WildRayZer: Self-supervised Large View Synthesis in Dynamic Environments

Supplementary Material

Section A details the full WildRayZer training pipeline, including RayZer pretraining, motion-mask learning, masked reconstruction, and joint training with copy–paste augmentation. Section B details the D-RE10K-iPhone benchmark and clarifies evaluation metrics. Section C presents extended qualitative visualizations, including motion-mask comparisons, failure analyses, and additional results on D-RE10K, D-RE10K-iPhone, and DAVIS [55].

A. Training Details

RayZer Pretraining. We first train a RayZer [27] using official release code with scene and pose latent representations. The encoder consists of 12 transformer layers with 8 additional geometry-specific layers, while the decoder uses 12 layers. We use a hidden dimension of $d = 768$ with attention heads of size 64. Both encoder and decoder employ QK normalization for training stability. Images are tokenized into 16×16 patches from 256×256 input resolution. The scene latent code has length 768, and camera poses are represented using 6D rotation representation. Importantly, we find smaller scene latent code yields better rendering quality compared to scene latent code length of 3084 when training with smaller batch size. Additionally, instead of relying solely on absolute positional embeddings, we introduce a dedicated embedding to distinguish input and target views. We also mix cases with 2, 3, and 4 input views during training so that RayZer can handle variable-length inputs without performance degradation.

We train on the RealEstate10K [100] dataset with 2 input views and 6 target views per scene. The model is trained for 100K steps with a batch size of 8 per GPU and gradient accumulation over 2 steps. We use the AdamW [42] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, learning rate of 4×10^{-4}, and weight decay of 0.05. The learning rate is warmed up over 6K steps. Gradient clipping is applied with a maximum norm of 1.0. We use mixed precision training (bfloat16) with TF32 enabled on H100.

The training objective combines L2 reconstruction loss (weight 1.0) and perceptual loss (weight 0.2) computed using VGG features. Input views are sampled with frame distances between 25 and 192 frames, with a scene scale factor of 1.35 applied during data augmentation.

Motion Mask Training Stage. In this stage, we train the motion mask predictor while keeping the pre-trained rendering model frozen. The goal is to learn to distinguish dynamic regions from static using self-supervised pseudo-labels derived from DINOv3 [70] features and SSIM [86].

We generate motion pseudo-labels by computing ground-truth and renderings differences semantically and structurally. For each pair of ground truth and rendering, we extract dense feature maps and compute cosine similarity between corresponding spatial locations. Then we derive a fused error map from SSIM dissimilarity map and DINO dissimilarity map, viewing as saliency map. Then we cluster DINO feature at patch level across input images and assign average saliency score based on the saliency map. Regions with low feature consistency across frames (similarity below a learned threshold) are labeled as dynamic, while consistent regions are labeled as static. We apply morphological operations (kernel size 3, 1 iteration) to expand the mask and filter out small components (area $< 0.25\%$ of image) to ensure coherent motion segments. Lastly, we use grabcut [78] to refine the boundary.

The motion mask predictor is trained on D-RE10K using binary cross-entropy loss between predicted masks and pseudo motion labels. We use a learning rate of 10^{-4} with AdamW optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 0.01) and batch size 32. Critically, we apply a PSNR-based sample filtering strategy: only samples with rendering PSNR > 17 dB are used for mask supervision. This prevents our model to learn from noisy pseudo-labels on challenging dynamic scenes where the frozen renderer struggles. We show detailed process of pseudo motion mask construction in Fig. 3.

Masked Latent Scene Reconstruction Stage. In this stage, we freeze the learned motion mask predictor and train the rendering model to reconstruct static scene content while explicitly providing masks on input images. This stage teaches the model to perform cross-view completion over masked areas. We implement a mask token strategy during training by masking 10% token, similar to standard MAE training but tailored for novel view synthesis. More specifically, we find clustered mask performances better compare to random token masking. The reconstruction loss is computed on the entire image.

Joint training with Copy-Paste Augmentation. In the final stage, we jointly train the motion-mask predictor and the rendering model with synthetic motion augmentation. Specifically, we adopt copy–paste augmentation [16] to insert controlled transient objects into static RE10K scenes, providing explicit supervision for the motion-mask predictor while simultaneously training the renderer to remain robust to these diverse transients. In addition, we mix D-RE10K sequences by masking the predicted motion regions, iteratively refining pseudo motion labels and excluding dynamic areas from the reconstruction loss.

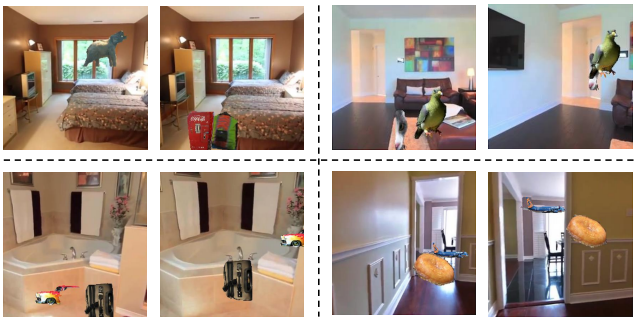

Figure 6. **Examples of Copy–paste mask augmentation.** We inject synthetic transient objects (*e.g.*, animals, household items, vehicles) into static RE10K scenes to simulate dynamic elements in otherwise static environments.

We detail the copy-paste augmentation here. We randomly paste 1-2 COCO objects (animals, vehicles, people) into 50% of training scenes occupying 25-35% of image size. Objects are blended using Gaussian smoothing ($\sigma = 3$) and positioned with 15% margin from image borders. With 80% probability, we use per-view random objects; otherwise, we paste the same object across all views in a sequence. This generates binary overlay masks M_{paste} indicating pasted regions.

In this stage, the motion mask predictor receives supervision from two sources: (1) DINOv3 pseudo-labels M_{DINO} for real dynamic content in D-RE10K, and (2) ground-truth paste masks M_{paste} for synthetic objects in augmented Static RE10K. We use binary cross-entropy loss with PSNR filtering (threshold 17 dB) to ensure supervision quality. The rendering loss combines masked reconstruction on both static regions and pasted regions:

$$\mathcal{L} = \mathcal{L}_{\text{masked}} + \lambda_{\text{mask}} \cdot \text{BCE}(M_{\text{pred}}, M_{\text{target}}), \tag{2}$$

where $M_{\text{target}} = M_{\text{DINO}}$ for dynamic scenes and $M_{\text{target}} = M_{\text{paste}}$ for augmented static scenes, and $\lambda_{\text{mask}} = 1.0$.

If we train only in the final stage, we observe many rendering artifacts due to inaccurate motion-mask predictions during early iterations. Therefore, adopting a progressive, multi-stage training strategy becomes essential for stability and reliable convergence.

B. Benchmarking Details

In this section, we provide additional details of the proposed D-RE10K-iPhone benchmark in Section B.1 and describe our masked image quality metrics in Section B.2. We will release our implementation as a reference.

B.1. D-RE10K-iPhone Benchmark Details

D-RE10K-iPhone comprises 50 real-world sequences featuring humans and vehicles. Each sequence contains 18 paired images captured from the same viewpoint: one *transient* frame with a moving object (*e.g.*, person, car) and one *clean* frame without it. An iPhone mounted on a tripod with a Bluetooth shutter is used to minimize pose differences between paired images. Each sequence require roughly 30 minutes to curate.

To reduce illumination-induced artifacts, we compare static background regions across each pair and discard pairs with noticeable lighting changes (e.g., sudden occlusions or reappearance of direct sunlight). For the v=2, 3, 4 sparse-view settings, input views are selected to span the full range of available camera angles in each sequence, ensuring that we evaluate genuine novel view synthesis rather than near-view interpolation.

B.2. Masked Image Quality Metrics

We compute standard image quality metrics (PSNR [17], SSIM [86], LPIPS [97]) over spatial regions specified by real-valued masks $M \in [0,1]^{H\times W}$, enabling separate evaluation of transient and static scene components.

Masked PSNR. Given a ground-truth image I, prediction $\hat{I}$, and mask M, we compute a masked mean squared error:

$$\text{MSE}_M = \frac{\sum_{i,j}(I_{ij} - \hat{I}_{ij})^2 M_{ij}}{\sum_{i,j} M_{ij}}, \tag{3}$$

$$\text{PSNR}_M = -10 \log_{10}(\text{MSE}_M). \tag{4}$$

This yields a localized measure of pixel-level fidelity within the masked region.

Masked SSIM. We first compute a standard SSIM map $S \in \mathbb{R}^{H'\times W'}$ using an 11×11 Gaussian window ($\sigma = 1.5$) with stability constants $C_1 = (0.01)^2$ and $C_2 = (0.03)^2$ for images in $[0,1]$. Let μ and σ denote local means and variances. The per-pixel SSIM is

$$S_{ij} = \frac{(2\mu_{ij}\hat{\mu}_{ij} + C_1)(2\sigma_{ij} + C_2)}{(\mu_{ij}^2 + \hat{\mu}_{ij}^2 + C_1)(\sigma_{ij}^2 + \hat{\sigma}_{ij}^2 + C_2)}. \tag{5}$$

The mask is downsampled to the SSIM resolution via area pooling (M'), and the masked SSIM is

$$\text{SSIM}_M = \frac{\sum_{i,j} S_{ij} M'_{ij}}{\sum_{i,j} M'_{ij}}. \tag{6}$$

Masked LPIPS. To obtain perceptually weighted masked errors, we use LPIPS in spatial mode, producing per-layer feature difference maps $D^{(\ell)} \in \mathbb{R}^{H_\ell \times W_\ell}$. For each feature level ℓ, we downsample the mask via area pooling to $M^{(\ell)}$ and compute

$$\text{LPIPS}_M = \sum_{\ell} \frac{\sum_{i,j} D^{(\ell)}_{ij} M^{(\ell)}_{ij}}{\sum_{i,j} M^{(\ell)}_{ij}}. \tag{7}$$

This applies perceptual masking consistently across all LPIPS feature scales.

Figure 7. **Motion Masks Comparisons.** We present motion masks comparisons on D-RE10K (row 1) and D-RE10K-iPhone (row 2) across Co-segmentation [1], MegaSAM [36], Segment Any Video [21], VideoCutler [84] and WildRayzer's motion mask predictions.

Implementation Details. All metrics are evaluated per image. For D-RE10K-iPhone's analysis, we report PSNR, SSIM, and LPIPS separately on transient regions (using $M_{\text{transient}}$) and on static regions (using $M_{\text{static}} = 1 - M_{\text{transient}}$). We notice different implementation can lead to different evaluation results, therefore we will release our implementations for these metrics as a reference.

C. Additional Visualizations

In this section, we provide additional visualization: (1) motion mask comparisions with state-of-art methods in Section C.1; (2) present WildRayzer's failure cases in Section C.2; (3) demonstrates additional WildRayzer's results on D-RE10K, D-RE10K-iPhone and DAVIS [55] benchmarks in Section C.3.

C.1. Motion Mask Visualization

We compare several popular motion-mask generators: co-segmentation based on DINO features [1], MegaSAM [36], Segment Any Video (SAV) [21], and the self-supervised instance segmentation model VideoCutler [84]. Even after careful hyperparameter tuning, training-free co-segmentation tends to either over-mask or under-mask large portions of the scene, which severely degrades rendering quality. MegaSAM provides useful cues, but under sparse-view inputs its masks are often noisy and inconsistent across views. SAV can be highly accurate when it selects the correct actor, but under sparse views it frequently segments the wrong object. VideoCutler, by design, returns salient instances rather than true movers, and thus does not distinguish dynamic from static objects; we visualize its first predicted mask for comparison. In contrast, our learned motion masks are the most faithful to true motion boundaries, despite being trained in a fully self-supervised manner without ground-truth motion labels. We demonstrate motion masks with one example from D-RE10K (top row) and D-RE10K-iPhone (bottom row).

C.2. Failure Cases

We illustrate common failure cases in Fig. 8. Because our pseudo motion masks do not enforce instance-level segmentation, they may capture only the moving parts of an object when other parts remain static across the input views. While this behavior can be consistent with the motion cue, it often degrades rendering quality and we treat it as a failure mode. We also observe under-segmentation, where the predicted motion mask is smaller than the true moving region (row 2). Finally, mask quality drops when the moving object occupies a large fraction of the input images (rows 3–4). Note we see large pose extrapolation as future works.

C.3. Additional Visualizations

We provide 12 additional qualitative examples to illustrate the behavior of WildRayZer beyond the main paper results in Fig. 9. The first three rows depict challenging dynamic indoor scenes from D-RE10K; in these cases, WildRayZer successfully removes transients while plausibly complete occluded structure across input views. The next two rows demonstrate cross-dataset generalization on unseen DAVIS sequences [55], showing that the learned motion masks and masked rendering transfer to videos with different content and capture conditions. The final row presents real-world examples from D-RE10K-iPhone under casual handheld capture, highlighting that our model maintains sharp static geometry and clean background reconstruction.

Figure 8. **Failure Cases.** Each row contains one failure case of WildRayzer. Motion mask only highlights moving parts dispite highlight television, it only highlights part of human body in row 1. Moreover, motion mask may miss small places such as feet in row 2. Row 3 and 4 show failure case when transient object is too big and predicted masks are smaller.

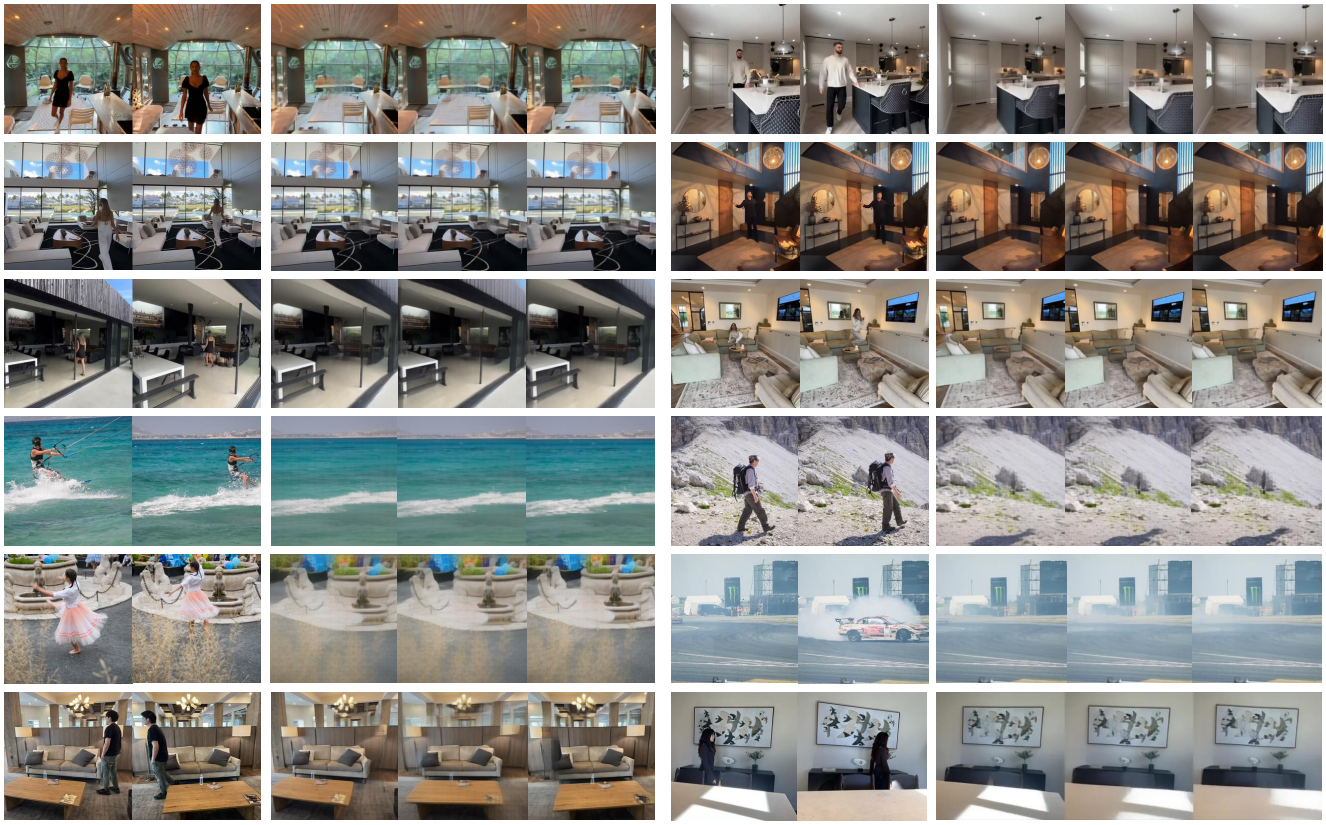

Figure 9. **Additional qualitative results.** We show 12 extra examples to illustrate WildRayZer's behavior across datasets. The first three rows are from D-RE10K, the next two rows demonstrate generalization to the unseen DAVIS dataset [55], and the last row shows additional real-world results on D-RE10K-iPhone.