Stereo4D: Learning How Things Move in 3D from Internet Stereo Videos

Supplementary Material

7. Stereo4D Statistics

We collected around 110k clips from 6,493 Internet VR180 videos. The videos were curated from public YouTube content using the tag "VR180", available under YouTube's Standard License. We have released derived geometry, motion data, and video links, under a CC license here.

Fig. 11 shows the camera translation distribution between the first and last frame of each clip. Fig. 12 shows, from left to right: (a) the distribution of x,y,z camera translations (log scale), (b) the distribution of rotations, (c) a sample of 5000 camera trajectories, viewed from above, colored by final camera orientation (red=right, cyan=left).

In Fig. 13, we measure the motion in terms of pixel displacement projected onto the image frame. Measuring motion in pixel-space emphasizes motion that occurs closer to the camera, since such motion yields larger pixel displacements, while naturally de-emphasizing motion further from the camera.

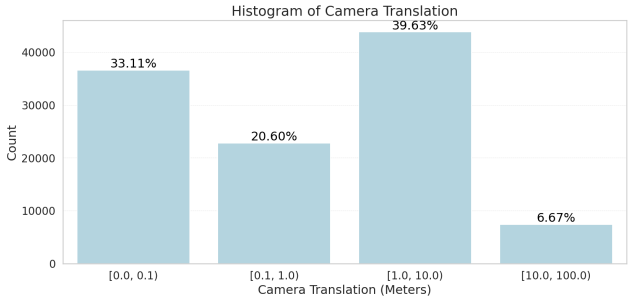

Figure 11. Camera statistics from Stereo4D. We measure the difference (in meters) of camera poses between the start and end frame of each video clip as calculated by SfM.

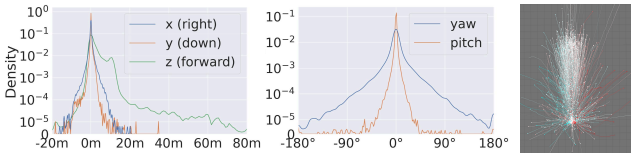

Figure 12. Camera statistics from Stereo4D. From left to right: a) the distribution of x,y,z camera translations (log scale), (b) the distribution of rotations, (c) a sample of 5000 camera trajectories, viewed from above, colored by final camera orientation (red=right, cyan=left)

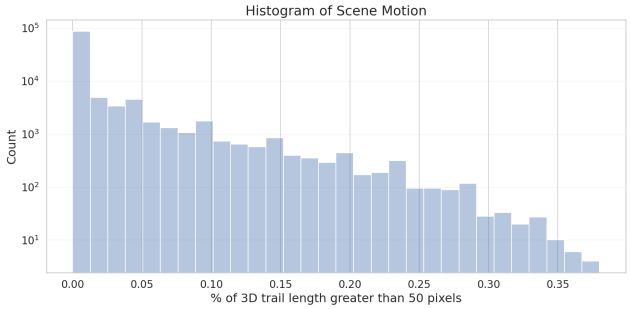

Figure 13. Scene motion statistics from Stereo4D. We measure scene motion in terms of pixel displacement projected onto the image frame. For each video, we measure the percentage of tracks that have 3D trail length greater than 50 pixels. The 3D trail length is measured by Eqn. 3.

8. Ablations

Track optimization. Fig. 15 shows a top-down view of Fig 3. We tried multiple stereo depth methods when developing our system, e.g., BiDAStereo [37] but it still shows jitter and drift (left). That said, as more advanced stereo methods become available, we can adopt them. Ablations (right): w/o $\mathcal{L}_{\text{static}}$ (static content reduces jitter from $\mathcal{L}_{\text{dynamic}}$ but still drifts); w/o $\mathcal{L}_{\text{dynamic}}$ (dynamic trajectories are distorted along camera rays by $\mathcal{L}_{\text{static}}$).

Effect of time gap for DynaDUSt3R. We evaluate DynaDUSt3R's capability with regard to time gap $\Delta_t = t_1 - t_0$ of input images. Fig. 14 shows motion error vs. input frame gap, across 1k Stereo4D pairs. As the time gap increases, the motion magnitude and uncertainty grows, leading to an increase in error.

DynaDUSt3R motion head ablation. DynaDUSt3R uses a separate motion head to predict motion for the pointmaps $M^{v\to t_q}$. Alternatively, one can also predict the deformed points with the same point head by conditioning on the time embedding. We compare 1) ours: predicting $M^{v\to t_q}$ with motion heads; and 2) directly regressing $P^{v\to t_q}$ with point heads and evaluate on the same Stereo4d test set in Table 1. Using a single head to predict motion (2) results in a drop in motion accuracy across all metrics: $\text{EPE}_{3D}\downarrow$ = (**0.1110** $\rightarrow$ 0.1401), $\delta^{0.05}_{3D}\uparrow$ = (**65.07** $\rightarrow$ 59.19), $\delta^{0.10}_{3D}\uparrow$ = (**75.18** $\rightarrow$ 69.73). This is likely due to decreased decoder capacity.

9. More qualitative comparisons

9.1. More results on held-out Stereo4D examples

Fig. 16 shows additional DynaDUSt3R predictions on the Stereo4D held-out test set, extending Fig. 7 from the main paper. Fig. 17 shows additional qualitative examples of mo-

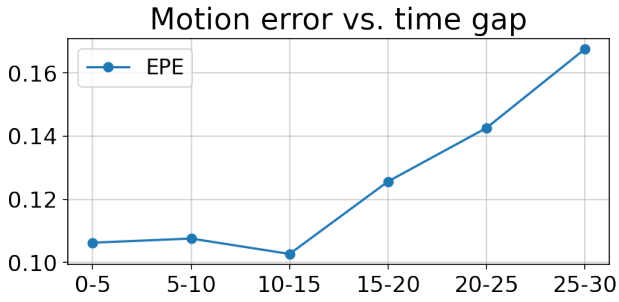

Figure 14. **Model performance with different input time gap**: Motion error vs. input frame gap. As the time gap increases, the motion magnitude and uncertainty grows, leading to an increase in error.

tion comparisons on Stereo4D test set, extending Fig. 8 from the main paper. Fig. 17 compares variants of Dyna-DUSt3R trained on different data sources. The model trained on PointOdyssey incorrectly predicts large 3D motions, while the model trained on Stereo4D makes more accurate motion predictions, closer to ground truth.

9.2. Additional track optimization examples

In Fig. 19, we illustrate estimated tracks for a video sequence featuring a forward-moving camera and vehicles driving towards the camera. Our initial 3D tracks derived directly from RAFT depth, BootsTAP 2D tracks, and SfM camera pose, show significant jitter for both dynamic (vehicle) and static (ground) points. However, after applying our track optimization, the ground points produce stable, static tracks, and vehicle tracks become smooth and coherent.

10. Dataset curation details

10.1. Equirectangular videos

The raw videos that we collect (see examples in Fig. 18) are natively stored in a cropped equirectangular format, which differs from a full 360° equirectangular projection as the horizontal field of view of the cropped format typically spans 180°—half of a full sphere. These videos often contain metadata specifying the horizontal and vertical field of view. For instance, metadata for a typical video might specify $\mathsf{start}_{\mathsf{yaw}} = -90.0°$, $\mathsf{end}_{\mathsf{yaw}} = 90.0°$, $\mathsf{start}_{\mathsf{tilt}} = -90.0°$, $\mathsf{end}_{\mathsf{tilt}} = 90.0°$; Since many VR180 videos are designed for an immersive VR experience, they are typically viewed with headsets. Hence, the baseline between the left and right cameras typically closely matches the average human interpupillary distance of 6.3 cm.

10.2. Structure from motion

For ease of processing with standard 3D computer vision pipelines, and to benefit from the wide FoV of the input videos, we convert the videos from their native format (equirectangular projections) to a fisheye format for camera pose estimation. We use a 140° field of view for these fisheye-projected videos, because many equirectangular videos have a black fade-out/feathering/vignetting effect applied at the boundary, as shown in Fig. 18. We found that using wider FoV frames significantly improves camera pose estimation in dynamic scenes. When using narrow FoV projections, dynamic objects are more likely to occupy a large fraction of the frame; when these dynamic foreground objects are rich in features, they can confuse camera tracking algorithms, leading to inaccurate camera poses that track the dynamic object rather than producing true camera motion with respect to the environment. In contrast, wide-angle fisheye videos capture more background regions, which tend to have stable features for tracking, yielding more reliable camera poses.

We first use ORB-SLAM2's stereo estimation mode [61] to identify trackable sequences within the videos, utilizing the method devised by Zhou *et al*. to divide videos into discrete, trackable shots [119]. For each given shot, consisting of frames $(I_i, \ldots, I_n)$, we estimate camera poses and rig calibration via an incremental global bundle adjustment algorithm similar to COLMAP [76]. We initialize the stereo rig calibration to be that of a rectified stereo pair with baseline 6.3 cm, but optimize for the calibration as part of the bundle adjustment process, as in practice the stereo rig can vary significantly from its nominal configuration. This process yields a camera position $\mathbf{c}_i$ and orientation $\mathbf{R}_i$ for each frame i (defined as the pose of the left camera), and a position $\mathbf{c}_r$ and orientation $\mathbf{R}_r$ for the right camera relative to the left (assumed to be constant throughout the shot).

10.3. Depth estimation

Depth estimation is first performed on a per-frame basis, with disparity maps computed independently for each frame.

We use the estimated camera rig calibration $\mathbf{c}_r, \mathbf{R}_r$ to rectify the original high resolution equirectangular video frames, ensuring that (1) the left and right views have centered principal points, (2) are oriented perpendicular to the baseline, and (3) pointing in a parallel direction. We then convert the equirectangular videos to perspective projections for downstream predictions.

Disparity is estimated from optical flow [84, 90] between the rectified left and right frames. The x-component of the optical flow is used as disparity, which is converted to metric depth using:

$$\mathsf{Depth} = \frac{\mathsf{baseline} \times f}{\mathsf{disparity}}. \tag{8}$$

Here $\mathsf{baseline} = 0.063$m, and f is the frame's focal length.

Outlier Rejection. Several criteria are applied to filter out unreliable pixels: *Inconsistency between left and right*

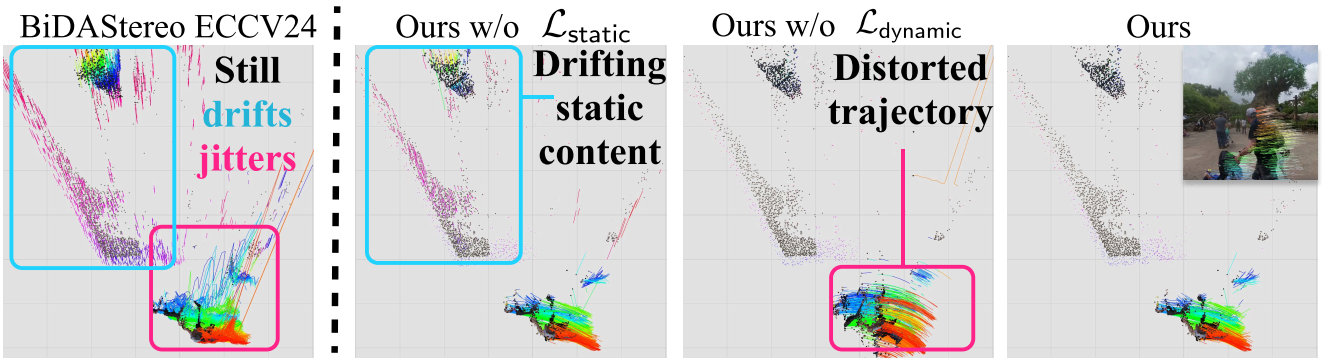

Figure 15. **Track optimization ablation**, extending Fig. 3. SOTA video stereo method, BiDAStereo [37], still shows jitter and drift (left). Ablations (right): w/o $\mathcal{L}_{\text{static}}$ static content reduces jitter from $\mathcal{L}_{\text{dynamic}}$ but still drifts; w/o $\mathcal{L}_{\text{dynamic}}$ dynamic trajectories are distorted along camera rays by $\mathcal{L}_{\text{static}}$.

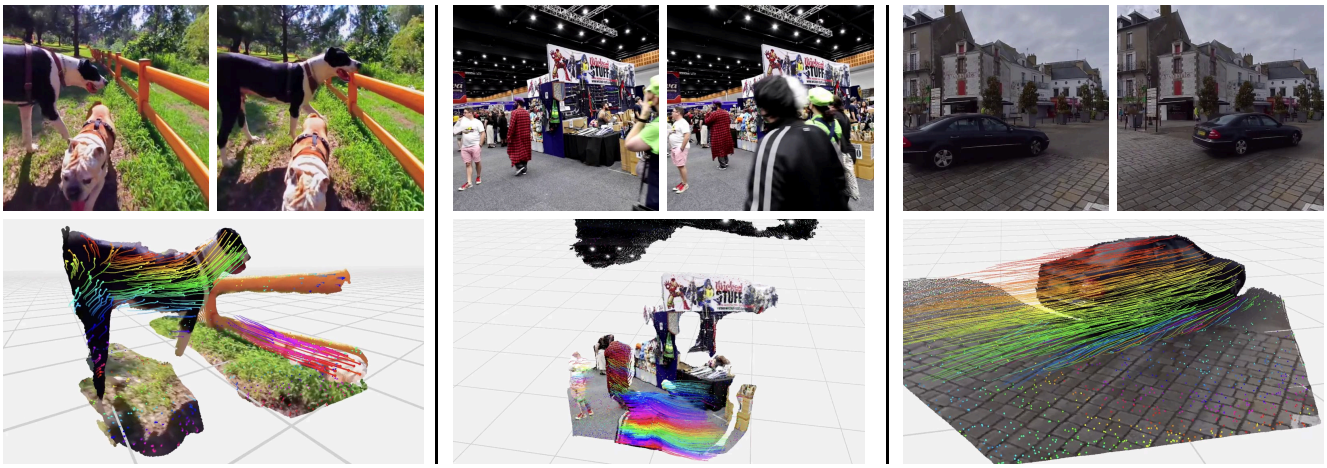

Figure 16. **More qualitative results on Stereo4D test set.** Extending Fig. 7, we visualize image pairs and corresponding dynamic 3D point clouds predicted by DynaDUSt3R trained on Stereo4D. Our method recovers accurate 3D shape and complex scene motion.

eyes: Disparity is rejected if the optical flow fails a cycle-consistency check with an error exceeding one pixel. *Depth values exceeding 20 meters* are considered invalid. Estimating accurate depth beyond a certain range requires sub-pixel disparity estimation, and therefore the resulting depths are usually very noisy. *Negative flow values* that shouldn't occur, but can, often due to errors in textureless regions. *Large vertical flow:* pixels with a y-component of flow exceeding one pixel are removed (as in our rectified stereo pairs correspondences should have the same y-value, and violating that epipolar constraint indicates uncertain matches). *Occlusion boundaries:* Depth gradients exceeding a threshold (threshold $= 0.3$) indicate occlusion boundaries and are rejected. For a pixel location (x, y), depth gradients are computed as:

$$\text{grad}_x = |\text{Depth}(x+1, y) - \text{Depth}(x-1, y)|,$$

$$\text{grad}_y = |\text{Depth}(x, y+1) - \text{Depth}(x, y-1)|.$$

Pixels are rejected if $\text{grad}_x > \text{threshold} \times \text{Depth}(x, y)$ or $\text{grad}_y > \text{threshold} \times \text{Depth}(x, y)$.

10.4. 2D tracks

We extract long-range 2D point trajectories using BootsTAP [17]. We run tracking on the left-eye video only. For every 10 frames, we uniformly initialize query points on image with stride 4. We then remove duplicated queries if earlier tracks fall within 1 pixel of a query point.

10.5. Choice of FoV and resolution for perspective projection

When converting the equirectangular videos to perspective projections, we use two FoVs: 60° and 120°. Both perspective videos are set to a resolution of 512×512, the maximum supported by BootsTAP. The 60° projection offers a higher sampling rate in scene units, which improves the accuracy of depth estimation and 2D tracks when measured in meters. Additionally, it has smaller perspective distortion near the

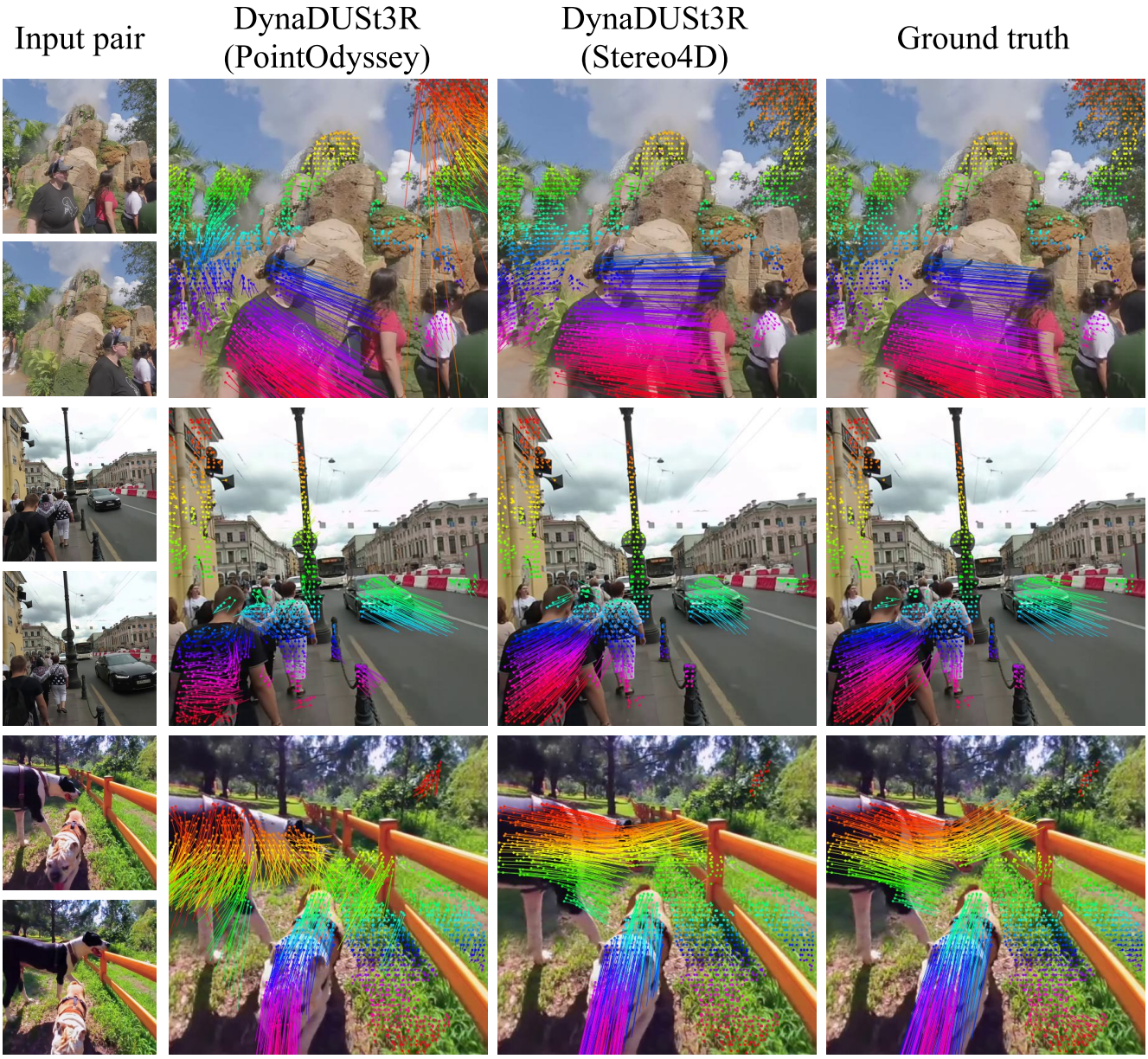

Figure 17. **More qualitative comparisons of 3D motion in the Stereo4D test set.** Extending Fig. 8, we compare variants of DynaDUSt3R trained on different data sources. The Stereo4D-trained model also makes more precise motion predictions than the PointOdyssey-trained model.

Figure 18. Example equirectangular stereo videos collected from the internet.

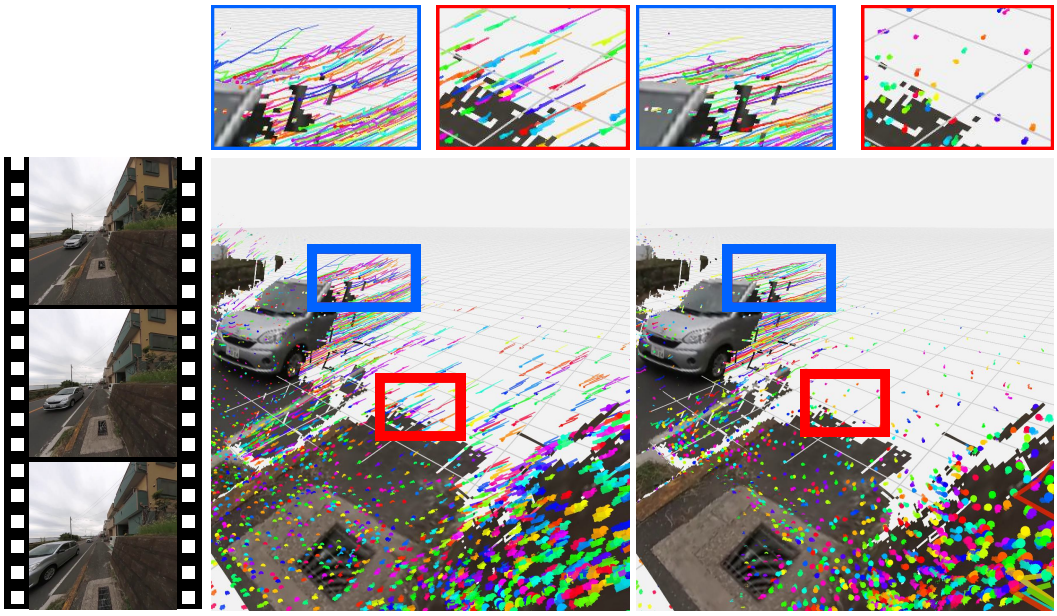

Figure 19. **Effect of Track Optimization.** We compare 3D tracks on a challenging walking tour video sequence. In this clip (left), the camera moves forward while vehicles drive toward the camera. We visualize the results across 16 frames, showing 3D trails left by both dynamic and static points. **Middle**: Our initial 3D tracks, created directly from RAFT, BootsTAP and SfM camera pose, also exhibit significant jitter for both dynamic (vehicle) and static (ground) points. **Right**: After applying our track optimization, the ground points yield stable, static tracks, and vehicle tracks become smooth and coherent.

image boundaries. In contrast, the 120° projection provides wider coverage, ensuring longer 2D tracks across the videos. This trade-off allows us to balance data quality with spatial coverage for downstream tasks, e.g. DynaDUSt3R. We take the union of the 3D tracks derived from each of these videos for DynaDUSt3R training supervision.

11. DynaDUSt3R training details

Dataloader. During training, we randomly sample two frames from the training videos that are at most 60 frames apart, at times t_0 and t_1, ($t_0 < t_1$). Additionally, we also sample one auxiliary frame in between, at time $t_{\text{aux}}, t_0 < t_{\text{aux}} < t_1$, for additional track supervision between the two input frames. During training, we add data augmentation by applying random crops and color jitter to the input images and cropping the ground truth pointmap and motionmap accordingly.

Training. The network takes input the two RGB images as well as query times $t_q = \{0, 1, \frac{t_{\text{aux}} - t_0}{t_1 - t_0}\}$ and predicts the pointmaps for the two input views and motionmaps for each query t_q. We supervise the network with losses defined in Eqn. 6 and 7. We initialize our network with the DUSt3R weights and initialize the motion head with the same weights as the point head. We finetune for 49k iterations with batch size 64, learning rate 2.5×10^{-5}, and optimized by Adam with weight decay 0.95.