WonderWorld: *Interactive* 3D Scene Generation from a Single Image

Supplementary Material

A. Overview

In this supplementary material, we show the following contents:

- Algorithms of WonderWorld (B)
- Details on guided depth diffusion (C)
- Further implementation details (D)
- Additional experiment results (E)

We also compile video results and interactive viewing examples of the generated virtual worlds in `https://kovenyu.com/WonderWorld/`. We strongly encourage the reader to view the project website.

B. Algorithms

We summarize the control loop of WonderWorld in Alg. 1 and the generation of FLAGS in Alg. 2 and Alg. 3.

C. Details on Guided Depth Diffusion

Accelerated depth guidance implementation. In our guided depth diffusion, we empirically observe that we do not need to use guidance in every denoising step. We set the guidance weights s_t such that the norm of the guidance signal is proportional to the norm of the predicted update. We use the Euler scheduler [26] with 30 steps for our depth diffusion, where we apply our guidance in only the last 8 steps. This significantly reduces runtime latency.

Relation to other guidance methods. The guidance technique has been used in sampling diffusion models with different guidance signals, such as text [19], features [12], and decoded features [39]. Yet, their goal is to control the semantic contents of generated images. Our guided depth diffusion targets a problem different from controllable image generation; we aim to estimate consistent depth that aligns with the existing depth geometry.

Tackling ground plane distortion. We note that our guided depth diffusion formulation is highly flexible and allows us to specify different depth constraints. For example, a significant geometric distortion is that the ground plane is often curved due to inaccurate camera intrinsic matrix and depth estimation. Thus, we add depth guidance for the ground plane by replacing the mask $\mathbf{M}_{\text{guide}}$ in Eq. 9 with a ground mask $\mathbf{M}_{\text{grd}}$ obtained from semantic segmentation, and replacing the depth of visible content $\mathbf{D}_{\text{guide}}$ with an analytically calculated flat ground depth $\mathbf{D}_{\text{grd}}$. To compute depth, we assume the height difference H_{cam} between the camera and the ground; then the depth of a ground pixel is $H_{\text{cam}} f_{\text{y}}/(p_{\text{y}} - y)$, where f_{y} is the focal length, y is the pixel y-coordinate, p_{y} is the y-principal point.

D. Further Experiment Details

Real photos. In our experiments, we use both real photos and synthetic stylized images. The following results use real photos as input: **(I)** "Holy Spirit Cathedral", "Ho Chi Minh City Hall", and "Marienplatz" in the *Interactive Scene Generation* section of the project website; **(II)** "Venice", "Main Square", "University Campus", "Arc de Triomf", "Segovia Cathedral", "Westlake", and "University Pathway" in the *Generated Virtual World* section of the project website; **(III)** The top example ("Venice") and bottom right example ("Main Square") in Fig. 1, the left example in Fig. 5, the 3rd example in Fig. 10, the 1st example in Fig. 11, the 1st example in Fig. 12, and the left example in Fig. 15.

Further implementation details. In single-view layer generation, we use an LLM to generate a structured scene description (Eq. 4). We use GPT-4 for this purpose, and the instruction prompt $\mathcal{J}$ is:

"You are an intelligent scene generator. Imagine you are wandering through a scene or a sequence of scenes, and there are 3 most significant common entities in each scene. The next scene you would go to is $\mathcal{U}$. Please generate the corresponding 3 most common entities in this scene. The scenes are sequentially interconnected, and the entities within the scenes are adapted to match and fit with the scenes. You must also generate a brief background prompt of about 50 words describing the scene. You should not mention the entities in the background prompt. If needed, you can make reasonable guesses. Please use the format below (the output should be JSON format): 'scene_name': ['scene_name'], 'entities': ['entity_1', 'entity_2', 'entity_3'], 'background': ['background prompt']",

where $\mathcal{U}$ is the user text input to specify the scene name. To generate the text prompt $\mathcal{T}$ in Eq. 4 for the first scene for inpainting the background layer and sky layer, we use a similar instruction to prompt the VLM (we use GPT-4V) to caption the input image, with the difference that we also ask the VLM to generate a style prompt $\mathcal{S}$. Then, we keep using the same style prompt $\mathcal{S}$ in Eq. 4 for the whole generation process. The "scene name" above is used to prompt the LLM to generate the next scene description. The "entities" above is used as the foreground prompt $\mathcal{F}$ in Eq. 4, and the "background prompt" is used as $\mathcal{B}$ in Eq. 4.

All generated scene images are 512×512 pixels. We set the camera focal length to $f_{\text{x}} = f_{\text{y}} = 960$ pixels for all scenes, while it is also possible to use off-the-shelf methods [24] for estimation. We post-process estimated depth using an efficient SAM [30, 38], similar to WonderJourney [67]. In practice, we generate the entire sky in the initial

Algorithm 1 WonderWorld control loop

```
**Input:** Initial scene image $\mathbf{I}_0$
**Output:** All generated scenes $\mathcal{G} = \{\mathcal{E}_0, \mathcal{E}_1, \ldots\}$
**Runtime output:** Real-time rendered image $\mathbf{I}_{\text{rend}}$
**Runtime user control:** Real-time camera pose $\mathbf{C}_{\text{rend}}$, generation camera pose $\mathbf{C}_{\text{gen}}$, (optional) user text prompt $\mathcal{U}$
1: $\mathbf{C}_{\text{rend}} \leftarrow$ 4x4 Identity matrix                          ▷ Initialize at origin
2: $\mathbf{C}_{\text{gen}} \leftarrow$ 4x4 Identity matrix                           ▷ Initialize at origin
3: $\mathbf{I}_{\text{scene}} \leftarrow \mathbf{I}_0$
4: $\mathbf{M} \leftarrow \mathbf{1}^{H \times W}$          ▷ Mask indicating which pixels are the current new scene
5: $\mathcal{T} \leftarrow \text{Captioning\_by\_VLM}(\mathbf{I}_{\text{scene}})$                 ▷ We use GPT4V
6: $\mathcal{G} \leftarrow \text{Generate\_FLAGS}(\mathbf{I}_{\text{scene}}, \mathbf{M}, \mathcal{T}, \emptyset)$                     ▷ Alg. 2
7: **in parallel do**
8:     **Thread 1:** Main control loop                          ▷ Async with generation
9:     **while** true **do**
10:        $\mathbf{I}_{\text{rend}} \leftarrow \text{Render}(\mathbf{C}_{\text{rend}}, \mathcal{G})$
11:        $\mathbf{C}_{\text{rend}} \leftarrow \text{Update\_by\_user}(\mathbf{C}_{\text{rend}})$          ▷ User can move, rotate, or stay static
12:    **end while**
13: **end parallel**
14: **in parallel do**
15:    **Thread 2:** Async generation signal (triggered event)
16:    $\mathbf{C}_{\text{gen}} \leftarrow \mathbf{C}_{\text{rend}}$
17:    $\mathbf{I}_{\text{partial}} \leftarrow \text{Render}(\mathbf{C}_{\text{gen}}, \mathcal{G})$                ▷ Partial rendered image
18:    $\mathbf{M} \leftarrow \text{Find\_empty\_pixels}(\mathbf{I}_{\text{partial}})$
19:    **if** $\mathcal{U}$ is empty **then**
20:        $\mathcal{U} \leftarrow \text{Propose\_by\_LLM}()$          ▷ We use GPT4 to propose a new scene name
21:        $\mathcal{T} \leftarrow \text{Generate\_by\_LLM}(\mathcal{U})$                                    ▷ Eq. 4
22:    **else**
23:        $\mathcal{T} \leftarrow \text{Generate\_by\_LLM}(\mathcal{U})$                                    ▷ Eq. 4
24:    **end if**
25:    $\mathbf{I}_{\text{scene}} \leftarrow \text{Outpaint}(\mathbf{I}_{\text{partial}}, \mathbf{M}, \mathcal{U})$
26:    $\mathcal{G} \leftarrow \text{Generate\_FLAGS}(\mathbf{I}_{\text{scene}}, \mathbf{M}, \mathcal{T}, \mathcal{G})$                   ▷ Alg. 2
27: **end parallel**
```

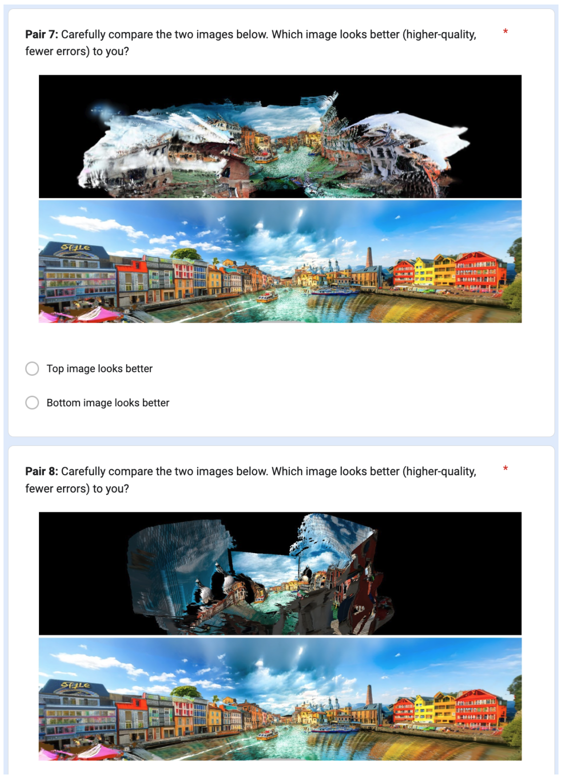
Algorithm 2 Generate FLAGS

```
**Input:** Scene image $\mathbf{I}_{\text{scene}}$, mask of new pixels $\mathbf{M}$, full text prompt $\mathcal{T} = \{\mathcal{F}, \mathcal{B}, \mathcal{S}\}$, existing scenes $\mathcal{G}$
**Output:** Extended scenes $\mathcal{G}$
1: $\mathbf{I}_{\text{fg}}, \mathbf{I}_{\text{bg}}, \mathbf{I}_{\text{sky}}, \mathbf{M}_{\text{fg}}, \mathbf{M}_{\text{bg}}, \mathbf{M}_{\text{sky}} \leftarrow$
   $\text{Generate\_layer\_images}(\mathbf{I}_{\text{scene}}, \{\mathcal{F}, \mathcal{B}, \mathcal{S}\})$   ▷ Sec. 3.1
2: $\mathbf{M}_{\text{init}} \leftarrow \mathbf{M} \odot \mathbf{M}_{\text{sky}}$
3: $\mathcal{L}_{\text{sky}} \leftarrow \text{Optimize\_layer}(\mathbf{I}_{\text{sky}}, \mathcal{G}, \mathbf{M}_{\text{init}})$        ▷ Alg. 3
4: $\mathcal{G} \leftarrow \mathcal{G} \cup \mathcal{L}_{\text{sky}}$          ▷ Add $\mathcal{L}_{\text{sky}}$ to the frozen $\mathcal{G}$
5: $\mathbf{I}'_{\text{bg}} \leftarrow \mathbf{M}_{\text{bg}} \odot \mathbf{I}_{\text{bg}} + (\mathbf{1} - \mathbf{M}_{\text{bg}}) \odot \mathbf{I}_{\text{sky}}$
6: $\mathbf{M}_{\text{init}} \leftarrow \mathbf{M} \odot \mathbf{M}_{\text{bg}}$
7: $\mathcal{L}_{\text{bg}} \leftarrow \text{Optimize\_layer}(\mathbf{I}'_{\text{bg}}, \mathcal{G}, \mathbf{M}_{\text{init}})$        ▷ Alg. 3
8: $\mathcal{G} \leftarrow \mathcal{G} \cup \mathcal{L}_{\text{bg}}$
9: $\mathbf{M}_{\text{init}} \leftarrow \mathbf{M} \odot \mathbf{M}_{\text{fg}}$
10: $\mathcal{L}_{\text{fg}} \leftarrow \text{Optimize\_layer}(\mathbf{I}_{\text{scene}}, \mathcal{G}, \mathbf{M}_{\text{init}})$     ▷ Alg. 3
11: $\mathcal{G} \leftarrow \mathcal{G} \cup \mathcal{L}_{\text{fg}}$
```

Algorithm 3 Optimize a FLAGS layer

```
**Input:** Reference image $\mathbf{I}_{\text{ref}}$, existing scenes $\mathcal{G}$, mask $\mathbf{M}_{\text{init}}$ to indicate which pixels are used to spawn surfels for this layer
**Output:** Layer $\mathcal{L}$
1: $\mathbf{D}_{\text{guide}}, \mathbf{M}_{\text{guide}} \leftarrow \text{Render\_partial\_depth}(\mathbf{C}_{\text{gen}}, \mathcal{G})$
2: $\mathbf{D}_{\text{scene}} \leftarrow \text{Guided\_depth\_diffusion}(\mathbf{I}_{\text{ref}}, \mathbf{D}_{\text{guide}}, \mathbf{M}_{\text{guide}})$
   ▷ Sec. 3.2
3: $\mathbf{N} \leftarrow \text{Estimate\_normal}(\mathbf{I}_{\text{ref}})$
4: $\mathbf{P}, \mathbf{C} \leftarrow \text{Unproject\_pixels}(\mathbf{I}_{\text{ref}}, \mathbf{D}_{\text{scene}}, \mathbf{M}_{\text{guide}})$   ▷ Eq. 6
5: $\mathbf{S} \leftarrow \text{Compute\_scales}(\mathbf{D}_{\text{scene}}, \mathbf{N}, \mathbf{K})$          ▷ Eq. 8
6: $\mathcal{L} \leftarrow \text{Initialize\_layer}(\mathbf{P}, \mathbf{N}, \mathbf{C}, \mathbf{S}, \mathbf{M}_{\text{init}})$
7: $\mathcal{L} \leftarrow \text{Optimize\_layer}(\mathcal{L}, \mathcal{G}, \mathbf{I}_{\text{ref}})$            ▷ Sec. 3.1
```

scene using SyncDiffusion [32] offline. To render the guidance mask $\mathbf{M}_{\text{guide}}$, we first render the FLAGS into the screen space, and accumulate the opacity. Then, we threshold the


Pair 7: Carefully compare the two images below. Which image looks better (higher-quality, fewer errors) to you? *

Top image looks better

Bottom image looks better

Pair 8: Carefully compare the two images below. Which image looks better (higher-quality, fewer errors) to you? *

Figure 9. Screenshot of human study survey.

accumulated opacity by 0.6 to find the visible mask $\mathbf{M}_{\text{guide}}$. We use the same method to find empty pixels in the partial rendered image $\mathbf{I}_{\text{partial}}$.

Human study details. We use Prolific to recruit participants for the human preference evaluation. For each experimental comparison, we recruit 204 participants from all over the globe. We use Google forms to present the survey. The survey is fully anonymized for both the participants and the host. Participants are shown top-by-bottom bird-eye rendering images of the same layout as in Fig. 5 with randomized top-bottom orders. For the ablation study, participants are shown side-by-side images. Participants are instructed to select one from two options: "Top is more visually compelling" or "Bottom is more visually compelling" The instruction is: "Carefully compare the two images below. Which image looks better (higher quality, fewer errors) to you?"

Since we compare to three baseline methods, each example forms three pairs. We use the four examples shown in Figure 5 and Figure 15, yielding 12 pairs in total. Each participant answers all 12 questions. We show a screenshot in Figure 9.

Depth estimation for baseline methods. For a fairer comparison, we also use Marigold for WonderJourney [67] in our experiments. Yet, LucidDreamer [8] requires metric depth, and Text2Room [20] requires depth inpainting, so we keep their original depth models.

Table 5. Time analysis for WonderWorld in generating a single scene on an A6000 GPU.

Outpainting	Layer generation	Depth	Normal	Optim.
2.1s	2.3s	2.5s	0.8s	1.9s

Table 6. Comparison of different depth alignment methods on our examples. The metric is the scale-invariant root mean square error (SI-RMSE) between the estimated depth and the existing depth.

w/o guided depth diffusion	Shift+Scale	Guided depth diffusion (ours)
0.36	0.21	0.08

E. Additional Results

We show additional baseline comparison results in Figure 15. We show additional qualitative results in Figure 11, 10, 12. To automate generation, we also use the panoramic camera paths. We use the LLM to generate the scene names.

We show different scenes using the same input image in Figure 13, and different styles in the same virtual world in Figure 14. In Table 5, we show a time analysis of WonderWorld for generating a single extrapolated scene.

Additional ablation study on guided depth diffusion. In Table 6, we show an ablation on guided depth diffusion. Besides "w/o guided depth diffusion" which does not have any treatment to align depth estimations, we further include a heuristic-based method "Shift+Scale" which uses the least square to solve for a shift value and a scale value that transforms the estimated depth to align with the existing depth. We use the same protocol as in the baseline comparison and main paper ablation study. We report the scale-invariant root mean square error (SI-RMSE) between the estimated depth and the visible existing depth. From Table 6, we observe that our guided depth diffusion provides much better alignment than the two variants.

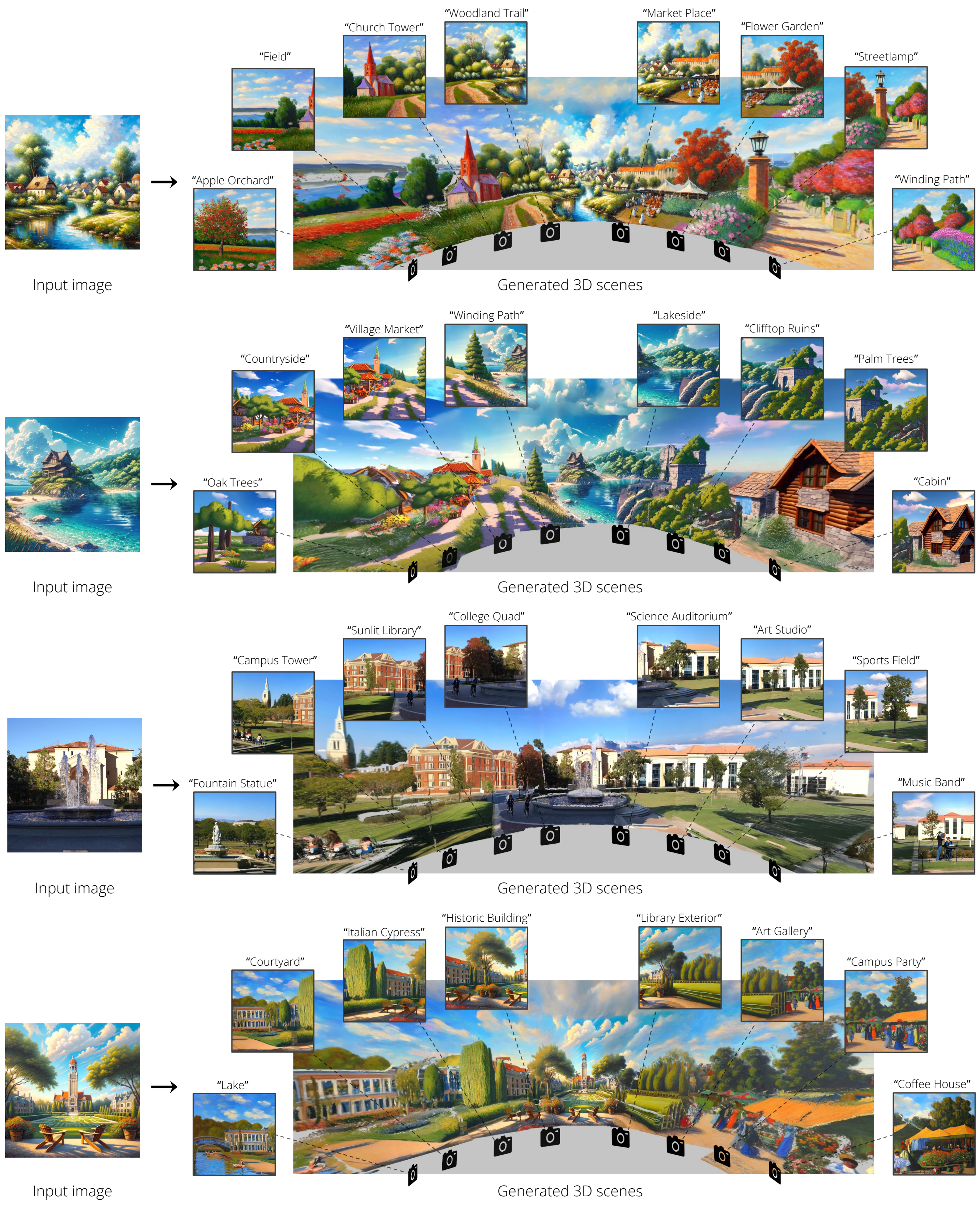


Figure 10. Qualitative examples. Each generated world consists of 9 scenes. The text prompts are generated by the LLM.

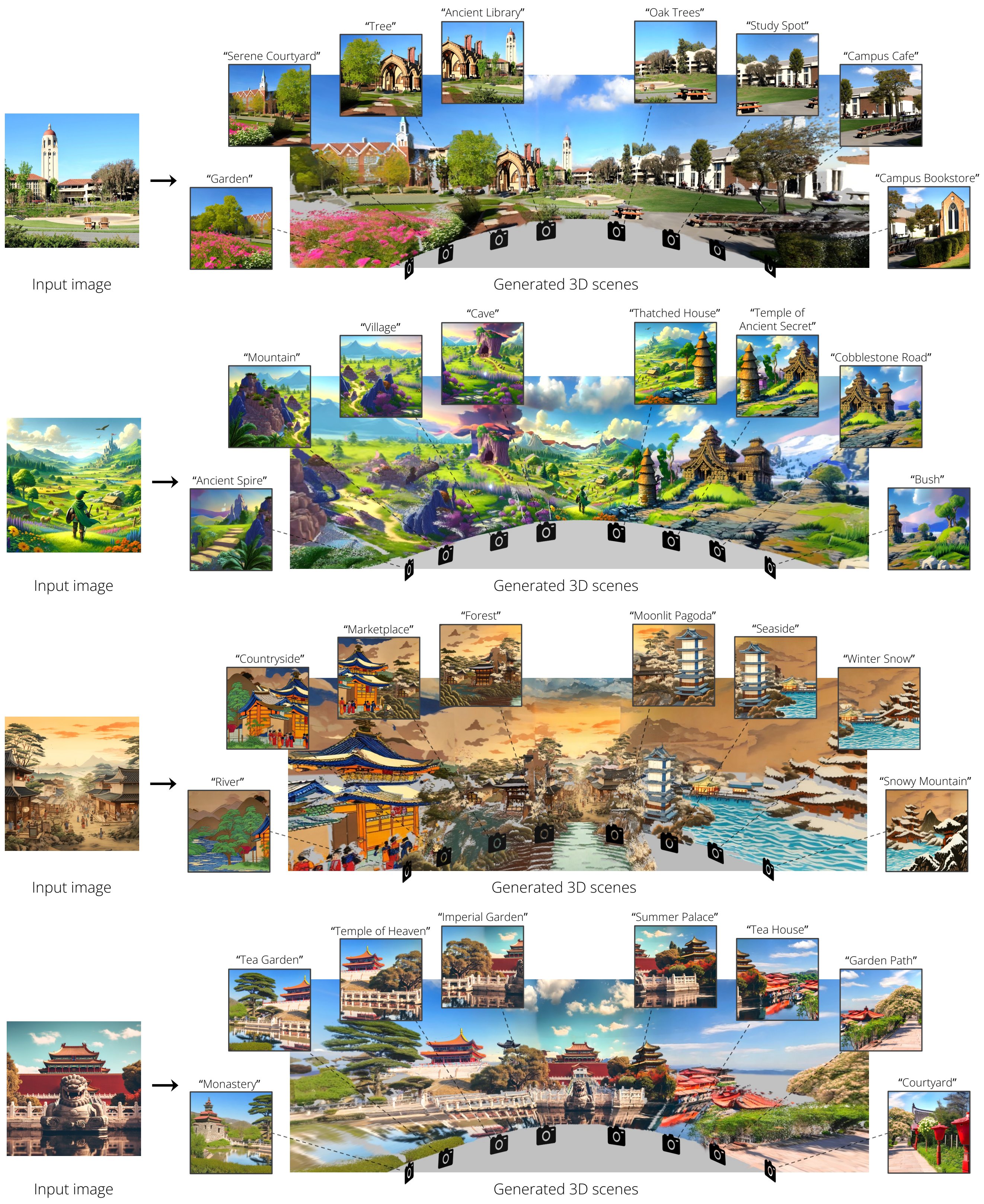


Figure 11. Qualitative examples. Each generated world consists of 9 scenes. The text prompts are generated by the LLM.

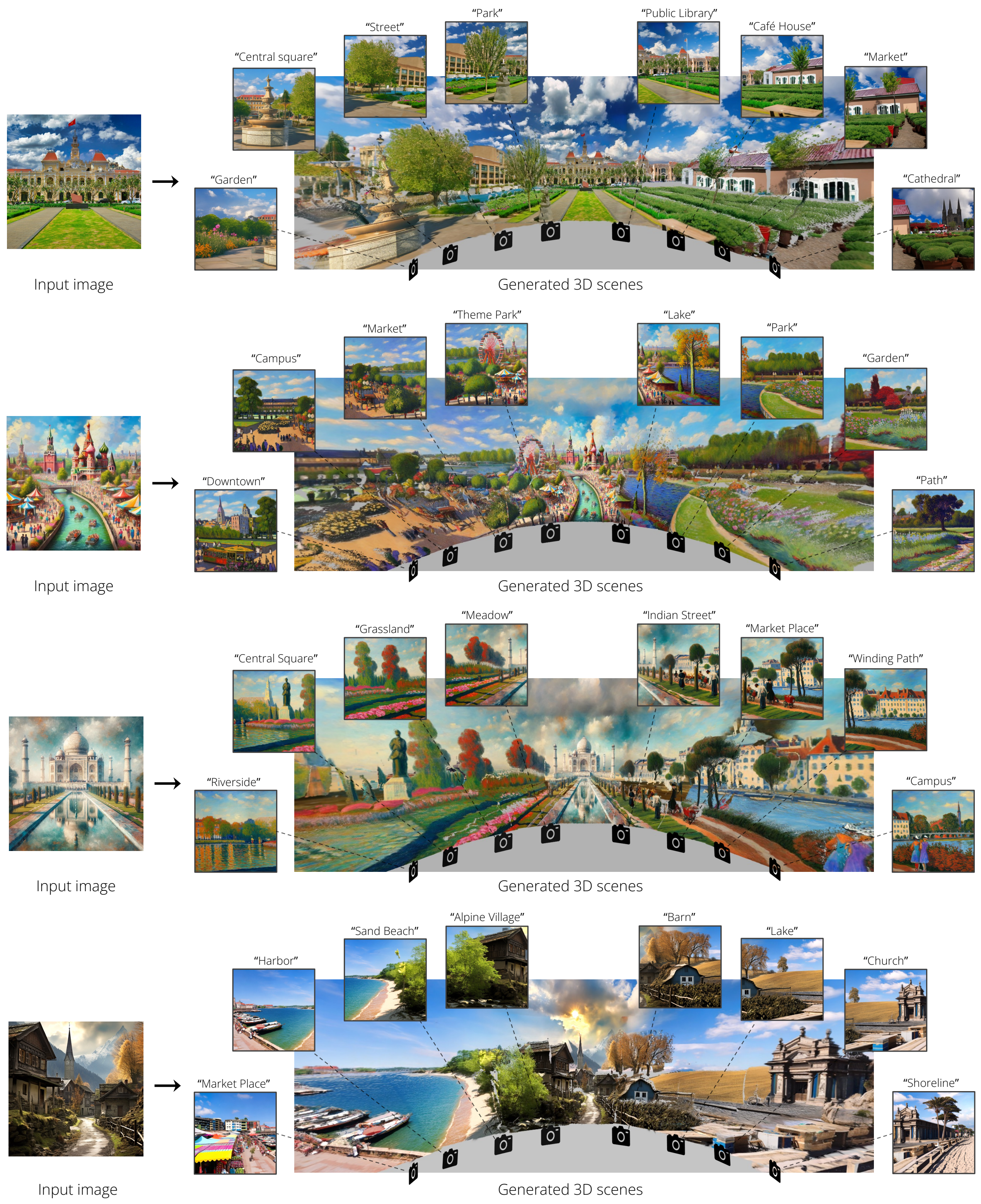


Figure 12. Qualitative examples. Each generated world consists of 9 scenes. The text prompts are generated by the LLM.

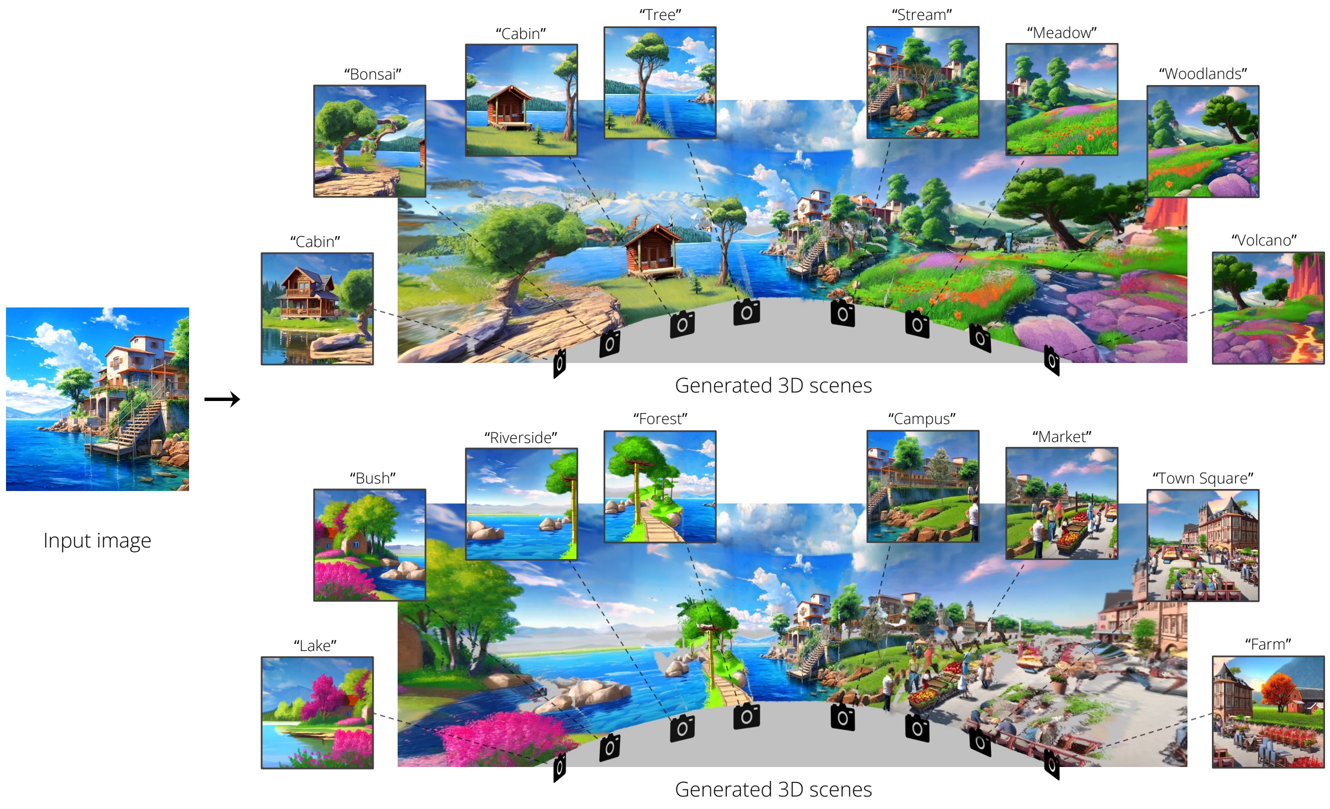


Figure 13. Diverse generation: Our WonderWorld allows generating different virtual worlds from the same input image.

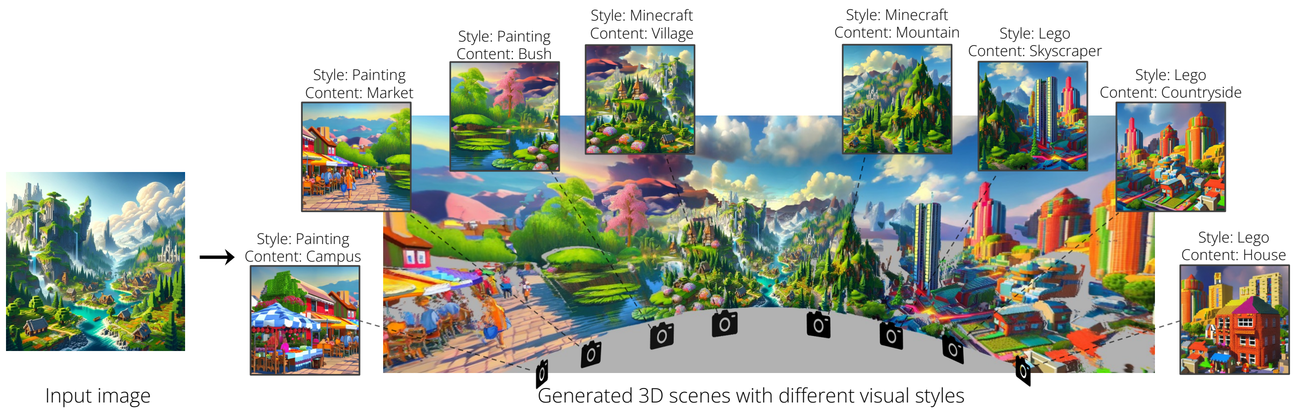

Input image
Style: Painting Content: Campus
Style: Painting Content: Market
Style: Painting Content: Bush
Style: Minecraft Content: Village
Style: Minecraft Content: Mountain
Style: Lego Content: Skyscraper
Style: Lego Content: Countryside
Style: Lego Content: House
Generated 3D scenes with different visual styles

Figure 14. WonderWorld allows users to specify different styles in the same generated virtual world, e.g., Minecraft, painting, and Lego styles.

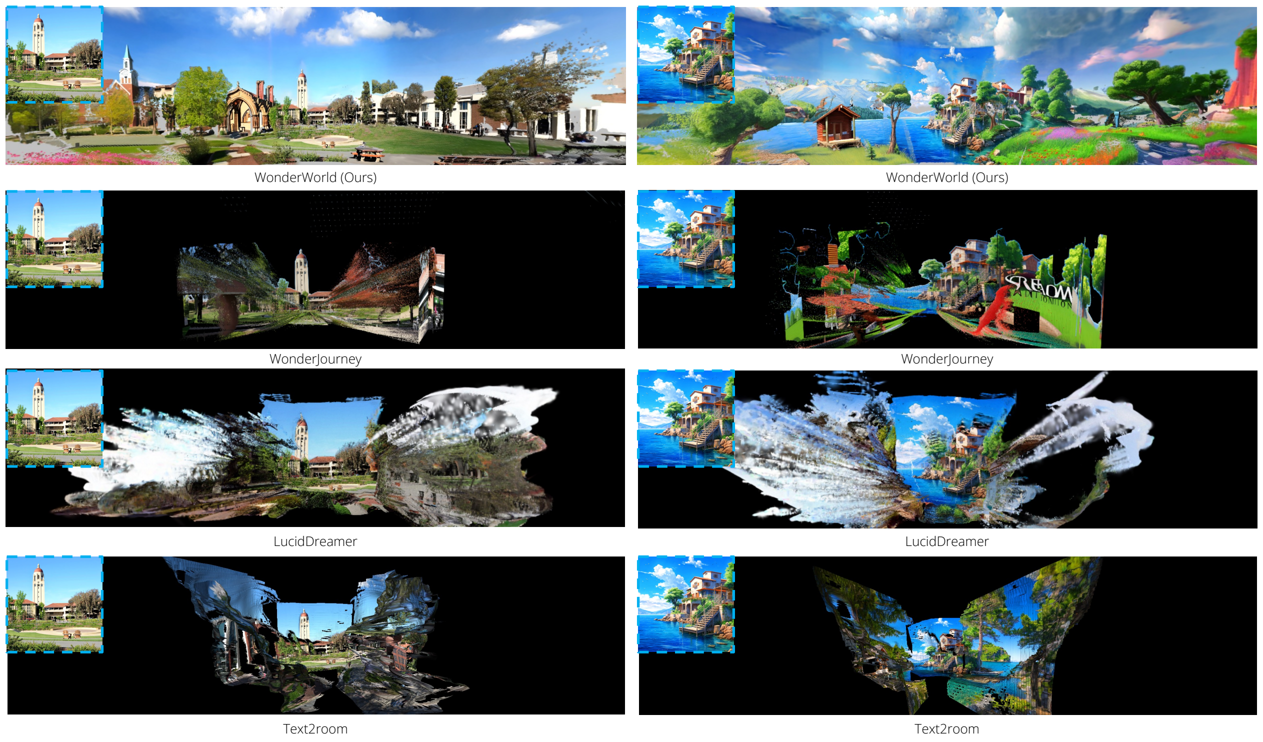


Figure 15. Baseline comparison. The inset with blur dashed bounding box is the input image.