WonderJourney: Going from Anywhere to Everywhere

Appendix

A. Overview

We compile a set of video results in our project website. **We strongly encourage the reader to see these video results**. Please use a modern browser such as Chrome, since we use advanced JavaScript libraries to control the carousels and video auto-play.

In the following, we summarize the contents in this supplementary document:

- [Section B] Additional qualitative results.
- [Section C] Longer "wonderjourneys". Each "wonderjourney" consists of 30 generated scenes.
- [Section D] Controlled "wonderjourneys" using user-provided descriptions (rather than LLM-generated descriptions), such as poems, story abstracts, and haiku.
- [Section E] Additional details on our renderer, camera paths, and depth processing.
- [Section F] Ablation study on white space ratio, the visual validation, and depth processing.
- [Section G] Additional details on the LLM and VLM we use.
- [Section H] Details on human preference evaluation setting.

B. Additional Results

We show additional results in Fig. 7 (going from anywhere) and Fig. 8 (going to everywhere).

C. Longer "Wonderjourneys"

We show examples of longer "wonderjourneys" in Fig. 9. We observe that the longer "wonderjourneys" allow including more diverse scenes with high visual quality.

D. Controlled "Wonderjourneys"

We may replace the LLM-generated scene descriptions with user-provided descriptions to control the generated "wonderjourneys". For example, one can use poems, haiku, or story abstracts. We show examples of classical Chinese poems, haiku, a nonsense poem "Jabberwocky" from *Alice's Adventures in Wonderland*, abstract of *Walden* by Henry David Thoreau, *Stopping by Woods on a Snowy Evening* by Robert Frost, and *Lines Written in Early Spring* by William Wordsworth in Fig. 10.

E. Details on Visual Scene Generation

Depth processing. As mentioned in the main paper, we find that depths of sky and distant pixels are estimated incorrectly. This is a general issue in monocular depth estimators, although we choose to use MiDaS v3.1 [35]. For the segmented sky pixels, we set the depth to 0.025. For distant pixels, we set the background far plane $F = 0.0015$. Since MiDaS is shift-invariant, we manually add a depth shift 0.0001 to ensure that the near objects do not collapse to the optical center due to extremely small depth values. Note that all these values (and all the depth estimator-related values below) are specific to MiDaS v3.1, and it may need to be changed for other depth estimators due to different normalization schemes used in their respective training procedures.

The disparity threshold T in depth refinement is set to 2. Empirically, we use the 30% and 70% depth values instead of $\min$ and $\max$ depth values within a segment to compute ΔD_j to improve robustness due to segmentation inaccuracy.

In new scene registration by depth consistency, we adapt MiDaS v3.1 for 200 iterations with learning rate 0.000001. In occlusion handling, we set the depth of disoccluders to 0.05. In scene completion, we set the depth of newly inpainted points to be the same as the depth of its valid nearest neighbor pixel.

Sky processing. Empirically, we find that sky segmentation is generally not accurate enough especially along the boundary of sharp shapes (such as tower spires) and complex shapes (such as tree leaves). Therefore, we use the following process to combat the inaccuracy. After using SAM to refine depth, we use an aggressively eroded sky segmentation to set depth values for sky pixels. Since SAM also segments sky slightly more accurately (although it often gives oversegments and it does not have semantic labels), we then use SAM to refine depth again to take advantage of the added accuracy. However, even SAM has difficulty in accurately segmenting complex sky-object boundaries. Thus, we dilate our outpainting mask a bit in the upper part of the image to cover the potentially inaccurately segmented boundary.

Rendering. We use a perspective pinhole camera model. To render the point cloud, we transform the points to a normalized device coordinate space and rasterize them to determine visibility. For each camera ray, we allow up to 8 points to reside in the z-buffer, and composite them using the following softmax-based function to avoid alias:

$$\texttt{disparity} = \frac{\sum_i \exp(\texttt{disparity}_i) * \texttt{color}_i}{\sum_i \exp(\texttt{disparity}_i)}, \quad (7)$$

where $i = 1, \cdots, 8$ indexes the points in the z-buffer. Higher

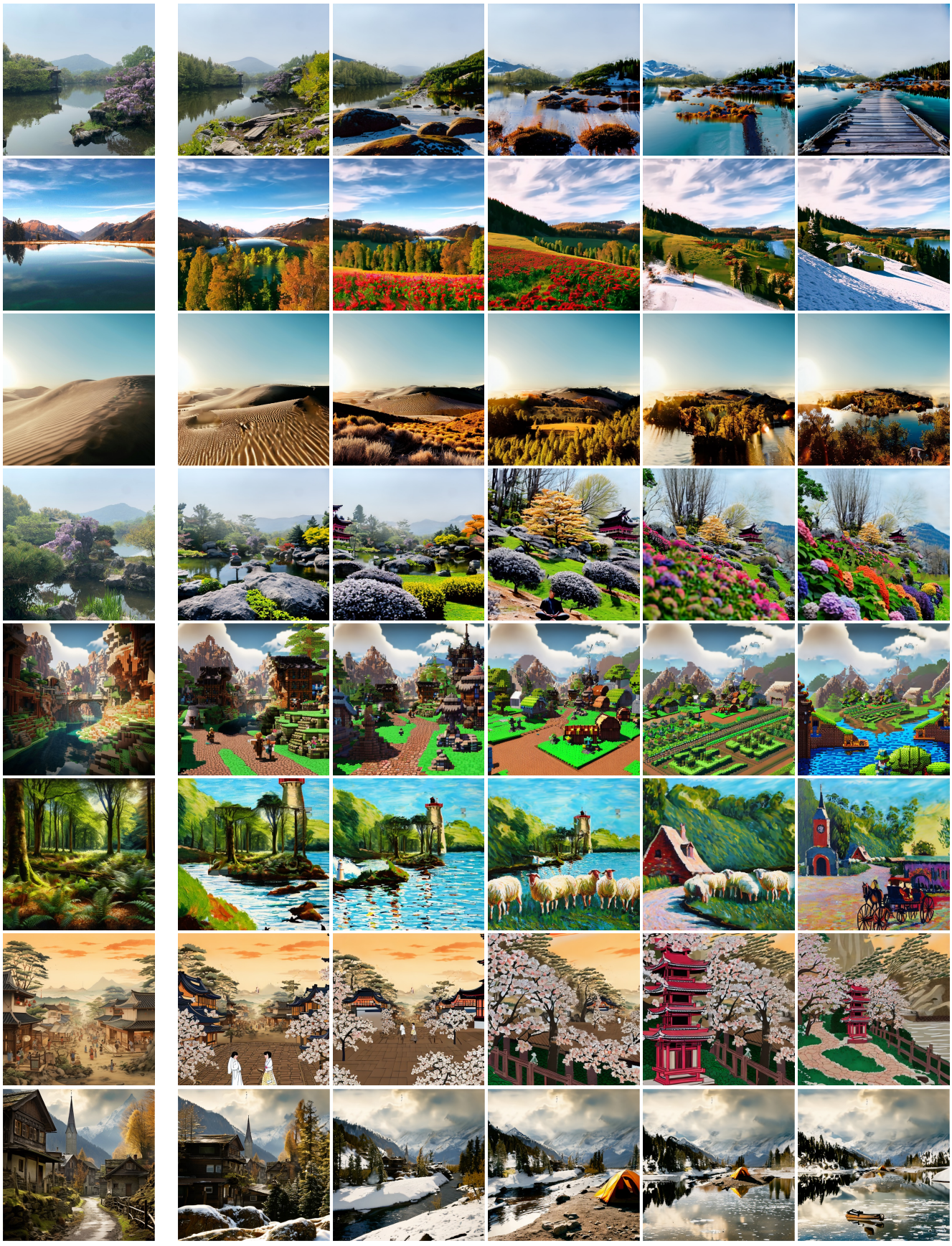

Figure 7. **Additional qualitative results** for "wonderjourneys". The inputs in the top four rows are real photos. The inputs in top four rows are real photos.

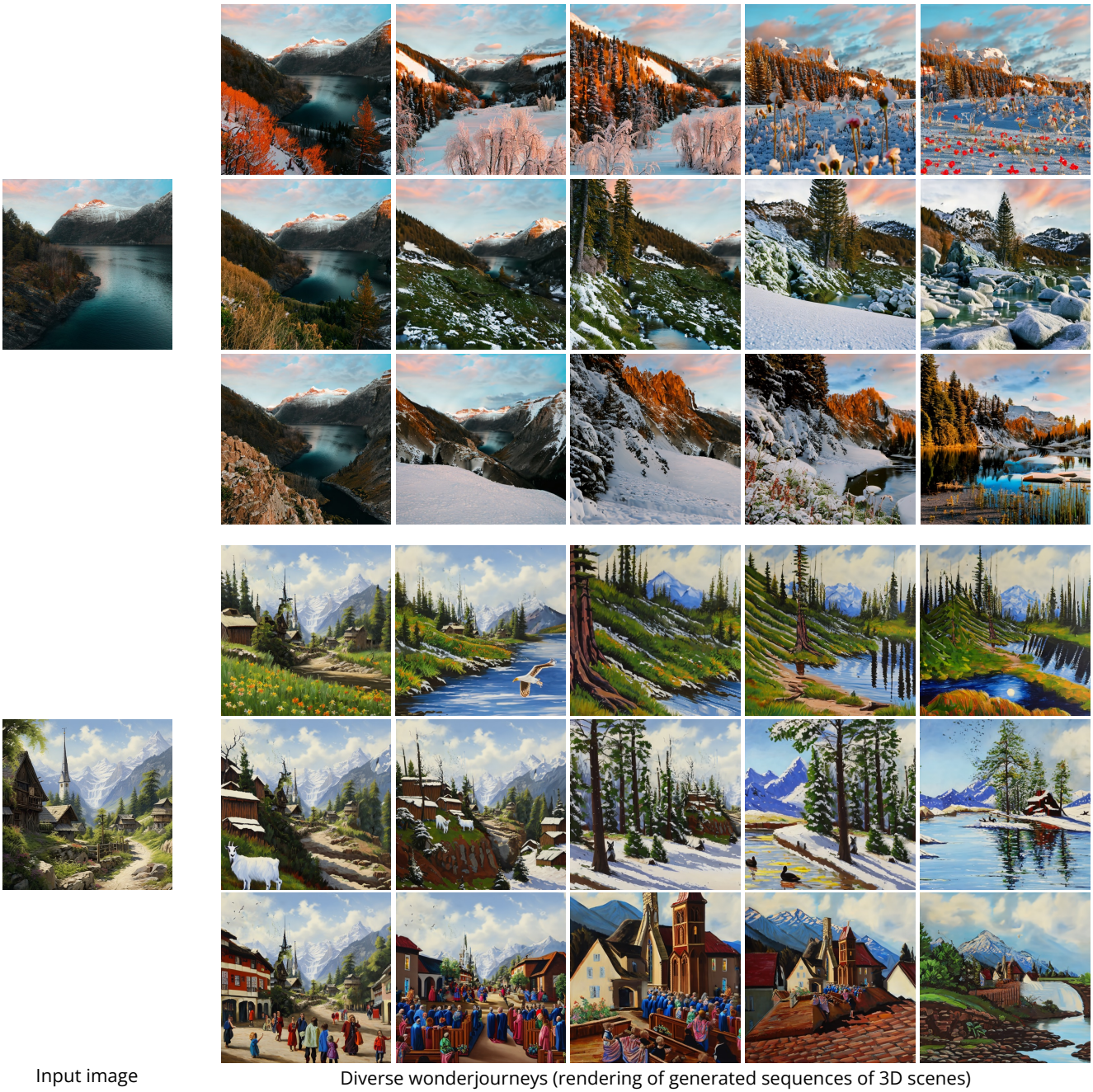

Figure 8. **Qualitative results for diverse** "wonderjourneys" generated from the same input image. The inputs in the top example is a real photo.

disparity values put higher weights on nearer points. At the boundary of objects, occluded points can also provide some blending for anti-aliasing. Our image resolution is 512×512.

Camera paths. To generate visual scenes, we move our camera either following a straight line, or to do a rotation. For the straight line, we use a camera movement of 0.0005 to the backward. For rotation, we move our camera with a rotation of 0.45 radians with a translation of 0.0001.

To generate the additional camera paths, we use the following rules: For a generated scene by rotation, we interpolate linearly among the camera rotation radians. For a generated scene by straight line, we linearly interpolate the translation. The additional cameras are also used in making our video results. Therefore, we also add a random sine perturbation to the height of the additional cameras. For the starting scene, the ending scene, and scenes right before

Figure 9. **Qualitative results for long "wonderjourneys"** generated from real input photos. Each long "wonderjourney" here consists of 30 scenes, yet we show 15 of them. See our website for video results.

and right after rotation happens, we add a linear acceleration process to make the video smoother.

F. Ablation Study

Outpainting white space. In Fig. 11, we show a comparison with the following variant, "Ours-slower". "Ours-slower" uses a $1/10$ speed and generating 10 scenes following a straight line camera path (see Section E for more details on the camera path), so that it ends up at the same camera location as "Ours". We use the same input image and the same scene description which includes "houses" in it. From Fig. 11 we observe that when we have insufficient empty space for outpainting, new objects like houses do not appear. This empirical observation may be due to that in the curated training set of the Stable Diffusion model, there may be few

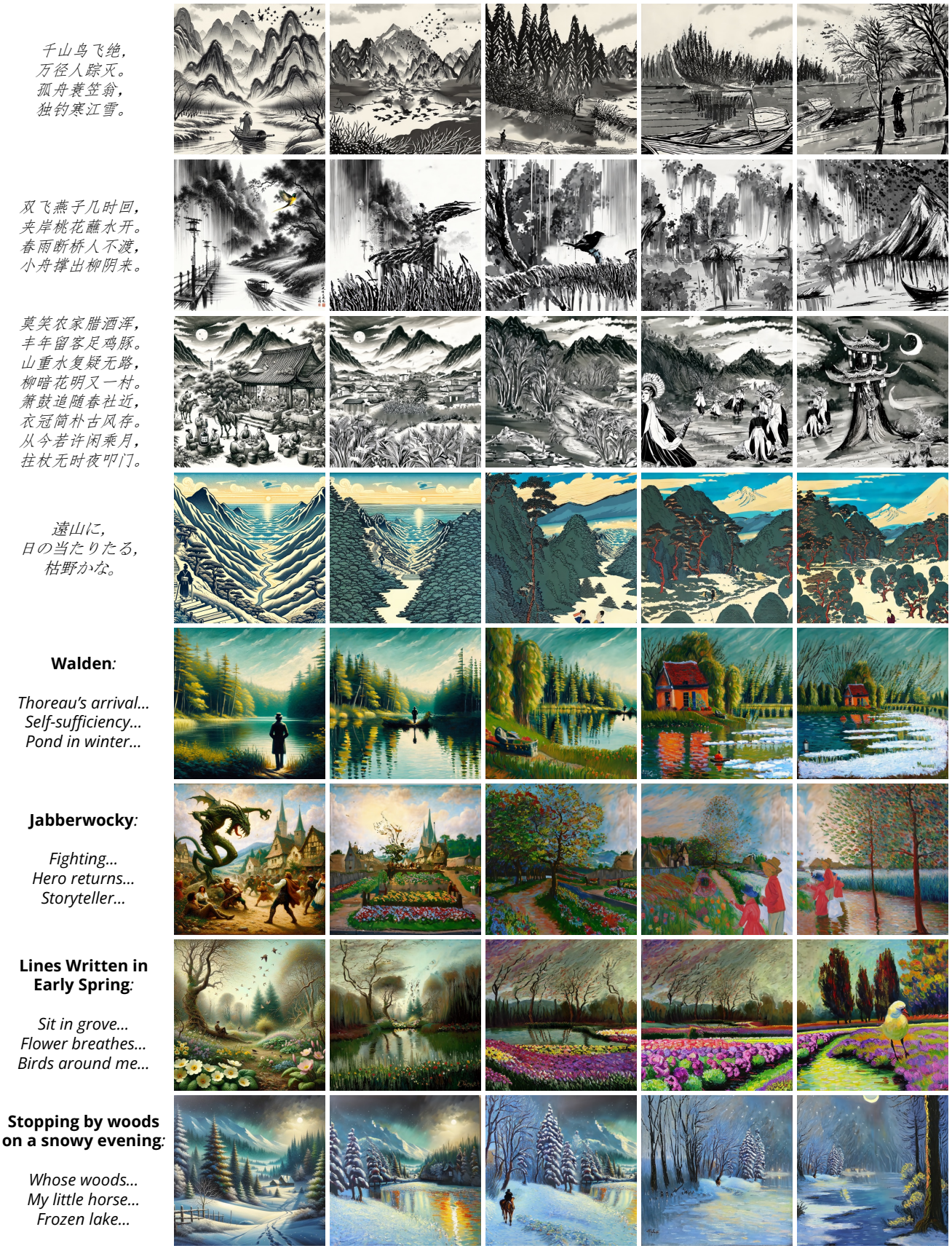

Figure 10. **Qualitative results for controlled "wonderjourneys"** generated from text descriptions (except for the "Jabberwocky" example where we manually pair it with an image). Each controlled "wonderjourney" here consists of $2N$ scenes where N is the number of parts of the text (e.g., a classical Chinese poem often has 4 or 8 parts). See our website for video results.

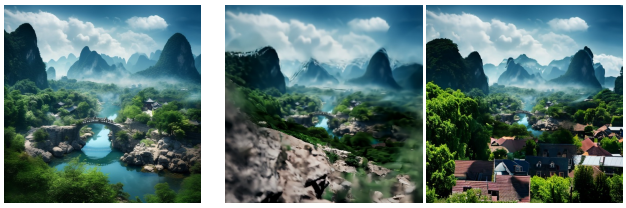

Figure 11. **Ablation comparison on sufficient outpainting space.** Both examples use the same scene descriptions including "houses" in it. Only "Ours" that has sufficient outpainting space can generate houses.

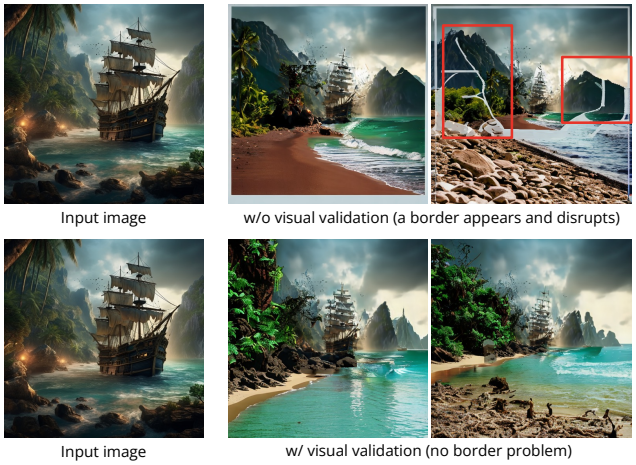

Figure 12. **Ablation comparison on using a VLM** to visually check if borders appear. Without visual validation, borders often appear and it disrupts the following scenes. Red boxes highlight the border disruptions.

partial cropped objects around the image borders. Thus, the outpainting model does not prefer generating partial objects along the borders.

Visual validation by a VLM. As mentioned in Section E, the painting frame and photo border can be a strong disruption. We show an example in Fig. 12, where the photo border appears and disrupts the next generated scene. The visual validation can effectively detect borders and launch a re-generation process to handle it.

Depth processing. Our depth processing is essential in generating geometrically coherent scenes. In Fig. 13, we show a visual comparison, where we show the rendered partial images using the original estimated depth and using our processed depth. Without our depth processing, the partial rendered images show strong distortions due to depth inaccuracy in object boundary, sky, and distant areas. For example, see the distortions in the sky (bottom example), the bird (middle example), and the tower (top example) in Fig. 13.

Figure 13. **Ablation comparison on our depth processing.** Zoom in to see more details. Without our depth processing, the rendered partial image demonstrates strong distortions. For examples, see the sky (bottom example), the bird (middle example), and the tower (top example).

G. Details on the LLM and VLM

We use GPT-4 for generating scene descriptions. Specifically, we use the following prompt $\mathcal{J}$ for the system call:

"You are an intelligent scene generator. Imaging you are flying through a scene or a sequence of scenes, and there are 3 most significant common entities in each scene. Please tell me what sequentially next scene would you likely to see? You need to generate the scene name and the 3 most common entities in the scene. The scenes are sequentially interconnected, and the entities within the scenes are adapted to match and fit with the scenes. You also have to generate a brief background prompt about 50 words describing the scene. You should not mention the entities in the background prompt. If needed, you can make reasonable guesses."

The input is the scene description memory $\mathcal{M}$ which is a collection of past and current scene descriptions $\mathcal{M}_i = \{\mathcal{S}_0, \cdots, \mathcal{S}_i\}$ as defined in Equation 2. In particular, $\mathcal{S}_i = \{S, O_i, B_i\}$ where S denotes a style, O_i denotes the object description, and B_i denotes the background description. We use a lexical category filter to extract nouns and adjectives. An actual scene description $\mathcal{S}_i$ looks like (the following is the actual text prompts of the "girl in wonderland" example in Fig. 1):

- *Scene 1: {Background: Alice in the wonderland. Entities: [' Alice', ' flowers', ' cat']; Style: Monet painting}*
- *Scene 2: {Background: luminous painting, way, vibrant bizarre croquet field, player, flamingo, mallet, Hedgehogs, balls, croquet, balls, life, domineering presence, atmosphere. Entities: ['flamingos', 'hedgehogs', 'The Queen of Hearts']; Style: Monet painting}*
- *Scene 3: {Background: scene, bizarre tea, party, great elm, tree, eccentric gentleman, top hat, presides, celebration, jittery hare, sleepy rodent, Cups, plates, assorted pastries, ancient misshapen, table, atmosphere, chaotic random bouts, nonsensical poetry, riddles. Entities: ['Mad Hatter', 'March Hare', 'Dormouse']; Style: Monet painting}*
- *Scene 4: {Background: impressionist strokes, endless checkerboard, landscape, animate chess, pieces, rules, game, sense, tension, serene, countryside, ambiance, trees, strange fruits, flowers. Entities: ['White Queen', 'Red Queen', 'Pawn']; Style: Monet painting}*

We use GPT-4V as the VLM for visual validation. The most significant unwanted effects are the painting frame and photo border that appear along some of the four boundaries of the outpainted image. We use the following system call:

"Along the four borders of this image, is there anything that looks like thin border, thin stripe, photograph border, painting border, or painting frame? Please look very closely to the four edges and try hard, because the borders are very slim and you may easily overlook them. If you are not sure, then please say yes."

We also use similar prompt for detecting out-of-focus blurry objects. If GPT-4V fails due to network connection or other practical reasons, we instead generate an 560×560 image and then center-crop it to bypass the frame problem. This is not as good as using visual validation, because it can lead to cropped partial foreground objects such as a half of a person, and the partial objects can become floaters due to low depth values when we move camera to generate new scenes.

H. Details on Human Preference Evaluation

We use Prolific[1] to recruit participants for the human preference evaluation. We use Google forms to present the survey. The survey is fully anonymized for both the participants and the host. We attach the anonymous survey link in the footnote[2] for reference.

[1] https://www.prolific.com/

[2] Comparison to InfiniteNature-Zero: https://forms.gle/mKxyJUT3qZLs2f8h9; Comparison to SceneScape: https://forms.gle/pt7NBj73Fnd5apjM8. Google forms require signing in to participate, but it does not record participant's identity.