RayTun3R: Online Camera Adaptation in 3D Foundation Models

Daniil Sinitsyn[1,2] **Nikita Araslanov**[1,2,3] **Daniel Cremers**[1,2]

[1]TU Munich [2]Munich Center for Machine Learning [3]University of Oxford

Abstract

Recent 3D foundation models, such as DUSt3R, MASt3R, VGGT, π^3, and Depth Anything 3, provide strong feed-forward depth and pose estimates on pinhole imagery, but degrade sharply under fisheye camera geometry. We show that this failure is partly caused by a pinhole camera bias in the positional encodings of pretrained 3D foundation models, and propose RayTun3R, a lightweight camera adaptation approach. It keeps the pretrained network fixed and adapts only lightweight components tied to token position and camera geometry. RayTun3R learns parameter-efficient residual corrections to absolute and rotary positional encodings, together with parameter-free tokenization and corrections to prediction-grid coordinates that remove residual pinhole assumptions. The resulting adapter contains only 10,752 trainable parameters and can be learned from a short temporal segment using geometric losses. Once adapted, RayTun3R transfers effectively to the remaining frames of the sequence without incurring additional runtime costs. Across diverse fisheye datasets with fields of view from 110° to 200°, our adapter reduces rotation error by 2–$12\times$ relative to the unadapted model, outperforms LoRA while using $\sim 14\times$ fewer trainable parameters, improves pose over adaptation-free baselines while avoiding their multi-view inference cost, and remains competitive on depth accuracy.

1 Introduction

Large 3D foundation models, such as DUSt3R [1], MASt3R [2], VGGT [3], π^3 [4], and Depth Anything 3 (DA3) [5], can recover depth, camera poses, and coarse 3D structure from a small set of images in near-instant feed-forward fashion. These models generalize well across everyday pinhole imagery; however, their robustness does not extend to changes in camera geometry. As Fig. 1 shows, fisheye input degrades geometry across the image, including near the optical center. A common workaround is a crop-based approach: splitting the fisheye image into multiple virtual pinhole views [6, 7], running inference with the foundation model on each view, and fusing the predictions. This allows the model to operate in a familiar input distribution, but also incurs additional computational costs and requires non-trivial prediction fusion without representing fisheye geometry directly.

We address this limitation with lightweight online adaptation of these pinhole-biased models. Our key observation is that pinhole and fisheye images assign different ray geometry to the same image-grid locations, making positional embeddings a natural target for online adaptation. By contrast, standard parameter-efficient fine-tuning (PEFT), such as LoRA [8], adapter layers [9], SSF [10], and calibration tokens [11], adapts generic feature transformations without isolating the camera-induced mismatch in the ray geometry.

Our adaptation method, *RayTun3R*, keeps the pretrained 3D model frozen and updates only lightweight components that encode how token locations in the image relate to camera rays. The

Code is available under the Apache 2.0 License at `https://github.com/DaniilSinitsyn/raytun3r`.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

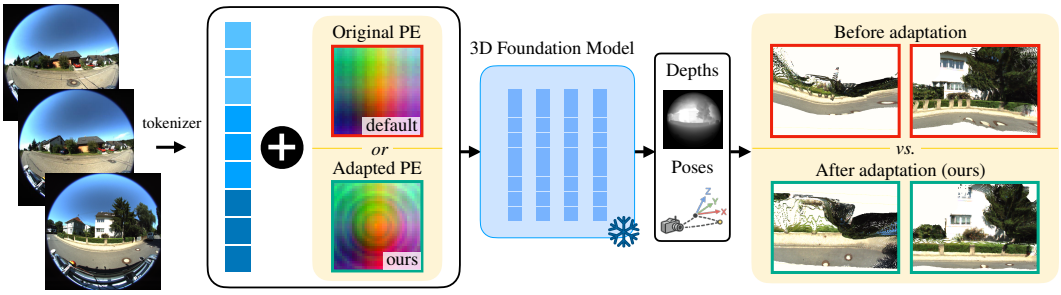


Figure 1: **RayTun3R overview.** The figure shows fisheye input, the original and adapted positional encodings, the frozen 3D foundation model, and the resulting geometry before and after adaptation. RayTun3R keeps the backbone frozen and adapts lightweight components tied to token location and camera geometry to recover more coherent depth, pose, and 3D structure.

learned parameters are small radial/angular lookup tables for absolute and rotary positional encodings; prediction-grid coordinates and patch-tokenization corrections are parameter-free.[1] RayTun3R comes at no inference-time overhead and adds only 10,752 trainable parameters, learned without ground truth on a short temporal segment. Across diverse fisheye datasets and three frozen backbones, it consistently improves pose and achieves competitive depth while using the full fisheye field of view.

Contributions. *(i)* We show that the local Jacobian of pretrained positional embeddings follows the geometry of the pinhole camera model, revealing a pinhole bias in the positional representation of 3D foundation models. *(ii)* Motivated by this analysis, we introduce a lightweight adapter for components tied to token locations in the image. It is runtime-efficient, adding only 10,752 trainable parameters on DA3-Small while keeping the pretrained model frozen. *(iii)* We empirically demonstrate that our adapter improves pose and depth accuracy of all foundation models tested (Depth Anything 3, VGGT, and π^3) on various fisheye datasets, substantially outperforming strong baselines, including LoRA and calibration tokens.

2 Related Work

We review feed-forward 3D models, camera-aware geometry, projection-based reuse of perspective backbones, and lightweight adaptation.

3D foundation models for image-based geometry. Classical reconstruction pipelines such as COLMAP [12], as well as learning-based methods for individual matching or pose-estimation components [13, 14], solve image-based geometry through modular stages. Recent 3D foundation models instead predict depth, pose, and coarse 3D structure in a feed-forward pass from only a few images [1, 2, 3, 4, 5]. Although these models differ in architecture and supervision, they are trained primarily on perspective imagery and therefore inherit pinhole-camera spatial priors.

Fisheye, omnidirectional, and arbitrary-camera geometry. A separate line of work handles non-pinhole geometry by exposing the camera model to the network, including fisheye depth, omnidirectional depth, and universal camera-aware 3D estimation [15, 16, 17, 18, 19, 20, 21, 22]. Other methods estimate or refine camera geometry explicitly, for example, through single-image calibration or radial-distortion averaging [23, 24]. Distortion-aware Transformers further modify tokenization, positional encoding, and dense decoding for wide-angle or fisheye images [25, 26, 27]. Camera-aware positional encodings are also closely related. PRoPE [28] represents camera intrinsics and extrinsics as relative positional encodings for multi-view transformers, showing that camera geometry is an effective conditioning signal for feed-forward 3D tasks. These works validate the importance of camera geometry, but require camera-aware training, new architectures, or a separate calibration/reconstruction procedure. Our goal is instead to adapt a frozen pinhole foundation model with a small online correction. Contemporaneous works also adapt foundation models to fisheye input. Fisheye3R [29] uses calibration tokens with masked attention, trained with supervision from synthetic fisheye distortions and optionally supervised fisheye data. PanoVGGT [30] trains a feed-forward reconstruction model for panoramic imagery with spherical-aware positional embeddings.

[1] We found directly distorting the positional embedding table to be suboptimal; see Appendix F.

FishRoPE [31] introduces a fisheye-aware angular formulation of RoPE for supervised fisheye detection and BEV segmentation; its geometry-specific positional modification is restricted to RoPE, while the full model also trains LoRA modules and task-specific heads. In contrast, RayTun3R adapts already trained 3D foundation models for depth and pose by correcting both absolute and rotary positional components when present, keeps the backbone frozen, and fits the residuals self-supervised from a short real fisheye sequence.

Projection-based adaptation to perspective models. Another strategy is to convert non-pinhole images into perspective-like views and reuse pinhole backbones. Examples include equirectangular or cubemap projection for depth estimation [18, 32], adaptive perspective slicing for VGGT-style models [33], and cubemap alignment through graph optimization [34]. These methods preserve the input distribution expected by the backbone, but duplicate computation across virtual views, require prediction fusion, and do not let the model represent the original fisheye geometry directly.

Parameter-efficient and test-time adaptation. Parameter-efficient fine-tuning methods such as LoRA [8], adapter layers [9], and SSF [10] update a small parameter subset while keeping the backbone fixed, but usually modify attention weights, MLP projections, prompts, or feature scales rather than the camera-dependent position-to-ray mapping. Calibration tokens [11] are more camera-aware and provide a strong fisheye baseline, but still express the correction through learned feature tokens. Test-time training methods optimize fast-adapting parameters online with self-supervised objectives [35, 36, 37, 38]. Unlike these methods, we do not train the backbone to support online optimization, and instead adapt only small positional corrections of an already frozen 3D model.

3 Preliminaries

Camera projection. We use a central camera model with projection κ and inverse projection κ^{-1}, mapping between camera rays and image coordinates in $\Omega \subset \mathbb{R}^2$. For a pinhole camera,

$$\kappa^{-1}(u, v) = \big((u - c_x)/f_x,\ (v - c_y)/f_y,\ 1\big)^\top, \tag{1}$$

where (f_x, f_y) are focal lengths and (c_x, c_y) is the principal point.

Fisheye cameras change this pixel-to-ray map. In common radially symmetric models, a ray with angle α to the optical axis lands at image radius $\rho(\alpha)$ from the principal point. The function $\rho(\alpha)$ can be a polynomial for the Kannala–Brandt (KB) model [39] or a rational function for the Enhanced Unified Camera Model (EUCM) [40]. Thus, unlike the pinhole case above, equal pixel displacements do not correspond to equal changes in viewing direction across the image.

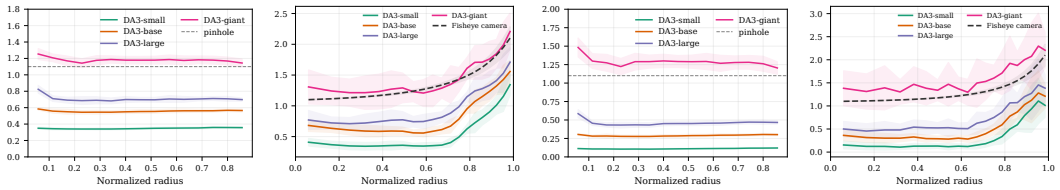
Patch tokens, positional embeddings and positional encoding. We study Transformer-based 3D foundation models, often initialized from DINOv2 [41]. The Vision Transformer (ViT) splits the image of size $H_{\text{img}} \times W_{\text{img}}$ into non-overlapping patches of size p, producing an $H \times W$ token grid with $H = H_{\text{img}}/p$ and $W = W_{\text{img}}/p$. Each patch is embedded into a C-dimensional token. During training (*e.g.* on pinhole images), ViTs learn a table of positional tokens, $P_A \in \mathbb{R}^{H \times W \times C}$, added to the image tokens. If the input resolution changes from $H \times W$, ViTs usually adjust P_A to the new grid size with interpolation (*e.g.* bicubic in DINOv2 [41]). However, the spatial structure of P_A would still reflect the camera geometry observed at training time.

Some ViTs also use rotary positional encoding (RoPE) [42] in self-attention. RoPE applies 2D rotations to pairs of query and key feature channels: For a token at position n, channel pair $(2j, 2j+1)$ is rotated by an angle proportional to $n\omega_j$, with fixed frequencies $\{\omega_j\}_j$. This makes attention depend on relative token offsets rather than on absolute coordinates alone. In image transformers, 2D axial RoPE applies the same construction separately along the vertical and horizontal grid coordinates. We denote this rotary positional mechanism by P_R.

On pinhole bias. Trained on pinhole images, 3D foundation models cannot reliably handle images with a different camera geometry, such as fisheye images. The geometric mismatch can be seen directly through the local backprojection Jacobian. Specifically, the Jacobian under the pinhole camera model,

$$J_{\kappa^{-1}} = \partial \kappa^{-1} / \partial (u, v), \tag{2}$$

is constant over the image. Therefore, a one-pixel displacement induces the same local change of $1/f_{\{x,y\}}$ in normalized image-plane coordinates everywhere. By contrast, for a radially symmetric



(a) pretrained σ_1 (b) adapted σ_1 (c) pretrained $\mathcal{G}_{\text{PE}}$ (d) adapted $\mathcal{G}_{\text{PE}}$

Figure 2: **Local Jacobian of the positional encoding in Depth Anything 3.** Pretrained positional embeddings have nearly radius-independent local geometry, consistent with a pinhole-like spatial prior (a, c). After adaptation on KITTI-360, the PE Jacobian becomes radius-dependent, matching the fisheye camera geometry more closely (b, d).

fisheye camera, the inverse projection can be written as

$$\kappa^{-1}(u, v) = \big(g(u, v)(u - c_x)/f_x,\ g(u, v)(v - c_y)/f_y,\ 1\big)^\top, \tag{3}$$

where g is a non-linear function of image radius (*e.g.* the pinhole model is a special case with $g \equiv 1$). Differentiating Eq. (3) yields

$$J_{\kappa^{-1}}(u, v) = g(u, v) \begin{pmatrix} 1/f_x & 0 \\ 0 & 1/f_y \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} (u - c_x)/f_x \\ (v - c_y)/f_y \\ 0 \end{pmatrix} \big(\partial_u g,\ \partial_v g\big), \tag{4}$$

which now depends on the image coordinates (u, v). Thus, the same image-grid displacement has a different geometric meaning at different radii. However, a positional encoding learned on pinhole images has only been exposed to the case of constant Jacobian, which leads to a pinhole bias.

Using Depth Anything 3 [5], we verify this bias by measuring the local Jacobian of the absolute positional embedding, $J_{\text{PE}} = \partial P_A / \partial(u, v) \in \mathbb{R}^{C \times 2}$, and summarizing it by its largest singular value σ_1 and local area element $\mathcal{G}_{\text{PE}} = \sqrt{\det(J_{\text{PE}}^\top J_{\text{PE}})}$ in Fig. 2. These quantities capture the strongest local variation and local scale of the embedding. Across all frozen model sizes, both are nearly flat as a function of normalized radius ρ (*cf*. Figs. 2a and 2c), matching the position-independent structure of a pinhole camera. After fitting our adapter on KITTI-360, the same curves bend toward the analytical fisheye reference (*cf*. Figs. 2b and 2d). This motivates RayTun3R: fine-tuning the positional representations that carry the camera bias rather than generic feature processing.

4 Method: RayTun3R

Our goal is to adapt a pretrained pinhole 3D foundation model to fisheye input while keeping the pretrained model frozen. Fisheye optics change the mapping between token locations in the image and viewing rays, so the spatial structure learned during pinhole pretraining no longer matches the input camera. Motivated by the PE Jacobian analysis, we therefore modify the components that encode how token locations in the image relate to geometry: absolute positional embeddings, rotary positional encoding when present, prediction-grid coordinates, and patch tokenization. The only learned parameters are small PE/RoPE lookup table corrections. Fig. 1 visualizes the main learned correction: The image content and backbone remain fixed, while the positional encoding is adapted to the fisheye camera geometry.

4.1 Problem setup

Let f_θ be a pretrained 3D foundation model that predicts depth and pose from one or more images. We consider pinhole-trained backbones, such as Depth Anything 3, π^3, and VGGT. Given a fisheye image stream $\{I_t\}$ and a known or estimated camera model κ, we learn a small set of adapter parameters ϕ while keeping all backbone weights θ fixed. The adapted model $f_{\theta,\phi}$ should produce accurate depth and pose on fisheye input without retraining the original backbone.

The only trainable parts are attached to positional mechanisms. In particular, we adapt the absolute positional embedding P_A and, when present, the rotary positional mechanism P_R defined in Sec. 3. All attention blocks, MLPs, prediction heads, and DPT weights remain frozen.

4.2 Adapting positional embeddings

The key difference between pinhole and fisheye geometry is radial, but real cameras and finite-resolution token grids may also have smaller orientation-dependent residuals. We therefore parameterize the correction in polar coordinates around the calibrated principal point. For each patch at coordinate u, v, let $\rho_{u,v} \in [0, 1]$ be its normalized radius and $\theta_{u,v} \in [0, 2\pi)$ its angle, both computed w.r.t. (c_x, c_y). We discretize radius into N_r bins and angle into N_θ bins, and learn a radial update $t_r(\rho)$ together with an angular update $\delta_\theta(\theta)$. Both updates are implemented as lookup tables and evaluated at continuous $\rho_{u,v}$ and $\theta_{u,v}$ by linear interpolation over the radial and angular bins. The corrected positional embedding is

$$P'(u, v) = P_A(u, v) + t_r(\rho_{u,v}) + \rho_{u,v}\, \delta_\theta(\theta_{u,v}). \tag{5}$$

The factor $\rho_{u,v}$ suppresses the angular term at the image center, where the angle is ill-defined, and lets it grow toward the periphery. We compute it as a patch radius, normalized to $[0, 1]$ by the grid boundary. All adapter parameters are initialized to zero, so adapter training starts exactly from the pretrained positional table.

Rotary positional encoding. For backbones that use RoPE, we learn a radial correction to the rotary angles. If $\omega(u, v)$ is the original rotary angle at patch (u, v), we add a correction from a learnable radial lookup table (shared across RoPE frequencies) with one parameter per bin:

$$\omega'(u, v) = \omega(u, v) + \Delta_r(\rho_{u,v}). \tag{6}$$

Prediction-grid coordinates. Some depth heads use a Dense Prediction Transformer (DPT)-style [43] head with a 2D sinusoidal prediction grid. We replace the regular prediction-grid coordinates with camera-aware coordinates obtained by undistorting each fisheye grid location through the calibrated fisheye-to-pinhole map. This does not change any model weights.

Patch tokenization. Patches outside the valid fisheye lens circle are replaced by the mean valid token (*border token*), avoiding artifacts from the invalid black region (*cf*. Fig. 1). We also locally undistort each patch before tokenization. Since fisheye distortion is position-dependent, especially near the boundary, each patch is resampled using the local linearization of the fisheye-to-pinhole map at its center following Qin and Li [44]. This parameter-free step makes each tokenized patch closer to a pinhole crop from the same viewing direction.

These learned and parameter-free changes add negligible inference overhead but improve stability and accuracy in practice.

4.3 Adapter training

We fit the adapter online on a short temporal segment of fisheye frames. Training uses windows of three frames $\{I_i\}_{i=1}^3$ sampled from this segment. For each window, the adapted model $f_{\theta,\phi}$ predicts a depth map D_i and camera pose T_i for each image. A predicted depth value defines

$$X_i(u, v) = D_i(u, v)\, \kappa^{-1}(u, v), \tag{7}$$

a 3D point at pixel (u, v), expressed in the coordinate frame of image I_i.

For each ordered image pair (I_i, I_j), let $m_{ij}(u, v)$ be the matched pixel in image I_j for pixel (u, v) in image I_i, and let $w_{ij}(u, v)$ be its confidence. To estimate the matching pixels, we use the correspondences produced by an off-the-shelf network, UFM [45], and their confidence scores. We penalize the reprojection error after transforming the predicted 3D point from I_i to I_j:

$$L_{\text{reproj}}(i, j) = \frac{1}{|\Omega|} \sum_{(u,v)\in\Omega} w_{ij}(u, v)\, \left\| \kappa\big(T_j T_i^{-1} X_i(u, v)\big) - m_{ij}(u, v) \right\|_1, \tag{8}$$

where Ω denotes the set of pixels inside the valid fisheye disc.

The reprojection loss couples depth and pose, but pose optimization can be unstable early in adaptation. Therefore, we add a fixed geometric pose target. From the same UFM matches, we estimate a relative pose $\tilde{T}_{ij} = (\tilde{R}_{ij}, \tilde{t}_{ij})$ once using MAGSAC++ [46]. We then penalize the angular difference between this estimate and the model-predicted relative pose $T_j T_i^{-1} = (\hat{R}_{ij}, \hat{t}_{ij})$:

$$L_{\text{pose}}(i, j) = \arccos\left(\frac{\operatorname{tr}\left(\tilde{R}_{ij}^\top \hat{R}_{ij}\right) - 1}{2} \right) + \arccos\left(\frac{\langle \tilde{t}_{ij}, \hat{t}_{ij} \rangle}{\|\tilde{t}_{ij}\|\, \|\hat{t}_{ij}\|} \right). \tag{9}$$

Since $\tilde{T}_{ij}$ is computed once from correspondences produced by an external matcher before adaptation, the adapter cannot influence its own pose pseudo-label. This fixed pose target stabilizes relative motion early in optimization, while L_{reproj} enforces depth consistency through the predicted geometry.

We further regularize depth with the standard edge-aware smoothness prior from self-supervised monocular depth estimation [47]:

$$L_{\text{smooth}}(i) = \frac{1}{|\Omega|} \sum_{(u,v)\in\Omega} e^{-|\partial_x I_i(u,v)|} \, |\partial_x D_i^*(u,v)| + e^{-|\partial_y I_i(u,v)|} \, |\partial_y D_i^*(u,v)|, \tag{10}$$

where $D_i^* = D_i / \text{mean}(D_i)$.

Finally, we keep the positional correction small and smooth. We regularize the adapted positional table $P' \in \mathbb{R}^{H\times W\times C}$ toward the initial table P_A:

$$\mathcal{L}_{\text{L2}} = \frac{1}{HW} \sum_{x,y} \|P'(x,y) - P_A(x,y)\|_2^2, \tag{11}$$

where x, y index token-grid locations. We further constrain neighboring positional corrections to vary smoothly using a total-variation penalty:

$$\mathcal{L}_{\text{TV}} = \frac{1}{HW} \sum_{x,y} \Big(\|P'(x+1,y) - P'(x,y)\|_2^2 + \|P'(x,y+1) - P'(x,y)\|_2^2 \Big), \tag{12}$$

where differences are taken only for valid neighbors. The total loss is

$$L = L_{\text{reproj}} + w_{\text{smooth}} L_{\text{smooth}} + w_{\text{L2}} \mathcal{L}_{\text{L2}} + w_{\text{TV}} \mathcal{L}_{\text{TV}} + w_{\text{pose}} \mathcal{L}_{\text{pose}}. \tag{13}$$

We use $w_{\text{pose}} = 1$, $w_{\text{smooth}} = 10$, $w_{\text{L2}} = 2$, and $w_{\text{TV}} = 20$ in all experiments. Appendix I ablates the contribution of the pose and PE regularization terms.

We do not use a photometric warping loss. Direct photometric alignment can be weak in low-texture regions and sensitive to initialization. The initial depth and pose can be far from optimum due to the model's pinhole bias. Following recent geometry and pose predictors such as AnyCam [48], we instead rely on UFM correspondences and optimize geometric consistency.

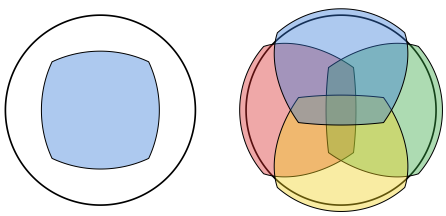
In summary, the only learned parameters are small radial/angular lookup tables for positional encodings; all backbone weights and prediction heads remain frozen, while the DPT-grid and patch-tokenization changes are parameter-free.

Implementation details Unless stated otherwise, images are resized to a maximum patch-aligned resolution of 504×504. We fit each adapter using batches of three-frame windows with Adam, learning rate 1×10^{-3}, and gradient clipping at norm 1.0. The PE adapter uses 20 radial bins and 8 angular bins; RoPE backbones additionally use 20 radial RoPE bins. All backbone weights remain frozen, and all residual adapter parameters are initialized to zero. After adaptation, inference requires a single forward pass per fisheye frame.

5 Experiments

Datasets. Fisheye datasets with reliable pose or depth ground truth remain less common than standard pinhole reconstruction benchmarks. We evaluate on five datasets covering outdoor driving (KITTI-360 [49], $185°$ FOV), handheld dual-fisheye video (TUM-VI [50], $195°$), posed DSLR fisheye scans (ScanNet++ [51], $115°$), multi-camera captures (ETH3D [52], $110°$), and dual-fisheye indoor/outdoor scenes (FIORD [53], $200°$). All datasets provide reference poses; ETH3D and ScanNet++ also provide dense depth.

Baselines. We compare against representative baselines from the two most relevant strategies for reusing pinhole-trained models on fisheye input: input-space projection and lightweight model adaptation. Center-PH undistorts each fisheye image into one forward-looking $110°$ virtual pinhole crop (Fig. 3a); we use



(a) Center-PH (b) Multi-PH

Figure 3: **Adaptation-free baselines.**

Table 1: **Pose and depth error across five datasets using DA3-Small.** Each entry reports a triple $R^\circ, t^\circ, d_{\text{reproj}}$, respectively. **Bold** and underline denote the best and second-best results, respectively. RayTun3R consistently offers low pose and depth error for the complete fisheye image (in contrast to Center-PH) and without extra inference-time costs (in contrast to Multi-PH).

Dataset	Vanilla			Center-PH			Multi-PH			LoRA			CalTok			**RayTun3R (ours)**		
ETH3D	8.59	15.16	15.98	3.46	13.70	10.92	3.31	13.68	13.48	2.18	10.74	9.02	2.48	13.21	11.94	**0.70**	**4.48**	**5.82**
KITTI-360	1.69	12.81	11.64	**0.79**	4.17	**3.10**	1.71	9.75	4.72	1.37	8.49	5.56	1.66	10.05	5.83	0.84	**2.92**	3.88
TUM-VI	10.41	23.23	57.01	3.33	29.24	**3.22**	2.99	25.60	4.92	3.38	13.63	3.83	3.84	16.17	9.61	**2.41**	**13.23**	3.81
ScanNet++	10.21	30.26	23.82	3.27	22.77	2.21	1.66	10.43	**1.63**	3.68	17.66	4.98	4.51	23.20	7.02	**1.11**	**5.78**	4.16
FIORD	18.20	29.50	75.30	6.92	23.40	**7.20**	6.30	18.90	15.60	7.75	12.20	12.10	20.40	22.20	25.20	**4.10**	**5.40**	9.00

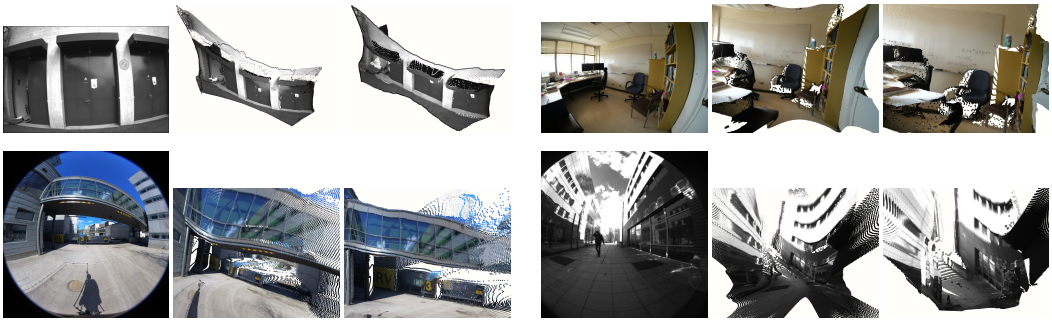


Figure 4: **Qualitative reconstructions.** Across datasets, the frozen model produces distorted or inconsistent geometry, while RayTun3R recovers more coherent structure after online adaptation.

110° to avoid severe center compression and peripheral stretching from wider rectilinear projections. Multi-PH fuses predictions from Center-PH and four additional virtual pinhole views back into the fisheye frame (Fig. 3b), improving coverage at roughly proportional extra cost.

For adaptation-based baselines, we include CalTok [11], the closest baseline to our setting because it adapts mostly frozen depth networks to fisheye cameras. CalTok adds $t \in \{4, 8\}$ learned auxiliary tokens to attention layers. We also include LoRA [8] as a strong generic PEFT baseline, updating QKV adapters with $r \in \{4, 8, 16\}$ and $\alpha \in \{8, 16, 32\}$. For the main comparison, we report the strongest ETH3D setting for each baseline: CalTok with $t = 4$ and LoRA with $r = 8, \alpha = 16$ (see Tab. 3, right). Unless specified otherwise, experiments use DA3-Small [5]; we also evaluate larger DA3 variants, π^3 [4], and VGGT [3].

Error metrics. For camera pose accuracy, we report angular rotation error R° and translation-direction error t°, both in degrees. Since ground-truth dense depth is not available for all fisheye datasets, we use a reprojection-based depth metric *with ground-truth pose* for the main comparison, denoted by d_{reproj}. When reliable ground-truth depth is available, we report scale-aligned depth error, AbsRel, following Eigen et al. [54], and the threshold accuracy $\delta_{1.25}$, which is the fraction of pixels whose predicted depth is within a factor 1.25 of the ground truth. Appendix A provides details on metric computation.

Evaluation protocol. For each sequence, we build a short adaptation set from the first 30 overlapping three-frame windows (stride of one frame, *i.e.*, 32 frames), filtering out nearly static windows with average optical-flow displacement below 2 pixels. We fit the adapter on this set and evaluate on the remaining frames of the sequence. Frames used for adaptation are excluded from evaluation.

We evaluate relative poses on consecutive image pairs. Note that short baselines can amplify translation-direction errors (t°) even when reprojection error (d_{reproj}) remains moderate. All methods use the same pair sampling. Unless stated otherwise, all methods use the calibration provided with each dataset; Appendix B evaluates sensitivity to predicted calibration.

Table 2: **Model family comparison under ground-truth calibration.** Each entry reports $R^\circ, t^\circ, d_{\text{reproj}}$, respectively; lower is better. **Bold** and underline denote the best and second-best results, respectively. Each dataset is represented by the first sequence of its evaluation split. Appendix E reports additional sequences.

Backbone	Method	ETH3D terrains			KITTI 0009			TUM-VI room6			ScanNet++ 3f15			FIORD Kitchen		
π^3 [4]	LoRA	1.40	4.9	1.4	2.22	3.4	2.7	9.01	22.4	6.8	3.11	16.1	2.1	4.59	4.62	**5.5**
	CalTok	1.49	7.5	1.3	0.93	11.5	4.6	12.3	24.7	10.0	38.1	17.2	9.1	27.24	11.96	7.7
	Center-PH	0.97	5.9	1.1	0.88	4.8	**1.5**	1.67	13.5	**1.2**	2.28	25.7	5.2	**1.11**	3.24	5.6
	Vanilla	4.65	4.4	3.9	3.24	14.2	5.4	9.12	27.4	31.6	6.17	19.7	38.6	15.36	11.63	28.7
	RayTun3R (ours)	**0.60**	**0.7**	**0.3**	**0.76**	**1.9**	1.9	**1.14**	**6.4**	2.0	**0.78**	**1.9**	**0.9**	3.49	**2.24**	5.6
VGGT [3]	LoRA	1.40	6.4	1.4	0.60	25.3	3.2	13.0	24.7	7.9	8.95	17.2	4.4	6.48	3.81	**6.1**
	CalTok	2.45	15.6	1.9	**0.52**	28.5	4.0	23.4	32.2	9.4	16.7	26.3	6.9	7.83	36.39	7.0
	Center-PH	0.97	7.4	2.9	0.82	4.3	3.0	1.13	10.5	**2.4**	2.45	27.3	6.1	6.43	10.11	7.58
	Vanilla	5.98	6.9	12.4	2.72	20.9	14.5	8.54	22.1	88.6	7.21	16.6	39.4	25.31	27.02	33.9
	RayTun3R (ours)	**0.53**	**1.0**	**1.2**	0.70	**3.9**	**2.2**	**1.12**	**7.7**	3.6	**0.93**	**6.0**	**3.2**	**6.19**	**3.72**	**6.1**

Table 3: **Left: Dense depth on ETH3D and ScanNet++.** We report the mean of per-scene means for *AbsRel* (lower is better) and $\delta_{1.25}$ (higher is better). **Right: Lightweight adaptation choices on ETH3D.** These results select the LoRA and CalTok settings used in the main comparison. All experiments use DA3-Small as the baseline.

	ETH3D		ScanNet++	
Method	AbsRel ↓	$\delta_{1.25}$ ↑	AbsRel ↓	$\delta_{1.25}$ ↑
Center-PH	0.111	0.867	**0.066**	**0.961**
LoRA	0.166	0.814	0.175	0.760
CalTok t=4	0.175	0.793	0.168	0.769
Vanilla	0.178	0.751	0.282	0.601
RayTun3R (ours)	**0.107**	**0.884**	0.108	0.886

Method	R° ↓	t° ↓	d_{reproj} ↓
Center-PH	3.46	13.70	10.92
CalTok t=4 (default)	2.48	13.21	11.94
CalTok t=8	2.63	12.83	15.77
LoRA r=4, α=8	2.47	12.18	10.85
LoRA r=8, α=16 (default)	2.18	10.74	9.02
LoRA r=16, α=32	3.01	11.99	10.68
RayTun3R (ours)	0.70	4.48	5.82

5.1 Main results

In Tab. 1, RayTun3R gives the lowest translation-direction error on all datasets and the lowest rotation error on ETH3D, TUM-VI, ScanNet++, and FIORD. Compared with CalTok, the closest fisheye-specific adaptation baseline, and LoRA, the strongest generic PEFT baseline, RayTun3R achieves substantially lower pose error and lower or comparable reprojection. This supports our view that camera adaptation is better handled by directly correcting camera-dependent positional components than by adapting generic feature transformations. Such transformations do not explicitly separate camera geometry from scene appearance. Compared with projection baselines, RayTun3R keeps the full fisheye field of view and single-view inference cost; Center-PH remains strong for depth on some moderate-FOV datasets, while Multi-PH improves coverage at a higher cost. Fig. 4 visually shows that, on fisheye inputs, adaptation produces more coherent geometry than the vanilla DA3 model.

Tab. 2 extends the comparison to π^3 and VGGT. Across both backbones, RayTun3R gives the lowest translation error on every sequence and consistently ranks among the best methods for rotation. It also improves substantially over CalTok and LoRA, indicating that the effect is not specific to DA3-Small. Appendix H reports additional results across DA3 model variants, π^3, and VGGT.

Together, Tabs. 1 and 2 support the core idea behind RayTun3R: Correcting camera-dependent positional components improves fisheye geometry across datasets and frozen 3D backbones.

Dense depth evaluation. For ETH3D and ScanNet++, which provide ground-truth depth, we report AbsRel and $\delta_{1.25}$ in Tab. 3 (left). RayTun3R improves over the vanilla model and lightweight adaptation baselines on both datasets. Center-PH remains strong on depth because it produces perspective images close to the backbone's pretraining distribution. However, Center-PH discards the peripheral fisheye content, yielding less accurate pose estimates: On ScanNet++, RayTun3R reduces Center-PH rotation from 3.27° to 1.11° and translation from 22.77° to 5.78°.

Adaptation regimes. Tab. 3 (right) compares lightweight adaptation choices on ETH3D. LoRA is the strongest generic PEFT baseline, with $r = 8, \alpha = 16$ giving the best overall trade-off, while CalTok is sensitive to the number of learned tokens and is best represented by $t = 4$. Both baselines improve over vanilla inference in some metrics, but their gains are less consistent than RayTun3R.

Table 4: **Component ablation and inference cost.** (a) Component ablation on KITTI-360 (drive 0000, cam02, 185° fisheye), adapted on the first 30 three-frame windows (32 frames) and evaluated on the first 500 frames, excluding the adaptation frames. (b) Inference cost on DA3-Small, measured per fisheye frame at 504×504 on one NVIDIA RTX A4000 and averaged over 1000 frames. Trainable parameter counts account for the DA3-Small backbone width, with token size $C = 384$.

(a) Component ablation on KITTI-360.

Configuration	$R^\circ \downarrow$	$t^\circ \downarrow$	$d_{\text{reproj}} \downarrow$
Patch undistortion (no learnable PE)	1.397	6.66	8.96
Naive remap of PE	0.810	12.93	11.53
Radial PE only	1.154	5.48	3.70
Radial + angular PE	1.038	4.21	3.39
RayTun3R w/o border token	1.061	4.45	3.17
RayTun3R w/o prediction-grid correction	1.094	4.78	3.64
RayTun3R w/o RoPE correction	0.942	5.56	3.27
RayTun3R (ours)	1.183	4.81	3.03

(b) Inference cost on DA3-Small

Method	ms/frame	Overhead	Params
Vanilla DA3	$\sim$100	baseline	0
Center-PH (single 110°)	$\sim$105	$+5\%$	0
Multi-PH (5 views)	$\sim$400	$+300\%$	0
LoRA (r=8)	$\sim$110	$+10\%$	147.5K
CalTok (t=4)	$\sim$105	$+5\%$	18.4K
RayTun3R (ours)	$\sim$100	$\approx 0\%$	10.8K

Discussion. These results support the PE-Jacobian analysis in Sec. 3: Fisheye transfer changes the position-to-ray mapping, so camera adaptation is better targeted through positional representations than generic feature transformations. Although the adapter is fitted on a specific sequence and can absorb some sequence-specific signal, its trainable capacity is restricted to PE/RoPE residuals, biasing adaptation toward camera geometry.

5.2 Design choices and runtime

Tab. 4a isolates the contribution of RayTun3R's components. The largest gain comes from the learned PE residual. *Radial PE only* already reduces reprojection error substantially, and *Radial + angular PE* further improves the pose–depth balance, indicating that the remaining mismatch is not purely radial. The *Naive remap of PE* baseline tests whether one can simply reuse the pretrained PE table at undistorted coordinates. It improves rotation, but gives poor translation and depth, showing that direct geometric re-indexing of the pretrained table does not produce consistent fisheye geometry. The parameter-free components address other places where pinhole assumptions enter the model. *Patch undistortion (no learnable PE)* helps, confirming that tokenization matters, but it remains far below the learned positional adapter. Removing the prediction-grid correction (pinhole DPT grid coordinates), the border token (out-of-lens patches kept as is), or the RoPE correction (Eq. (6)) has a smaller effect; the *Full model* gives the lowest reprojection error, although some ablations achieve slightly lower pose error.

The extended ablation in Appendix J shows the same pattern on ETH3D and FIORD: Learned PE/RoPE residuals drive most of the improvement, while parameter-free corrections provide smaller, dataset-dependent changes.

Tab. 4b reports inference cost after adaptation. RayTun3R adds only small positional lookup tables: 20 radial and 8 angular bins for PE, plus 20 radial RoPE bins when applicable. These tables are evaluated inside the original forward pass, giving negligible overhead over DA3-Small at inference time. By contrast, MULTI-PH requires five virtual pinhole views and costs about $4\times$ more per frame; LoRA adds around 10% latency and uses $\sim14\times$ more trainable parameters, while CalTok has 18.4K trainable parameters with modest overhead. Appendix G reports adaptation and evaluation compute.

6 Conclusion and Limitations

RayTun3R adapts 3D foundation models to fisheye images while preserving the priors acquired from pinhole pretraining. The method corrects the components linking token locations in the image to geometry — primarily, absolute positional embeddings, with additional RoPE, prediction-grid, and patch-tokenization corrections. Across fisheye datasets and frozen backbones, RayTun3R improves pose accuracy while remaining parameter- and runtime-efficient.

Limitations. *(i)* RayTun3R computes a camera-specific correction; a different fisheye camera or distortion profile requires new adaptation. Within the same physical camera, the adapter can be transferred across sequences (Appendix C). However, the gap to a target-trained adapter can be

noticeable (*e.g.*, on TUM-VI), possibly because, with a small adaptation budget, some scene-specific appearance information leaks into the adapter. Transfer under substantial changes in scene appearance has not been systematically studied. *(ii)* RayTun3R assumes a principal point and mostly radial distortion; it does not explicitly cover strong tangential or non-radial optics. *(iii)* RayTun3R requires camera parameters, although Appendix B shows that predicted calibration from off-the-shelf networks such as AnyCalib [23] remains accurate. *(iv)* Our work focuses on fisheye images. In principle, RayTun3R can be applied to any camera model, since it adapts the positional encodings to the geometry of the target camera. We leave the evaluation on other camera models, such as panoramic or equirectangular inputs, for future work. *(v)* The training set for RayTun3R needs several views with sufficient inter-frame displacement. Unlike the frozen model, adaptation therefore cannot operate on a single image. With small or degenerate motion, the self-supervised constraints become weak because large depth or translation-direction errors can induce only small reprojection errors. *(vi)* The backbone and calibration studies (Tabs. 2 and 5) use the first sequence of each evaluation split, complemented by further sequences in Appendix E. Only the DA3-Small comparison in Tab. 1 covers all sequences.

Broader impact and asset-license details are provided in Appendices K and L.

Acknowledgments and Disclosure of Funding

This work was supported by the European Research Council (ERC) Advanced Grant "SIMULACRON" (grant agreement No. 884679), the DFG project CR 250/26-1 "4D-YouTube", the GNI Project "AI4Twinning", and the Munich Center for Machine Learning (MCML). N. A. was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project No. 563662485. The authors declare no competing interests.

References

[1] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D vision made easy. In *CVPR*, pages 20697–20709, 2024.

[2] Bardienus Pieter Duisterhof, Lojze Zust, Philippe Weinzaepfel, Vincent Leroy, Yohann Cabon, and Jerome Revaud. MASt3R-SfM: A fully-integrated solution for unconstrained structure-from-motion. In *3DV*, 2025.

[3] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *CVPR*, pages 5294–5306, 2025.

[4] Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Permutation-equivariant visual geometry learning. In *ICLR*, 2026.

[5] Haotong Lin, Sili Chen, Jun Hao Liew, Donny Y. Chen, Zhenyu Li, Yang Zhao, Sida Peng, Hengkai Guo, Xiaowei Zhou, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth Anything 3: Recovering the visual space from any views. In *ICLR*, 2026.

[6] Yahui Wang, Shaojun Cai, Shi-Jie Li, Yun Liu, Yangyan Guo, Tao Li, and Ming-Ming Cheng. CubemapSLAM: A piecewise-pinhole monocular fisheye SLAM system. In *ACCV*, pages 34–49, 2018.

[7] Jialei Xu, Wei Yin, Dong Gong, Junjun Jiang, and Xianming Liu. SDGE: Stereo guided depth estimation for 360° camera sets. In *IROS*, pages 11179–11186, 2024.

[8] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

[9] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *ICML*, pages 2790–2799, 2019.

[10] Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Scaling & shifting your features: A new baseline for efficient model tuning. In *NeurIPS*, 2022.

[11] Suchisrit Gangopadhyay, Jung-Hee Kim, Xien Chen, Patrick Rim, Hyoungseob Park, and Alex Wong. Extending foundational monocular depth estimators to fisheye cameras with calibration tokens. In *ICCV*, pages 5198–5209, 2025.

[12] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, pages 4104–4113, 2016.

[13] Paul-Edouard Sarlin, Ajaykumar Unagar, Måns Larsson, Hugo Germain, Carl Toft, Viktor Larsson, Marc Pollefeys, Vincent Lepetit, Lars Hammarstrand, Fredrik Kahl, and Torsten Sattler. Back to the feature: Learning robust camera localization from pixels to pose. In *CVPR*, pages 3247–3257, 2021.

[14] Sergei Solonets, Daniil Sinitsyn, Lukas von Stumberg, Nikita Araslanov, and Daniel Cremers. An analytical solution to Gauss-Newton loss for direct image alignment. In *ICLR*, 2024.

[15] Robert McCraith, Lukas Neumann, Andrew Zisserman, and Andrea Vedaldi. Monocular depth estimation with self-supervised instance adaptation. *arXiv:2004.05821 [cs.CV]*, 2020.

[16] Sheng Xie, Daochuan Wang, and Yun-Hui Liu. OmniVidar: Omnidirectional depth estimation from multi-fisheye images. In *CVPR*, pages 21529–21538, 2023.

[17] Jiaxi Deng, Yushen Wang, Haitao Meng, Zuoxun Hou, Yi Chang, and Gang Chen. OmniStereo: Real-time omnidirectional depth estimation with multiview fisheye cameras. In *CVPR*, pages 1003–1012, 2025.

[18] Yuliang Guo, Sparsh Garg, S. Mahdi H. Miangoleh, Xinyu Huang, and Liu Ren. Depth Any Camera: Zero-shot metric depth estimation from any camera. In *CVPR*, pages 26996–27006, 2025.

[19] Guoyang Zhao, Yuxuan Liu, Weiqing Qi, Fulong Ma, Ming Liu, and Jun Ma. FisheyeDepth: A real scale self-supervised depth estimation model for fisheye camera. In *ICRA*, pages 3780–3787, 2025.

[20] Luigi Piccinelli, Yung-Hsu Yang, Christos Sakaridis, Mattia Segu, Siyuan Li, Luc Van Gool, and Fisher Yu. UniDepth: Universal monocular metric depth estimation. In *CVPR*, pages 10106–10116, 2024.

[21] Luigi Piccinelli, Christos Sakaridis, Mattia Segu, Yung-Hsu Yang, Siyuan Li, Wim Abbeloos, and Luc Van Gool. UniK3D: Universal camera monocular 3D estimation. In *CVPR*, pages 1028–1039, 2025.

[22] Mu Hu, Wei Yin, Chi Zhang, Zhipeng Cai, Xiaoxiao Long, Hao Chen, Kaixuan Wang, Gang Yu, Chunhua Shen, and Shaojie Shen. Metric3D v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):10579–10596, 2024.

[23] Javier Tirado-Garín and Javier Civera. AnyCalib: On-manifold learning for model-agnostic single-view camera calibration. In *ICCV*, pages 8044–8055, 2025.

[24] Daniil Sinitsyn, Linus Härenstam-Nielsen, and Daniel Cremers. PRaDA: Projective radial distortion averaging. In *CVPR*, pages 21902–21912, 2025.

[25] Akshaya Athwale, Arman Afrasiyabi, Justin Lagüe, Ichrak Shili, Ola Ahmad, and Jean-François Lalonde. DarSwin: Distortion aware radial Swin transformer. In *ICCV*, pages 5929–5938, 2023.

[26] Akshaya Athwale, Ichrak Shili, Émile Bergeron, Ola Ahmad, and Jean-François Lalonde. DarSwin-UNet: Distortion aware architecture. In *WACV*, pages 8670–8680, 2025.

[27] Dianyi Yang, Jiadong Tang, Yu Gao, Yi Yang, and Mengyin Fu. Sector patch embedding: An embedding module conforming to the distortion pattern of fisheye image. *arXiv:2303.14645 [cs.CV]*, 2023.

[28] Ruilong Li, Brent Yi, Junchen Liu, Hang Gao, Yi Ma, and Angjoo Kanazawa. Cameras as relative positional encoding. In *NeurIPS*, 2025.

[29] Ruxiao Duan, Erin Hong, Dongxu Zhao, Eric Turner, Alex Wong, and Yunwen Zhou. Fisheye3R: Adapting unified 3D feed-forward foundation models to fisheye lenses. In *ECCV*, 2026.

[30] Yijing Guo, Mengjun Chao, Luo Wang, Tianyang Zhao, Haizhao Dai, Yingliang Zhang, Jingyi Yu, and Yujiao Shi. PanoVGGT: Feed-forward 3D reconstruction from panoramic imagery. In *CVPR*, 2026.

[31] Rahul Ahuja, Mudit Jain, Bala Murali Manoghar Sai Sudhakar, Venkatraman Narayanan, Pratik Likhar, Varun Ravi Kumar, and Senthil Yogamani. FishRoPE: Projective rotary position embeddings for omnidirectional visual perception. *arXiv:2604.10391 [cs.CV]*, 2026.

[32] Ning-Hsu Wang and Yu-Lun Liu. Depth Anywhere: Enhancing 360 monocular depth estimation via perspective distillation and unlabeled data augmentation. In *NeurIPS*, 2024.

[33] Jiayi Yuan, Haobo Jiang, De Wen Soh, and Na Zhao. VGGT-360: Geometry-consistent zero-shot panoramic depth estimation. *arXiv:2603.18943 [cs.CV]*, 2026.

[34] Dongki Jung, Jaehoon Choi, Yonghan Lee, and Dinesh Manocha. RPG360: Robust 360 depth estimation with perspective foundation models and graph optimization. In *NeurIPS*, 2025.

[35] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, pages 9229–9248, 2020.

[36] Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T. Freeman, and Hao Tan. Test-time training done right. In *ICLR*, 2026.

[37] Xingyu Chen, Yue Chen, Yuliang Xiu, Andreas Geiger, and Anpei Chen. TTT3R: 3D reconstruction as test-time training. In *ICLR*, 2026.

[38] Tao Xie, Peishan Yang, Yudong Jin, Yingfeng Cai, Wei Yin, Weiqiang Ren, Qian Zhang, Wei Hua, Sida Peng, Xiaoyang Guo, and Xiaowei Zhou. Scal3R: Scalable test-time training for large-scale 3D reconstruction. *arXiv:2604.08542 [cs.CV]*, 2026.

[39] Juho Kannala and Sami S. Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1335–1340, 2006.

[40] Bogdan Khomutenko, Gaëtan Garcia, and Philippe Martinet. An enhanced unified camera model. *IEEE Robotics and Automation Letters*, 1(1):137–144, 2016.

[41] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

[42] Byeongho Heo, Song Park, Dongyoon Han, and Sangdoo Yun. Rotary position embedding for vision transformer. In *ECCV*, pages 289–305, 2024.

[43] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, pages 12179–12188, 2021.

[44] Xuebin Qin and Shigang Li. Tracking feature points of fisheye full-view image by normalized image patch. *IEEJ Transactions on Electronics, Information and Systems*, 132(9):1516–1523, 2012.

[45] Yuchen Zhang, Nikhil Keetha, Chenwei Lyu, Bhuvan Jhamb, Yutian Chen, Yuheng Qiu, Jay Karhade, Shreyas Jha, Yaoyu Hu, Deva Ramanan, Sebastian Scherer, and Wenshan Wang. UFM: A simple path towards unified dense correspondence with flow. In *NeurIPS*, 2025.

[46] Daniel Barath, Jana Noskova, Maksym Ivashechkin, and Jiri Matas. MAGSAC++, a fast, reliable and accurate robust estimator. In *CVPR*, pages 1304–1312, 2020.

[47] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, pages 270–279, 2017.

[48] Felix Wimbauer, Weirong Chen, Dominik Muhle, Christian Rupprecht, and Daniel Cremers. AnyCam: Learning to recover camera poses and intrinsics from casual videos. In *CVPR*, pages 16717–16727, 2025.

[49] Yiyi Liao, Jun Xie, and Andreas Geiger. KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2D and 3D. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3292–3310, 2023.

[50] David Schubert, Thore Goll, Nikolaus Demmel, Vladyslav Usenko, Jörg Stückler, and Daniel Cremers. The TUM VI benchmark for evaluating visual-inertial odometry. In *IROS*, pages 1680–1687, 2018.

[51] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. ScanNet++: A high-fidelity dataset of 3D indoor scenes. In *ICCV*, pages 12–22, 2023.

[52] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *CVPR*, pages 3260–3269, 2017.

[53] Ulas Gunes, Matias Turkulainen, Xuqian Ren, Arno Solin, Juho Kannala, and Esa Rahtu. FIORD: A fisheye indoor-outdoor dataset with LiDAR ground truth for 3D scene reconstruction and benchmarking. In *SCIA*, pages 3–17, 2025.

[54] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NeurIPS*, 2014.

A Evaluation metrics

For each frame pair (i, j), let $(\hat{R}_{ij}, \hat{t}_{ij})$ be the predicted relative pose and $(R^\star_{ij}, t^\star_{ij})$ the ground-truth pose. We report angular rotation and translation-direction errors in degrees:

$$R_{\mathrm{err}}(i,j) = \arccos\left(\frac{\mathrm{tr}\Big((R^\star_{ij})^\top \hat{R}_{ij}\Big) - 1}{2}\right), \qquad t_{\mathrm{err}}(i,j) = \arccos\left(\frac{\langle t^\star_{ij}, \hat{t}_{ij}\rangle}{\|t^\star_{ij}\|\,\|\hat{t}_{ij}\|}\right). \tag{14}$$

Ground-truth depth is not available for all fisheye datasets and is often sparse or sensor-projected. Therefore, we use a reprojection-based depth metric for the main comparison. Let

$$X_i(u) = D_i(u)\,\kappa^{-1}(u) \tag{15}$$

be the 3D point induced by predicted depth at pixel u. For each image pair, we resolve the global scale ambiguity with a single scalar s, and define

$$d_{\mathrm{reproj}}(i,j) = \min_{s>0} \frac{1}{|\Omega|} \sum_{u\in\Omega} \big\|\kappa\big(R^\star_{ij}\big(s\,X_i(u)\big) + t^\star_{ij}\big) - u_{ij}(u)\big\|_2 . \tag{16}$$

d_{reproj} uses ground-truth pose, so it evaluates depth independently of predicted pose. In practice, s is estimated per pair in closed form by robustly minimizing the distance between high-confidence matched 3D points.

When reliable ground-truth depth $D^\star_i$ is available, we additionally report scale-aligned depth error following Eigen et al. [54]. With a scalar $s^\star_i$ fitted to the ground-truth depth, we define

$$d_{AbsRel}(i) = \frac{1}{|\Omega_i|} \sum_{u\in\Omega_i} \frac{\big|s^\star_i D_i(u) - D^\star_i(u)\big|}{D^\star_i(u)}. \tag{17}$$

We also report threshold accuracy $\delta_{1.25}$, the fraction of pixels whose predicted depth is within a factor 1.25 of ground truth:

$$\delta_{1.25}(i) = \frac{1}{|\Omega_i|} \sum_{u\in\Omega_i} \mathbf{1}\left(\max\left(\frac{s^\star_i D_i(u)}{D^\star_i(u)}, \frac{D^\star_i(u)}{s^\star_i D_i(u)}\right) < 1.25\right). \tag{18}$$

B Predicted calibration

When ground-truth intrinsics are unavailable, we replace them with per-image predictions from AnyCalib [23] on one representative sequence per dataset and re-run adaptation. Tab. 5 shows that our method remains useful under approximate camera calibration and outperforms the competitors LoRA and CalTok consistently and significantly across all settings. The errors vary across sequences; however, RayTun3R exhibits higher robustness compared to the larger failures observed for LoRA and CalTok on TUM-VI, ScanNet++, and FIORD.

Table 5: **Ground-truth vs. AnyCalib calibration with DA3-Small.** We evaluate one representative sequence per dataset. For each table entry we report the triple comprising R°, t°, and d_{reproj} (lower is better). RayTun3R outperforms both LoRA and CalTok — often by a significant margin — also with approximate camera calibration estimated by AnyCalib.

Method	ETH3D terrains			KITTI 0009			TUM-VI room6			ScanNet++ 3f15			FIORD Kitchen		
RayTun3R (ours) (GT)	0.48	0.9	1.7	0.69	2.4	2.9	1.94	11.2	3.6	0.40	2.2	1.7	3.1	2.5	5.5
RayTun3R (ours) (AnyCalib)	1.02	2.3	2.6	1.14	2.5	3.1	1.16	7.0	4.4	0.75	4.6	3.1	4.9	3.0	5.2
LoRA (GT)	3.62	3.4	4.0	0.61	2.8	4.6	2.96	12.6	2.8	4.22	23.0	2.9	7.7	14.6	10.1
LoRA (AnyCalib)	2.30	4.7	3.2	2.32	2.8	4.8	6.60	20.4	16.6	3.52	17.5	4.9	10.2	14.2	10.8
CalTok (GT)	3.41	4.8	4.5	1.09	3.1	4.4	3.79	14.9	4.6	3.09	20.0	4.4	15.8	15.5	9.0
CalTok (AnyCalib)	2.15	4.4	3.4	1.90	8.5	5.8	7.21	42.0	38.1	4.44	24.7	5.5	30.3	59.9	10.6

C Cross-sequence transfer

We evaluate an adapter fitted on one sequence on the remaining sequences recorded by the same physical camera, without further optimization, and report the average in Tab. 6. On KITTI-360, the transferred adapter nearly matches an adapter fitted on the target sequence; on TUM-VI, the gap is larger, in particular for t° and d_{reproj}. In both cases, the transferred adapter substantially improves over the vanilla model.

Table 6: **Cross-sequence transfer with DA3-Small.** *Transfer*: adapter fitted on one sequence and evaluated on the remaining sequences of the same camera (averaged). *Target-trained*: adapter fitted on the evaluated sequence. Lower is better.

Dataset	Metric	Transfer	Target-trained	Vanilla
KITTI-360	R°	0.94	0.84	1.69
	t°	2.98	2.92	12.81
	d_{reproj}	3.90	3.88	11.64
TUM-VI	R°	3.21	2.41	10.41
	t°	20.1	13.2	23.2
	d_{reproj}	6.1	3.8	57.0

D Additional analyses

We report three additional analyses with DA3-Small. Tab. 7a varies the number of three-frame windows used for adaptation on KITTI-360 drive 0009. The adapter is fitted on the first windows of the sequence and evaluated on the remaining frames. A single window (three frames) already improves all metrics over the frozen model, but most of the gain requires more windows, with the best results at 30 windows.

Tab. 7b starts from the AnyCalib calibration of the same sequence (Appendix B), perturbs the focal length, the distortion parameters, or the principal point, and re-runs adaptation with the perturbed calibration. Translation and reprojection errors remain well below the frozen model in all settings; only a $\pm 20\%$ focal error raises the rotation error slightly above it. Note that the adapter is re-fitted under each perturbed calibration; the robustness of a fixed adapter to calibration changes after adaptation is not evaluated.

Tab. 8 splits the depth error by normalized image radius into center, middle, and periphery. For KITTI-360, we evaluate against sparse LiDAR depth projected into the fisheye image. The largest improvements occur in the periphery, where fisheye distortion is strongest, consistent with RayTun3R compensating for the radial mismatch between pinhole and fisheye geometry.

E Additional sequences for the backbone and calibration studies

Tabs. 2 and 5 evaluate one sequence per dataset, taken deterministically as the first sequence of each evaluation split. Tabs. 9 and 10 report results on further sequences per dataset.

F Can we simply undistort fisheye patch coordinates?

An alternative to adapting the model is to undistort each fisheye patch coordinate and to reuse the original 2D positional embedding table. This fails for wide-angle fisheye images because undistortion samples the pinhole grid highly non-uniformly: Many fisheye patches collapse onto a small central region of the original table, making their embeddings nearly indistinguishable, as Fig. 5 illustrates. The corresponding *Naive remap of PE* row in Tab. 4a confirms that this re-indexing alone gives poor translation and depth.

We therefore learn a lightweight residual correction directly on the positional embeddings, while keeping the pretrained backbone frozen.

Table 7: **Adaptation length and calibration perturbation with DA3-Small** on KITTI-360 drive 0009. (a) Number of three-frame adaptation windows. (b) Perturbed calibration used for re-adaptation, starting from AnyCalib. Lower is better.

(a) Adaptation length.

Windows	Frames	$R^\circ \downarrow$	$t^\circ \downarrow$	$d_{\text{reproj}} \downarrow$
1	3	1.31	5.52	6.86
2	4	1.36	4.72	6.04
5	7	1.56	6.07	4.98
10	12	1.40	4.41	4.21
20	22	1.26	3.31	3.52
30	32	0.69	2.40	2.90
Vanilla		2.23	5.75	9.91

(b) Calibration perturbation.

Calibration error	$R^\circ \downarrow$	$t^\circ \downarrow$	$d_{\text{reproj}} \downarrow$
Focal $\pm 10\%$ (avg.)	1.36	2.14	3.73
Focal $\pm 20\%$ (avg.)	2.44	2.57	6.55
Distortion $\pm 10\%$ (avg.)	1.43	2.08	3.91
Principal point ± 20 px	0.68	2.89	2.57
None (AnyCalib)	1.14	2.53	3.10
Vanilla	2.23	5.75	9.91

Table 8: **Depth error by image region with DA3-Small.** Pixels are grouped by normalized image radius. KITTI-360 uses sparse projected LiDAR depth. Each entry reports AbsRel for Vanilla / RayTun3R; lower is better.

Region	ScanNet++	KITTI-360
Center ($< 1/3$)	0.19 / **0.09**	0.32 / **0.17**
Middle ($1/3$–$2/3$)	0.14 / **0.10**	0.21 / **0.15**
Periphery ($> 2/3$)	0.60 / **0.18**	2.54 / **0.29**
Mean	0.29 / **0.11**	0.92 / **0.20**

G Computational resources

The experiments were run on NVIDIA RTX A6000 and NVIDIA RTX A4000 GPUs, with independent scenes, sequences, and baselines executed in parallel when possible. A typical full train-and-evaluate run takes about 2–3 hours per ETH3D, ScanNet++, or FIORD scene, and about 3–5 hours per KITTI-360 drive or TUM-VI sequence because of longer sequences. The reported experiments required approximately 180–250 total GPU-hours. Additional preliminary development and failed configurations required extra compute but are not included in this estimate.

H ETH3D across backbones

Tab. 11 reports mean ETH3D results across scenes for five frozen backbones: DA3-Small, DA3-Base, DA3-Large, π^3, and VGGT. RayTun3R gives the lowest mean rotation and translation error for every backbone. Center-PH is competitive on depth for DA3-Base, DA3-Large, and π^3, but its pose errors remain consistently higher. Vanilla inference degrades more strongly under fisheye geometry, especially for pose.

I Loss-component ablation

To justify the adaptation objective in Eq. (13), we run an ablation on the ETH3D `terrains` sequence with different combinations of the loss components. Tab. 12a shows that the full objective is best or tied on this sequence. Removing the pose loss mainly degrades pose accuracy, while removing either the ℓ_2 or TV regularizer generally weakens pose or reprojection performance.

J Additional component ablations

Tab. 13 separates learned positional updates from parameter-free camera-model corrections. On ETH3D terrains and FIORD Kitchen, learned PE/RoPE residuals are the dominant source of improvement. Parameter-free corrections alone provide only small gains on ETH3D and degrade FIORD Kitchen, but they improve the learned adapter when combined with it. This supports our claim that the learned positional residual is the main adaptation mechanism, while tokenization and prediction-grid corrections act as complementary camera model components.

Table 9: **Additional sequences for the backbone comparison** (Tab. 2). Each entry reports $R^\circ, t^\circ, d_{\text{reproj}}$, respectively; lower is better. **Bold** denotes the best result per backbone.

	π^3									VGGT								
Sequence	**RayTun3R**			LoRA			CalTok			**RayTun3R**			LoRA			CalTok		
ETH3D delivery	**0.61**	**1.65**	**0.85**	1.48	9.97	3.08	7.18	79.94	25.75	**0.20**	**1.05**	**1.30**	2.82	15.42	3.60	4.73	54.01	10.97
ETH3D electro	**0.72**	**3.65**	**1.58**	3.98	18.43	5.50	4.00	42.55	4.11	**1.15**	**8.96**	**1.94**	2.42	13.51	4.43	5.07	56.25	2.29
KITTI 0003	**0.78**	**5.90**	**4.34**	1.31	37.98	6.98	1.13	91.87	9.74	**1.02**	**4.83**	**7.19**	2.15	10.99	17.36	1.14	100.71	13.63
KITTI 0005	**0.91**	**4.84**	**2.56**	1.00	18.05	3.67	1.32	97.53	4.35	**1.36**	6.55	**2.73**	1.80	**5.29**	5.58	2.89	69.75	8.71
ScanNet++ 04c2	**1.10**	**8.29**	**2.27**	4.31	19.67	3.59	14.88	61.52	5.19	**1.12**	**8.65**	**2.59**	2.43	19.40	6.84	8.22	76.96	11.25
ScanNet++ 0a5c	**1.68**	**14.41**	**2.92**	17.30	61.22	7.87	13.52	58.84	5.20	**1.46**	**14.75**	**3.84**	4.68	22.53	5.57	4.98	34.39	6.25
TUM-VI room1	**3.04**	**20.07**	**4.27**	8.68	31.26	20.53	19.10	85.62	13.05	**1.89**	**12.41**	**5.10**	6.64	27.70	35.37	23.06	83.70	39.15
TUM-VI room2	**1.83**	**7.82**	**2.45**	6.40	33.74	7.94	23.59	94.26	8.87	**1.60**	**6.25**	**4.43**	5.61	16.97	11.02	14.92	72.84	9.07
FIORD meetingroom	**6.36**	**10.33**	**4.43**	90.13	37.80	16.06	96.92	72.14	23.06	5.88	**10.29**	**6.67**	8.05	16.34	7.80	**3.71**	32.06	13.75
FIORD outcorridor	4.25	**13.58**	**7.56**	**1.56**	14.22	41.10	6.01	74.11	34.60	3.74	6.28	**7.64**	**1.55**	**5.21**	11.58	3.70	37.18	18.30

Table 10: **Additional sequences for ground-truth *vs*. AnyCalib calibration** (Tab. 5), DA3-Small. Each entry reports $R^\circ, t^\circ, d_{\text{reproj}}$; lower is better.

	RayTun3R						LoRA						CalTok					
Sequence	GT			AnyCalib			GT			AnyCalib			GT			AnyCalib		
ETH3D delivery	0.60	1.72	2.52	0.62	3.99	2.37	0.96	4.58	4.55	0.98	7.46	4.62	1.59	8.48	5.61	1.36	11.95	4.94
KITTI 0003	0.83	3.21	4.17	0.96	3.09	5.21	3.27	5.08	4.79	1.63	15.99	12.92	4.12	9.29	5.94	2.10	18.58	17.61
KITTI 0005	0.83	2.71	2.31	1.00	2.41	3.03	0.96	2.74	3.15	1.17	3.20	6.77	1.35	3.96	3.57	1.58	6.36	8.96
TUM-VI room1	2.24	13.45	4.83	2.10	11.20	6.34	3.32	13.40	3.25	7.71	30.61	9.81	4.10	17.61	9.64	9.47	37.16	54.59
TUM-VI room2	2.06	6.52	3.82	1.69	5.00	4.50	2.24	8.01	3.69	4.64	12.64	12.44	2.97	7.21	7.09	3.93	27.98	35.82
ScanNet++ 04c2	0.74	6.01	3.26	0.98	5.64	4.46	2.13	17.66	3.63	2.75	19.67	7.39	2.40	23.45	3.51	3.03	20.12	8.42
ScanNet++ 0a5c	1.26	13.42	4.89	1.73	12.68	8.70	2.54	24.75	4.64	5.20	24.95	12.31	4.51	29.30	13.95	5.97	27.27	14.51
FIORD meetingroom	7.91	7.59	8.80	6.26	5.79	11.50	14.71	8.65	9.46	34.22	43.11	14.56	50.74	42.41	17.10	76.36	55.53	16.78
FIORD outcorridor	2.28	7.13	12.73	2.52	8.45	7.28	2.41	12.55	12.92	2.25	10.40	9.77	3.66	13.70	22.22	2.50	10.26	12.07

Tab. 12b separates absolute PE and RoPE adaptation on ETH3D terrains and evaluates the sensitivity to the number of radial (N_r) and angular (N_θ) bins. *Absolute PE* is the main learned component on DA3-Small: using *RoPE only* remains close to the unadapted model, while *Absolute PE* alone nearly matches the full model. The bin-count sensitivity shows that increasing the radial resolution beyond 20 bins gives only marginal gains, while removing angular bins substantially degrades performance. This indicates that the remaining camera-geometry mismatch contains an orientation-dependent component, not only a radial one.

K Broader impact

This work improves geometric perception for fisheye cameras, which are common in robotics, autonomous systems, AR/VR, and indoor mapping. More reliable geometry from wide-field cameras can lead to safer navigation and scene understanding, especially when standard pinhole assumptions fail. At the same time, like other 3D perception methods, improved reconstruction and pose estimation should be deployed with attention to privacy, consent, and safety in applications involving people or private spaces. Because fisheye cameras are common in vehicles, mobile robots, doorbells, and indoor monitoring devices, easier camera adaptation can also lower the barrier to deploying 3D mapping systems in private or shared spaces. Responsible deployment should include consent, calibration validation, and application-specific safety checks.

L Existing assets and licenses

Tab. 14 summarizes the external datasets, pretrained model weights, and code used in this work. We use these assets only for non-commercial academic research and evaluation, and do not redistribute datasets or pretrained checkpoints.



(a) Original PE (b) Undistortion pattern (c) Naive sampling

Figure 5: **Naive undistortion-based PE reuse.** (a) Original DA3-Small PE table, shown as ℓ_2 distance to the center embedding. (b) Locations reached by undistorted and rescaled fisheye coordinates. (c) Resulting PE lookup. Many fisheye patches collapse onto a small central region of the table, producing nearly identical embeddings and losing positional resolution.

Table 11: **Mean ETH3D results across backbones.** For each frozen backbone, entries are ordered as R°, t°, AbsRel, and $\delta_{1.25}$. Lower is better except for $\delta_{1.25}$. **Bold** marks the best method within each backbone and metric.

Method	DA3-Small				DA3-Base				DA3-Large				π^3				VGGT			
	R	t	AbsRel	δ	R	t	AbsRel	δ	R	t	AbsRel	δ	R	t	AbsRel	δ	R	t	AbsRel	δ
Baseline	8.59	15.16	0.178	0.751	8.27	12.24	0.147	0.794	6.36	13.94	0.135	0.828	2.66	11.30	0.250	0.642	3.19	11.52	0.285	0.557
Center-PH	3.46	13.70	0.111	0.867	1.85	9.36	**0.082**	**0.911**	1.56	9.32	**0.075**	**0.941**	1.08	10.46	**0.156**	0.772	1.17	8.98	0.228	0.623
RayTun3R (ours)	**0.70**	**4.48**	**0.107**	**0.884**	**0.54**	**3.26**	0.089	0.910	**0.51**	**2.96**	0.083	0.925	**0.66**	**2.48**	0.175	**0.863**	**0.96**	**4.82**	**0.139**	**0.834**

Table 12: **Loss, PE/RoPE, and bin-count ablations.** (a) Ablation of loss components; we study the effect of pose supervision and the two PE regularizers from Eq. (13). (b) PE/RoPE split and bin-count sensitivity on ETH3D terrains; each entry reports $R^\circ, t^\circ, d_{\text{reproj}}$, respectively; lower is better.

(a) Ablation of loss components.

Loss components			ETH3D terrains		
Pose	L2	TV	$R^\circ \downarrow$	$t^\circ \downarrow$	$d_{\text{reproj}} \downarrow$
✓	✓	✓	0.48	0.9	1.7
×	✓	✓	0.68	1.1	1.7
✓	×	✓	0.81	1.0	1.8
✓	✓	×	0.58	1.3	1.9
×	×	✓	0.65	1.4	1.8
×	✓	×	0.61	1.5	1.7
✓	×	×	0.78	1.5	1.8
×	×	×	0.75	1.8	1.8

(b) PE/RoPE split and bin-count sensitivity.

Configuration	ETH3D terrains		
Absolute PE only (no RoPE)	0.68	**0.9**	**1.6**
RoPE only (no Absolute PE)	19.52	7.8	9.6
Both (full)	**0.48**	**0.9**	**1.6**
N_r=10, N_θ=8	0.72	0.9	1.7
N_r=20, N_θ=8 (default)	0.48	**0.9**	1.6
N_r=40, N_θ=8	**0.47**	**0.9**	**1.5**
N_r=20, N_θ=0 (radial only)	2.82	3.3	3.6

Table 13: **Learned positional residuals vs. parameter-free camera-model corrections.** We report pose rotation error R°, translation-direction error t°, and reprojection error d_{reproj} on ETH3D terrains and FIORD Kitchen. Lower is better for all metrics. *Param.-free* denotes patch undistortion, prediction-grid coordinate correction, and border-token handling. **Bold** marks the best value per dataset and metric.

Configuration	Details	ETH3D terrains			FIORD Kitchen		
		$R^\circ \downarrow$	$t^\circ \downarrow$	$d_{\text{reproj}} \downarrow$	$R^\circ \downarrow$	$t^\circ \downarrow$	$d_{\text{reproj}} \downarrow$
Vanilla	no PE/RoPE, no param.-free corrections	19.87	8.2	10.7	28.09	20.7	14.2
Param.-free only	patch undist., grid correct., border token	17.78	7.3	10.9	39.04	36.3	19.2
Learned only	PE+RoPE, no param.-free corrections	0.67	1.0	1.8	4.64	4.1	5.8
RayTun3R Full	PE+RoPE + param.-free corrections	**0.48**	**0.9**	**1.6**	**3.10**	**2.5**	**5.5**

Table 14: **Existing assets used in this work.** External datasets and pretrained models used for evaluation or adaptation.

Asset	Type	Use	License / terms
KITTI-360 [49]	Dataset	Evaluation	CC BY-NC-SA 3.0; scripts MIT
TUM-VI [50]	Dataset	Evaluation	Data CC BY 4.0; code BSD-2-Clause
ScanNet++ [51]	Dataset	Evaluation	Terms of Use; non-commercial research/education
ETH3D [52]	Dataset	Evaluation	CC BY-NC-SA 4.0
FIORD [53]	Dataset	Evaluation	CC BY 4.0
DA3 [5]	Model/code	Backbone	Apache-2.0; CC BY-NC 4.0 for Large/Giant
π^3 [4]	Model/code	Backbone	Code BSD-3-Clause; weights CC BY-NC 4.0
VGGT [3]	Model/code	Backbone	Checkpoint-specific license; non-commercial checkpoint
UFM [45]	Model/code	Matches	Code BSD-3-Clause; checkpoint CC BY-NC-SA 4.0

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the scope, assumptions, method components, and empirical claims. The limitations section further bounds the claims to fisheye adaptation, radial distortion assumptions, calibration inputs, and fisheye imagery.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a dedicated limitations paragraph in Sec. 6, covering camera-specific adaptation, radial distortion assumptions, calibration, and scope beyond fisheye cameras.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not present formal theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Secs. 3 and 5 and the appendix describe the camera models, adapter parameterization, losses, baselines, datasets, evaluation protocol, and metrics needed to reproduce the main experimental setup.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The paper uses public datasets and pretrained models and states that code will be released, but an anonymized code package with full reproduction scripts is not included in the submission.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the datasets, local-pair protocol, adaptation-window construction, optimizer, learning rate, image resolution, adapter size, initialization, baselines, and evaluation metrics in the method, appendix, and experiments sections.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The paper reports aggregate results over datasets, sequences, or scenes, but does not include error bars, confidence intervals, or formal statistical significance tests.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper reports inference timing and states the GPU type, typical train/evaluation runtime per dataset, parallel execution setup, and approximate total GPU-hours for the reported experiments in Sec. 5.2 and Appendix G.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses public datasets and pretrained vision models for geometric perception experiments, and we are not aware of deviations from the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: Appendix K discusses positive uses for robotics, autonomous systems, AR/VR, and mapping, and notes privacy, consent, and safety considerations for deployment in applications involving people or private spaces.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release a high-risk generative model, scraped dataset, or other asset requiring special misuse safeguards.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Appendix L lists all external datasets, pretrained models, and code assets used in the experiments, together with citations, use in this work, and license or terms-of-use information.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce a new dataset, benchmark, or pretrained model asset. The proposed method is an adaptation procedure evaluated on existing datasets and pretrained models.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing experiments or newly collected human-subject data.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The work does not involve crowdsourcing experiments or newly collected human-subject data requiring IRB or equivalent review.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs are not used as an important, original, or non-standard component of the core method or experiments. Any writing or editing assistance does not affect the scientific methodology.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.