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

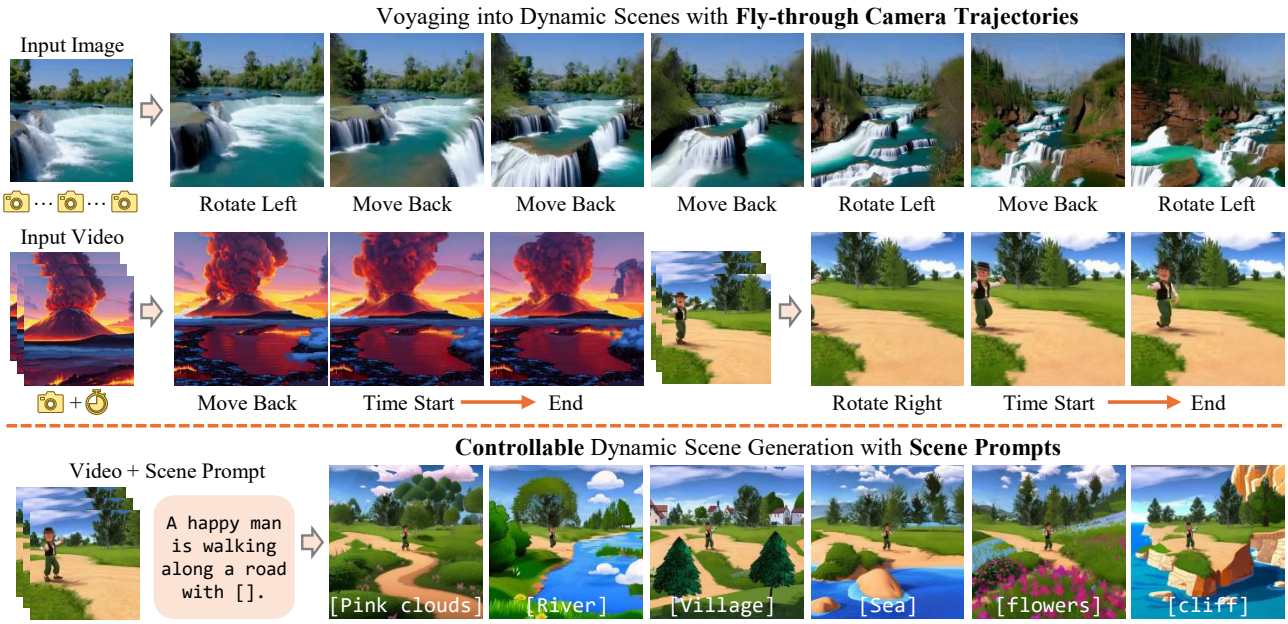

Figure 1. DynamicVoyager generates 4D point clouds of perpetual dynamic scenes by our dynamic scene outpainting process. Given a fixed viewpoint video (or an image with a motion prompt) with dynamic scene prompts and fly-through camera trajectories, DynamicVoyager can generate dynamic scenes along the trajectories (**Top**) and control scene generation contents (**Bottom**).

Abstract

The problem of generating a perpetual dynamic scene from a single view is an important problem with widespread applications in augmented and virtual reality, and robotics. However, since dynamic scenes regularly change over time, a key challenge is to ensure that different generated views be consistent with the underlying 3D motions. Prior work learns such consistency by training on multiple views, but the generated scene regions often interpolate between training views and fail to generate perpetual views. To address this issue, we propose DynamicVoyager, which reformulates dynamic scene generation as a scene outpainting problem with new dynamic content. As 2D outpainting models struggle at generating 3D consistent motions from a single 2D view, we enrich 2D pixels with information from their 3D rays that facilitates learning of 3D motion consistency. More specifically, we first map the single-view video input to a dynamic point cloud using the estimated video depths. We then render a partial video of the point cloud from a novel view and outpaint the missing regions using ray information (e.g., the distance from a ray to the point cloud) to generate 3D consistent motions. Next, we use the outpainted video to update the point cloud, which is used for outpainting the scene from future novel views. Moreover, we can control the generated content with the input text prompt. Experiments show that our model can generate perpetual scenes with consistent motions along fly-through cameras. Project page: https://tianfr.github.io/DynamicVoyager.

1. Introduction

Perpetual scene generation [27, 30, 58] aims to create a virtual 3D scene as the camera moves along arbitrary tra-

jectories, typically starting from a single view observation. While recent studies [27, 57, 58] have achieved significant progress in perpetual generation of *static* scenes by leveraging image outpainting models [23, 24, 41], these approaches cannot generate scenes with *dynamic* content (*e.g.,* waving hands, flowing rivers). Such dynamic content has important applications in augmented reality (AR), virtual reality (VR) [3, 4] and robotics [7]. For example, AR/VR game designers need to build a perpetual dynamic scene for players to explore and interact, while roboticists use virtual scenes containing commonly seen dynamic objects in natural environments for training embodied agents [9, 52] by self-exploration.

A key challenge in generating perpetual dynamic scenes is to ensure that the generated dynamic content has 3D consistent motions: *i.e.,* the motions observed in any two views must correspond to the same underlying 3D dynamics. Generating consistent motions from a single view is inherently ambiguous due to the limited 3D motion information contained in a 2D image. Previous dynamic scene generation methods [26, 28, 42, 50, 56] address this challenge by learning from multiple views surrounding the scenes so that 3D motion information can be implicitly learned from the 2D motions observed in these views. However, such a learning strategy severely restrains models from creating perpetual dynamic content, because the generated dynamic regions often interpolate between training views and fail to generate perpetual views, as exemplified in Figure 2.

In this paper, we propose DynamicVoyager, a novel approach to perpetual dynamic scene generation that reformulates the task as a scene outpainting problem, enabling the synthesis of new dynamic content in previously unseen regions of the scene. Rather than relying on multiple 2D views as input, DynamicVoyager learns 3D-consistent motion dynamics from single-view observations by treating *pixels as rays*, thereby enriching each pixel with the contextual information of its corresponding camera ray.

More specifically, given an input image, we exploit image-to-video diffusion models [14, 54] to synthesize a fixed pose video from that image. To initialize a dynamic scene from the video input, we estimate the video depth maps and backproject the fixed viewpoint video frames into 3D space as dynamic point clouds with the estimated depths. After that, we move the camera to a partially unseen area of the scene and rasterize the reconstructed point clouds into an incomplete video from that view. To outpaint the video with 3D consistent motions, we consider pixels as rays to complement the video input with 3D scene information. For pixels in the visible area, we obtain 3D motion information by sampling ray depth maps from the point clouds. For pixels in the unseen area, we backproject rays from pixels and use the distance between these rays and the point cloud to infer the 3D spatial relationship between

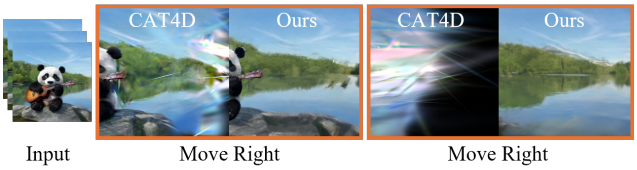

Figure 2. Failure examples of previous dynamic scene generation methods. While the generated dynamic scenes from previous works [42, 49, 50] are strictly bounded by the input views, DynamicVoyager successfully generates dynamic scenes with large camera motions by the proposed scene outpainting process.

the unseen area and the visible area. We then outpaint the incomplete video with the sampled ray depth and ray distance maps for the visible and unseen parts, respectively. Finally, we estimate the depth maps of the outpainted video and update the unseen area of the dynamic point clouds. Figure 1 visualizes examples of dynamic scenes generated by our method using input from a video with a fixed camera pose or a single image. It also shows that we can control the perpetual generation with input scene prompts.

To summarize, our contributions are as follows:

1. We reformulate the dynamic scene generation problem as a scene outpainting problem, so that our generated scene can be explored from a single view to any place through fly-through camera trajectories.
2. We generate view-consistent motions in 3D space by treating pixels as rays, which allows us to use 3D information to enrich the outpainting model.
3. We present experiments showing that, unlike other bounded scene generation methods, our model can generate perpetual dynamic scenes and control the generation with scene prompts.

2. Related Work

Static scene generation. With the development of vision foundation models [18, 20, 37, 40, 47], many works have started to generate static scenes from input images [2, 6, 8] or text prompts [13, 31]. Early efforts focused on indoor scene generation [2, 6, 13, 25] or object-centric scene generation [8]. These methods firstly generate multiview images of the target scene and fuse the multiview images by using 3DGS [19] or NeRF [32, 43, 44] representations. However, they can only generate static scenes with limited ranges of camera motions. When the camera moves drastically, these methods fail to generate unseen regions with consistent and clear scene structures. To overcome this issue, later studies have started to explore *perpetual scene generation* [30] that allows larger camera movements [17, 27, 30, 35, 57, 58]. Notably, WonderJourney [58] and WonderWorld [57] propose to generate an infinite world with diverse contents by using a combination of state-of-the-art depth estimation

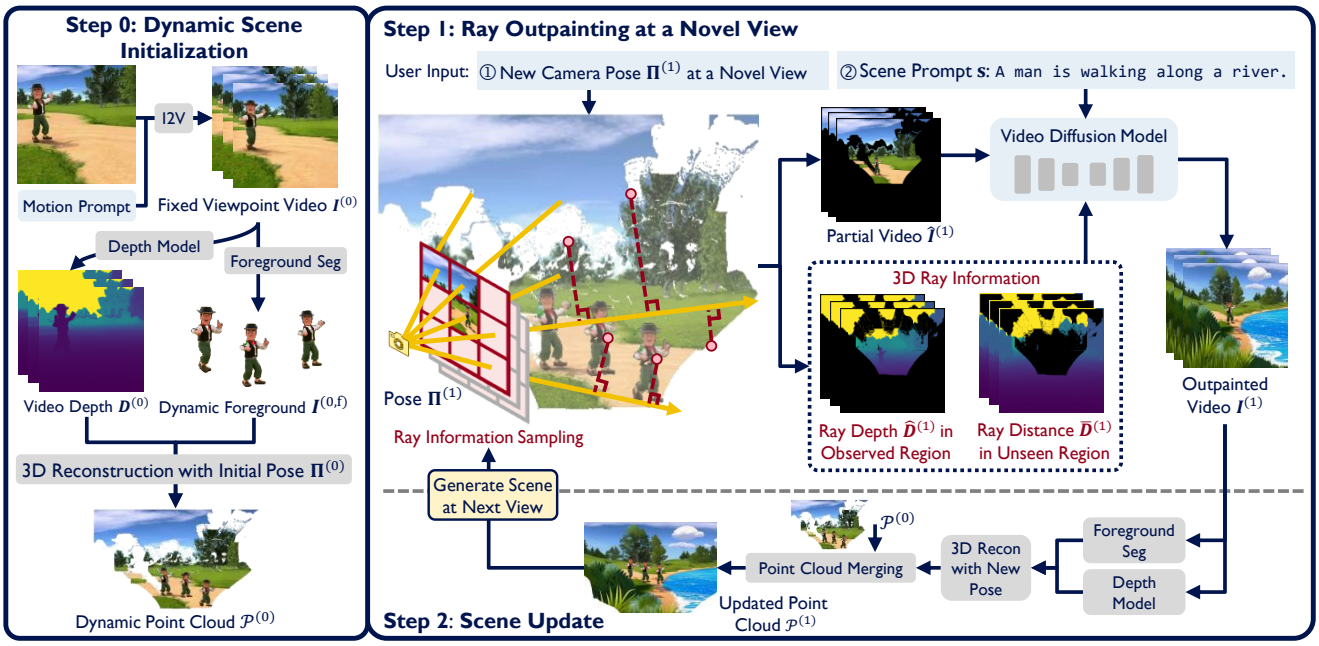

Figure 3. The overview of DynamicVoyager. First, for an initial camera pose, we build the dynamic point clouds from the input image by employing the image-to-video diffusion model [54], depth model [37] and foreground segmentation model [18] (§3.1). Then given a new camera pose, we render the partial video and the corresponding ray depth and ray distance maps. We consider pixels as rays to enrich the pixel information of the partial video with the corresponding ray depth and ray distance information for outpainting the video with consistent motions (§3.2). Finally, we update the dynamic point cloud with the outpainted video and the corresponding video depth (§3.3).

[37], image segmentation [20] and object detection models [18]. However, these methods can only generate diverse scenes with static objects. In this paper, we target the problem of generating dynamic scenes with new dynamic contents with fly-through camera trajectories.

Dynamic scene generation. Following the great success of static scene generation methods, recent studies have started to address the dynamic scene generation problem [26, 28, 34, 38, 42, 49, 50, 55, 56]. Due to the limited resources of real-world dynamic scene data, many researchers try to leverage the pretrained video diffusion models [14, 39, 51, 54] to generate multiple views of dynamic scenes. A key challenge in this setting is maintaining 3D motion consistency in multiple generated views. CAT4D [49] and DimensionX [42] learn to generate 3D consistent motion from multiview datasets [5, 29]. However, as the multiview images in these datasets are taken from cameras that are very close to each other, the generated dynamic scenes in these methods are strictly limited by the input images or videos. 4K4DGen [26] animates 3D consistent motion by building a 3D consistent noise space in the diffusion models. GEN3C [39] achieves camera control of dynamic scenes with 4D point cloud representations. However, since these methods only interpolate novel views surrounding the input image or videos, they still cannot generate new dynamic contents in the unseen regions of the input images. In this paper, we consider dynamic scene generation as a scene outpainting problem so that our method can generate new dynamic content with fly-through camera motions in the unseen regions of the input video.

Video diffusion models. Our method relies on the power of the recent video diffusion models [51, 54] to outpaint dynamic scenes with diverse contents. While some researchers propose video diffusion models with controllable camera trajectories [1, 10, 11, 15, 22, 45, 53, 60], these trajectories are bounded by the regions of the input views (a failure case is shown in Figure 8). Besides, since the outputs of these methods are generated videos without any 4D scene structures, they always encounter view inconsistency issues when fusing 4D representation with the output videos. Instead, our method adopts 4D point clouds as the dynamic scene representation and outpaints the dynamic point clouds with video outpainting models, where the unseen regions in the input video can be generated by the outpainting procedure, and the motion consistency in different views can be achieved by introducing ray information in 3D space.

3. Approach

In this section, we present the proposed DynamicVoyager approach for generating perpetual dynamic scenes, which is summarized in Figure 3. DynamicVoyager consists of

three steps, described in §3.1, §3.2 and §3.3, respectively:

- *Dynamic Scene Initialization:* DynamicVoyager accepts either a video with a fixed camera pose or an image with a prompt describing object motions as input—in the latter case, it first generates a video with a fixed pose. A depth map is then estimated from each frame of the video. We segment the video into a dynamic foreground and a static background, reconstruct the point clouds separately, and aggregate these into one dynamic point cloud.
- *Ray Outpainting:* At a new given pose, the goal is to generate a video consistent with the existing dynamic point cloud and a given scene prompt. We first rasterize the point cloud at the new pose, which is consistent by construction. That said, the rasterization only fills in part of the image, so the question is how to outpaint the rest *consistently*. To do so, we propose to guide the outpainting process with 3D information by providing the model with ray depth in the observed region and the distance of rays to the point cloud in the unseen region.
- *Dynamic Scene Update:* We update the dynamic point cloud using the depth maps of the video frames from the new pose. We repeat the steps in §3.2 and §3.3 to iteratively update the dynamic point cloud from novel poses.

3.1. Dynamic Scene Initialization

We begin by describing how to generate an initial dynamic point cloud given a video at a fixed camera pose $\mathbf{\Pi}^{(0)}$ or an image with a motion prompt. Notation-wise, a superscript (i) denotes objects related to the i^{th} camera pose $\mathbf{\Pi}^{(i)}$.

For illustration, suppose we are given an image $\boldsymbol{I}_0^{(0)} \in \mathbb{R}^{h\times w\times 3}$, where $h, w, 3$ are respectively height, width and number of color channels, as well as a prompt describing the desired motion for the video. To generate a video at the fixed pose $\mathbf{\Pi}^{(0)}$, we employ a pretrained image-to-video diffusion model [54] by providing both $\boldsymbol{I}_0^{(0)}$ and the motion prompt, where the motion prompt is further prepended with `Camera is strictly fixed`. We let $\boldsymbol{I}^{(0)} = \{\boldsymbol{I}_t^{(0)}\}_{t=0}^{N-1}$ denote the generated video, where $\boldsymbol{I}_t^{(0)}$ denotes its t^{th} frame.

Given the generated or user-input fixed viewpoint video $\boldsymbol{I}^{(0)}$, we reconstruct the underlying dynamic scene, represented as a 4D point cloud $\mathcal{P}^{(0)} = \{\boldsymbol{p} = (\boldsymbol{x}, t, \boldsymbol{c})\}$, where $\boldsymbol{x} \in \mathbb{R}^3$, $t \in \mathbb{R}$ and $\boldsymbol{c} \in \mathbb{R}^3$ denote the 3D position, timestamp and color respectively. Since the objects in the video foreground and the scene in the background have different dynamic natures, our framework reconstructs point cloud $\mathcal{P}^{(0,\text{f})}$ for the foreground scene and $\mathcal{P}^{(0,\text{b})}$ for background separately. To this end, we extract the binary video foreground masks $\boldsymbol{M}^{(0)} = \{\boldsymbol{M}_t^{(0)} \in \{0,1\}^{h\times w}\}_{t=0}^{N-1}$ with the foreground segmentation method [18] to segment the video $\boldsymbol{I}^{(0)}$ into foreground video $\boldsymbol{I}^{(0,\text{f})} = \{\boldsymbol{I}_t^{(0,\text{f})}\}_{t=0}^{N-1}$ and background video $\boldsymbol{I}^{(0,\text{b})} = \{\boldsymbol{I}_t^{(0,\text{b})}\}_{t=0}^{N-1}$. We employ the depth model [37] on video $\boldsymbol{I}^{(0)}$ to obtain the video depth maps $\boldsymbol{D}^{(0)} = \{\boldsymbol{D}_t^{(0)} \in \mathbb{R}^{h\times w}\}_{t=0}^{N-1}$. Next, we describe the foreground and background scene initialization steps.

Figure 4. Qualitative ablation studies of the proposed foreground layer (§3.1) and background completion (§3.3) strategy.

Scene foreground initialization. We exploit the depth maps $\boldsymbol{D}^{(0)}$ and foreground masks $\boldsymbol{M}^{(0)}$ to obtain the foreground depth maps $\boldsymbol{D}^{(0,\text{f})} = \{\boldsymbol{D}_t^{(0,\text{f})}\}_{t=0}^{N-1}$. Since the video is captured with a fixed camera, we initialize the video camera pose $\mathbf{\Pi}^{(0)}$ as the center of the dynamic scene. We reconstruct the initial foreground point cloud by employing the foreground video frames $\boldsymbol{I}^{(0,\text{f})}$, foreground depth maps $\boldsymbol{D}^{(0,\text{f})}$ and the initial camera pose $\mathbf{\Pi}^{(0)}$,

$$\mathcal{P}_t^{(0,\text{f})} = \phi([\boldsymbol{I}_t^{(0,\text{f})}, \boldsymbol{D}_t^{(0,\text{f})}], \mathbf{\Pi}^{(0)}, t),\ t \in \{0, ..., N\text{–}1\}\,, \quad (1)$$

where $\mathcal{P}_t^{(0,\text{f})}$ denotes the foreground point cloud at timestamp t. ϕ denotes the mapping from an image with its depth map to the point cloud. We obtain the foreground point cloud at all timestamps $\mathcal{P}^{(0,\text{f})} = \bigcup_{t=0}^{N-1} \mathcal{P}_t^{(0,\text{f})}$.

Scene background initialization. Although the background geometry of a dynamic scene remains static over time, its appearance can vary across frames due to dynamic textures such as flowing water, moving clouds, or changing lighting. Hence, we model the background as having constant depth but varying color across time. Also, notice that foreground objects may occlude parts of the background at certain timestamps, making some background pixels unobservable at those frames. To address this, we leverage foreground masks to identify non-occluded regions and compute the background depth at each pixel location. Specifically, let $\boldsymbol{D}_t^{(0)}(x, y)$ and $\boldsymbol{M}_t^{(0)}(x, y)$ denote the depth and foreground mask values at pixel (x, y) of the video frame $\boldsymbol{I}_t^{(0)}$. The background depth can be computed element-wise as

$$\boldsymbol{D}^{(0,\text{b})}(x,y) = \frac{\sum_{t=0}^{N-1} \boldsymbol{D}_t^{(0)}(x,y) \cdot \left(1 - \boldsymbol{M}_t^{(0)}(x,y)\right)}{\sum_{t=0}^{N-1} \left(1 - \boldsymbol{M}_t^{(0)}(x,y)\right)}\,, \quad (2)$$

where $\boldsymbol{D}^{(0,\text{b})}$ is the refined background depth map. We employ the same mapping function ϕ to map the background regions on the video frames to the 4D point clouds,

$$\mathcal{P}_t^{(0,\text{b})} = \phi([\boldsymbol{I}_t^{(0,\text{b})}, \boldsymbol{D}^{(0,\text{b})}], \mathbf{\Pi}^{(0)}, t), t \in \{0, ..., N\text{–}1\}. \quad (3)$$

In this way, the obtained background point clouds $\mathcal{P}^{(0,\text{b})} = \bigcup_{t=0}^{N-1} \mathcal{P}_t^{(0,\text{b})}$ have time-invariant positions and time-varying colors. Finally, we obtain the scene point cloud by

merging the foreground and background point clouds,

$$\mathcal{P}^{(0)} = \mathcal{P}^{(0,\mathrm{f})} \cup \mathcal{P}^{(0,\mathrm{b})}, \tag{4}$$

which is then introduced for outpainting the scene at a novel view with 3D consistent motions.

3.2. Ray Outpainting at a Novel View

To outpaint the scene at a given novel pose $\mathbf{\Pi}^{(1)}$, we follow the assumption of previous works [21, 57, 58] that the view of $\mathbf{\Pi}^{(1)}$ overlaps moderately with that of $\mathbf{\Pi}^{(0)}$ to allow for outpainting a substantial portion of the scene. The question is, how can we outpaint a dynamic scene containing consistent motions with the existing dynamic point cloud $\mathcal{P}^{(0)}$?

To address this question, we first rasterize the point cloud at $\mathbf{\Pi}^{(1)}$. As shown in the top row of Figure 5, we obtain a partial video whose observed region is the projection of the portion of the point cloud visible from $\mathbf{\Pi}^{(1)}$. To handle the unseen region, a naive approach would be to directly outpaint the partial video. However, this typically leads to inconsistencies across the boundary of observed and unseen regions, as shown in Figure 5. Our intuition is that pixels in each video frame only provide 2D information, which is not enough to reconstruct the 3D motion. To address this issue, we consider the 3D ray from the camera origin through each pixel. If the pixel is in the observed region, we compute its depth. If the pixel is in the unseen region, we compute the distance from its ray to the point cloud. Both the depth and distance maps serve as guidance for how to outpaint a pixel while respecting information from the existing point cloud.

Ray information sampling in observed region. Given a novel pose $\mathbf{\Pi}^{(1)}$, we first rasterize the dynamic point cloud $\mathcal{P}^{(0)}$: at every time step $t \in \{0, \cdots, N-1\}$, we compute

$$(\hat{\boldsymbol{I}}_t^{(1)}, \hat{\boldsymbol{D}}_t^{(1)}, \hat{\boldsymbol{M}}_t^{(1)}) = \varphi(\mathcal{P}_t^{(0)}, \mathbf{\Pi}^{(1)}). \tag{5}$$

Here, φ denotes the image rasterization function. $\hat{\boldsymbol{M}}_t^{(1)}$ is a binary mask, where $\hat{\boldsymbol{M}}_t^{(1)}(x, y) = 1$ if at least one 3D point from $\mathcal{P}_t^{(0)}$ hits the image at pixel (x, y) during rasterization and 0 otherwise. Correspondingly, $\hat{\boldsymbol{I}}_t^{(1)}$ is the rasterized partial video and $\hat{\boldsymbol{D}}_t^{(1)}$ the rasterized *ray depth* map (to be distinguished from the depth map generated by depth models as in §3.1). A hat on the notation emphasizes that the signals are not observed for all locations, and we denote $\hat{\boldsymbol{D}}^{(1)} = \{\hat{\boldsymbol{D}}_t^{(1)}\}_{t=0}^{N-1}$, $\hat{\boldsymbol{M}}^{(1)} = \{\hat{\boldsymbol{M}}_t^{(1)}\}_{t=0}^{N-1}$ and $\hat{\boldsymbol{I}}^{(1)} = \{\hat{\boldsymbol{I}}_t^{(1)}\}_{t=0}^{N-1}$. A few remarks are in order. First, by doing the rasterization, the observed regions are obviously consistent with the point cloud. Second, the ray depth maps encode the 3D information of the observed scene, which as we shall see will be included as guidance for outpainting. Third, for the regions not observed at $\mathbf{\Pi}^{(1)}$, we introduce the distance between rays and the point cloud to complement the 3D information in these regions, as described next.

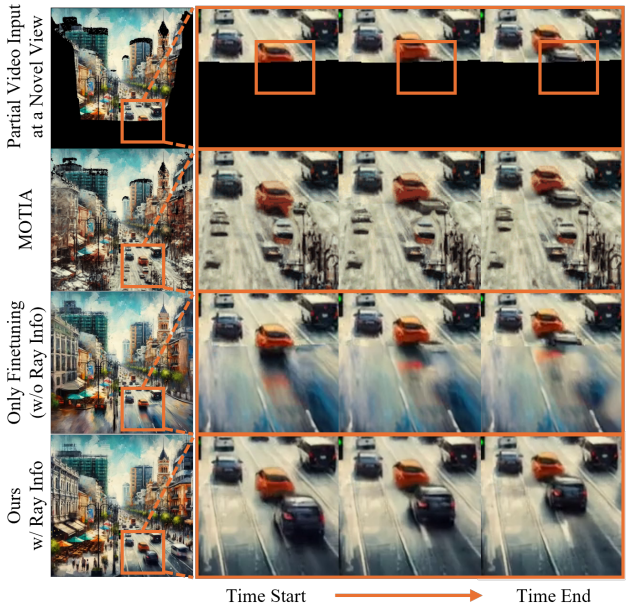

Figure 5. Ablation studies of scene outpainting with ray information. We also compare with the 2D outpainting model MOTIA [46]. Detailed visualization demonstrates that with ray information, our model successfully outpaints the dynamic scenes with consistent motions.

Ray-point-cloud distance computation in unseen region. Fix the reference coordinate system at the origin of a camera. Let $\boldsymbol{r} \in \mathbb{R}^3$ be a unit vector denoting the direction of a ray from the origin pointing at a pixel location, and $\boldsymbol{p} \in \mathbb{R}^3$ a point in the 3D space. We recall that the distance between the ray and the 3D point can be computed as

$$\mathrm{dist}_{\mathrm{r2p}}(\boldsymbol{r}, \boldsymbol{p}) = \sqrt{\|\boldsymbol{p}\|_2^2 - (\boldsymbol{r}^\top \boldsymbol{p})^2}. \tag{6}$$

With the above said, we can compute the distance between camera rays and the 4D point cloud for the complementary 3D information of the unseen regions. Namely, we compute

$$\bar{\boldsymbol{D}}_t^{(1)}(x, y) = \min_{\boldsymbol{p} \in \mathcal{P}_t^{(0)}} \mathrm{dist}_{\mathrm{r2p}}\left(\boldsymbol{r}^{(1)}(x, y), \boldsymbol{p} - \boldsymbol{o}^{(1)}\right), \tag{7}$$

where $\boldsymbol{r}^{(1)}(x, y) \in \mathbb{R}^3$ is the unit-norm ray vector starting from the origin of $\mathbf{\Pi}^{(1)}$ pointing at pixel (x, y), and $\boldsymbol{o}^{(1)}$ is the camera center of $\mathbf{\Pi}^{(1)}$, both in the reference coordinate system of $\mathbf{\Pi}^{(0)}$. As we will see, such a distance will be used to guide the outpainting.

Ray outpainting. To outpaint the dynamic scene at the novel pose $\mathbf{\Pi}^{(1)}$ from the partial video, we desire a few properties for the outpainted video: 1) the outpainted video needs to have a static pose fixed at $\mathbf{\Pi}^{(1)}$ and be consistent with the existing dynamic point cloud in the observed region, 2) the user should be able to control the generation

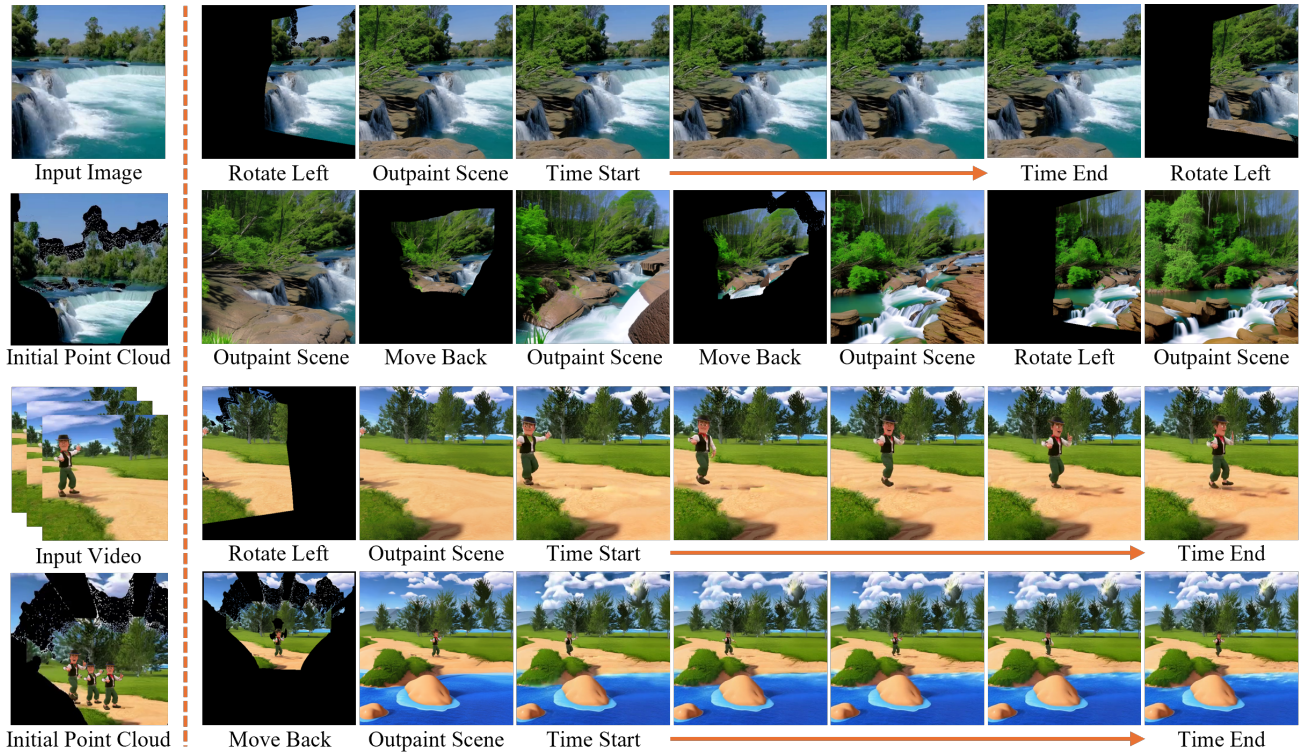

Figure 6. Voyaging into perpetual dynamic scenes with long camera trajectories from a single image or fixed viewpoint video. We iteratively show the unseen regions through large camera motions, the dynamic scene outpainting results and the consistent motions from novel views. By enriching the pixel information with ray contexts, our method successfully builds large dynamic scenes from input views.

in the outpainted region (i.e., the unseen region), and 3) the motions in the outpainted region need to be consistent with the motions in the observed region. To fulfill properties 1) and 2), we design our video outpainting model to take as input the extracted partial video $\hat{\boldsymbol{I}}_t^{(1)}$, as well as a prompt narrating the desired motions in the generated scene. For property 3), we propose to employ ray depth map $\hat{\boldsymbol{D}}^{(1)}$ and ray distance map $\bar{\boldsymbol{D}}^{(1)}$ as guidance on 3D information in the observed region and how much the outpainted ray in the unseen region should respect the existing point cloud. To sum up, our video outpainting model accepts the partial video $\hat{\boldsymbol{I}}^{(1)}$, scene prompt $\mathbf{s}$, ray depth map $\hat{\boldsymbol{D}}^{(1)}$, ray distance map $\bar{\boldsymbol{D}}^{(1)}$ and generates an outpainted video $\boldsymbol{I}^{(1)} = \{\boldsymbol{I}_t^{(1)}\}_{t=0}^{N-1}$. Next we describe how to train such a model.

Training the outpainting model. We collect a small video dataset for training (details in *supplementary material*), and reconstruct point clouds from it to compute ray information for supervision. More specifically, given a training video $\boldsymbol{I}$, we extract depth maps $\boldsymbol{D}$ using the depth model [37] and backproject the video from a camera $\boldsymbol{\Pi}$ at the world center to reconstruct a point cloud. We then move the camera closer to the scene and filter out the 3D points that are no longer visible from the current viewpoint. Then we move the camera back to the original camera pose $\boldsymbol{\Pi}$ and compute a partial video $\hat{\boldsymbol{I}}$, a partial depth map $\hat{\boldsymbol{D}}$, and a ray distance map $\bar{\boldsymbol{D}}$ following §3.2. To train the model, we add noise to $\boldsymbol{I}$ to obtain z_τ at step τ, and employ ControlNet [59] to inject ray information. Let $\theta_\mathrm{v}, \theta_\mathrm{c}$ denote the training parameters of the video diffusion model and ControlNet model. The training objective is

$$\mathcal{L} = \mathbb{E}_{z_0,\boldsymbol{I},\hat{\boldsymbol{I}},\tau,\hat{\boldsymbol{D}},\bar{\boldsymbol{D}},\mathbf{s},\epsilon}\left\|\epsilon - \epsilon_{\theta_\mathrm{v},\theta_\mathrm{c}}(\boldsymbol{I},\hat{\boldsymbol{I}},z_\tau,\tau,\hat{\boldsymbol{D}},\bar{\boldsymbol{D}},\mathbf{s})\right\|_2^2, \quad (8)$$

where $\epsilon \sim \mathcal{N}(0,1)$ is the diffusion noise.

3.3. Dynamic Scene Update

Now that we have an outpainted video $\boldsymbol{I}^{(1)}$ at the new pose $\boldsymbol{\Pi}^{(1)}$, the next step is to produce a new dynamic point cloud and merge it with the previous one $\mathcal{P}^{(0)}$.

Point cloud merging. We exploit the depth model [37] to obtain the video depth maps $\boldsymbol{D}^{(1)}$ of the outpainted video $\boldsymbol{I}^{(1)}$. Noting that there is a depth inconsistency problem between the estimated depths and the rendered ray depth map $\hat{\boldsymbol{D}}^{(1)}$ in the mask regions $\hat{\boldsymbol{M}}^{(1)}$, we follow common practices [57, 58] to align the video depth maps $\boldsymbol{D}^{(1)}$ with the ray depth maps $\hat{\boldsymbol{D}}^{(1)}$ by finetuning the depth estimation model. After obtaining the aligned depth maps from the finetuned depth model, we follow §3.1 to initialize foreground point cloud $\mathcal{P}^{(1,\mathrm{f})}$ and background point cloud

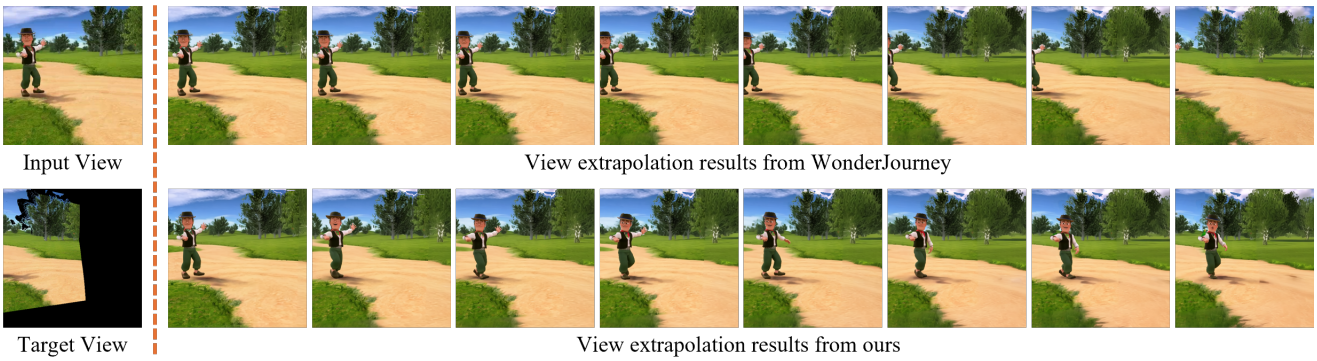

Figure 7. View extrapolation results. While Wonderjourney [58] only extrapolates views of a static character in 3D space, DynamicVoyager renders extrapolated views with 3D consistent motions by leveraging the ray information for scene outpainting.

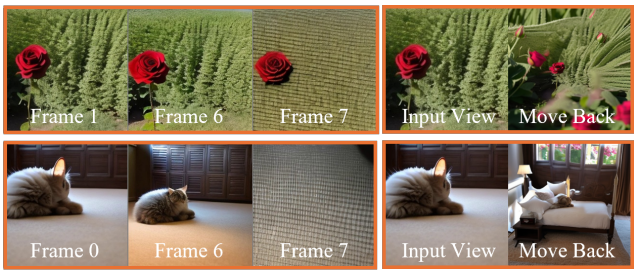

Figure 8. Visual comparisons with video diffusion model MotionCtrl [48]. Full details are in *supplementary material*. DynamicVoyager outpaints the scenes with large camera motions.

$\mathcal{P}^{(1,\mathrm{b})}$. Then we merge the new point clouds with the previous one to update the dynamic scenes,

$$\mathcal{P}^{(1)} = \mathcal{P}^{(1,\mathrm{f})} \cup \mathcal{P}^{(1,\mathrm{b})} \cup \mathcal{P}^{(0)}, \tag{9}$$

where $\mathcal{P}^{(1)}$ is the updated point cloud at the pose $\mathbf{\Pi}^{(1)}$. In this way, we could generate perpetual dynamic scenes along fly-through camera trajectories by looping §3.2 and §3.3 with new camera poses and scene prompts.

Background completion. In practice, we find that the missing background point clouds occluded by the moving foreground in the scene cause inconsistency when rendering along fly-through camera trajectories (as shown in Figure 4). We hence propose to employ our video outpainting model to generate background videos of the occluded parts. After that, we extract depth maps of the background videos and reconstruct the background point clouds with the extracted depth maps. Details are in *supplementary material*. We also update the scene with these point clouds.

4. Experiments

In this section, we first describe the details of our experimental setup, we then discuss the results of the scene generation with fly-through cameras and controllable scene generation. Finally, we conduct ablation studies of our model.

Table 1. Quantitative results of controllable dynamic scene generation on the dynamic degree (DD), factual consistency (FC) [12] and CLIP [36] scores. While WonderJourney [58] only controls static scene generation with scene prompts, our model achieves controllable dynamic scene generation with higher performance.

methods	WonderJourney [58]	ours
CLIP-SIM ↑ / DD ↑ / FC ↑	24.98 / 2.23 / 1.30	**25.23 / 3.13 / 1.32**

4.1. Experiment Setups

Data. We trained our outpainting model with the OpenVid dataset [33], which contains high-resolution videos with detailed video prompts. In order to train with daily dynamic scene videos, we sampled 5,000 videos that contain outdoor dynamic scenes, including natural scenes (such as waterfalls, rivers and moving clouds) and urban scenes (such as walking people and moving cars). Additional data curation details can be found in the *supplementary material*.

Training details. We employed the LoRA [16] to finetune the `CogvideoX-5B-I2V` model [54]. The LoRA rank was 256 and the learning rate was 1×10^{-4} with a cosine schedule. The video resolution is $16\times512\times512$, and hence $N = 16$ and $h = w = 512$. For calculating the ray information, we used the depth model [37] to extract the depth map of each video frame. Then, we followed the state-of-the-art video outpainting model MOTIA [46] and exploited the same mask proportion for outpainting.

Camera details. To set up dynamic scenes with fly-through camera trajectories, we followed the common practices in [21, 35, 58], where cameras move along a straight line or rotate. In the former case, we moved the camera backward by 0.0005 units between two adjacent camera poses, where one unit corresponds to the normalized 3D coordinate defined in PyTorch3D, and in the latter case, we rotated the

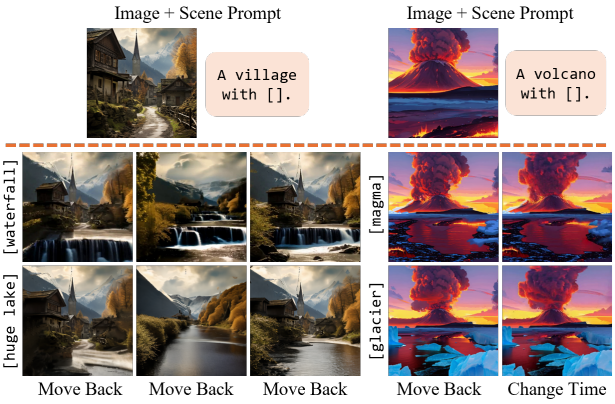

Figure 9. Controllable scene generation from input images with scene prompts. Full details are shown in *supplementary material*. DynamicVoyager successfully controls the dynamic scene outpainting content with the corresponding scene prompt.

camera by 0.45 radians. We generated camera paths by linearly interpolating translation and approximating the rotation using uniform angular steps.

4.2. Scene Generation with Fly-through Cameras

Figure 6 shows the dynamic scene generation results with long fly-through trajectories. We tested our model on both the background dynamics (water flowing in the waterfall) and the foreground dynamics (person walking on the path). Compared to the initial point cloud, our model generated perpetual dynamic scenes by exploiting the proposed scene outpainting process. As there is no direct baseline for perpetual dynamic scene generation, we compare our method with the perpetual static scene generation method WonderJourney [58] in Figure 7, state-of-the-art 4D scene generation model CAT4D [49] in Figure 2 and camera-controllable video diffusion model MotionCtrl [48] in Figure 8. These figures demonstrate that our model can better generate perpetual dynamic scenes with 3D consistent motions by exploiting our ray outpainting model.

4.3. Scene Generation Controlled by Text

To test the controllable dynamic scene generation ability of our model, we generated dynamic scenes from village and volcano images with scene prompts. As shown in Figure 9, for the village image, we control the outpainting content by introducing two different text prompts: `waterfall` and `huge lake`. For the volcano image, we prompted the model to generate `glacier` and `magma`. Our model successfully controls the perpetual dynamic scene generation with the given scene prompts. Following the common practices in [21, 56–58], we employed the CLIP [36] scores between the novel view images and the input scene prompts and the dynamic degree (DD) and factual consistency (FC) indices proposed by VideoScore [12] to thoroughly evaluate the generation quality of the video. Table 1 shows that, compared to WonderJourney, our model achieves better performance by controlling dynamic scene generation.

Table 2. Quantitative results of 3D consistent motions. VQ, TC and TA respectively denote the visual quality, temporal consistency and text-to-video alignment matrices [12]. Our model achieves better performance by achieving 3D consistent motions.

methods	CLIP-SIM ↑	VQ ↑	TC ↑	TA ↑	FC ↑
MOTIA [46]	23.80	2.281	1.852	2.656	1.859
w/o ray info	23.12	2.562	1.984	2.672	1.977
DynamicVoyager	**24.70**	**2.578**	**2.234**	**2.938**	**2.188**

Table 3. Quantitative ablation studies of the proposed foreground layer (§3.1) and background completion (§3.3) strategy.

methods	CLIP-SIM ↑	VQ ↑	TC ↑	TA ↑	FC ↑
w/o fg layer	23.88	1.891	0.871	2.219	1.102
w/o bg completion	24.62	1.906	0.871	2.547	1.195
DynamicVoyager	**25.48**	**1.977**	**1.031**	**2.562**	**1.320**

4.4. Ablation Study

Ray outpainting. To fairly evaluate the proposed ray outpainting model, we conducted experiments on five scenes containing common dynamic objects: cars, people, water and clouds. For each scene, we sampled 16 novel view images from the generated dynamic point cloud at 16 different timestamps along the fly-through camera trajectories. As there are no 3D video outpainting baselines, we chose the 2D outpainting model MOTIA [46] for comparisons. Figure 5 shows that by introducing the ray information for outpainting, our model successfully outpaints 3D consistent motions at the outpainting boundaries. In this way, our ray outpainting model achieves better performance in Table 2.

4D representation. We tested the proposed foreground representation and background completion strategy in Figure 4 and Table 3. It can be seen that our model renders consistent foregrounds with foreground layers and generates a plausible background with the background completion strategy.

5. Conclusion

We proposed DynamicVoyager, a new model for perpetual dynamic scene generation, which reformulates this task as an outpainting problem. By considering pixels as rays and conditioning on ray depth and distance maps, DynamicVoyager can generate scenes with 3D consistent motions. Experimental results demonstrate the superiority of our model.

Limitations. Challenges remain in handling reflections, shadows, fine structures, multi-view rendering, and depth discontinuities, which can be explored in future work.

Acknowledgments

The authors sincerely thank Prof. Kostas Daniilidis, Dr. Jiahui Lei, Qiao Feng, Chen Wang, Ziqing Xu, and Uday Kiran for their generous and insightful feedback on this work. We also acknowledge the support of a Penn Engineering graduate fellowship, Penn startup funds, and the Research Collaboration on the Mathematical and Scientific Foundations of Deep Learning under grants NSF 2031985 and Simons 814201.

References

[1] Sherwin Bahmani, Ivan Skorokhodov, Aliaksandr Siarohin, Willi Menapace, Guocheng Qian, Michael Vasilkovsky, Hsin-Ying Lee, Chaoyang Wang, Jiaxu Zou, Andrea Tagliasacchi, et al. VD3D: Taming large video diffusion transformers for 3D camera control. In *ICLR*, 2025. 3

[2] Miguel Angel Bautista, Pengsheng Guo, Samira Abnar, Walter Talbott, Alexander Toshev, Zhuoyuan Chen, Laurent Dinh, Shuangfei Zhai, Hanlin Goh, Daniel Ulbricht, et al. Gaudi: A neural architect for immersive 3D scene generation. In *NeurIPS*, 2022. 2

[3] Michael Broxton, John Flynn, Ryan Overbeck, Daniel Erickson, Peter Hedman, Matthew Duvall, Jason Dourgarian, Jay Busch, Matt Whalen, and Paul Debevec. Immersive light field video with a layered mesh representation. In *ToG*, 2020. 2

[4] Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. High-quality streamable free-viewpoint video. In *ToG*, 2015. 2

[5] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *CVPR*, 2023. 3

[6] Terrance DeVries, Miguel Angel Bautista, Nitish Srivastava, Graham W Taylor, and Joshua M Susskind. Unconstrained scene generation with locally conditioned radiance fields. In *ICCV*, 2021. 2

[7] C Gan, J Schwartz, S Alter, M Schrimpf, J Traer, J De Freitas, J Kubilius, A Bhandwaldar, N Haber, M Sano, et al. ThreeDWorld: A platform for interactive multi-modal physical simulation. In *NeurIPS*, 2021. 2

[8] Ruiqi Gao*, Aleksander Holynski*, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul P. Srinivasan, Jonathan T. Barron, and Ben Poole*. CAT3D: Create anything in 3D with multi-view diffusion models. In *NeurIPS*, 2024. 2

[9] Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, Xiaodi Yuan, Pengwei Xie, Zhiao Huang, Rui Chen, and Hao Su. ManiSkill2: A unified benchmark for generalizable manipulation skills. In *ICLR*, 2023. 2

[10] Yuwei Guo, Ceyuan Yang, Anyi Rao, Zhengyang Liang, Yaohui Wang, Yu Qiao, Maneesh Agrawala, Dahua Lin, and Bo Dai. AnimateDiff: Animate your personalized text-to-image diffusion models without specific tuning. In *ICLR*, 2024. 3

[11] Hao He, Yinghao Xu, Yuwei Guo, Gordon Wetzstein, Bo Dai, Hongsheng Li, and Ceyuan Yang. CameraCtrl: Enabling camera control for text-to-video generation. In *ICLR*, 2025. 3

[12] Xuan He, Dongfu Jiang, Ge Zhang, Max Ku, Achint Soni, Sherman Siu, Haonan Chen, Abhranil Chandra, Ziyan Jiang, Aaran Arulraj, Kai Wang, Quy Duc Do, Yuansheng Ni, Bohan Lyu, Yaswanth Narsupalli, Rongqi Fan, Zhiheng Lyu, Yuchen Lin, and Wenhu Chen. VideoScore: Building automatic metrics to simulate fine-grained human feedback for video generation. In *EMNLP*, 2024. 7, 8

[13] Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. Text2Room: Extracting textured 3D meshes from 2D text-to-image models. In *ICCV*, 2023. 2

[14] Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. CogVideo: Large-scale pretraining for text-to-video generation via transformers. In *ICLR*, 2022. 2, 3

[15] Chen Hou, Guoqiang Wei, Yan Zeng, and Zhibo Chen. Training-free camera control for video generation. In *ICLR*, 2025. 3

[16] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022. 7

[17] Ronghang Hu, Nikhila Ravi, Alexander C Berg, and Deepak Pathak. Worldsheet: Wrapping the world in a 3d sheet for view synthesis from a single image. In *ICCV*, 2021. 2

[18] Jitesh Jain, Jiachen Li, Mang Tik Chiu, Ali Hassani, Nikita Orlov, and Humphrey Shi. OneFormer: One transformer to rule universal image segmentation. In *CVPR*, 2023. 2, 3, 4

[19] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D gaussian splatting for real-time radiance field rendering. In *TOG*, 2023. 2

[20] 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. 2, 3

[21] Hanyang Kong, Dongze Lian, Michael Bi Mi, and Xinchao Wang. DreamDrone: Text-to-image diffusion models are zero-shot perpetual view generators. In *ECCV*, 2024. 5, 7, 8

[22] Zhengfei Kuang, Shengqu Cai, Hao He, Yinghao Xu, Hongsheng Li, Leonidas Guibas, and Gordon Wetzstein. Collaborative video diffusion: Consistent multi-video generation with camera control. In *NeurIPS*, 2024. 3

[23] Black Forest Labs. Flux. https://github.com/black-forest-labs/flux, 2024. 2

[24] Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, Kyle Lacey, Yam Levi, Cheng Li, Dominik Lorenz, Jonas Müller, Dustin Podell, Robin Rombach, Harry Saini, Axel Sauer, and Luke Smith. Flux.1 kontext: Flow matching for in-context image generation and editing in latent space, 2025. 2

[25] Jiabao Lei, Jiapeng Tang, and Kui Jia. RGBD2: Generative scene synthesis via incremental view inpainting using RGBD diffusion models. In *CVPR*, 2023. 2

[26] Renjie Li, Panwang Pan, Bangbang Yang, Dejia Xu, Shijie Zhou, Xuanyang Zhang, Zeming Li, Achuta Kadambi, Zhangyang Wang, Zhengzhong Tu, et al. 4K4DGen: Panoramic 4D generation at 4K resolution. In *ICLR*, 2025. 2, 3

[27] Zhengqi Li, Qianqian Wang, Noah Snavely, and Angjoo Kanazawa. Infinitenature-Zero: Learning perpetual view generation of natural scenes from single images. In *ECCV*, 2022. 1, 2

[28] Huan Ling, Seung Wook Kim, Antonio Torralba, Sanja Fidler, and Karsten Kreis. Align your gaussians: Text-to-4D with dynamic 3D gaussians and composed diffusion models. In *CVPR*, 2024. 2, 3

[29] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. DL3DV-10K: A large-scale scene dataset for deep learning-based 3D vision. In *CVPR*, 2024. 3

[30] Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite Nature: Perpetual view generation of natural scenes from a single image. In *ICCV*, 2021. 1, 2

[31] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. SyncDreamer: Generating multiview-consistent images from a single-view image. In *ICLR*, 2024. 2

[32] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2

[33] 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. 7

[34] Zijie Pan, Zeyu Yang, Xiatian Zhu, and Li Zhang. Efficient4D: Fast dynamic 3D object generation from a single-view video. *arXiv preprint arXiv 2401.08742*, 2024. 3

[35] Stefan Popov, Amit Raj, Michael Krainin, Yuanzhen Li, William T. Freeman, and Michael Rubinstein. CamCtrl3D: Single-image scene exploration with precise 3D camera control. In *3DV*, 2025. 2, 7

[36] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 7, 8

[37] 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. 2, 3, 4, 6, 7

[38] Jiawei Ren, Liang Pan, Jiaxiang Tang, Chi Zhang, Ang Cao, Gang Zeng, and Ziwei Liu. DreamGaussian4D: Generative 4D gaussian splatting. *arXiv preprint arXiv:2312.17142*, 2023. 3

[39] Xuanchi Ren, Tianchang Shen, Jiahui Huang, Huan Ling, Yifan Lu, Merlin Nimier-David, Thomas Müller, Alexander Keller, Sanja Fidler, and Jun Gao. GEN3C: 3d-informed world-consistent video generation with precise camera control. In *CVPR*, 2025. 3

[40] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 2

[41] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10684–10695, 2022. 2

[42] Wenqiang Sun, Shuo Chen, Fangfu Liu, Zilong Chen, Yueqi Duan, Jun Zhang, and Yikai Wang. DimensionX: Create any 3D and 4D scenes from a single image with controllable video diffusion. *arXiv preprint arXiv:2411.04928*, 2024. 2, 3

[43] Fengrui Tian, Shaoyi Du, and Yueqi Duan. MonoNeRF: Learning a generalizable dynamic radiance field from monocular videos. In *ICCV*, 2023. 2

[44] Fengrui Tian, Yueqi Duan, Angtian Wang, Jianfei Guo, and Shaoyi Du. Semantic Flow: Learning semantic fields of dynamic scenes from monocular videos. In *ICLR*, 2024. 2

[45] Basile Van Hoorick, Rundi Wu, Ege Ozguroglu, Kyle Sargent, Ruoshi Liu, Pavel Tokmakov, Achal Dave, Changxi Zheng, and Carl Vondrick. Generative camera dolly: Extreme monocular dynamic novel view synthesis. In *ECCV*, 2024. 3

[46] Fu-Yun Wang, Xiaoshi Wu, Zhaoyang Huang, Xiaoyu Shi, Dazhong Shen, Guanglu Song, Yu Liu, and Hongsheng Li. Be-your-outpainter: Mastering video outpainting through input-specific adaptation. In *ECCV*, 2024. 5, 7, 8

[47] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D vision made easy. In *CVPR*, 2024. 2

[48] 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 generation. In *ACM SIGGRAPH*, 2023. 7, 8

[49] Rundi Wu, Ruiqi Gao, Ben Poole, Alex Trevithick, Changxi Zheng, Jonathan T Barron, and Aleksander Holynski. Cat4D: Create anything in 4D with multi-view video diffusion models. *arXiv preprint arXiv:2411.18613*, 2024. 2, 3, 8

[50] Yiming Xie, Chun-Han Yao, Vikram Voleti, Huaizu Jiang, and Varun Jampani. SV4D: Dynamic 3D content generation with multi-frame and multi-view consistency. *arXiv preprint arXiv:2407.17470*, 2024. 2, 3

[51] Jinbo Xing, Menghan Xia, Yong Zhang, Haoxin Chen, Wangbo Yu, Hanyuan Liu, Gongye Liu, Xintao Wang, Ying Shan, and Tien-Tsin Wong. Dynamicrafter: Animating open-domain images with video diffusion priors. In *ICLR*, 2024. 3

[52] Jihan Yang, Shusheng Yang, Anjali W. Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in Space: How multimodal large language models see, remember and recall spaces. *arXiv preprint arXiv:2412.14171*, 2024. 2

[53] Shiyuan Yang, Liang Hou, Haibin Huang, Chongyang Ma, Pengfei Wan, Di Zhang, Xiaodong Chen, and Jing Liao.

Direct-a-Video: Customized video generation with user-directed camera movement and object motion. In *SIGGRAPH*, 2024. 3

[54] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. CogVideoX: Text-to-video diffusion models with an expert transformer. In *ICLR*, 2024. 2, 3, 4, 7

[55] Yuyang Yin, Dejia Xu, Zhangyang Wang, Yao Zhao, and Yunchao Wei. 4DGen: Grounded 4D content generation with spatial-temporal consistency. *arXiv preprint arXiv:2312.17225*, 2023. 3

[56] Heng Yu, Chaoyang Wang, Peiye Zhuang, Willi Menapace, Aliaksandr Siarohin, Junli Cao, Laszlo A Jeni, Sergey Tulyakov, and Hsin-Ying Lee. 4Real: Towards photorealistic 4D scene generation via video diffusion models. In *NeurIPS*, 2024. 2, 3, 8

[57] Hong-Xing Yu, Haoyi Duan, Charles Herrmann, William T Freeman, and Jiajun Wu. WonderWorld: Interactive 3D scene generation from a single image. *arXiv preprint arXiv:2406.09394*, 2024. 2, 5, 6

[58] 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, 2, 5, 6, 7, 8

[59] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023. 6

[60] Yuyang Zhao, Chung-Ching Lin, Kevin Lin, Zhiwen Yan, Linjie Li, Zhengyuan Yang, Jianfeng Wang, Gim Hee Lee, and Lijuan Wang. GenXD: Generating any 3D and 4D scenes. In *ICLR*, 2025. 3