Stepper: Stepwise Immersive Scene Generation with Multiview Panoramas

Supplementary Material

A. Overview

This supplementary material will provide **additional results** in Appendix B, including several video results, a user study, and further ablation studies. We also provide more details regarding the **technical implementation** in Appendix C, the **datasets** in Appendix D, and the **evaluation protocol** in Appendix E.

B. Additional Results

B.1. Video Results

To highlight the differences between our method and the baselines, we generate the same scene with every method and render trajectories from it. These renderings are provided on our project page at fwmb.github.io/stepper.

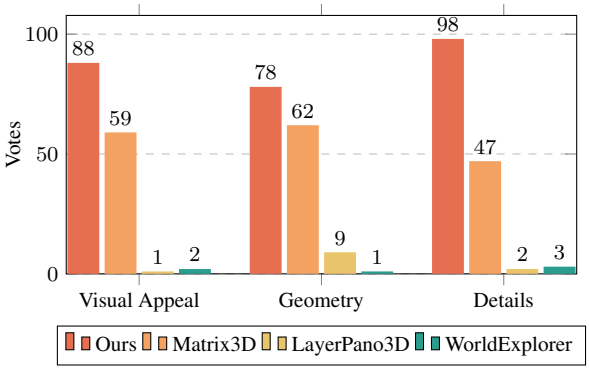
B.2. User Study

Additionally to our quantitative evaluation, we also perform a user-study. Here, $n = 10$ participants, who were not familiar with the project, are shown five scenes with three different camera trajectories each (15 samples). For every sample, we randomly arrange our method and the three baselines. The participants then pick the best video for the three categories 1) *most visually appealing*, 2) *most accurate geometry*, and 3) *most details*. A screenshot of the study form can be seen in Fig. 5.

We collected 150 votes per category and report the results in Tab. 1, which support our qualitative and quantitative findings from the main paper. Overall, our method clearly produces the most appealing scenes. Matrix-3D is a strong competitor, however, it falls short especially on the *most details* category. This aligns with our insight that panorama video models cannot operate on resolutions high enough to obtain sharp results. The other two baselines LayerPano3D and WorldExplorer are far behind. We observe that for LayerPano3D, the layering often fails to properly disentagle the context of the scene, leading to strong blending artifacts. WorldExplorer generally struggles to retain scene consistency when moving further away from the origin, resulting in distorted images and Gaussian artifacts. In general, the user study confirms the results from our qualitative and quantitative evaluation in the main paper and further proves the efficiency of our new method.

B.3. Ablations

Auto-regressive stepping. To test the robustness of our model, we auto-regressively apply our model for a large number of steps and visualize the results in Fig. 2. As can be seen, our model is able to predict meaningful novel views even for further away steps. However, after the third or fourth step, the visual quality starts to degrade slightly and artifacts can appear. For our main experiments, we opt to perform $n = 2$ steps in every direction (front, right, back, left), as this ensures maximum quality of our generated novel panorama views, while covering a large area. We believe that enabling the model to do even more steps would be a promising direction for future work.



Votes	Visual Appeal	Geometry	Details
WorldExplorer	2	1	3
LayerPano3D	1	9	2
Matrix3D	59	62	47
Ours	**88**	**78**	**98**

Table 1. **User study results.** $n = 10$ participants cast votes on the most *visually appealing*, *most accurate geometry* and *most details* categories for 15 video comparisons, yielding 150 votes per category in total.

Effectiveness of MapAnything. The three baselines methods that we compare against rely on a monocular depth predictor to obtain initial scene geometry from the initial panorama. In contrast, we first generate several novel panorama views using our model, and then apply MapAnything as our reconstruction model. To test whether the usage of multiple panorama views has an effect on the reconstruction capabilities of MapAnything, we test it in different configurations, as can be seen in Fig. 1. When applying MapAnything only on a single cubemap (a), MapAnything fails to properly align the pointmaps of the different views. Sampling more views from the initial panorama (b) helps, but there are still several problematic parts (e.g. bushes in the back). Multiple cubemaps from panoramas from auto-regressive stepping (c) provide a strong stereo signal and result in very consistent and accurate geometry reconstruc-

Our stages (single step = 30s)		*Ours vs Baselines*	
1. Gen. $n = 8$ novel panos	$n\times$30s	WorldExplorer	7h
2. MapAnything	20s	Matrix-3D	40min
3. Filtering	2min	LayerPano3D	32min
4. 3DGS optimization	8min	**Ours**	**15min**

Table 2. **Inference time - Panorama to 3D scene.**

tion. However, we find that MapAnything struggles with textureless regions with cameras facing downwards. This is fixed by again sampling more views from the different generated panoramas (d).

Runtime analysis. While our pipeline is not optimized for inference speed yet, it is significantly faster than the baselines, as shown in details in Tab. 2. There are several possible optimizations: *1)* The 3DGS optimization can be replaced by a feed-forward model like AnySplat. *2)* The panorama generation model currently does 50 diffusion steps. Using improved solvers and distillation, this can be reduced significantly. *3)* Scene expansion could be done *on-demand*, only generating novel panoramas in areas where needed.

C. Technical Details

For all of our experiments, we use the same multi-view panorama generation model. It was obtained via finetuning a pre-trained CubeDiff model, which itself is based on a variant of the popular Latent Diffusion Model architecture. The finetuning uses the default diffusion loss for 90000 steps. The learning rate is set to $8e^{-5}$, with a linear ramp-up from 0 during the first 10000 steps. During finetuning and inference of the model, we remove all text-conditioning. Generating a single novel-view panorama with 50 diffusion steps takes around 30 seconds. For MapAnything, we rely on the commercial checkpoint under Apache license. To improve the visual quality of the renderings, we add a skybox texture to the 3DGS scene where MapAnything predicts infinite distance.

D. Datasets

Training data. As described in the main paper, we adapt the Infinigen framework to procedurally generate synthetic multi-view panorama data, as can be seen in Fig. 3.

Indoor scenes are rooms of different categories (livingroom, bathroom, bedroom, kitchen, *etc*.) and require about 1 hour to generate on a standard CPU machine. Outdoor scenes are also sampled from different biomes (river, mountains, forest, cave, arctic, *etc*.). However, due to them requiring more, complex assets like trees and bushes, their generation takes up to 3 hours on CPU. In total, we generate 4730 indoor and 3252 outdoor scenes

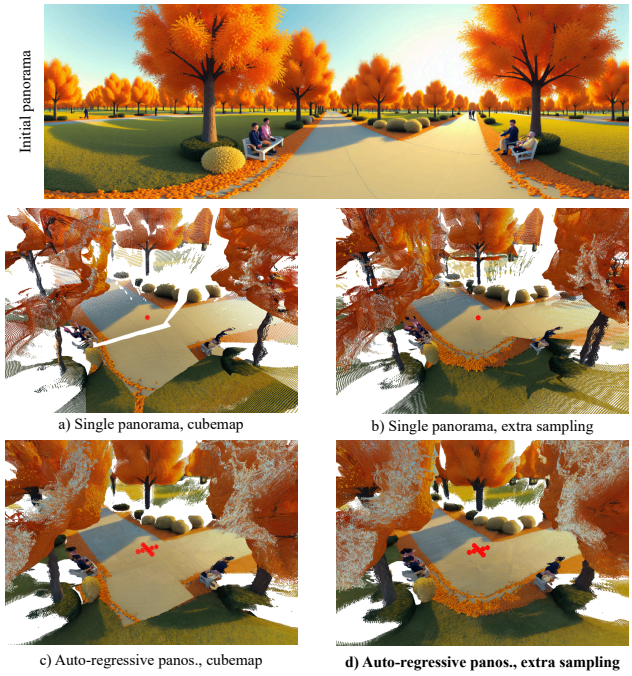


Figure 1. **Impact of Multi-View Panoramas** We depict the effect of using various options of panorama input for MapAnything and found that the auto-regressive expansion and sampling yield the most complete and accurate outputs.

During the generation process of a scene, a number of positions are sampled for panoramic cameras. For an indoor scene, the pipeline places up to $n = 20$ cameras at random locations in the room. To reduce the numbers of assets that have to be generated to cover the bigger outdoor scenes, we only place $n = 4$ cameras. Sometimes, the placement algorithm can only find a smaller number of cameras. For every placed camera, we render its view and the views $d = 0.25m$ to the right and to the left. The two pairs [[left, center], [center, right]], as well as the mirrored pair [[center, left], [right, center]], where we rotate the panorama by $180°$, are rendered. Therefore, every placed camera yields $m = 4$ pairs of left-right panoramas with our desired baseline. In total, we generate 187156 indoor and 49372 outdoor panorama pairs. Due to the high resolution of 4096×2048 and raytracing, the rendering process takes around 70 seconds per panorama on an A100 GPU.

Testing data. We curate a test set of 16 synthetic scenes, for which we render an initial panorama and eight novel-view panoramas from different locations at a resolution of 4096×2048. For each panorama, we also compute the corresponding depth map. The scenes are obtained using Blender (6), Infinigen indoors (5) and Infinigen outdoors (5) and are visualized in Fig. 4. For the Blender scenes, we adapt the following sources, which are all under Cre-

ative Commons licenses: Arctic ships[1], Barbershop[2], Classroom[3], Mountain cabin[4], Valley[5], Blue room[6].

Qualitative examples. We rely on a number of qualitative examples from other papers[7][8] to demonstrate the generalization capabilities of our method.

E. Evaluation Details

Setup. For every test scene, we generate a 3DGS scene with every baseline. As a first step, we transform every scene such that its scene origin and rotation line up with the ground-truth scene. This can be achieved using the camera configurations from the individual reconstructions. Then, we align the scale of every scene with the scale of the ground-truth scale. To this end, we render a depth map in all four horizontal directions with a 90 degree field-of-view and a resolution of 1024×1024 for the initial panorama (at the origin of the scene). After filtering out invalid depths, the scene scale is determined via the median of all scales between the rendered and ground-truth depths. Our scenes tend to contain around 2.5×10^6 Gaussians, Matrix-3D's scenes around 10×10^6 Gaussians, LayerPano3D's scenes around 3×10^6 Gaussians, and WorldExplorer's scenes around 0.4×10^6 Gaussians.

Rendering. Using the alignment and scaling from above, we render the six cubefaces for each of the 9 panorama (initial + 8 novel-view panoramas) with a field of view of 90 degrees and a resolution of 1024×1024 once from the 3DGS scene and once from the ground-truth panoramas using resampling. The 3DGS rendering is performed using the open-source `gsplat` library. For every rendered cubeface (54 in total), we compute the NVS metrics SSIM, LPIPS, and PSNR and report the average.

[1] https://download.blender.org/archive/gallery/blender-splash-screens/blender-3-2/

[2] https://svn.blender.org/svnroot/bf-blender/trunk/lib/benchmarks/cycles/barbershop_interior/

[3] https://www.blender.org/download/demo-files/

[4] https://www.blenderkit.com/get-blenderkit/28cb035a-d333-4e7f-a2b5-543629cdd982/

[5] https://www.blenderkit.com/get-blenderkit/28cb035a-d333-4e7f-a2b5-543629cdd982/

[6] https://blog.polyhaven.com/blue-wall-scene-file/

[7] https://ys-imtech.github.io/projects/LayerPano3D/

[8] https://katjaschwarz.github.io/worlds/

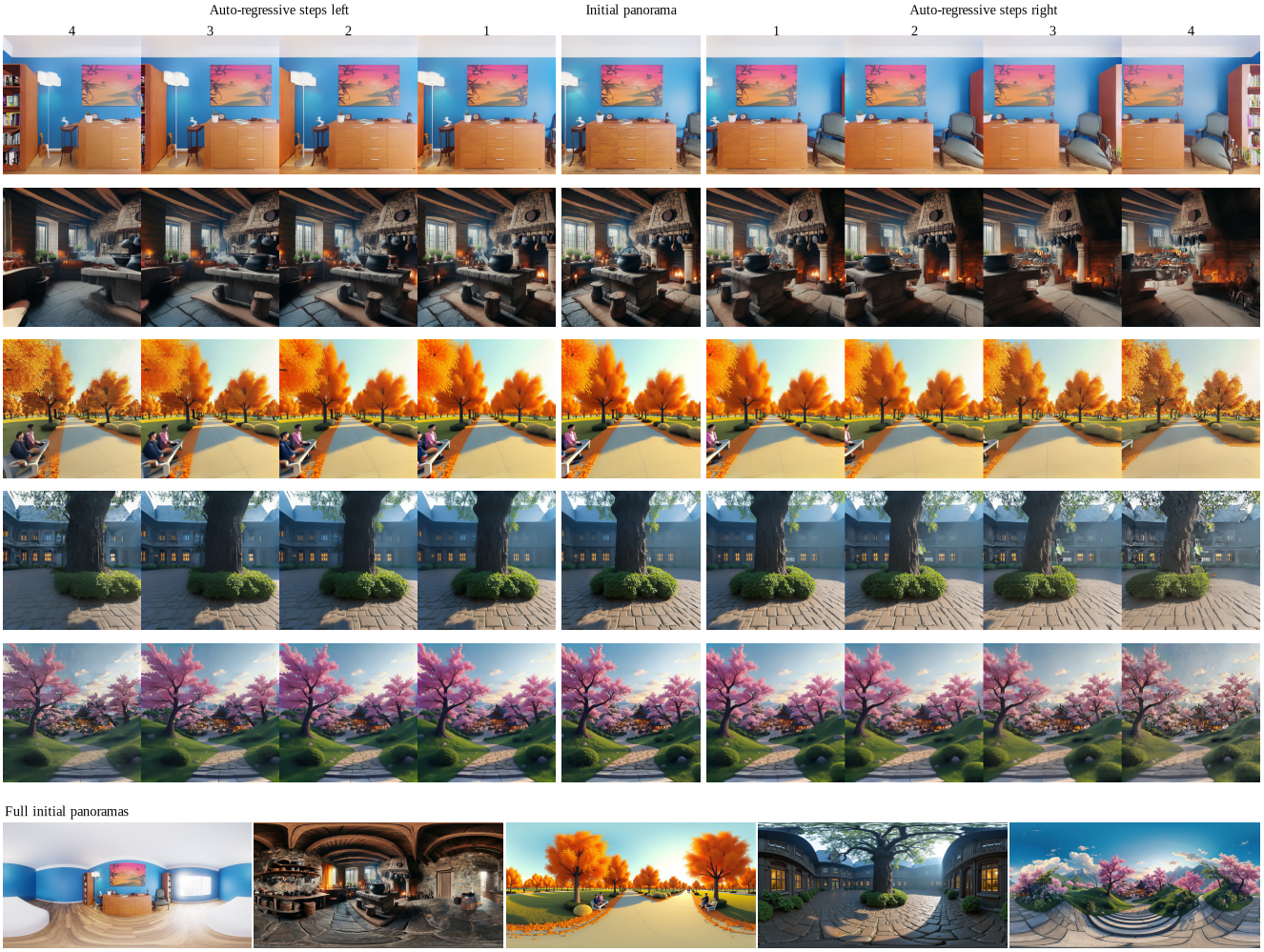


Figure 2. **Auto-regressive steps.** We perform a number of auto-regressive steps to generate novel-view panoramas from the initial panorama to the left and right for several examples. From the generated panoramas, we visualize the forward-facing view to highlight the generated details. A single step corresponds to a step length of $0.25m$.

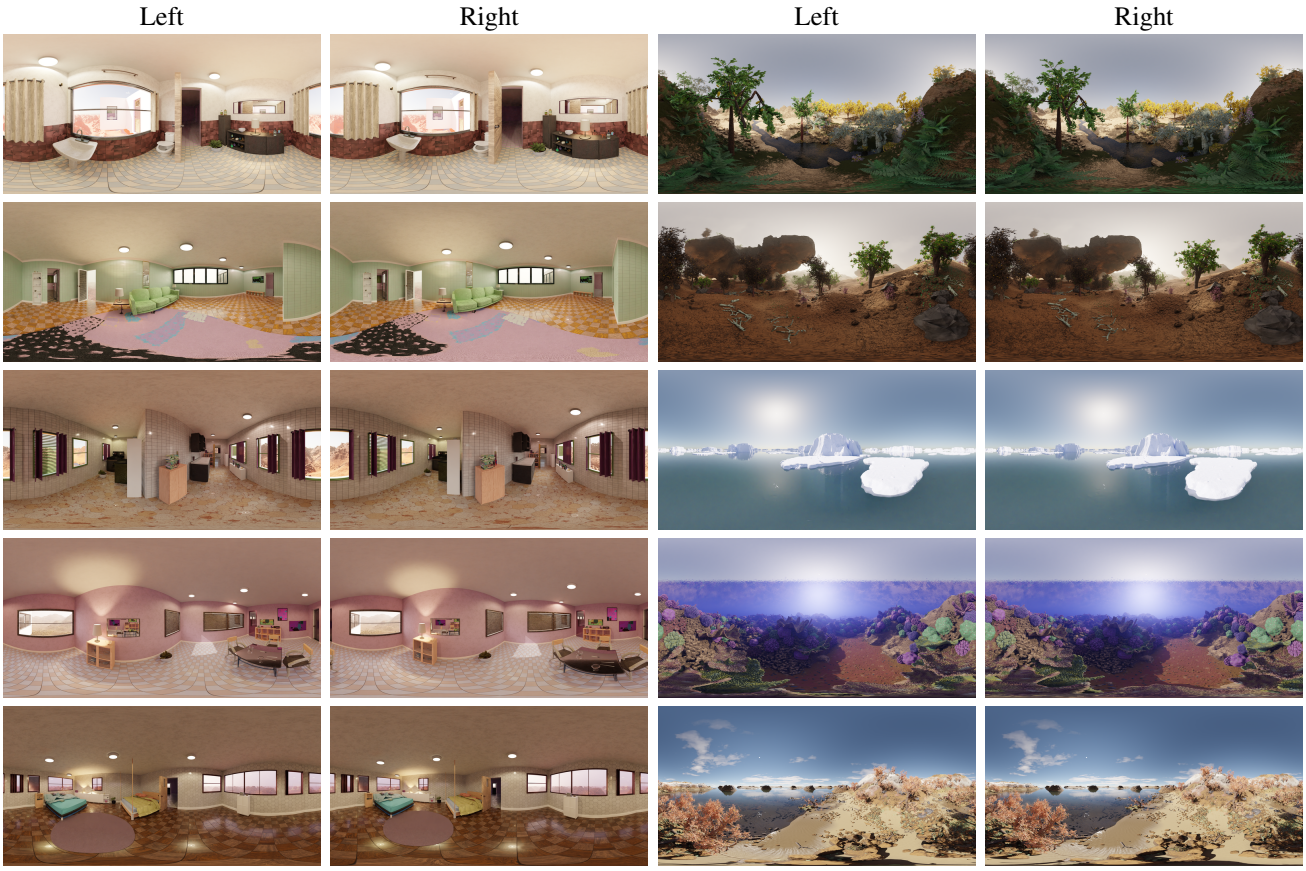


Figure 3. **Multi-view panoramas pairs generated with Infinigen.**

Figure 4. **Input panoramas used for testing.**

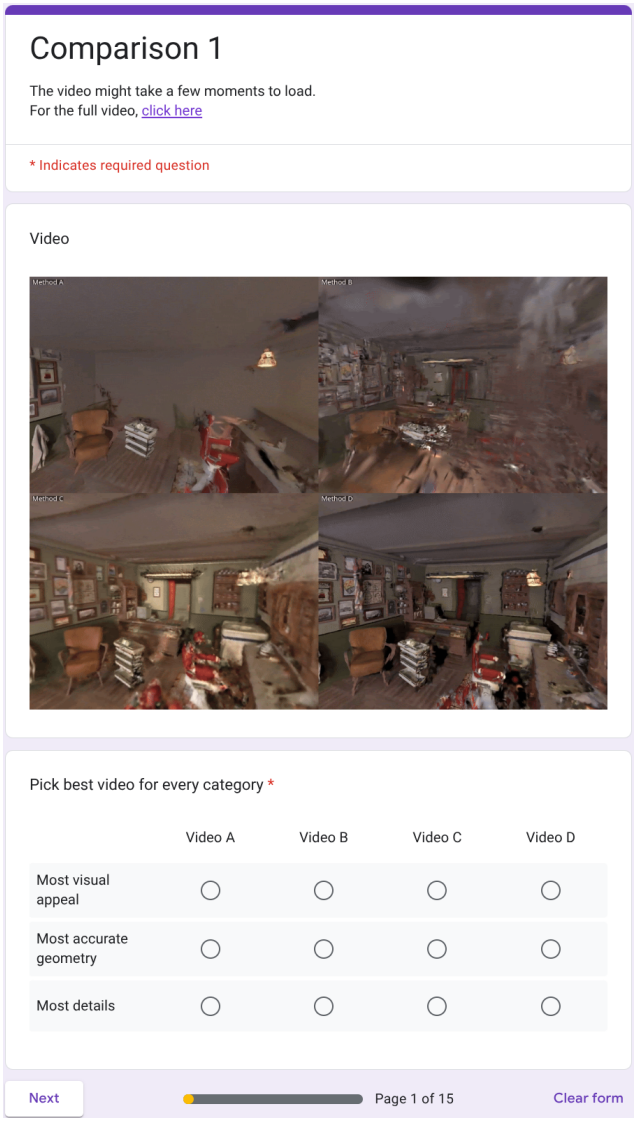


Figure 5. **User study form.** Participants fill out this randomized form and cast a single vote per category.