Supplementary Material : Scaling Transformer-Based Novel View Synthesis Models with Token Disentanglement and Synthetic Data

Nithin Gopalakrishnan Nair[1*] Srinivas Kaza[2*] Xuan Luo[2] Vishal M. Patel[1]
Stephen Lombardi[2] Jungyeon Park[2]
[1] Johns Hopkins University [2] Google
{ngopala2,vpatel36}@jhu.edu {srinivaskaza,xuluo,salombardi,jungyeonp}@google.com
https://scaling3dnvs.github.io

1. Design choices

We provide further details of the exact transformer model used here. **Transformer blocks** We find the claims regarding the naive transformer architecture to be unstable for image generative tasks to be true. We use QK-Norm to stabilize the transformer block. We use 24 transformer blocks with an embedding dimension of 1024. In addition to this, different from LVSM, we use attention biases at all layers and include the bias for the last transformer block, as we find this design choice particularly stable with linear learning rate decay. We use a patch size of 8 for all experiments.

1.1. Enhancing 3D generative models for 3D consistent generation

The use of diffusion models has been widely explored for generating 3D scenes. Multiple works in the literature adapt pretrained text-to-image and image-to-video models for 3D-consistent scene generation. Most of these works condition the diffusion model on camera parameters and learn the conditional distribution of multiple views given the camera poses. Given the ability to cherry-pick and sample through the diffusion model multiple times, these models produce high-quality results. However, existing 3D scene generation models cannot mass-produce synthetic data for fine-tuning substream models for high-fidelity generation. Until now, no generalizable models with high-fidelity results have been proposed that can directly utilize the data generated by diffusion models. We argue that this drawback is caused by a lack of analysis of the inference-time generation process of diffusion models. Although extensive studies have been performed on different training strategies for 3D-consistent generation using diffusion models, much less effort has been put into improving inference-time generation quality.

Most 3D generative models generate N views of a scene, each of dimension $(H \times W \times C)$, in parallel to preserve 3D consistency. The generation process starts with random isotropic Gaussian noise of dimension $N \times H \times W \times C$, which undergoes a diffusion process of T steps. This either converts it into a latent representation, which is then decoded by a VAE decoder to produce multiview images, or generates images directly. These multiview images are further used to train a NeRF or a Gaussian Splat model to generate novel views of the scene. When the diffusion model generates high-quality, 3D-consistent images, this framework works perfectly. However, in reality, diffusion models are sensitive to input noise. Even for the simple case of image generation, different noise inputs produce different quality results. Recent works have shed light on inference-time scaling laws for generation, claiming that the quality of diffusion model outputs can be controlled by selecting the correct input noise via rejection sampling. Similar claims have been made for video generation models, where performance improves significantly by refining the input noise schedule.

To understand this, consider a toy example: Suppose we want to generate an image (I_1) using the diffusion model conditioned on a text prompt. Starting with Gaussian noise N_1, if we want to generate another image (I_2) close to (I_1), the desired noise is most likely closer to N_1. Previous works have demonstrated enhanced video generation results by selecting starting noises that are close across different frames.

In our case, we use the image-to-multiview variant of CAT3D as the base model for generating multiview images. For choosing the initial noise, we follow a specific heuristic. Specifically, we ensure that the noise across different views remains 3D-consistent. CAT3D is a multiview diffusion model that generates eight views simultaneously, conditioned on the camera poses. CAT3D allows conditioning on a particular view to generate the remaining views. Given the view to be conditioned, we select a random noise for this view, denoted as V_1, with its noise represented as N_1 and the corresponding rotation-translation matrices denoted as R_1, t_1. To estimate the starting noise for other views, we perform a warping operation on N_1, denoted by:

$$N_i = warp(N_1, inv([R_i, t_i])[R_1, t_1]) \tag{1}$$

where the warp operation transforms the coordinates of N_1 to N_i. However, we noticed that such a warping operation fails in regions outside the scene. To handle these cases while enhancing consistency, we marginally modify the noise. Specifically, for these cases, we assign the noise as:

$$N_2 = \alpha N_1 + (1 - \alpha)\mathcal{N}(0, I) \tag{2}$$

Thus, our effective starting noise is defined as:

$$N_{final} = \begin{cases} N_1, \text{overlapping regions} \\ N_2, \text{non overlapping regions} \end{cases}$$

We perform the effective noising operation parallely with respect to the reference view. First we take view 1, warp to view 2. then add noise, then we Although we use CAT3D, our method is generalizable across any 3D scene generation model.

Understanding the value that synthetic data from generative models can bring, we propose a method to enhance diffusion-based 3D generative models to produce high-quality, 3D-consistent results.

1.2. Loss functions

Similar to LVSM, we utilize Mean Square Error (MSE) loss for training our network. Instead of using Perceptual Loss, we utilize LPIPS loss for training. Given the ground-truth target view of dimension $\hat{I} \in \mathbb{R}^{H \times W \times C}$ and the reconstructed target view I, the effective objective function used for optimization is defined as:

$$L = MSE(I, \hat{I}) + \lambda \cdot LPIPS(I, \hat{I}) \tag{3}$$

where λ is a scaling factor set to 0.5 for all experiments.

1.3. Emergent Properties

One surprising **emergent property** of our newly proposed transformer block is its ability to disentangle the source and target tokens, which allows it to scale better for synthetic data compared to a naive transformer block. We present these results in Figure X, where we observe significant improvements. We hypothesize that this emergent property arises because synthetic data is generally prone to artifacts and out-of-distribution noise. When transformer blocks cannot distinguish between source and target tokens, the model learns using both real and synthetic data, leading to reconstructions that inherit these artifacts. However, in our case, only the relevant information from clean images is used for backpropagation, allowing the model to utilize useful context from synthetic data while discarding artifacts during token fusion for target view generation.

2. Limitations

Our model struggles when regions occluded in the source images become visible in the target view. As shown in **??**, when a new object enters the scene, the model hallucinates the affected region. We argue that this phenomenon is inherently ill-posed and lacks a definitive solution. Additionally, the model uses a token size of 8 for all blocks, resulting in 1024 tokens per source image, which demands significant memory. We leave further architectural optimizations, such as hierarchical transformers and more efficient networks like linear attention and state-space models (e.g., Mamba [1], [2]), for future work.

3. Failure cases of our method

We notice that our method contains two main failure modes (1) when an new object comes into the view in between the conditioned frames. (2) When too many shiny artifacts are present in the image

References

[1] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. 2

[2] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model, 2024. 2

Figure 1. **Figure illustrating results from DL3DV dataset trained with our synthetic data**. The first 2 images represent the input views. third presents results of LVSM, Fourth represents our results and fifth the ground truth

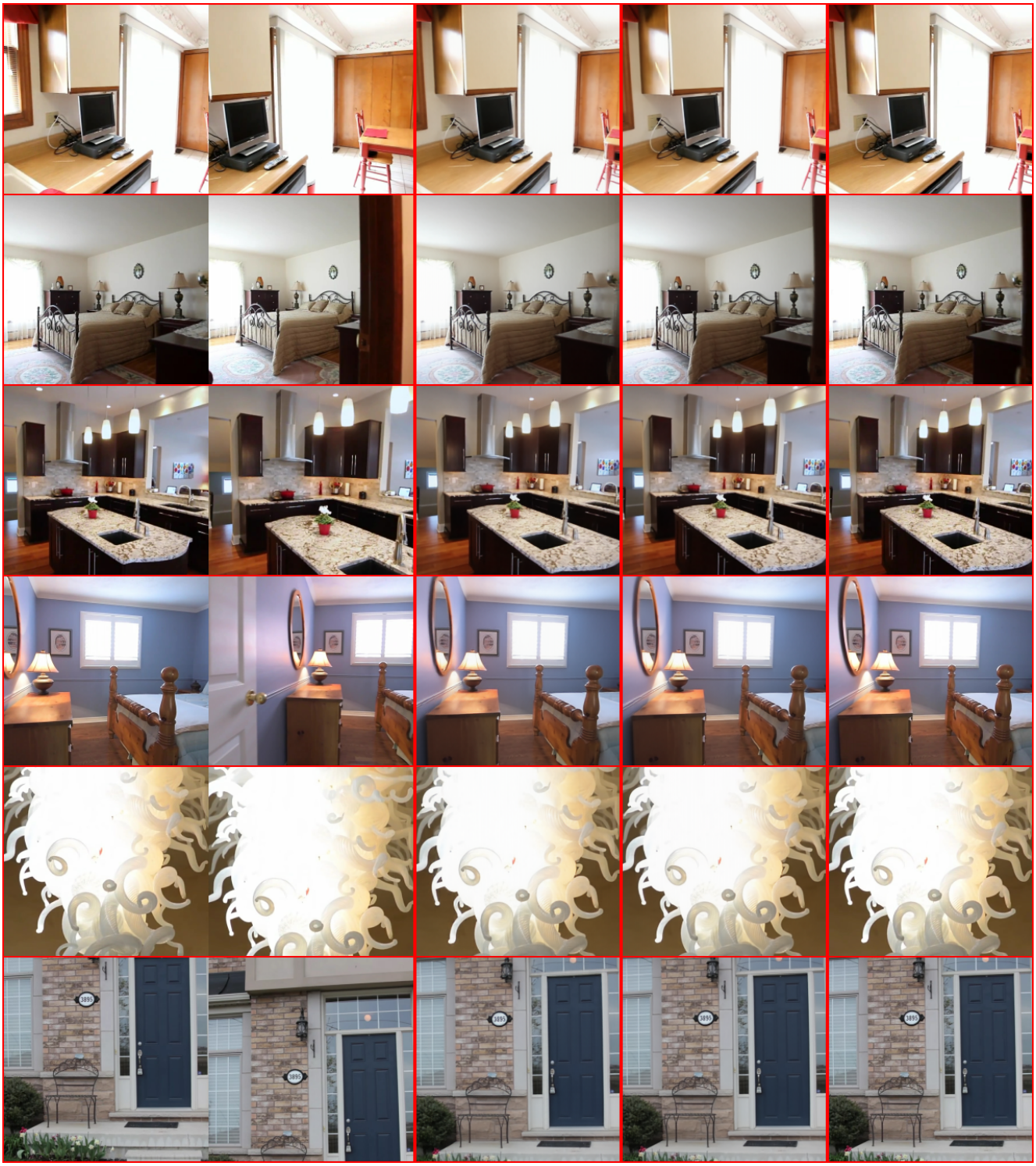

Figure 2. **Figure illustrating results from Re10k dataset trained with our synthetic data**. The first 2 images represent the input views. third presents results of LVSM, Fourth represents our results and fifth the ground truth

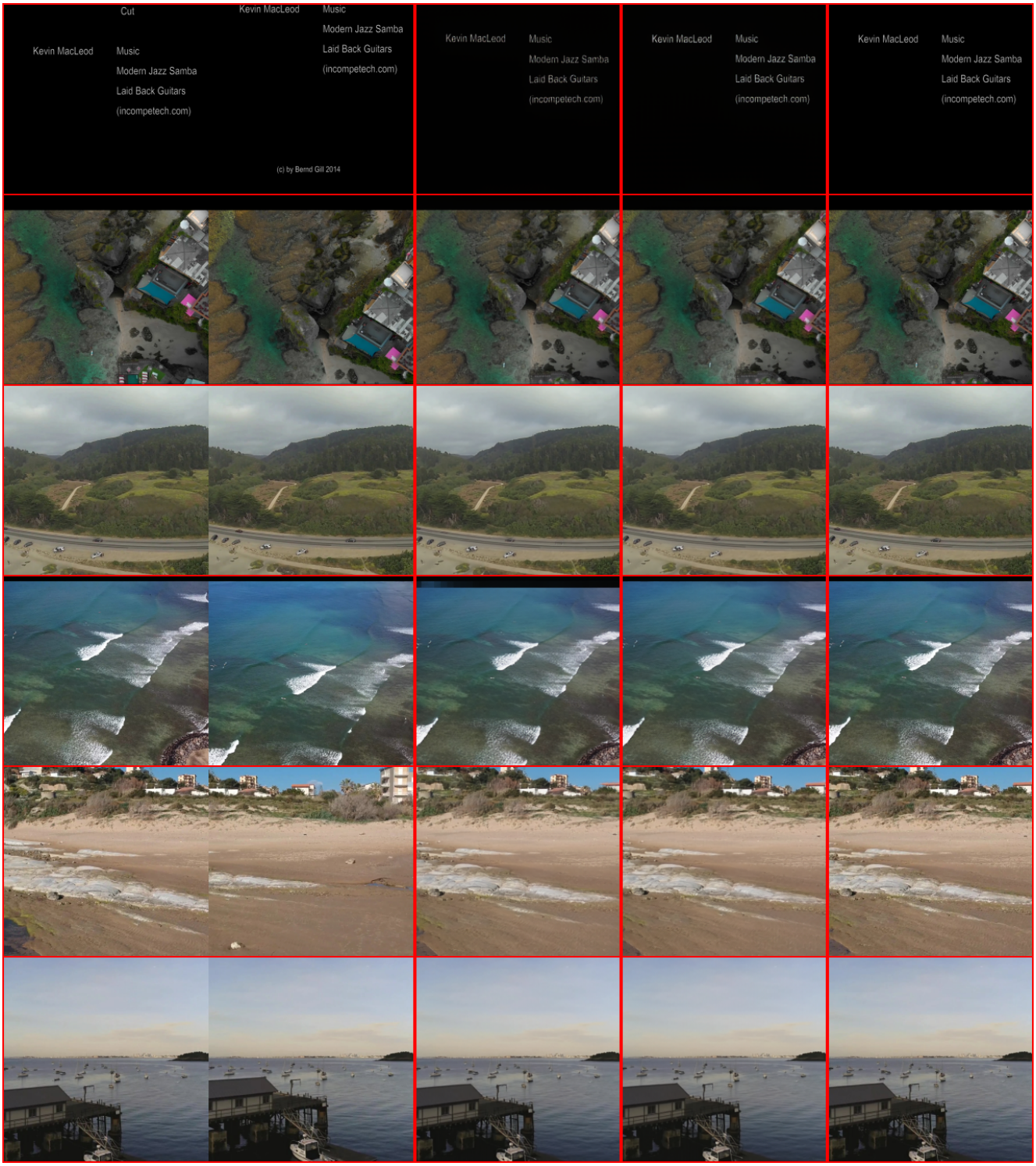

Figure 3. **Figure illustrating results from ACID dataset trained with our synthetic data**. The first 2 images represent the input views. third presents results of LVSM, Fourth represents our results and fifth the ground truth

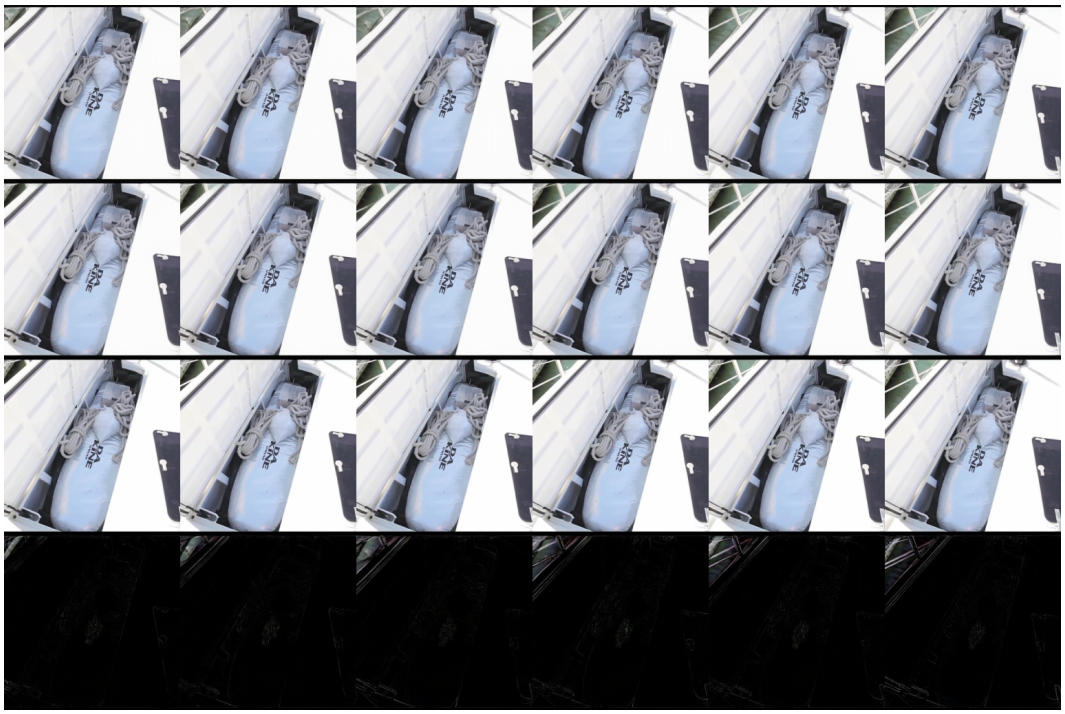

Figure 4. **Figure illustrating the regions where our method works better than LVSM for Re10K dataset**. The figures are in the order Row 1:- LVSM, Row 2:- OURS Row 3:- GT, Row 4:- Difference between LVSM and Ours

Figure 5. **Figure illustrating the regions where our method works better than LVSM for Re10K dataset**. The figures are in the order Row 1:- LVSM, Row 2:- OURS Row 3:- GT, Row 4:- Difference between LVSM and Ours

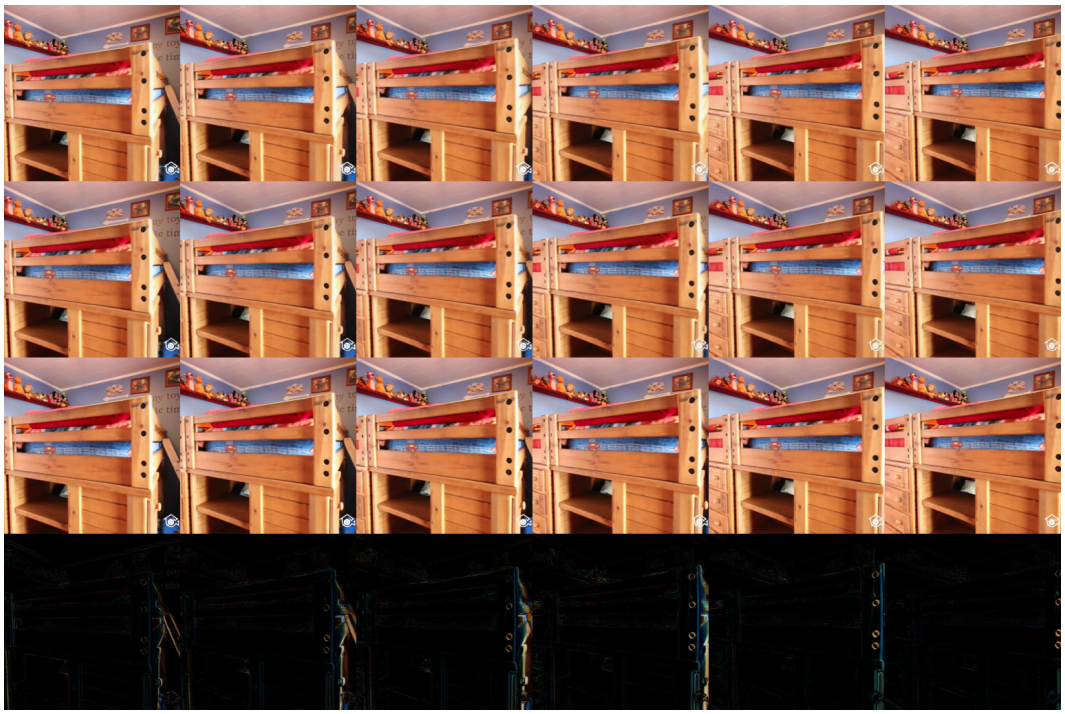

Figure 6. **Figure illustrating the regions where our method works better than LVSM for Re10K dataset**. The figures are in the order Row 1:- LVSM, Row 2:- OURS Row 3:- GT, Row 4:- Difference between LVSM and Ours

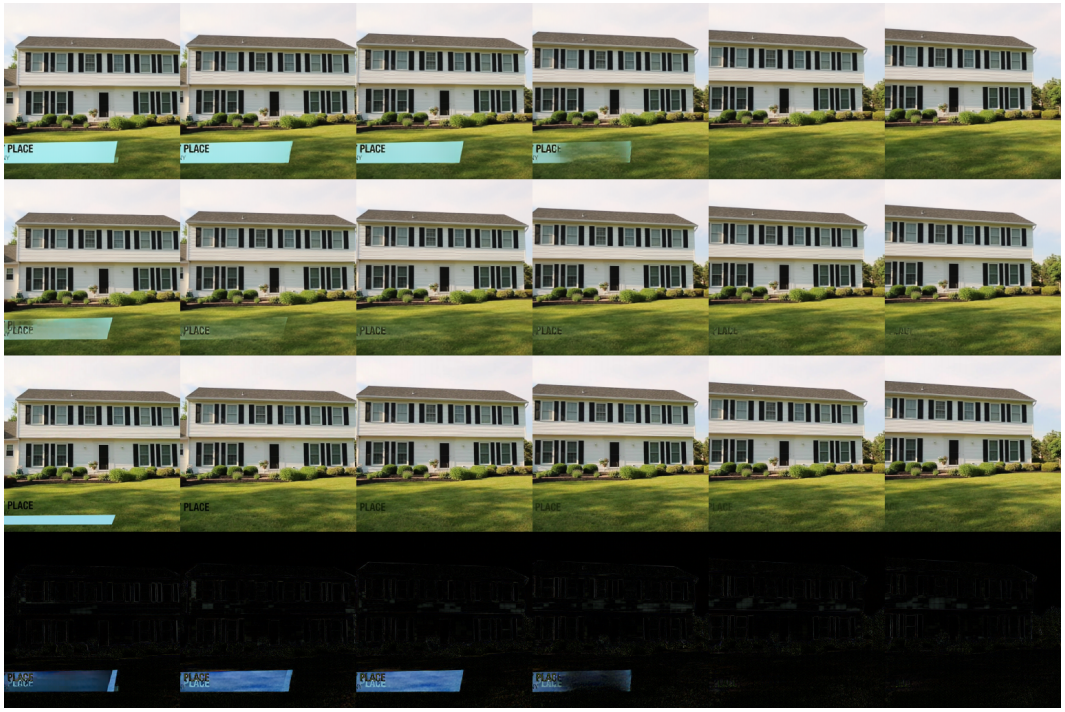

Figure 7. **Figure illustrating the regions where our method works better than LVSM for Re10K dataset**. The figures are in the order Row 1:- LVSM, Row 2:- OURS Row 3:- GT, Row 4:- Difference between LVSM and Ours

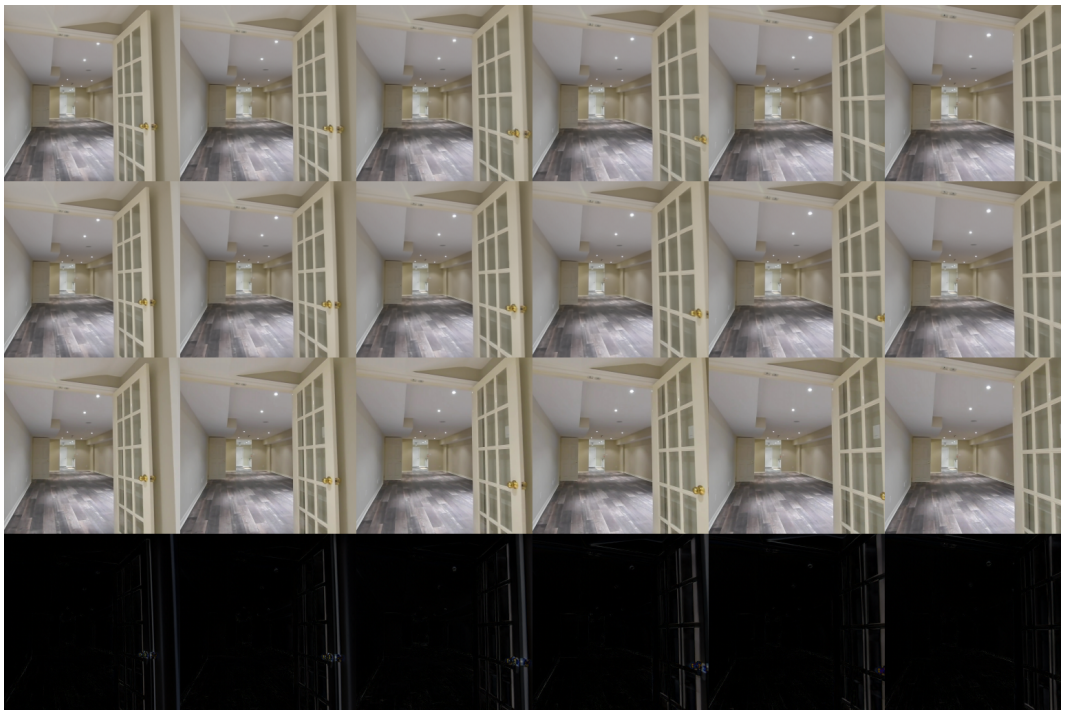

Figure 8. **Figure illustrating the regions where our method works better than LVSM for Re10K dataset**. The figures are in the order Row 1:- LVSM, Row 2:- OURS Row 3:- GT, Row 4:- Difference between LVSM and Ours

Figure 9. **Figure illustrating the regions where our method works better than LVSM for Re10K dataset**. The figures are in the order Row 1:- LVSM, Row 2:- OURS Row 3:- GT, Row 4:- Difference between LVSM and

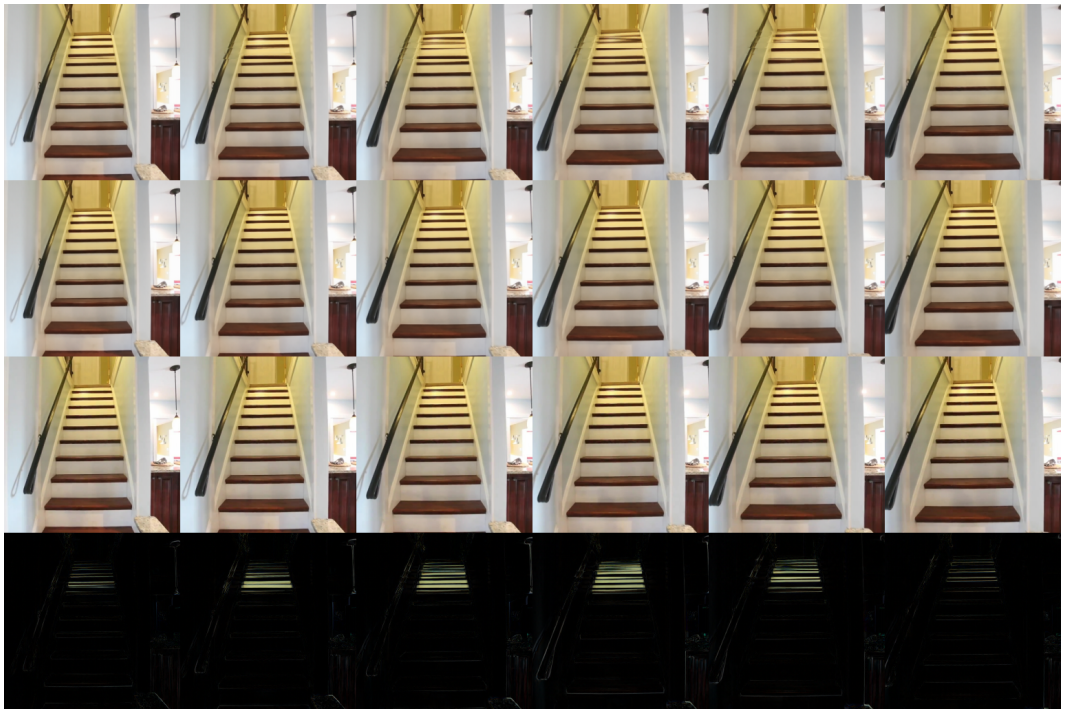

Figure 10. **Figure illustrating the regions where our method works better than LVSM for Re10K dataset**. The figures are in the order Row 1:- LVSM, Row 2:- OURS Row 3:- GT, Row 4:- Difference between LVSM and Ours

Figure 11. **Figure illustrating failure cases of our method**. Our method fails to perform well if there are occluded objects coming into the scene. Figure ordering is OURS, GT, DIFF

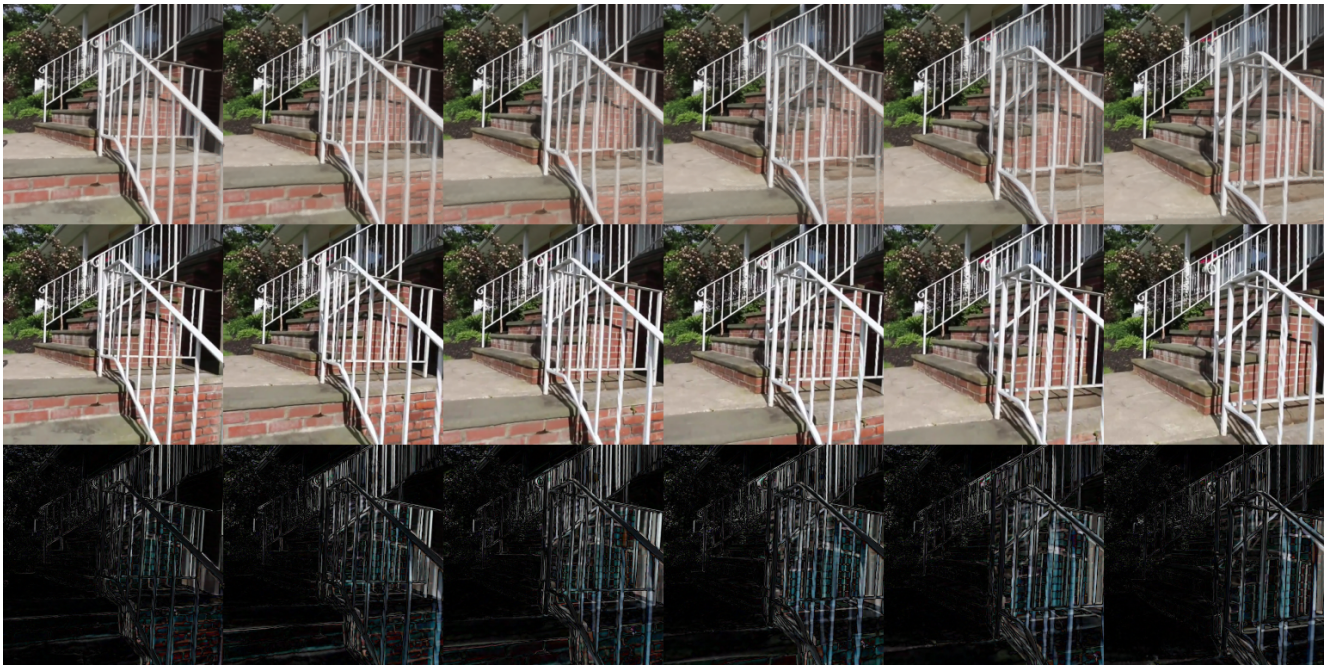

Figure 12. **Figure illustrating failure cases of our method**. Our method fails to perform well if there are occluded objects coming into the scene. Figure ordering is OURS, GT, DIFF

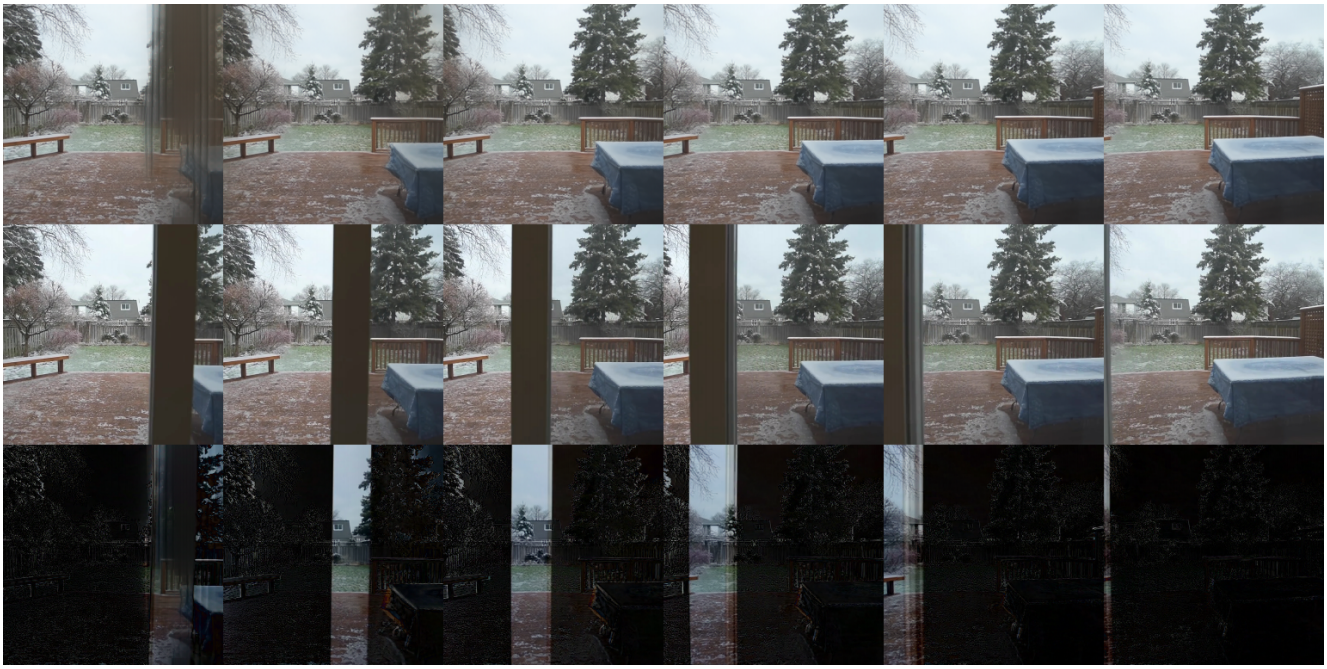

Figure 13. **Figure illustrating failure cases of our method**. Our method fails to perform well if there are occluded objects coming into the scene. Figure ordering is OURS, GT, DIFF

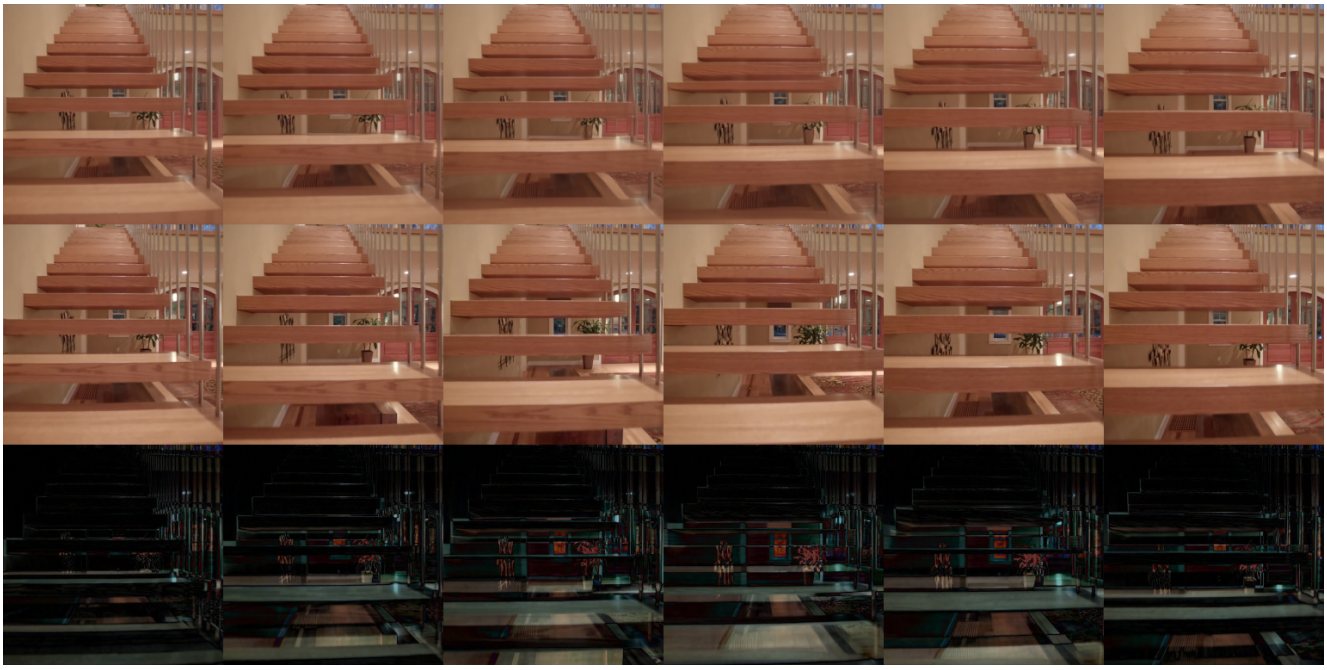

Figure 14. **Figure illustrating failure cases of our method**. Our method fails to perform well if there are occluded objects coming into the scene. Figure ordering is OURS, GT, DIFF

Figure 15. **Figure illustrating failure cases of our method**. Our method fails to perform well if there are occluded objects coming into the scene moreover, our method also fails to reconstruc tproperly when there are some shiny obejcts in the scene. Figure ordering is OURS, GT, DIFF