

# Appendix

## A. Additional Baseline Comparisons

To highlight the challenge of sparse-view 4D reconstruction, we compare with five additional baselines for monocular 4D reconstruction and sparse-view reconstruction. Our method remains state-of-the-art.

Method	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Description
MV-MonST3R [77]	12.64	0.475	0.574	Monocular 4D reconstruction
MV-MegaSAM [38]	14.25	0.578	0.412	Monocular 4D reconstruction
ViewCrafter [75]	24.97	0.834	0.135	Multi-view diffusion
DNGaussian [34]	27.34	0.897	0.103	Sparse-view reconstruction
InstantSplat [13]	29.11	0.909	0.082	Sparse-view reconstruction
MonoFusion	<b>30.64</b>	<b>0.930</b>	<b>0.055</b>	Ours

**Baseline: Monocular 4D reconstruction.** We compare with MonST3R, as well as MegaSAM which claims better results than MonST3R. For each method, we report the best result among using a single-view video, concatenating the 4 views into longer video, or interleaving the 4 views (simulating a “rotating” camera). MonST3R and MegaSAM fail in sparse-view scenarios, especially with large viewpoint shifts. When given concatenated or interleaved sparse-view videos as input, both models simply copy the input frames onto flat planes: notice the image corners in visual results. Given a single input video, both models are missing large regions due to occlusions and unseen geometry past the video boundary. Without non-trivially handling images from multiple wide-baseline input views, MonST3R and MegaSAM alone cannot produce view-consistent reconstructions. **Baseline: Multi-view diffusion**. We use ViewCrafter, a video diffusion method for novel-view synthesis, to novel views at each timestep. ViewCrafter’s rendered views are worse than our method’s outputs, due to grid-like rendering artifacts, missing black regions of the scene, additional hallucinations, and imperfect alignment between different input views. **Baseline: Sparse-view methods**. We compare with alternative sparse-view methods InstantSplat and DNGaussian. Although they do not handle dynamic scenes, we run them independently per frame. Both methods suffer from blurry reconstructions and missing details, although InstantSplat benefits from cross-view consistency coming from DUST3R. A qualitative analysis of all baselines for  $5^\circ$  and  $45^\circ$  novel view synthesis on the ExoRecon dataset is available in Fig. 10 and 11.

## B. Training View Renderings

To build confidence in our implementations, we validate every baseline we run by verifying that each method looks reasonable at training views (Fig. 9). It is worth noticing that in each iteration of optimization, we sample a batch of frames out of the video to optimize the overall loss. As the loss is optimized as a global minimum averaged over all frames, it is possible that some artifacts remain for certain frames.

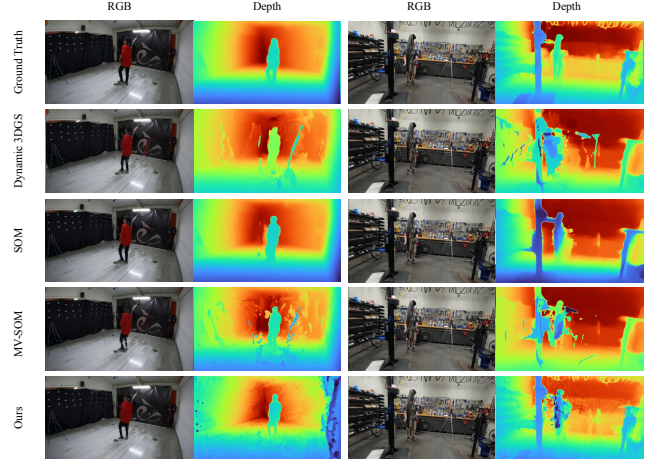


Figure 9. **Training view results.** We visualize the rasterized RGB image and depth map from each method for the dancing (left) and bike repair (right) sequences. All methods are capable of producing reasonable training views and depth maps. It is worth noticing that in each iteration of optimization, we sample a batch of frames out of the video to optimize the overall loss. As the loss is optimized as a global minimum averaged over all frames, it is possible that some artifacts remain for certain frames.

## C. Training Details

In Tab. 4, we report the learning rate and loss weights of Gaussians in our optimization process. These hyperparameters are shared across every scene that we evaluated on. Specifically,  $\mathcal{L}_{\text{smooth.bases}}$  enforces smooth motion bases by penalizing high accelerations in rotations and translations.  $\mathcal{L}_{\text{smooth.tracks}}$  promotes smooth object tracks by penalizing large accelerations in object positions across frames.  $\mathcal{L}_{\text{depth.grad}}$  aligns the gradients of the predicted and ground truth depth maps to preserve structural details.  $\mathcal{L}_{\text{z.accel}}$  penalizes high accelerations along the depth axis to reduce jitter in depth estimation.  $\mathcal{L}_{\text{scale.val}}$  constrains the variance of scale parameters of Gaussians to achieve consistent representations.

Table 4. Training hyper-parameters: learning rates (left) and loss weights (right) shown row-by-row

Parameter	Learning Rates			Loss Weights	
	FG LR	BG LR	Motion LR	Loss Param.	Weight
means	$1.6 \times 10^{-4}$	$1.6 \times 10^{-4}$	—	$w_{\text{rgb}}$	7.0
opacities	$1 \times 10^{-2}$	$1 \times 10^{-2}$	—	$w_{\text{mask}}$	5.0
scales	$5 \times 10^{-3}$	$1 \times 10^{-3}$	—	$w_{\text{feat}}$	7.0
quats	$1 \times 10^{-3}$	$1 \times 10^{-3}$	—	$w_{\text{smooth.bases}}$	0.1
colors	0	$1 \times 10^{-2}$	—	$w_{\text{depth.reg}}$	1.0
feats	$1 \times 10^{-3}$	$1 \times 10^{-3}$	—	$w_{\text{smooth.tracks}}$	2.0
motion.coefs	$1 \times 10^{-3}$	—	—	$w_{\text{scale.var}}^1$	0.01
rots	—	—	$1.6 \times 10^{-4}$	$w_{\text{z.accel}}$	1.0
transls	—	—	$1.6 \times 10^{-4}$	$w_{\text{track}}$	2.0

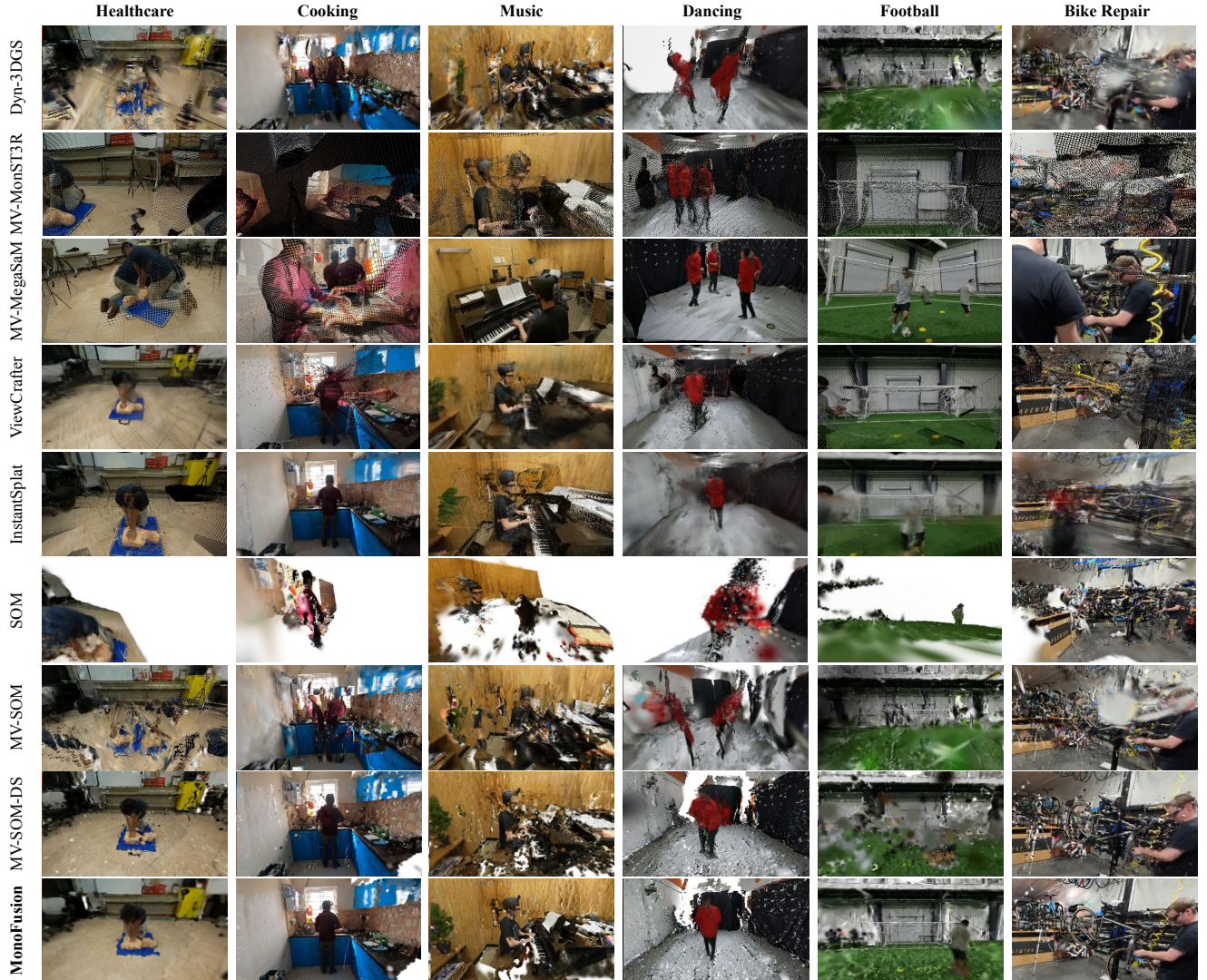


Figure 10. **Qualitative results of 45° extreme novel view synthesis results on ExoRecon.** For each of the baseline methods considered in the main paper and in the appendix, we visualize the rasterized RGB image from each method for a 45° novel-view from one of the training views. Results suggest that prior work that is tuned for a dense camera setup struggles to work in the sparse-view case, and methods that are tuned for a monocular input cannot naively address a multi-view setup. Our proposed MonoFusion, successfully uses priors in the form of monocular depth and feature-based motion clustering to achieve the best of both worlds.

## D. Alternative design choice

We explore higher-quality and metric-scale depth initialization, by replacing UniDepth in MV-SOM with improved metric depth estimators. Our results suggest that simply improving the quality of metric-scale depth in MV-SOM is not enough.

Method	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	IOU $\uparrow$	AbsRel ( $\downarrow$ )
w/ UniDepth [48]	26.91	0.890	0.138	0.845	0.474
w/ Metric3Dv2 [22]	27.29	0.894	0.132	0.847	0.462
w/ UniDepthV2 [49]	27.65	0.900	0.103	0.872	0.356
<b>MonoFusion</b>	<b>30.64</b>	<b>0.930</b>	<b>0.055</b>	<b>0.963</b>	<b>0.290</b>

## E. Limitations and Future Work

We address two key limitations of our work. First, like previous methods, we rely heavily on 2D foundation models to estimate priors (e.g. depth and dynamic masks) for gradient-based differentiable rendering optimization. Thus, imprecise priors can harm the downstream rendering process. In addition, the current pipeline requires a user prompt to specify dynamic masks for each moving object [3], which can be labor-intensive for complex scenes and may fail in some cases (Fig. 12). To solve this, distilling dynamic masks from foundation models or inferring dynamic masks



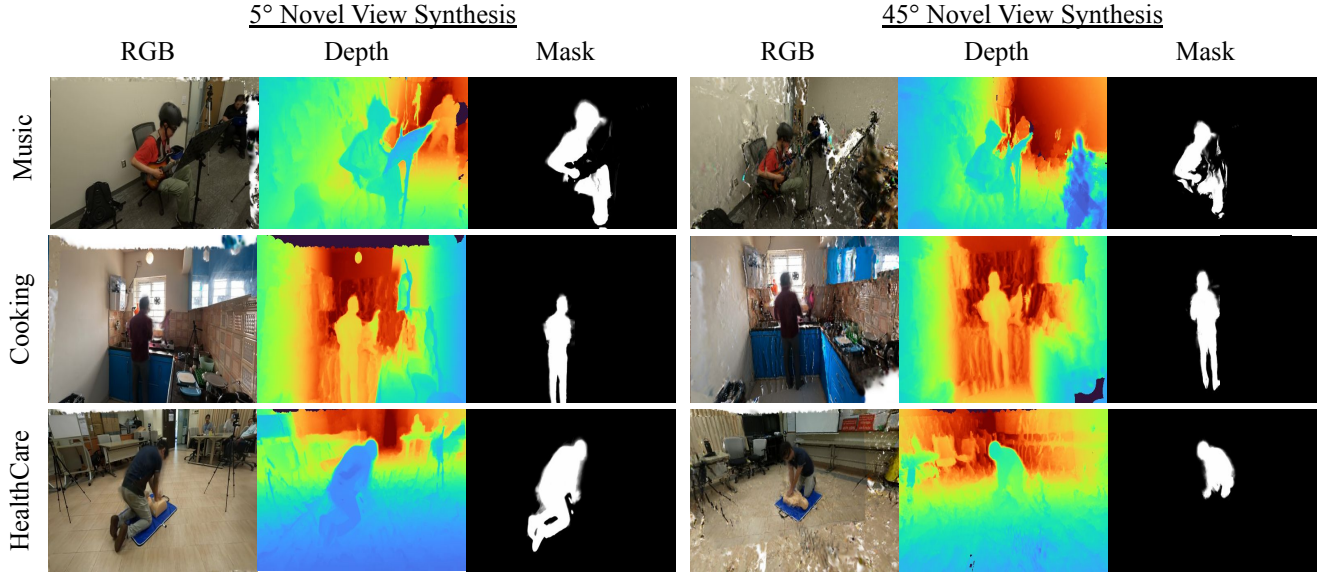


Figure 11. **Novel view synthesis results from more video sequences.** In each row, we visualize the rasterized RGB image, depth map, and foreground mask from our method for various diverse scene including music (top), cooking (middle), and healthcare (bottom). We include results for 5° (left) and 45° (right) novel view synthesis results. Notably, the rendered RGB and depth maps produce consistent reconstructions and plausible geometry.

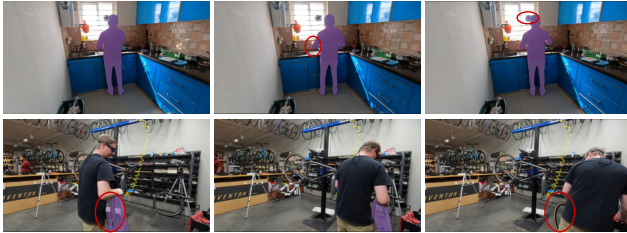


Figure 12. **Failure example of SAM-V2.** We qualitatively inspect the SAM-V2 dynamic foreground masks on the kitchen (top) and bike repair (bottom) scenes. The dynamic mask is highlighted in purple, and failures in dynamic mask estimation are highlighted in red circle. We observe that SAM-V2 can miss important body parts (e.g. the person’s hands) or get confused by the background (as shown in top row). Long-term occlusion will also lead to tracking failure (as shown in bottom row). These failure cases suggest that dynamic mask tracking in complex scenes remains an open challenge.

from image level priors (as in [77]) could be beneficial.

Second, most off-the-shelf feed-forward depth estimation networks are trained on simple scene-level datasets, with few dynamic movers (e.g. people) in the foreground. In practice, we observe that the depth of humans in dynamic scenes is often incorrect when observed from other views (Fig. 13). For example, DUS3R often estimates the depth of a human to be the same as the depth of surrounding walls, causing the human to blend into the background.

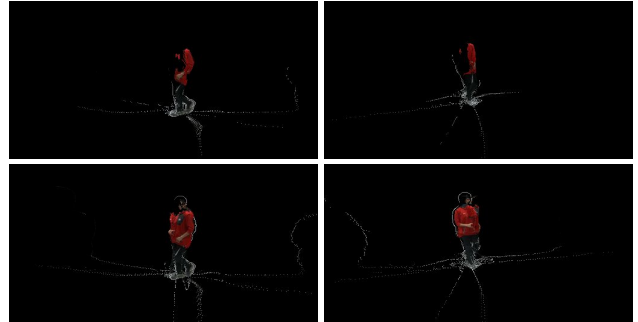


Figure 13. **Visualization of foreground projection for different checkpoints.** Here we show the projection by known cameras and ground-truth foreground masks, using the point cloud from DUST3R (top row) and MonST3R (bottom row) for two selected cameras (each column represents one camera). Notably, although MonST3R is fine-tuned on temporal frame sequences instead of multi-view information, MonST3R benefits from the presence of dynamic foreground movers in its fine-tuning dataset and thus gives a better foreground result.

We believe that these fundamental problems with the depth predictions *cannot* be solved by any alignment in the output space. To mitigate this issue, we plan to further fine-tune DUST3R or MonST3R on existing dynamic human datasets.