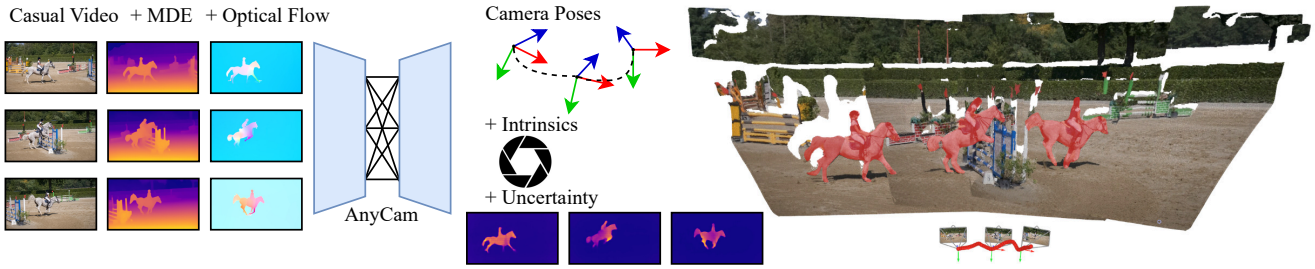

Figure 1. **AnyCam.** Given a casual video and pretrained monocular depth estimation (MDE) and optical flow networks, AnyCam outputs camera poses, camera intrinsics, and uncertainty maps in a single forward pass. The uncertainty maps represent probable movement in the scene. By using a novel loss formulation, AnyCam can be trained on a large corpus of unlabelled videos mostly obtained from YouTube.

Abstract

Estimating camera motion and intrinsics from casual videos is a core challenge in computer vision. Traditional bundle-adjustment based methods, such as SfM and SLAM, struggle to perform reliably on arbitrary data. Although specialized SfM approaches have been developed for handling dynamic scenes, they either require intrinsics or computationally expensive test-time optimization and often fall short in performance. Recently, methods like Dust3r have reformulated the SfM problem in a more data-driven way. While such techniques show promising results, they are still 1) not robust towards dynamic objects and 2) require labeled data for supervised training. As an alternative, we propose ***AnyCam****, a fast transformer model that directly estimates camera poses and intrinsics from a dynamic video sequence in feed-forward fashion. Our intuition is that such a network can learn strong priors over realistic camera poses. To scale up our training, we rely on an uncertainty-based loss formulation and pre-trained depth and flow networks instead of motion or trajectory supervision. This allows us to use diverse, unlabelled video datasets obtained mostly from YouTube. Additionally, we ensure that the predicted trajectory does not accumulate drift over time through a lightweight trajectory refinement step. We test AnyCam on established datasets, where it delivers accurate camera poses and intrinsics both qualitatively and quantitatively. Furthermore, even with trajectory refinement, AnyCam is significantly faster than existing works for SfM in dynamic settings. Finally, by combining camera information, uncertainty, and depth, our model can produce high-quality 4D pointclouds. For more details and code, please check out our project page: fwmb.github.io/anycam*

1. Introduction

Estimating camera motion and intrinsics from a casual, dynamic video is a long-standing problem in 3D computer vision. Such scene parameters serve as the basis for a plethora of more complex applications, ranging from novel view synthesis to reconstruction. In particular, an efficient and robust solution to this problem would unlock the huge corpus of online video data from sources like YouTube to train 3D foundation models. To this day, the availability of 3D data is one of the main limitations when creating general-purpose 3D models. [33]

Traditional SfM (and SLAM) systems like COLMAP [32] excel at reconstruction in settings where a static scene is captured from many well-distributed camera angles. However, they generally fail when there are dynamic parts in the scene, the camera follows a suboptimal trajectory, or the image quality suffers from artifacts. This makes them unsuitable for casual videos. Recently, a number of works have proposed to incorporate certain deep learning-based components into or instead of the classic SfM pipeline to make it more robust. For example, Dust3r [44] proposes a network to predict dense 3D pointmaps from multiple

views. Camera pose and intrinsics can then be recovered by aligning the point maps from every view. FlowMap [34] reformulates the problem by learning per-frame depth and aligning them based on ow and point tracks. While these proposed approaches show promising results, they are still restricted to static scenes and fail when dynamic objects are too prominent. There are methods speci cally designed to deal with dynamic scenes, e.g. LeapVO [5] and ParticleSfM [56]. However, they are generally trained in a supervised way and require calibrated camera intrinsics. Given the general lack of intrinsics information in casual videos, the applicability of such methods in this domain is limited.

This work presents AnyCam, a novel method targeted explicitly for robust camera motion and intrinsics estimation in dynamic casual videos. Unlike most existing approaches, we utilize a transformer-based model to directly predict relative camera poses and intrinsics for a sequence of video frames. Crucially, this end-to-end formulation allows our network to learn strong priors over plausible camera poses and to become robust towards imperfect inputs. Additionally, we design a training scheme that can ingest raw videos and does not require any labels. Our formulation effectively models data uncertainty and automatically lters out dynamic objects. This enables training on a collection of datasets obtained from YouTube and other casual video sources, resulting in strong generalization capabilities. Furthermore, we present a lightweight test-time re nement step to avoid long-term drift.

Our experiments thoroughly test AnyCam's zero-shot capabilities in camera pose estimation and intrinsics recovery. We achieve state-of-the-art results on several challenging dynamic benchmarks and perform on par with methods trained fully supervised with labeled data. Additionally, by combining camera information, uncertainties, and depth maps, we can create high- delity 4D pointclouds.

2. Related Work

2.1. Foundational Models for Depth and Flow

In the tasks of monocular depth (MDE) and ow estimation, well-generalizable foundation models have replaced early deep learning approaches [8, 12, 13] in the last years. For MDE, DepthAnything [50] uses a data engine to construct a large corpus of automatically annotated data to learn relative depth estimation. Additional ne-tuning allows for metric depth estimates. DepthAnythingV2 [51] netunes the previous model using synthetic data for better performance. Metric3D [53] and Metric3Dv2 [16] transform images to canonical camera intrinsics with a xed focal length. DepthPro [1] proposes a two-stage training curriculum with a second stage solely on synthetic data to sharpen boundary predictions. DepthCrafter [17] leverages a conditional diffusion model to predict temporally consistent depth maps for videos. In this work, we utilize UniDepth [28] for metric MDE, which uses a geometric invariance loss on different image augmentation to enforce consistency.

RAFT [38] presented the state of the art for optical ow estimation for a long time. It improved previous methods by introducing a recurrent look-up operator on correlation volumes to iteratively re ne ow predictions without needing coarse-to- ne ow pyramids. GMFlow [46] avoids correlation volumes and instead leverages the properties of transformers for global matching on feature maps. This removes the need for iterative steps to improve runtime performance. UniMatch [47] extends GMFlow network by tasks of disparity and depth prediction to enable cross-task transfer learning of a single transformer network.

We rely on both off-the-shelf MDE and Optical Flow networks to bene t from strong geometric priors during training and inference.

2.2. SfM and SLAM

For many decades, the problem of recovering camera parameters and geometry from images has been formulated as the Structure-from-Motion (SfM) pipeline [14, 24, 26]. While many different implementations of the SfM pipeline exist, COLMAP [32] has emerged as the standard due to its robustness and exibility. One of the drawbacks of SfM methods is their high computational cost. Simultaneous Location and Mapping (SLAM) [9, 10, 22, 23] approaches employ a similar pipeline to SfM but focus on the ef cient processing of consecutive video frames. In recent years, these classical optimization-based approaches were enhanced by learned components [2, 7, 19, 21, 29–31, 52]. However, relying on epipolar geometry [15] or photometric consistency [10] makes them susceptible to high error on highly dynamic scenes. The strong focus on self driving data provided datasets with mostly static environments [4, 11, 36], an assumption that does not hold for casual videos.

2.3. Learning Based SfM and SLAM

Largely learning-based methods started to replace classical SLAM and SfM systems due to improved robustness [39]. DROID-SLAM extends the framework of RAFT [38] by an update operator on both depth and pose estimates. A nal differentiable bundle adjustment (BA) layer produces the - nal pose estimates. ParticleSfM [56] utilizes dense correspondences inside a BA framework to optimize poses. The dense correspondences are initialized from optical ow, and dynamic points are ltered using trajectory-based motion segmentation. CasualSAM [55] predicts both depth and movement from images to get frame-to-frame motion. A global optimization aligns the scale of the prediction and re nes the poses. Dust3R [44] is a dense multi-view stereo method that regresses point coordinates between an image pair. This allows it to be extended to either SfM or

SLAM. FlowMap [34] proposes to reconstruct a scene by over tting a depth network to it and aligning depth maps via correspondences from ow or point tracking. LEAP-VO [5] combines visual and temporal information of video sequences to improve the tracking accuracy of points and identify occluded and dynamic points. A sliding window bundle adjustment then optimizes the poses. The concurrent work of MonST3R [54] netunes Dust3r on mostly synthetic data to generalize it to dynamic scenes. While these works achieve impressive progress, they generally obtain poses from aligning depth and point maps or by optimizing them per-scene. This makes it hard to inject prior information about camera motion. In contrast, our method uses a neural network to predict a trajectory, which can effectively learn priors over realistic camera motions.

3. Method

In the following, we rst describe our transformer-based model AnyCam, which directly predicts camera pose and intrinsics for video sequences. Further, we introduce an uncertainty-based training scheme, which enables training on large-scale dynamic video datasets without any ground truth labels, and a lightweight test-time re nement strategy.

3.1. Preliminaries

As input, our pipeline receives a sequence of video frames $I^i \in ([0;1]^3)^{;i \in \mathcal{N}}$, where $\mathcal{N} = \{1;\dots;n\}$ denotes the set of frame indices and $= \{1;\dots;H\} \times \{1;\dots;W\}$ denotes the pixel lattice. Using off-the-shelf depth and optical ow predictors, we further obtain dense depth maps $D^i \in \mathbb{R}_+^{1\ H\ W}$, as well as dense optical ow $F^{i!\ j} \in \mathbb{R}^{2\ H\ W}$ from any frame i to j. We generally assume a simpli ed pinhole model that is constant for the duration of a sequence. Therefore, camera intrinsics for a video are modeled via a single focal length $f \in \mathbb{R}_+$. Let $_f(x) : \mathbb{R}^3 \to \mathbb{R}^2$ be the corresponding projection function that maps a point $x \in \mathbb{R}^3$ in the camera's coordinate system to the respective pixel $p \in \mathbb{R}^2$ on the image plane parametrized by focal length f. Accordingly, $_f^{-1}(p;d) : (\mathbb{R}^2;\mathbb{R}_+) \to \mathbb{R}^3$ denotes the unprojection function where d is the depth value for p. Let $P^{i!\ j} \in SE3$ describe the relative camera pose, consisting of a rotation and translation, between frames i and j. To map a pixel location p from frame i to frame j given the pixel's depth value d_p in frame i, we write:

$$p' = \ _f \ P^{i!\ j} \ _f^{-1}(p;d_p) \tag{1}$$

3.2. Transformer for Camera Prediction

Camera pose and intrinsics are crucial for many 3D tasks. While SfM and SLAM can be highly accurate in well-posed, static environments, they are very sensitive to outliers. Therefore, dynamic objects pose a signi cant challenge to them. Not only do they make convergence to the correct solution more dif cult, but they also often lead to catastrophic failures like loss of tracking or degenerate solutions. Different techniques were introduced to make the underlying optimizations more robust. These range from robust cost functions and RANSAC, to ltering out objects that could possibly move using off-the-shelf segmentation models. Nevertheless, these techniques are either hand-crafted, struggle to generalize across many domains, or present a signi cant computational overhead.

This paper aims to build a robust camera pose estimation system that can readily be applied to any casual video. We base our design on three key insights:

1. The space of plausible camera poses in videos is much more constrained than in general multi-view settings.
2. Strong, data-driven priors can help resolve ambiguities introduced by dynamic objects.
3. Incorporating information from all pixels instead of selected keypoints makes detecting outliers easier.

Pose from a feed-forward network. For a natural video captured by a single camera, the space of plausible camera movements is well-constrained and continuous. This is especially true when also considering the temporal context of videos. However, this prior information is hard to inject into classical SfM and SLAM approaches [45]. Hand-crafted priors often fail to generalize to different domains and struggle to capture the full variety of plausible motions. On the other hand, when trained on large and diverse datasets, neural networks can learn strong priors over the distribution of plausible camera poses. This knowledge makes them robust to small outliers or noisy measurements and provides reliable generalization abilities. Therefore, we choose a neural network for the prediction of frame-to-frame poses.

We rely on a transformer architecture, which as input receives frames I^i, depth predictions D^i, and ow predictions $F^{i!\ i+1}$ for a short sequence. First, a backbone extracts features for each timestep separately. Stacked self-attention layers allow the model to then exchange information between the frames, producing n pose tokens $^{i!\ i+1}; i \in \{1;\dots;n\ 1\}$ and updated feature maps. Additionally, a single sequence token ^{seq} is predicted.

To decode the features, a frame prediction head $H^P = (H^P;H\ )$ outputs the 6DOF camera poses $P^{i!\ i+1}$ and pixel-wise uncertainty maps $^i \in \mathbb{R}_+$ for every timestep. A sequence head H^{seq} additionally determines a single set of camera intrinsics for the video. In the following, we explain how to robustly design these decoder heads.

Robust intrinsics hypotheses. Camera intrinsics, which in our case are parametrized by focal length f, are notoriously dif cult to disentangle from the camera movement. For example, the same motion pattern in the optical ow can be described by mostly rotation if the camera has a

Figure 2. Architecture. AnyCam processes a sequence of frames from a casual video with corresponding depth maps and optical flow. A backbone extracts feature maps per image. Information sharing between frames is enabled by multiple attention layers that process the features of all sequence images. The transformer architecture outputs one pose token $t_f^{i\to j}$ per timestep and an additional sequence token t^{seq}. The pose tokens are processed using multiple intrinsic hypotheses $\{f_1, \dots, f_m\}$, parametrized by frame prediction heads $(H_f^P; H_f)$. The sequence head H^{seq} predicts the likelihood scores of the different hypotheses. The model is trained end-to-end via a reprojection loss, a pose consistency loss between forward and backward pose predictions, and a KL-divergence loss.

wide field of view or by mostly translation, if the field of view is narrow. Because camera intrinsics have such a major effect, designing a robust system to recover them is vital. While focal length could, in principle, also be predicted by the network directly, we find that it makes training unstable, and the network does not converge to a meaningful result

Instead of considering the focal length as a free variable during training, we reformulate it as a property of the model itself. Considering a fixed set of m candidate focal lengths $\{f_1, \dots, f_m\}$, we train m individual frame prediction heads $\{H_{f_1}, \dots, H_{f_m}\}$. Every head H_f predicts pose $P_f^{i\to i+1}$ and uncertainty map U_f^i under the assumption that the given sequence was captured with a camera of focal length f. For every candidate prediction, we later compute an individual loss. The sequence head H^{seq} learns likelihood scores $P = (p_{f_1}, \dots, p_{f_m})$ for the different candidates. The focal length and corresponding poses with the highest likelihood are the final output of the model.

$$f_{final} = \arg\max_{f \in (f_1, \dots, f_m)} P \tag{2}$$

This idea is similar to [34], but we use candidates as part of the model itself rather than only at the loss level.

3.3. Dynamics-aware Pose Training

We aim to train our model on large, unlabelled datasets. Here, our loss formulation follows two main objectives:

1. Leveraging multi-view information to recover local camera motion between adjacent frames.
2. Using the context of longer sequences to learn realistic long-range camera motion patterns.

Uncertainty-aware flow loss. Using the predicted relative pose $P_f^{i\to j}$ between two frames i and j, and the corresponding depth map D^i, we can project all pixel locations p_{uv} from frame i into frame j. Subtracting the original pixel locations from the projected pixels locations p'^0_f yields the induced optical flow $\hat{F}_f^{i\to j}$:

$$\begin{aligned}\hat{F}_{f;uv}^{i\to j} &= p'^0_{f;uv} - p_{uv} \\ &= \pi_f\left(P_f^{i\to j}\, \pi_f^{-1}(p_{uv}; d_{uv})\right) - p_{uv}\end{aligned} \tag{3}$$

Assuming a static world, the induced flow $\hat{F}_f^{i\to j}$ will match the reference optical flow $F^{i\to j}$ optimally if the predicted pose $P_f^{i\to j}$ and depth D^i is as close to the real pose as possible. However, dynamic objects lead to inconsistencies in the reference optical flow, which would deteriorate gradients during optimization. From a statistical point of view, these inconsistencies cannot be captured via our flow induction formulation. Thus, we choose to model them via so-called aleatoric uncertainty and use the predicted uncertainty map U_i. For a pair of frames i and j, this leads us to the following loss function:

$$\tilde{F}_{f;uv}^{i\to j} = \hat{F}_{f;uv}^{i\to j} - F_{uv}^{i\to j} \tag{4}$$

$$L_f\left(F^{i\to j}\right) = \frac{1}{|j|}\sum_{uv} 2 \ln\left(\frac{1}{\sqrt{2}\,\sigma_{f;uv}^i}\right) \exp\left(\frac{\sqrt{2}\,\tilde{F}_{f;uv}^{i\to j}}{\sigma_{f;uv}^i}\right) \tag{5}$$

which is then summed up over every pair of neighboring frames within the sequence:

$$L_f(F) = \sum_{i=1}^{n-1} L_f\left(F^{i\to i+1}\right) \tag{6}$$

Intuitively, the model learns to downweight areas in the input frame where there will likely be a high loss, i.e. dynamic objects. In turn, the pose supervision signal mostly comes from areas that can likely be captured via the induced

ow, i.e. static parts. In practice, the uncertainty map does not only make pose training more stable, but it can also be used in downstream tasks as a motion segmentation map. Finally, due to its dense nature, the loss is robust to small outliers, such as local inaccuracies of the reference ow.

Learning camera motion patterns. While the ow loss L_f^F provides a strong training signal per frame, the model could, in theory, minimize the loss without considering adjacent frames. However, it is important for the model to also rely on the context of the entire sequence when predicting camera poses. For example, optical ow predictions for some frames could be inaccurate, or dynamic objects could occlude the relevant static part. We introduce an additional loss term and a dropout training strategy to make the model consider the entire sequence.

We rst leverage the fact that when reversing the sequence, the relative poses $P_f^{j\leftarrow i}$ between frames are the same as the inverted poses of the original sequence $(P_f^{i\leftarrow j})^{-1}$. The model becomes more robust towards inaccurate inputs by enforcing consistency between the forward and backward pose predictions during training. We rely on a simple L1 loss, where I_4 is the 4×4 identity matrix:

$$L_f^{\#} = \sum_{i=1}^{n-1} \left\| P_f^{i\leftarrow i+1} \, P_f^{i+1\leftarrow i} - I_4 \right\|_{1;1} \tag{7}$$

Furthermore, we utilize the temporal dependency of poses within a sequence. Given the overall trajectory, the model should learn strong motion priors. To encourage this, we apply a dropout scheme during training by setting values in the pose tokens to zero with probability p_{drop}. This encourages the network to exchange information between frames in the self-attention layers. The network needs to learn realistic motion patterns to overcome the noise injected by dropout. Note that applying dropout on the pose tokens does not affect the uncertainty map.

Finding the best candidate. While the ow loss L_f^F is very effective for learning pose and uncertainty, it also serves as a helpful proxy. It indicates how well the model can reconstruct the observed optical ow using the predicted poses $P_f^{i\leftarrow j}$ and corresponding focal length. After the model has converged during training, it will predict (close to) the optimal poses assuming focal length f. L_f^F now is mainly dependent on f. Thus, the magnitude of L_f^F during training re ects how close f is to the true but unknown actual focal length. We leverage this fact and train the sequence head H^{seq} to predict which of the focal candidates will yield the lowest ow loss.

In practice, we nd that, while correct on average, arg min on the different loss terms is too noisy to enable stable training. Therefore, we rst convert the (inverted) ow loss terms to probability scores using softmax and then optimize the Kullback-Leibler divergence KL_{div}. Let $P = (p_{f_1}; \dots; p_{f_m})$ be the output of H^{seq}:

$$L^{Intr} = KL_{div}\left(P; \mathrm{softmax}(L_{f_1}^F; \dots; L_{f_m}^F)\right) \tag{8}$$

Final loss. By combining all loss terms, we obtain the nal training objective, through which our model learns poses, uncertainties, and intrinsics from unlabelled video sequences:

$$L = \sum_{k=1}^{m} \lambda_F L_{f_k}^F + \lambda_{\#} L_{f_k}^{\#} + \lambda_{Intr} L^{Intr} \tag{9}$$

3.4. Test-time Re nement

Our model predicts relative camera poses for a sequence of frames. This allows our model to be directly used as a visual odometry system by chaining the relative poses together:

$$P^j = \prod_{i=1}^{j} P^{i\leftarrow i+1} \tag{10}$$

While this works well for short to medium-length sequences, it is prone to accumulating drift over a longer time. Small inaccuracies in earlier relative poses will propagate throughout the trajectory.

We perform lightweight test-time re nement using bundle adjustment (BA) [41] to overcome this issue. By chaining multiple optical ow maps $F^{i\leftarrow i+1}; F^{i+1\leftarrow i+2}; \dots$, pixels can be tracked through multiple frames. We initialize the BA system with our predicted trajectory and then optimize over coarse pixel tracks in a sliding-window fashion. Crucially, the BA optimization requires the predicted uncertainty maps to downweigh pixel tracks on dynamic objects. For more details on the BA optimization, please refer to the supplementary material.

4. Experiments

In the following, we thoroughly evaluate the overall performance of our model regarding camera pose estimation and intrinsics recovery. We put a special focus on testing the zero-shot generalization capabilities of our network. Further, we also validate and demonstrate the effectiveness of our design choices.

4.1. Setup

Data. Our training formulation requires no labeled data and is robust towards dynamic objects and suboptimal image quality. This is a key bene t of our method, as dynamic videos with 3D labels are scarce. To highlight this strength, we rely on a diverse mix of datasets (YouTube-VOS [48], RealEstate10K [57], WalkingTours [42], OpenDV [49],

Category	Method	No Supervis.	Approx. Runtime	Sintel			TUM-RGBD (dynamics)		
				ATE#	RPE_{trans} #	RPE_{rot} #	ATE#	RPE_{trans} #	RPE_{rot} #
w/ intrinsics	DROID-SLAM [39]		< 2min	0.175	0.084	1.912	-	-	-
	DPVO [40]		< 2min	0.115	0.072	1.975	-	-	-
	LeapVO [5]		< 2min	0.089	0.066	1.250	0.068	0.008	1.686
w/o intrinsics	ParticleSfM [56]		< 20min	0.129	0.031	0.525	-	-	-
	MonST3R* [54]		< 2min	0.108	0.042	0.732	0.063	0.009	1.217
	Robust-CVD [18]	3	?	0.360	0.154	3.443	0.153	0.026	3.528
	CasualSAM [55]	3	> 1h	0.141	0.035	0.615	0.071	0.010	1.712
	Ours w/o ref.	3	< 20sec	0.099	0.045	0.567	0.095	0.025	1.050
	Ours	3	< 2min	0.078	0.031	0.427	0.056	0.005	0.927

Table 1. Pose estimation in dynamic environments. Absolute trajectory error (ATE) and relative pose error for translation (RPE_{trans}) and rotation (RPE_{rot}) on the Sintel and TUM-RGBD (dynamics) datasets. We compare against other learning-based VO/SLAM systems. AnyCam achieves competitive performance against systems trained in a supervised manner as well as methods that have ground truth camera intrinsics available. We consistently outperform Robust-CVD and are on par with CasualSAM, which has a significantly higher runtime than AnyCam. *excluded from the evaluation as concurrent work. Many measurements are taken over from [54].

	Dataset	Source	Domain	Dyn.	Seq.	Frames
Train	YouTube-VOS	YouTube	Diverse	3	3471	87K
	RealEstate10K	YouTube	Indoor	7	6929	90K
	WalkingTours	YouTube	Outdoor	3	278	53K
	OpenDV	YouTube	Driving	3	532	105K
	EpicKitchens	GoPro	Ego	3	167	100K
Test	Sintel	Blender	Synthetic	3	14	629
	TUM-RGBD	MoCap	Indoor	3	8	720
	Davis	YouTube	Diverse	3	90	6118
	Aria EA	Glasses	Ego	3	30	19200
	Waymo	Sensor Rig	Driving	3	64	3067

Table 2. Dataset Mix: We train our method on a diverse dataset mix obtained mostly from YouTube. Evaluation happens on unseen datasets. For many datasets, custom splits were used.

EpicKitchens [6]) based on YouTube or individual GoPro capture, as shown in Tab. 2. None of the datasets has ground truth 3D labels (note that for some, Colmap was later applied to obtain proxy labels, but we do not use them). Following existing works on pose estimation in dynamic environments [5, 54, 56], we perform evaluation on the Sintel [3] and the dynamic subset of TUM-RGBD [35]. Furthermore, we test AnyCam qualitatively on three other datasets: Davis [27] (diverse videos from YouTube), Waymo [37] (autonomous driving), and Aria Everyday Activities [20].

Implementation details. We rely on the recent UniDepth [28] and UniMatch [47] methods to obtain depth and flow maps. Our model is implemented in PyTorch and is initialized with a pretrained DinoV2 [25] ViT backbone. We train AnyCam in two stages with different sequence lengths, first 2 and then 8 frames. All frames are sampled at a resolution of 336×336 from the videos in the datasets. During inference, we crop the input video to squares of 336×336 and pass sequences of up to 100 frames to the model at once. The model is configured to use 32 focal length candidates ranging from $0.1H$ to $3.5H$ where H is the image height. Training converges after 250k iterations at a batch size of 16 (seq. len 2) and 4 (seq. len 8) per GPU, and takes around two days on two NVIDIA A100 40GB GPUs per stage. For a video of 50 frames, it takes around 1) 15 seconds to obtain flows and depths, 2) 5 seconds for AnyCam to predict a trajectory, and 3) 90 seconds for test-time refinement. For more implementation details and hyperparameters, please refer to the supplementary material.

4.2. Camera Pose Estimation

We first test AnyCam's ability to recover camera trajectory in challenging environments and compare it to state-of-the-art methods in that domain. We group methods by the amount of data they require both during training and test time. SLAM and SfM systems for dynamic environments, like DPVO [40] and LeapVO [5], are trained with ground truth motion or trajectory data and mostly require camera intrinsics during test time. Methods like CasualSAM [55] require neither special training data nor intrinsics but are slow because they rely on much more costly test-time optimization. In comparison, AnyCam works in a feed-forward way, and optional test-time refinement is very lightweight.

Our model achieves strong quantitative results throughout all benchmarks as shown in Tab. 1. The low relative pose errors for translation (RPE_{trans}) and rotation (RPE_{rot}), even without test-time refinement, confirm that our model learns meaningful motion priors during training and generalizes to other datasets. In particular, this is noteworthy compared to specialized SLAM systems like LeapVO, which result in significantly higher local pose errors.

However, when only relying on feed-forward predictions, our model can suffer from drift, denoted by a slightly higher absolute trajectory error (ATE). This is a result of chaining the predicted relative poses $P_{i,j}$, where small errors in earlier frames accumulate over time. Our test-time

Figure 3. Qualitative results on various datasets.Red: forward-pass prediction, Green: re ned trajectory, Yellow: GT (if available). AnyCam is able to predict high-quality pose estimates on challenging scenes in dynamic environments. The uncertainty maps show objects with a high likelihood of movement, such as persons or cars, that would produce inconsistencies in the induced optical ow. Pose re nement with bundle adjustment further aligns the trajectory towards reducing the error compared to the ground truth poses.

re nement effectively corrects for this by leveraging longer-range dependencies between frames. As a result, the drift is reduced, and AnyCam matches the ATE for some SLAM systems like LeapVO, which requires camera intrinsics.

As can be seen in Tab. 1, AnyCam also generalizes to different domains like autonomous driving or egocentric vision. In particular, it works well in settings where classical SfM or SLAM systems are dif cult to operate in. For example, in the rst sequence of Aria Everyday Activities, a person enters a house via an opening door. Not only does the lighting change drastically, but also the 3D environment transforms as the door opens. Still, the camera motion trajectory is recovered correctly. During the second Waymo sequence, a vehicle moves in stop-and-go traf c. As can be observed in the uncertainty maps, AnyCam correctly identi es the moving vehicles on the left side as moving, while the uncertainty for the bus on the right is low. This indicates that the model actually relies on motion cues and not only on semantics, which would be the case when ltering out (potentially) moving objects with segmentation models.

Method	$AFE_{(px)}$ #	$RFE_{(\%)}$ #
UniDepth	447.4	0.357
Dust3r	434.0	0.364
Ours	252.2	0.181

Table 3. Camera intrinsics estimation. We measure the mean absolute focal error (AFE) and mean relative focal error (RFE) across all sequences of the Sintel dataset.

4.3. Intrinsics Recovery

For most casual videos, particularly when they originate from the internet, camera intrinsics are not available. Nevertheless, they are crucial for high-quality pose estimation and reconstruction. Therefore, we also evaluate the accuracy of the recovered focal length and report the results in Tab. 3. UniDepth predicts depth maps and focal length from a single image. Our superior performance against UniDepth (which provides the depth maps for AnyCam) suggests that our video-based training and inference offer bene ts for fo-