WonderJourney: Going from Anywhere to Everywhere

Hong-Xing Yu¹ Haoyi Duan¹ Junhwa Hur² Kyle Sargent¹ Michael Rubinstein²
William T. Freeman² Forrester Cole² Deqing Sun² Noah Snavely² Jiajun Wu¹ Charles Herrmann²
¹Stanford University ²Google Research

Figure 1. We propose WonderJourney—generating a sequence of diverse yet coherent 3D scenes from a text description or an arbitrary image such as photos or generated art (“from anywhere”). WonderJourney can generate various journeys (which we refer to as “wonderjourneys”) for a fixed input, potentially ending “everywhere” (Fig. 4). We show rendered images along the generated sequence of 3D scenes. We strongly encourage the reader to see video examples at https://kovenyu.com/WonderJourney/.
Abstract

We introduce WonderJourney, a modular framework for perpetual 3D scene generation. Unlike prior work on view generation that focuses on a single type of scenes, we start at any user-provided location (by a text description or an image), and generate a journey through a long sequence of diverse yet coherently connected 3D scenes. We leverage an LLM to generate textual descriptions of the scenes in this journey, a text-driven point cloud generation pipeline to make a compelling and coherent sequence of 3D scenes, and a large VLM to verify the generated scenes. We show compelling, diverse visual results across various scene types and styles, forming imaginary “wonderjourneys”. Project website: https://kovenyu.com/WonderJourney/.

“No, no! The adventures first, explanations take such a dreadful time.” – Alice’s Adventures in Wonderland

1. Introduction

In “Alice’s Adventures in Wonderland”, the story begins with Alice falling down the rabbit hole and emerging into a strange and captivating Wonderland. In her journey through this wonderland, Alice encounters many curious characters such as the Cheshire Cat and the Mad Hatter, and peculiar scenarios, such as the tea party and the rose garden – eventually ending at the royal palace. These characters and settings combine to form a compelling world that has fascinated countless readers over the years. In this paper, we follow in this creative tradition and explore how modern computer vision and AI models can similarly generate such interesting and varied visual worlds, which users can journey through, much like Alice did in her own adventures in Wonderland.

Toward this goal, we introduce the problem of perpetual 3D scene generation. Unlike previous work on perpetual view generation [21, 24] that only generates a single type of scene, such as nature photos, our objective is to synthesize a series of diverse 3D scenes starting at an arbitrary location specified via a single image or language description. The generated 3D scenes should be coherently connected along a long-range camera trajectory, traveling through various plausible places. The generated 3D scenes allow rendering a fly-through video through a long series of diverse scenes to simulate the visual experience of a journey in an imaginary “wonderland”. We show examples in Fig. 1.

The main challenges of perpetual 3D scene generation center around generating diverse yet plausible scene elements. These scene elements need to support the formation of a path through coherently connected 3D scenes. They should include various objects, backgrounds, and layouts that fit in observed scenes and transit naturally to the next, yet unobserved, scenes. This generation process can be broken down into determining what objects to generate for a given scene; where to generate these objects; and how these scenes connect to each other geometrically. Determining what elements to generate calls for semantic understanding of a scene (e.g., a lion might not be a great fit for a kitchen); determining where to generate them calls for common sense regarding the visual world (e.g., a lion should not be floating in the sky); further, generating these elements in a new connected scene requires geometric understanding (e.g., occlusion and disocclusion, parallax, and appropriate spatial layouts). Notably, these challenges are absent in prior work on perpetual view generation, as they tend to focus on a single domain [24] or a single scene [11].

Our proposed solution, WonderJourney, addresses each of the above challenges in perpetual 3D scene generation with its own module. WonderJourney leverages an LLM to generate a long series of scene descriptions, followed by a text-driven visual scene generation module to produce a series of colored point clouds to represent the connected 3D scenes. Here, the LLM provides commonsense and semantic reasoning; the vision module provides visual and geometric understanding and the appropriate 3D effects. In addition, we leverage a vision-language model (VLM) to verify the generation and launch a re-generation when it detects undesired visual effects. Our framework is modular. Each of the modules can be implemented by the best pretrained models, allowing us to leverage the latest and future advancements in the rapidly growing vision and language research. Our main contributions are as follows:

• We study perpetual 3D scene generation and propose WonderJourney, a modular framework that decomposes this problem into core components: an LLM to generate scene descriptions, a text-driven visual module to generate the coherent 3D scenes, and a VLM to verify the generated scenes.

• We design a visual scene generation module that leverages off-the-shelf text-to-image and depth estimation models to generate coherent 3D point clouds. Our module handles boundary depth inaccuracy, sky depth inaccuracy, and disocclusion unawareness.

• We show compelling visual results and compare WonderJourney with SceneScape [11] and InfiniteNature-Zero [21] in a user study, which shows that WonderJourney produces interesting and varied journeys.

2. Related Work

Perpetual view generation. The seminal work on perpetual view generation is Infinite Images [18] which synthesizes the effect of navigating a 3D world by stitching and rendering images according to camera motions. Later works, such as Infinite Nature [24] and InfiniteNature-Zero [21], learn to auto-regressively generate next view based on the current view. Follow-up works improve global 3D consistency [5] and visual quality [4]. A recent work, SceneScapes [11],
explores text-driven perpetual view generation by gradually constructing a single cave-like scene represented by a lengthy mesh. While these methods study perpetual generation, they are restricted to a single domain such as nature photos [21, 24] or a single scene [11].

**3D scene generation.** Considerable progress has recently been made in text-to-3D or image-to-3D generation, many of which focus on objects without background [9, 20, 22, 26, 27, 33, 34]. These works typically leverage a 2D image prior from an image generation model, e.g., an image diffusion model [36], and then build a 3D representation, such as a NeRF [42], by distilling the supervision of the 2D image generation model [33]. Other works on 3D object generation focus on learning a 3D generative model directly from 2D images [6, 7, 13, 14, 29, 30, 32, 37, 39]. Several works also focus on generating a single 3D scene with background [2, 10]. For example, Text2Room [15] generates a room-scale 3D scene from a single text prompt, using textured 3D meshes for their scene representation. Other relevant works have focused on generating (sometimes called “reconstructing”) a scene from limited observations, such as a single image. Long-Term Photometric Consistent NVS [43] generates single scenes from a source image by auto-regressively generating with a conditional diffusion model. GeNVS [8] and Diffusion with Forward Models [40] use an intermediate 3D representation but are trained and evaluated on each scene separately. ZeroNVS [38] synthesizes a NeRF of a scene from a single image. Their work focuses on generating a single scene while ours targets generating a coherently connected sequence of diverse scenes.

**Text-guided video generation.** The idea of scene generation has also been explored in video generation. Several concurrent works like TaleCrafter [12] and others [16, 23, 25] also discuss the task of creating a series of videos which follow an LLM-generated story or script. However, all these works use different scenes as different clips in a video, resulting in hard cuts, while our perpetual 3D scene generation aims at generating sequences of coherently connected scenes.

### 3. Approach

Our goal is to generate a potentially endless sequence of diverse yet coherently connected 3D scenes, which requires both geometric understanding of 3D scenes and visual common sense and semantic understanding of scene structures. To tackle this challenging task, we propose WonderJourney. The main idea is to generate a text description of the visual elements the next scene would contain, and then employ a text-guided visual generation module to make that 3D scene.

WonderJourney is a modular framework that decomposes this task into scene description generation, visual scene generation, and visual validation, as in Fig. 2. Given an input image or text, we first pair it with the other modality by generating the image with a text-to-image model or by generating the description with a Vision Language Model (VLM). Then we generate the next-scene description by a Large Language Model (LLM). A visual scene generation module takes in the next-scene description and the current scene image to generate the next 3D scene represented by a colored point cloud. This generation process is then checked by a VLM to make sure there is no unwanted effects, or it gets regenerated. We note that our framework is modular such that each module can be implemented with the latest (pretrained) models and thus can easily leverage the quick advances of large language and vision models.

#### 3.1. Scene description generation

We propose an auto-regressive scene description generation process, i.e., the scene description generation takes a set of past and current scene descriptions as input and predicts the subsequent scene description:

$$S_i = g_{\text{description}}(\mathcal{J}, \mathcal{M}_i),$$

(1)
New scene point cloud \rightarrow \text{Current scene image} \rightarrow \text{Estimated depth} \rightarrow \text{Refined depth} \rightarrow \text{Point cloud} \rightarrow \text{Occlusion-resolved depth} \rightarrow \text{Refined depth} \rightarrow \text{New scene image}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig3.png}
\caption{The visual scene generation module. Each arrow represents a parametric vision model (e.g., a depth estimator) or an operation (e.g., rendering). Our fully modular design easily benefits from advances in the corresponding research topics.}
\end{figure}

where \( S_i \) denotes the \( i \)th scene description, and \( g_{\text{description}} \) denotes the scene description generator which is implemented by an LLM that takes two inputs: 1) the task specification \( T = \text{"You are an intelligent scene generator. Please generate 3 most common entities in the next scene, along with a brief background description."} \); and 2) the scene description memory \( \mathcal{M}_i = \{ S_0, S_1, \cdots, S_i \} \) which is a collection of past and current scenes. The latest description memory is:

\[
\mathcal{M}_{i+1} = \mathcal{M}_i \cup \{ S_{i+1} \}. \tag{2}
\]

We define the scene description \( S_i = \{ S, O_i, B_i \} \), which consists of a style \( S \) that is kept consistent across scenes, objects in the scene \( O_i \), and a concise caption \( B_i \) describing the background of the scene. We allow the LLM to output natural language descriptions, and then use a lexical category filter to process the raw text of \( O_i \) and \( B_i \) such that we only keep nouns for entities and adjectives for attributes. Empirically this generates more coherently connected scenes compared to requiring the LLM to directly output this structured description.

### 3.2. Visual scene generation

Since we want the generated next scene to be coherent with past scenes geometrically and semantically, we formulate our visual scene generation as a conditional generation problem, taking both the next-scene description and the 3D representation of the current scene as conditions:

\[
P_{i+1} = g_{\text{visual}}(I_i, S_{i+1}), \tag{3}
\]

where \( P_i \) denotes a colored point cloud that represents the next 3D scene, and \( I_i \) denotes the image of current scene. The visual scene generator \( g_{\text{visual}} \) consists of learning-free operations such as perspective unprojection and rendering, as well as components that use parametric (pretrained) vision models, including a depth estimator, a segmentation-based depth refiner, and a text-conditioned image outpainter. We show an illustration in Fig. 3.

#### Lifting image to point cloud

Given the current scene represented by an image \( I_i \), we lift it to 3D by estimating depth and unproject it with a pinhole camera model. We use MIDAS v3.1 [35], one of the state-of-the-art depth estimators, in our experiments. However, we find that existing monocular depth estimators share two common issues. First, depth discontinuity is not well modeled, witnessed by previous work [1, 28, 41], resulting in overly smooth depth edges across object boundaries. Second, the depth of the sky is always underestimated, also observed by previous work [21, 24]. To address these two issues, we introduce a depth refinement process that leverages pixel grouping segments and sky segmentation.

#### Depth refinement

To enhance the depth discontinuity across object boundaries, we model scene elements with frontal planes when the elements have a limited disparity range. We use SAM [19] to generate pixel grouping segments \( \{ \text{seg}_j \}_{j=1}^{N_s} \) where \( \text{seg}_j \in \{0, 1\}^{H \times W} \) is a segment mask, sorted in descending order according to the size of the segment \( \|	ext{seg}_j\| \). We iteratively refine the estimated depth:

\[
\text{depth[seg}_j] \leftarrow \begin{cases} 
\text{median(depth[seg}_j]), & \text{if } \Delta D_j < T, \\
\text{depth[seg}_j], & \text{otherwise,}
\end{cases} \tag{4}
\]

for \( j = 1, \cdots, N_s \), where \( \text{depth} \in \mathbb{R}^{H \times W} \) is initialized with the estimated monocular depth, \( \text{median}() \) is a function that returns the median value of the input set, \( \Delta D_j = \max(\text{disparity[seg}_j]) - \min(\text{disparity[seg}_j]) \) denotes the disparity (the reciprocal of depth) range within the segment \( \text{seg}_j \). We keep the estimated depth of segments with high disparity range as they do not fit to a frontal-plane, such as roads. Note that the idea of frontal-plane modeling...
has also been explored in 3D Ken-Burns [31] with selected semantic classes such as car and people. In contrast, as we target at general scenes with diverse styles, we use the criterion of the disparity range for keeping estimated depth instead of selected semantic classes.

To handle the sky depth which is always underestimated, we use OneFormer [17] to segment sky region and assign a high depth value to it. However, this results in inaccurate depth estimates along the sky boundary; if we were to naively use the output segmentation, these errors result in accumulated severe artifacts in later scenes. To resolve it, we simply remove points along the sky boundary. Besides, we find that depth at distant pixels are generally not reliable. Thus, we also set a far background plane with depth $F$ that cuts off all pixels’ depth beyond it.

**Description-guided scene generation.** To generate a new scene that is connected to the current scene, we place a camera $C_{i+1}$ with an appropriate distance to the current camera $C_i$. As shown in Fig. 3, we render the partial image $I_{i+1}$ (more details on the camera and the renderer are in Appendix E) and outpaint it with a text-guided outpainter to

---

**Figure 4.** Qualitative results for diverse journeys generated from the same input image, showing that WonderJourney can go everywhere. The input in the top example is a real photo.
generate a new scene image $I_{i+1}$:

$$I_{i+1} = g_{\text{outpaint}}(\hat{I}_{i+1}, S_{i+1}),$$

where we use the Stable Diffusion model [36] for $g_{\text{outpaint}}$ in our experiments. Note, we purposefully place the new camera at a location that creates enough empty space in the rendered image. We empirically find that text-guided outpainters tend to avoid generating partial objects, likely due to their curated training image datasets, which tend to not include truncated or partial objects. Leaving too little empty space therefore results in just a simple extrapolation of the partial image $\hat{I}_{i+1}$ and a lack of adherence to the text prompt $S_{i+1}$, especially in regards to new objects. After generating the new scene image, we lift it to 3D by estimating and refining depth for it, and we obtain the new point cloud $P_{i+1} = P_i \cup P^\prime_{i+1}$ where $P^\prime_{i+1}$ denotes the additional points from unprojecting the outpainted pixels.

**New scene registration by depth consistency.** However, as the depth estimator is unaware of geometry constraints, the depth for points in $P^\prime$ generally do not align with $P_i$. Thus, we adapt the depth estimator by a depth alignment loss:

$$L_{\text{depth}} = \max(0, D^*_\text{bg} - D^*_\text{fg}) + \|D^*_\text{fg} - D^\prime_{\text{fg}}\|,$$

where $D^*_{\text{bg}}$ denotes the analytically computed depth of background pixels from $I_i$, $D^*_{\text{fg}}$ denotes the estimated depth for pixels corresponding to $D^*_{\text{bg}}$, $D^\prime_{\text{fg}}$ denotes the computed depth of all other visible pixels from $I_i$, and $D^\prime_{\text{fg}}$ denotes the estimated depth for pixels corresponding to $D^*_{\text{fg}}$.

**Occlusion handling by re-rendering consistency.** Another geometric inconsistency is that disocclusion regions can have a lower depth values than their occluders, as the depth estimator is not aware of this 3D geometric constraint. We highlight the wrongly estimated disocclusion depth in the refined depth in Fig. 3. To resolve this issue, we re-render the new scene $P_{i+1}$ at the camera $C_i$ and detect all inconsistent pixels. At each inconsistent pixel, we move back all the rasterized additional points from $P^\prime_{i+1}$ that have lower depth values than the one point from $P_i$. This removes the disocclusion inconsistency and guarantees that the disocclusion comes after the occluder.

**Scene completion.** We obtain the final point cloud $P_{i+1}$ by adding more points to $P_{i+1}$. We add points by repeating the following “complete-as-you-go” process: we place an additional camera along a camera trajectory connecting the new scene to the current scene, render a partial image at that camera, outpaint the image, and add the additional points to the point cloud. Note that in our visual scene generation formulation in Equation 3, one can replace the image input $I_i$ with the point cloud $P_i$ from the current scene, forming a persistent scene representation. This allows a trade-off between 3D persistence and empirical requirements. In practice, maintaining a large point cloud leads to prohibitively many points that require a large amount of GPU memory when generating a long trajectory of high-resolution scenes. Thus in our experiments we take the image formulation.

### 3.3. Visual validation

Empirically, in a large portion of generated photos and paintings, a painting frame or a photo border appears, disrupting the geometry consistency. Additionally, there are often unwanted blurry out-of-focus objects near the borders of the generated images. Thus, we propose a validation step to identify and reject these undesired generated scenes.

We formulate this as a text-based detection problem, where our objective is to detect a set of predefined undesirable effects in the generated scene image. We reject and regenerate the scene image if any unwanted effect is detected. Specifically, right after we generate a new scene image $I_{i+1}$, we immediately feed it to a VLM and prompt it with the query $J^{\text{detect}} = "Is there any $X_t$ in this image?"$ where $X_t \in \{X_1, \ldots, X_T\}$ is an unwanted effect specified by text, such as “photo border”, “painting frame”, or “out-of-focus objects”. If any unwanted effect is detected, we regenerate $I_{i+1}$ with a new description $S_{i+1}$ or a new random seed.

### 4. Experiments

**Dataset and baselines.** Since the perpetual 3D scene generation is a new task without an existing dataset, we use a mixture of photos taken by ourselves, copyright-free photos from online, and generated examples, for evaluation in our experiments. We perform the pairing process by DALL-E 3 [3] for text-to-image pairing. We consider two state-of-the-art perpetual view generation methods as our baseline: image-based InfiniteNature-Zero and text-based SceneScape.

**Qualitative demonstration.** We show qualitative examples of the generated journey across different scenes and different styles in Fig. 1 and Fig. 5. These results show that WonderJourney is able to generate diverse yet coherently connected scenes from various types of input images, i.e., it can go from anywhere. We show more examples in the Appendix. We further show examples of diverse generation samples from the same input in Fig. 4. These diverse generated journeys suggest that WonderJourney supports going to different destinations at each run.

**Additional evaluations.** We show additional qualitative results in Appendix B, longer “wonderjourneys” (up to 30 scenes) in Appendix C, controlled “wonderjourneys” (i.e., using user-provided full text, such as poems and haiku, instead of LLM-generated text guidance) in Appendix D, and ablation studies in Appendix F.

**Human preference evaluation.** Since a main application of WonderJourney is for creative and entertainment purposes, we focus on human preference evaluation as our quanti-
Figure 5. From diverse starting scenes with different styles, WonderJourney generates a sequence of diverse yet coherent 3D scenes, showing that it can go from anywhere to everywhere (e.g., nature, village, city, indoor, or fantasy). The inputs in top two rows are real photos. We strongly encourage the reader to see the video results in the project website.
Table 1. Human preference of ours over baseline on diversity, visual quality, scene complexity, and overall interesting-ness.

<table>
<thead>
<tr>
<th></th>
<th>Div.</th>
<th>Qual.</th>
<th>Compl.</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours over InfiniteNature-Zero</td>
<td>92.7%</td>
<td>94.9%</td>
<td>91.5%</td>
<td>88.6%</td>
</tr>
<tr>
<td>Ours over SceneScape</td>
<td>88.8%</td>
<td>83.4%</td>
<td>80.0%</td>
<td>90.3%</td>
</tr>
</tbody>
</table>

Figure 6. Comparison with InfiniteNature-Zero [21] and SceneScape [11]. Note that InfiniteNature-Zero is trained on nature photos, so we only compare with it using photorealistic nature images as input. 

Table 1. Human preference of ours over baseline on diversity, visual quality, scene complexity, and overall interesting-ness.

We introduce WonderJourney to generate a long sequence of diverse yet coherently connected 3D scenes starting at any user provided location. WonderJourney achieves compelling, diverse visual results across various scene types and different styles, enabling users to journey through their own adventures in the generated “wonderjourneys”.

Acknowledgments. This work was supported by NSF RI #2211258, ONR N00014-23-1-2355, and ONR YIP N00014-24-1-2117. The work was done in part when Hong-Xing Yu was a student researcher at Google and has been supported by gift funding and GCP credits from Google.
References


[25] Chang Liu, Haoming Wu, Yujie Zhong, Xiaoyun Zhang, and Weidi Xie. Intelligent grimm–open-ended visual storytelling


