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

Supplementary Material

A. Overview

In this supplementary, we explain more details on implementation in Appendix C and the test-time refinement process in Appendix D. We also consider limitations and future work in Appendix E and ethical implications in Appendix F.

B. Code & Project Page

We release our code base for training, evaluation, and visualization under github.com/Brummi/anycam. Additionally, we provide interactive 3D results and more details on our project page under fwmb.github.io/anycam.

C. Further Implementation Details

Focal Length Candidates. In our model, we configure $m = 32$ distinct focal length candidates. For every candidate f_i, we train individual prediction heads $\mathcal{H}_{f_i}$. Focal length is not linearly related to rotation and translation magnitudes. Empirically, we find that the following formula, which combines linear and exponential spacing, provides a good distribution of focal length candidates.

$$\delta_i = \frac{i}{m-1} \tag{11}$$

$$f_i^{\text{exp}} = \exp\left(\delta_i \log(f_{min}) + (1-\delta_i)\log(f_{max})\right) \tag{12}$$

$$f_i^{\text{lin}} = \delta_i f_{min} + (1-\delta_i) f_{max} \tag{13}$$

$$f_i = 0.75 \cdot f_i^{\text{exp}} + 0.25 \cdot f_i^{\text{lin}} \tag{14}$$

We define $f_{min} = 0.1H$ and $f_{max} = 3.5H$, where H represents the height of the input image in pixels, yielding the distribution which can be seen in Fig. 5. In this way, the model can make predictions independent of the pixel size.

Camera Pose Parametrization. The prediction heads do not directly output the $P \in \mathbb{R}^{4\times4}$. To ensure that the pose matrix is in SE3, we predict translation $t \in \mathbb{R}^3$ and rotation $R \in \text{SO3}$ separately. R is parametrized via the axis-angle representation, *i.e.* the model predicts three values for the different axis rotations. We find that the axis-angle representation is significantly more stable than the quaternion representation and it converges faster. When using quaternions, it usually happens that a small number of (random) prediction heads does not converge to meaningful results.

Training Stabilization. Our training datasets cover a diverse range of datasets, which all have varying scales. E.g. driving datasets depict scenes and movements much larger compared to video sequences captured from VR glasses. When naively training on all five datasets from the start, the model does not converge to a meaningful solution. We hypothesize that the different scales introduce noise that hinders the optimization process. To overcome this issue, we first undergo a warmup phase, during which datasets are introduced one-by-one. First, the model is trained for 10,000 steps on RealEstate10K, then for another 10,000 steps on RealEstate10K and EpicKitchens, and so on until all datasets have been introduced. Through this strategy, the model can already roughly estimate the camera pose and then only is adapted to a different scale.

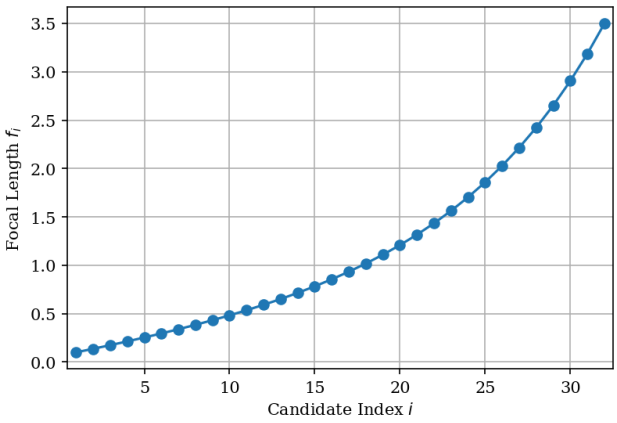

Figure 5. **Focal Length Candidates.** Linear-exponential distribution of focal length candidates relative to the image height.

Loss Configuration. The model is trained using the Adam optimizer at a learning rate of $\epsilon = 1e^{-4}$. After 100,000 steps, the learning rate is reduced to $\epsilon = 1e^{-5}$. We use $\lambda_{\sigma\mathbf{F}} = 1$, $\lambda_{\uparrow\downarrow} = 1$, and $\lambda_{Intr} = 1$. Since the flow loss values tend to have a very small magnitude, we set the temperature of the softmax operator in $\mathcal{L}_{Intr}$ to 100. Note that we also detach the flow losses in $\mathcal{L}_{Intr}$ and only pass gradients to the sequence head. This ensures that the different candidate heads remain independent of each other. Finally, we also apply L2 weight decay with a factor of 0.01 on the pose tokens to avoid overflow issues when training with mixed precision.

Model Architecture. We adapt the DinoV2 based DepthAnything model to predict both a pixel aligned map and tokens for pose and intrinsics. For both our backbone and UniDepth, we rely on Vit-S.

D. Test-Time Refinement Details

The main objective of our test-time refinement strategy is to reduce drift over longer time frames. To ensure geometric consistency over time, we apply bundle adjustment (BA) and optimize the camera trajectory in a sliding window fashion.

Setup. Since we primarily care about long-range dependencies, we apply BA with a stride of 3 frames. After the optimization is complete, the remaining poses are then interpolated and combined with the original predictions. During BA, we sample a uniform 16×16 grid of points per frame and then track them for 8 consecutive frames. Tracking is performed by chaining optical flow maps and we additionally accumulate uncertainty per tracked point. The uncertainty of a tracked point at a specific frame is the sum of uncertainties from all previous frames of the track. Intuitively, this means that a track only has low uncertainty as long as it does not encounter a pixel that has high uncertainty. For every point track, we optimize a single 3D point anchored in the first frame of the track and parametrize it by inverse depth. The point is initialized via the predicted depth that was also used as input by the AnyCam model. Thus, a track T_{fxy} starting in frame f at grid location x, y is defined by

$$T_{fxy} = ((\mathbf{p}_1, \dots \mathbf{p}_8), (\sigma_1, \dots, \sigma_8), d^{-1}) \tag{15}$$

where $\mathbf{p}_j$ are the pixel locations in the consecutive frames, σ_j are the corresponding uncertainty values, and d^{-1} is the inverse depth of the anchor point. In total, we optimize 1) the camera poses, 2) the inverse depths of the anchor points, and 3) a single focal length value.

Optimization. The main objective of our optimization process is to minimize the reprojection error for every track. In fact, we rely on a similar formulation as the flow loss $\ell_{f,uv}^{\mathbf{F}^{i\to j}}$ used for training AnyCam. To completely filter out very dynamic objects, we define a maximum uncertainty $\sigma_{\text{max}} = 0.05$ and ignore all points that exceed this threshold. All others are weighted accordingly in a linear fashion. Let $\mathbf{T} = \{T_{000}, \dots\}$ be the set of all tracks in the sequence:

$$\mathcal{L}_T^{Repr} = \sum_{i=2}^{8} \left\| \pi_f(\mathbf{P}^{1\to i} \pi_f^{-1}(\mathbf{p}_1, 1/d^{-1})) - \mathbf{p}_i \right\|_1 \cdot (\sigma_{\text{max}} - \sigma_i) \tag{16}$$

$$\mathcal{L}^{Repr} = \frac{1}{|\mathbf{T}|} \sum_{T \in \mathbf{T}} \mathcal{L}_T^{Repr} \tag{17}$$

Note that the uncertainties are not optimized during test-time refinement. Additionally, we apply smoothness term to encourage straight trajectories. Let n be the total number of frames in the sequence:

$$\mathcal{L}^{Smooth} = \frac{1}{n-2} \sum_{i=1}^{n-2} \left\| \left(\mathbf{P}_f^{i\to i+1} \right)^{-1} \mathbf{P}_f^{i+1\to i+2} - \mathbf{I}_4 \right\|_{1,1} \tag{18}$$

The final cost function is obtained by combining both terms, with $\lambda_{Smooth} = 0.1$:

$$\mathcal{L}_{BA} = \mathcal{L}^{Repr} + \lambda_{Smooth} \mathcal{L}^{Smooth} \tag{19}$$

We implement the entire BA process in PyTorch and use the Adam optimizer with a learning rate of $1e^{-4}$.

Sliding Window. Optimizing the entire sequence at once is both costly and can lead to instabilities. Therefore, we apply BA in a sliding window fashion. We define our window to be $w = 8$ frames wide and use an overlap of $o = 6$. That means we begin by optimizing the first 8 frames, and then shift the window by $w - o = 2$ to optimize frame 3 to 10. Note that we freeze the poses that have already been optimize and only adapt poses 9 and 10. For every sliding window, the optimization is performed for 400 steps. This is repeated until the end of the sequence is reached. In the end, we perform 5000 steps of global BA, where we consider all poses.

E. Limitations & Future Work

Reliance on pretrained model. AnyCam uses both pretrained depth and optical flow models during training and inference. While UniDepth and UniMatch show really strong performance, they can fail in rare cases. Depending on the severity of the failure, the accuracy of AnyCam can then get compromised. Typical failure cases include poor optical flow predictions when there are challenging lighting conditions, or inaccurate depth predictions when the input image does not have any scene context. Note that many errors in the input can still be dealt with due to our uncertainty formulation. Similarly, even though depth prediction are very consistent in scale as UniDepth is a metric depth model, the depths can have small flickering. This becomes visible when aggregating multiple depth maps over a longer time.

For future work, it would make sense to design the model to be reliant exclusively on images as input. Furthermore, we plan to add a system which adapts the scale and shift of the depth maps to be consistent among each other to allow for more accurate 4D reconstruction.

Drift over longer time. Our test-time refinement already greatly improves the drift problem. However, even then we only use tracks of length 8. To overcome drift on a global scale, our system would require a global scene representation or other techniques like keyframes. Many existing

SLAM and SfM systems [40, 43] provide inspiration for that.

Unnatural camera motion During training, AnyCams learns to translate images, flow, and depth of a sequence to a realistic camera motion. Our training data is very diverse, covering a wide range of realistic motions, and our experiments show that AnyCam can generalize very well. Still, due to its nature as a neural network, the model can fail when encountering very uncommon / unnatural camera motions.

To improve generalization even further, we plan to train the model on even more datasets. This can be achieved easily as our training pipeline is able digest any kind of unlabelled videos.

F. Ethical Considerations

Our training data is partially made up of videos obtained from public sources like YouTube. These videos can contain identifying information like faces, number plates, etc.. To remove such information, faces have been blurred in many datasets, *e.g*. WalkingTours, before usage in our project. Additionally, since our model only predicts camera poses and uncertainty, the output does not allow to infer the identity of persons in the input data.

While we also aim to build a data mix that covers different geographical regions and domains, it is nevertheless possible that the model learns a bias. For example, driving data in the OpenDV dataset is mostly from the US, China, and Europe. Our model might struggle in driving environments that are very different from this training data.

Finally, despite showing strong performance, we cannot provide accuracy guaranties for the predictions of AnyCam. Therefore, it should not (yet) be used in safety-critical applications.