Supplementary Material for Voyaging into Perpetual Dynamic Scenes from a Single View

Fengrui Tian Tianjiao Ding Jinqi Luo Hancheng Min René Vidal
University of Pennsylvania
{tianfr,tjding,jinqiluo,hanchmin,vidalr}@upenn.edu

We organize the supplementary material as follows.
• §1 provides an additional video for better visualization of the generated dynamic scenes, implementation code of DynamicVoyager, and a table of notations used in the paper.
• §2 presents more implementation details of the proposed ray outpainting model and 4D point cloud reconstruction.
• §3 introduces the details of the background completion strategy.
• §4 presents more scene generation results and details of camera-controllable video diffusion model MotionCtrl [4].

1. Video, Code, and Notation Table

We encourage readers to watch the video in the supplementary material to better understand our visualization results. Moreover, we provide the implementation code in the supplementary material; our code and pretrained models will be publicly available. For convenience, we summarize the notations in the paper in Table 1.

2. Implementation Details

We present further implementation details and the dataset filtering process here.

Scene generation details. Inspired by WonderJourney [5], we adjust the depth maps with SAM [1] to promote spatio-temporal consistency of the depth for each object in the scene.

Dataset filtering process. We used the keywords in Table 2 to filter videos in OpenVid [2]. After filtering, we randomly selected 5×10^3 videos to fine-tune our video outpainting models.

3. Background Completion

As described in the method section of the main paper, in practice, we find that background regions occluded by the dynamic foregrounds can become exposed when rendering the dynamic scene with a fly-through camera trajectory. Since these regions are fully occluded by the foregrounds under camera pose $\mathbf{\Pi}^{(\mathrm{i})}$, the missing parts cause inconsistent colors in 3D when rendering the dynamic scene with fly-through cameras. To address this issue, we propose to employ our outpainting model to fill the missing colors in the background. More specifically, after obtaining the fixed viewpoint video $\boldsymbol{I}^{(\mathrm{i})}$, the depth maps $\boldsymbol{D}^{(\mathrm{i})}$ and foreground masks $\boldsymbol{M}^{(\mathrm{i})}$, we obtain the background video $\boldsymbol{I}^{(\mathrm{i,b})}$ by employing foreground masks on the video $\boldsymbol{I}^{(\mathrm{i})}$. For the regions occluded by the foreground in the video $\boldsymbol{I}^{(\mathrm{i,b})}$, we sample the ray information following (5), (6) and (7) in the main paper. Then we exploit our ray outpainting model to inpaint the video colors at the occluded regions with the sampled ray information. After that, we obtain the background depths of these regions with a depth estimation model [3] and reconstruct the background point cloud on these regions. Finally, we merge these occluded background point clouds into our dynamic scene point cloud.

4. More Dynamic Scene Generation Results

Space-time interpolation. Figure 1 presents visualizations of space-time interpolation results. In this figure, a cartoon cat is playing guitar with moving clouds. It can be seen that our model successfully renders dynamic scenes with fly-through cameras, while WonderJourney [5] only interpolates in the static scene.

Controllable scene generation. Figure 2 reports additional results on controllable scene generation. We control the village scene to generate `cascade waterfall` and `huge lake` separately. It can be seen that our model successfully controls scene generations with fly-through cameras.

MotionCtrl comparison details. We chose to compare with MotionCtrl [4], which is a camera-controllable video diffusion model presented in the figure of the main paper. As MotionCtrl only generates videos from text prompts, we use the first frame generated by MotionCtrl as the input of our model. To test the camera control performance in the MotionCtrl model, we evenly sample 16 camera translations from $[0, 0, 0]$ to $[0, 0, 40]$ and introduce them into MotionCtrl models. The results are shown in Figure 3. As Mo-

Table 1. Notations.

Notation	Description
$\mathbf{\Pi}^{(i)}$	Camera pose for the i^{th} view
$\boldsymbol{I}^{(i)}$	Video frames at pose $\mathbf{\Pi}^{(i)}$: $\{\boldsymbol{I}_t^{(i)} \in \mathbb{R}^{h\times w\times 3}\}_{t=0}^{N-1}$
$\boldsymbol{I}_t^{(i)}$	Video frame at timestamp t and pose $\mathbf{\Pi}^{(i)}$
$\boldsymbol{D}^{(i)}$	Depth maps at pose $\mathbf{\Pi}^{(i)}$: $\{\boldsymbol{D}_t^{(i)} \in \mathbb{R}^{h\times w}\}_{t=0}^{N-1}$
$\boldsymbol{D}_t^{(i)}$	Depth map at timestamp t and pose $\mathbf{\Pi}^{(i)}$
$\boldsymbol{M}^{(i)}$	Binary foreground masks at pose $\mathbf{\Pi}^{(i)}$: $\{\boldsymbol{M}_t^{(i)} \in \{0,1\}^{h\times w}\}_{t=0}^{N-1}$
$\boldsymbol{M}_t^{(i)}$	Binary foreground mask at timestamp t and pose $\mathbf{\Pi}^{(i)}$
$\mathcal{P}^{(i)}$	4D point cloud constructed until pose $\mathbf{\Pi}^{(i)}$
$\mathcal{P}_t^{(i)}$	4D point cloud at timestamp t constructed until pose $\mathbf{\Pi}^{(i)}$
$\boldsymbol{p}$	Point in 4D point cloud: $\boldsymbol{p} = (\boldsymbol{x}, t, \boldsymbol{c})$ with position $\boldsymbol{x}$, timestamp t, color $\boldsymbol{c}$
$\boldsymbol{x}$	3D position of a point, $\boldsymbol{x} \in \mathbb{R}^3$
$\boldsymbol{c}$	Color of a point, $\boldsymbol{c} \in \mathbb{R}^3$
$\mathcal{P}^{(i,\text{f})}, \mathcal{P}^{(i,\text{b})}$	Foreground, background point cloud until pose $\mathbf{\Pi}^{(i)}$
$\boldsymbol{I}^{(i,\text{f})}, \boldsymbol{I}^{(i,\text{b})}$	Foreground, background video frames at pose $\mathbf{\Pi}^{(i)}$
$\boldsymbol{D}^{(i,\text{f})}, \boldsymbol{D}^{(i,\text{b})}$	Foreground, background depth maps at pose $\mathbf{\Pi}^{(i)}$
ϕ	Mapping function from 2D image and depth to 3D point cloud
φ	Image rasterization function
$\hat{\boldsymbol{I}}^{(i)}$	Rasterized (partial) video at pose $\mathbf{\Pi}^{(i)}$ given point cloud $\mathcal{P}^{(i-1)}$
$\hat{\boldsymbol{D}}^{(i)}$	Rasterized ray depth maps at pose $\mathbf{\Pi}^{(i)}$ given point cloud $\mathcal{P}^{(i-1)}$
$\hat{\boldsymbol{M}}^{(i)}$	Binary masks for observed regions in rasterization at pose $\mathbf{\Pi}^{(i)}$
$\boldsymbol{r}^{(i)}(x,y)$	Unit-norm ray vector from camera origin at pose $\mathbf{\Pi}^{(i)}$ to pixel (x,y)
$\boldsymbol{o}^{(i)}$	Camera center of pose $\mathbf{\Pi}^{(i)}$
dist_{r2p}	Distance function between a ray and a point
$\bar{\boldsymbol{D}}_t^{(i)}$	Distance map between rays and point cloud at timestamp t
$\mathbf{s}$	Scene prompt describing desired motion
z_τ	Noisy video at noise step τ
θ_{v}	Training parameters of video diffusion model
θ_{c}	Training parameters of ControlNet

Table 2. Filter keywords.

keywords	mountain, forest, river, lake, ocean, canyon, meadow, hill, city, trees, cloud, wind, nature grassland, beach, snow, waterfall, stream, wildlife, pond, dunes, island, rainforest, sunset, sunrise, storm, rain, thunderstorm snowstorm, lightning, rainbow, twilight, dawn, dusk, star, aurora, moon, downtown, skyline, street, road, pedestrians, crowd transport, bridge, tower, skyscraper, urban, bustling, bus, metropolis, coast, beachfront, harbor, port, pier, boat, yacht dock, marina, waterfront, seaside, swimming, surf, sailboat, sun, shore, tidal, waves, bay, lagoon, village, town, countryside rural, greenhouse, sunflower field, landscape, sky, lake, water, sea, seas, cityscape, pedestrian, pedestrians, cars, traffic, people

tionCtrl learns from training videos with relatively limited camera movements, it fails to generate videos with large camera motions.

References

[1] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *ICCV*, 2023. 1

[2] Kepan Nan, Rui Xie, Penghao Zhou, Tiehan Fan, Zhenheng Yang, Zhijie Chen, Xiang Li, Jian Yang, and Ying Tai. Openvid-1M: A large-scale high-quality dataset for text-to-video generation. In *ICLR*, 2025. 1

[3] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. In *IEEE TPAMI*, 2020. 1

[4] Zhouxia Wang, Ziyang Yuan, Xintao Wang, Yaowei Li, Tianshui Chen, Menghan Xia, Ping Luo, and Ying Shan. Motionctrl: A unified and flexible motion controller for video genera-

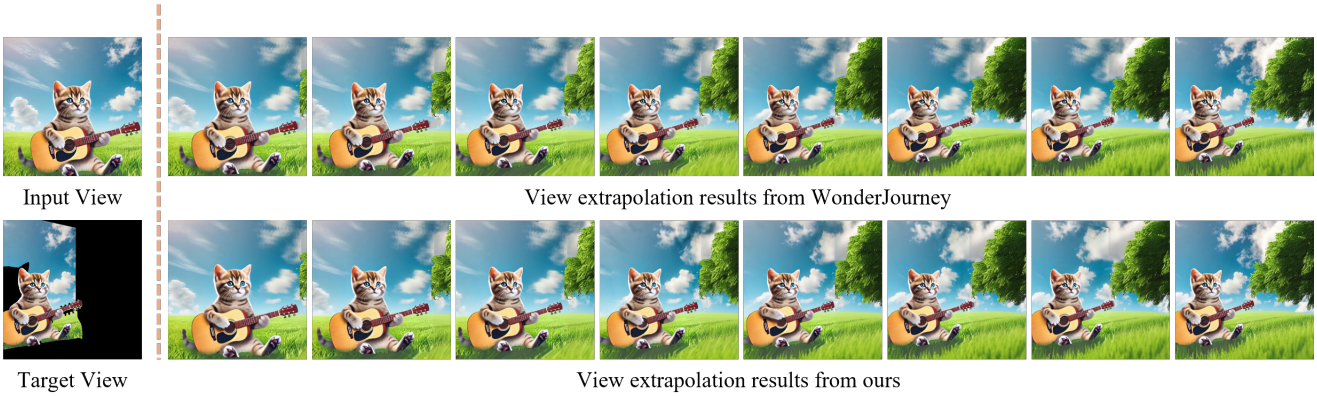

Figure 1. Space-time interpolation comparisons between DynamicVoyager and WonderJourney [5].

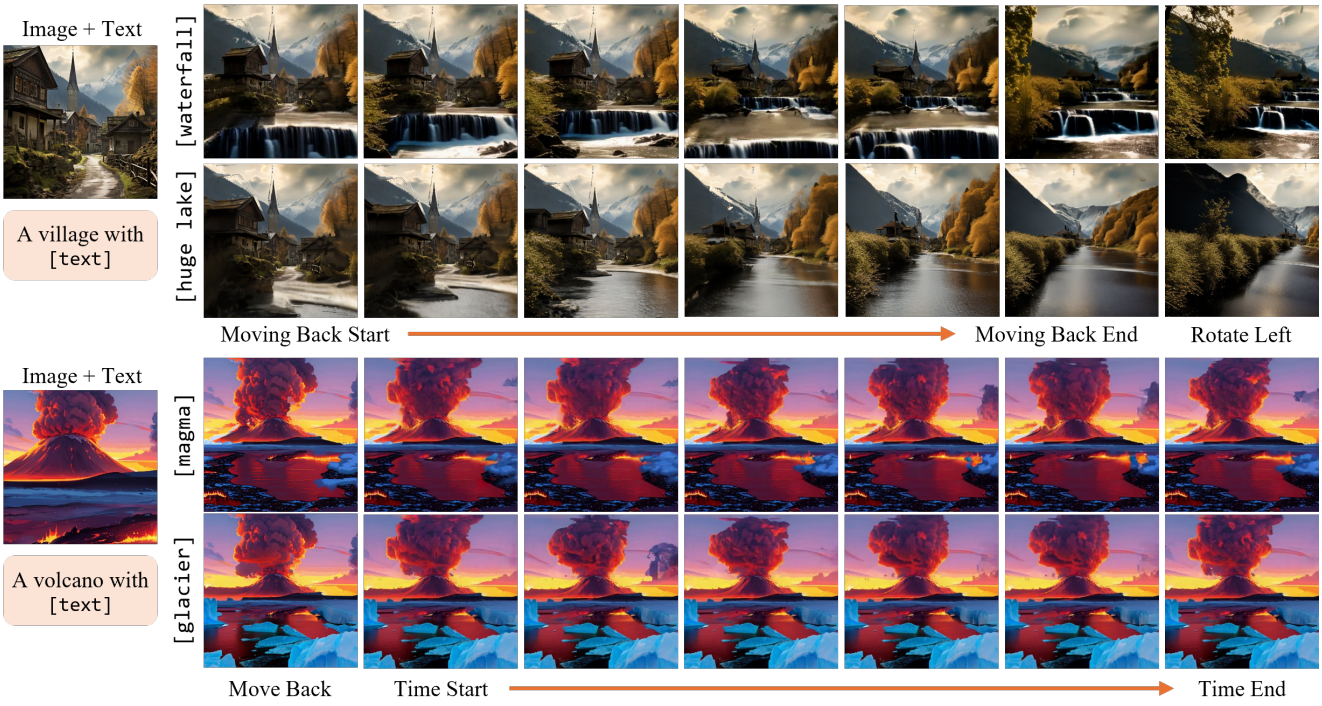

Figure 2. Controllable scene generation from input images with scene prompts. Given an image as input, DynamicVoyager successfully controls the dynamic scene outpainting content with the corresponding text prompt.

tion. In *ACM SIGGRAPH*, 2023. 1, 4

[5] Hong-Xing Yu, Haoyi Duan, Junhwa Hur, Kyle Sargent, Michael Rubinstein, William T Freeman, Forrester Cole, Deqing Sun, Noah Snavely, Jiajun Wu, et al. WonderJourney: Going from anywhere to everywhere. In *CVPR*, 2024. 1, 3

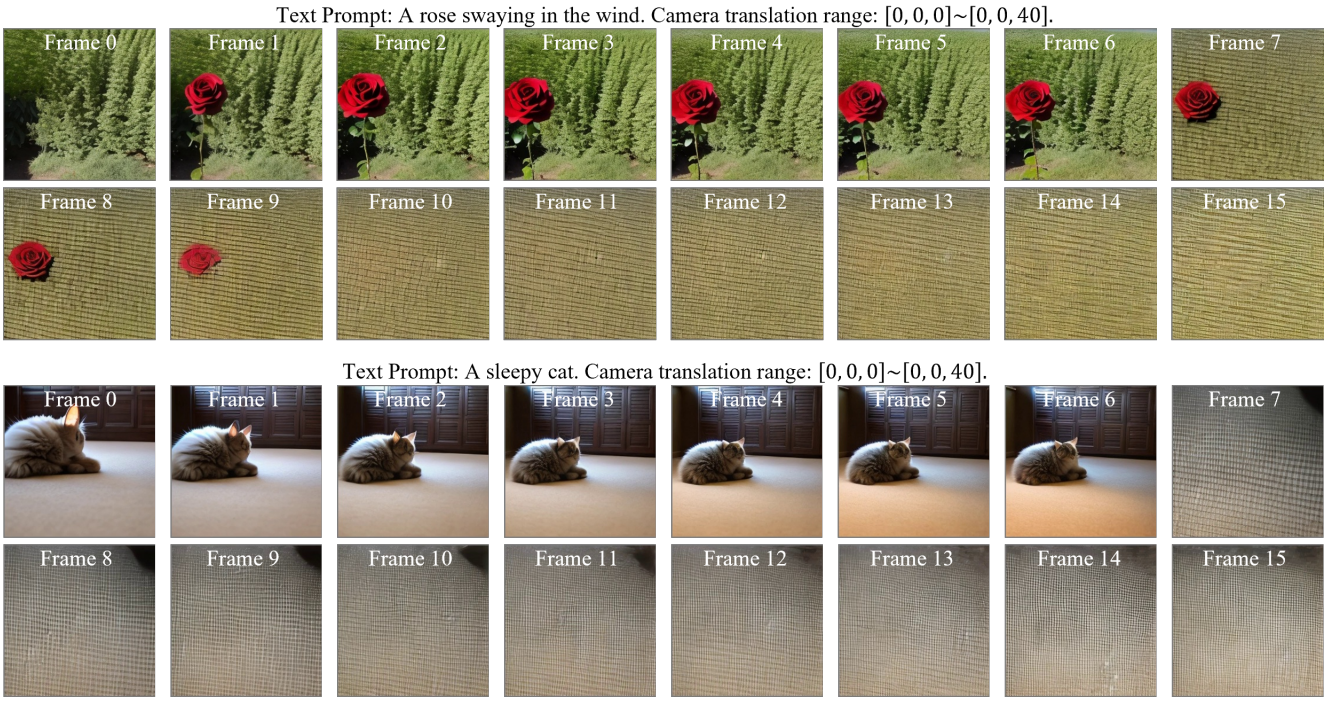

Figure 3. Detailed results of MotionCtrl [4] with large camera translation inputs. It can be seen that the model fails to generate videos with large camera motions.