# Theme Worlds

Seamless Wrap-Around Panoramas from Stable Diffusion v1.5, via a From-Scratch MultiDiffusion Sampler

**Marlen Melis** · KAIST AIC.20000 *Introduction to AI Computing* (Fall 2026) · Mini-Project Write-Up

**Figure 1:** Three finished worlds, each 2048 × 512 px from the submitted notebook and *nothing but* Stable Diffusion v1.5 — no source images, no fine-tuning. Top to bottom: volcanic_lava_world + flooded_metropolis (seed 201), arctic_ice_cave + cozy_village_snowfall (seed 1002, region-conditioned), fantasy + medieval (seed 1001). In each, the left and right edges are the *same* edge — cylinders, not strips (Table 3).

## 1 Goal and concept

One run of my notebook produces a wide panorama of an imagined world whose **left and right edges join perfectly**. Roll the image sideways by any amount, wrap it around a cylinder or use it as a 360° backdrop, and no seam appears anywhere. The user picks **one or two** themes from a 50-entry menu I wrote (plus a `custom` slot); with two themes the notebook builds a *single* world that fuses them — lava rivers running through a drowned city, neon towers encrusted in coral, an ice cave that opens onto a lantern-lit village and closes back into ice as you travel around the loop.

This is harder than it sounds. SD v1.5 was trained at 512 × 512; asked directly for 2048 × 512 it paints roughly four 512-ish scenes glued together, with duplicated skylines and warped geometry, because nothing in its training distribution has that shape — and nothing in the model makes the left edge agree with the right edge, so a panorama generated normally is a *strip* with an obvious join.

On top of the course rules I set two constraints: **inference time only** (no fine-tuning, no LoRA) and **zero source images** — 0 of the 10 permitted. Every submitted pixel comes from the stock SD v1.5 weights; the contribution is the sampling schedule around them, written out in the notebook rather than imported from a library.

## 2 Method

### 2.1 MultiDiffusion, re-implemented

The backbone is MultiDiffusion [1]. A single shared latent canvas $x_t$ of size $\frac{W}{8} \times \frac{H}{8}$ is kept. At every denoising step the canvas is cut into overlapping 64 × 64-latent windows (= 512 × 512 px, exactly SD v1.5's native shape); one ordinary DDIM step $\Phi$ runs on each window independently; the results are averaged back:

$$x_{t-1} = \Big(\textstyle\sum_i \mathcal{S}_i^\top \, \Phi\big(\mathcal{S}_i x_t\big)\Big) \oslash \Big(\textstyle\sum_i \mathcal{S}_i^\top \mathbf{1}\Big), \qquad (1)$$

where $\mathcal{S}_i$ crops window $i$ and $\oslash$ is elementwise division, so every latent cell becomes the average of every window opinion that covered it. Each window is a problem the UNet was trained to solve; the averaging is what forces neighbours to agree on one scene over the 50 steps. No weight is touched and no tensor exceeds 64 × 64 latents, which is why 2048 × 512 fits on a free T4.

### 2.2 Extension 1: circular windows

The published method tiles a strip. I make the window list *wrap*: columns run $\{0, s, 2s, \dots\}$ across the full latent width and each window indexes its columns modulo that width (`arange(w0,w1) % W`), so a window starting near the right edge continues onto the left edge. Both ends therefore share windows from step 1 and are denoised as one continuous surface: **nothing is stitched afterwards**, the loop is a property of the sampling. That fixes the latents; the VAE decoder has a receptive field of its own, so before decoding I wrap 8 latent columns around each side and crop the corresponding 64 px off afterwards.

### 2.3 Extension 2: multi-branch theme conditioning

The obvious way to combine two themes is to concatenate them into one prompt. I measured what that costs: CLIP's text encoder has a hard **77-token** context and silently discards the rest, and a fused two-theme prompt (cyberpunk + medieval) measures **97 tokens** — so **20 tokens are dropped without warning**, and what gets cut is the tail, i.e. the quality suffix, every time.

So I never concatenate. My sampler accepts $K$ conditioning *branches*, each with its own embedding *per window*, plus

a weight matrix; one UNet call carries the unconditional pass and all $K$ branches, whose classifier-free guidance [5] directions are added:

$$\hat{\epsilon} = \epsilon_u + g \sum_{k=1}^{K} w_k \big(\epsilon_k - \epsilon_u\big), \qquad \textstyle\sum_k w_k = 1. \tag{2}$$

Four strategies fall out, differing *only* inside `build_conditioning` — everything downstream is shared, which is what makes Fig. 2 a fair comparison. `compose` (default) sets $K$=2 so both prompts act on every pixel, combined in noise space as in composable diffusion [2]; neither prompt is ever truncated because neither is ever concatenated. `regions` keeps $K$=1 but gives every window its own interpolated embedding along a circular raised-cosine field $w(u) = \mathrm{clip}\big[\big(\frac{1}{2}(1+\cos 2\pi u) - \frac{1}{2}\big)\sigma + \frac{1}{2}\big]$, so each theme owns one side of the loop with two blended hand-off bands. `blend` uses one averaged embedding everywhere, and `single_prompt` is the naive concatenation, kept as a control.

### 2.4 Fitting a free T4

MultiDiffusion calls `scheduler.step()` once per window per timestep, out of order, so the scheduler must be stateless: DDIM [4] is, SD v1.5's default PNDM is not — it would silently corrupt its own history — and the notebook swaps it and asserts the swap. The UNet runs in fp16 but the fusion accumulator is fp32, because averaging up to eight overlapping fp16 predictions per cell is exactly where rounding shows up as banding. Windows are batched (halved automatically when two branches are active, to keep the real UNet batch constant), the stride is exposed (16 rather than the reference's hard-coded 8, halving the work), and VAE decoding falls back to tiled decoding on OOM. Initial noise is drawn on the **CPU** generator, whose RNG is identical across GPUs, so a seed reproduces the same image on any machine (§5.5). The safety checker is not loaded: it generates nothing, costs ∼1.2 GB and returns black images on false positives, a real risk for stylised landscape art.

## 3 Data and resources

**Source or training images: none** — zero of the ten permitted. There is no image input, no upload step and no `data.zip`, and nothing was fine-tuned. The 50-theme menu, the quality suffix and the negative prompt are my own text; Table 1 lists every pretrained weight the notebook loads and where it comes from. Software: `diffusers` (a weight loader and `DDIMScheduler` only — the sampler is written out in the notebook), `transformers`, `torch`, `numpy`, `PIL`. All compositing and labelling happens inside the notebook with PIL; no image is edited in an external tool and no text label is ever asked of the model.

## 4 Experimental setup

Table 2 gives the settings. Each run renders `NUM_CANDIDATES` seeds, saves every candidate at full resolution, and writes a JSON log with the checkpoint id, scheduler config, resolved prompts, canvas, steps, guidance, window/stride, every seed and the measured scores, so any image can be regenerated exactly. Selection is quantitative: CLIP ViT-L/14 [6] scores six square crops taken *around the loop* (a panorama is four times wider than CLIP's square input), a theme's score is its best crop, and the candidate's score is the *worst* of the two theme scores — penalising a candidate that quietly dropped a theme. Anything above a seam ratio of 4.0 is disqualified.

**Table 1:** Every pretrained weight loaded by the notebook.

| Weight | Role and source |
|---|---|
| SD v1.5 (fp16) | **Generates every submitted pixel.** `stable-diffusion-v1-5/ stable-diffusion-v1-5` on Hugging Face, `https://huggingface.co/stable-diffusion-v1-5/stable-diffusion-v1-5`, loaded with `from_pretrained`. Ships its own CLIP ViT-L/14 *text* encoder, the VAE and the scheduler config. |
| CLIP ViT-L/14 | **Scores finished images only; generates nothing.** `openai/clip-vit-large-patch14`, `https://huggingface.co/openai/clip-vit-large-patch14`. Used in Step 9 to rank candidates — the "pretrained reward model" carve-out. Setting `USE_CLIP_RANKING=False` disables it and the notebook still runs. |
| *(not loaded)* | The SD safety checker is explicitly disabled. No SDXL/FLUX, no third-party SD v1.5 checkpoint, LoRA or DreamBooth model exists anywhere in the notebook. |

**Table 2:** Settings for the submitted panoramas.

| | |
|---|---|
| Canvas | 2048 × 512 (also 1536 × 768), multiples of 64 |
| Window / stride | 64 × 64 latents (512 px) / 16 cells ⇒ 16 windows per step, every cell covered ≥4× |
| Sampler | DDIM, 50 steps, CFG 7.5; `compose` ($K$=2, $\lambda$=0.5) or `regions` (sharpness 2.0) |
| Runtime | fp16 UNet/VAE with an fp32 fusion accumulator; 4 windows per UNet call (2 when $K$=2); seeds `BASE_SEED`+$i$ drawn on the CPU generator; ±8 latent columns of circular padding before decoding, then cropped |
| Prompt | `"{theme}, seamless wide panorama, sweeping vista, single horizon, highly detailed, atmospheric, cinematic lighting, digital matte painting"` (≤55 tokens) |
| Negative | seam, stitching artifact, repeating/tiling pattern, duplicate objects, split screen, multiple horizons, repeated skyline, text, watermark, border (66 tok.) |

## 5 Results and analysis

### 5.1 Does it actually loop?

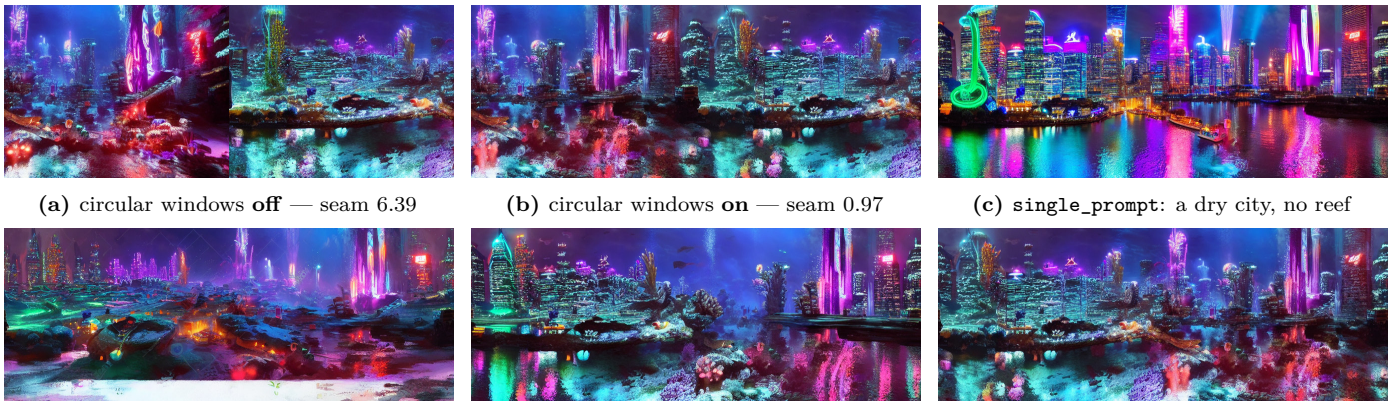
The claim needs a number, not an impression. I define the **seam ratio** as the mean absolute difference between the first and last image columns divided by the mean absolute difference between *neighbouring* columns:

$$R = \overline{|I_{:,0} - I_{:,W-1}|} \;/\; \overline{|I_{:,j+1} - I_{:,j}|}. \tag{3}$$

$R \approx 1$ means the wrap-around join is statistically indistinguishable from any other pair of adjacent columns — there is no special column — whereas a strip merely cropped to size blows up. Across **28 panoramas in 12 configurations** the measured value averages **1.12** (median 1.13, s.d. 0.23, worst case 1.62; 26 of 28 below 1.5), Table 3.

### 5.2 Control 1: is the circular machinery doing anything?

The honest test is the same theme, seed and steps with window wrapping switched off; the notebook runs this pair automatically. Over **12 paired runs** the seam ratio is **7.94** on average with wrapping *off* (range 5.01–14.63) versus **1.05** with it *on* (0.72–1.36): a mean **7.6**× improvement on identical inputs, never below 5.4×. Fig. 2(a–b) is a representative pair: without the wrap the two ends are



**(a)** circular windows **off** — seam 6.39 **(b)** circular windows **on** — seam 0.97 **(c)** `single_prompt`: a dry city, no reef

**(d)** `blend`: a washed-out midpoint **(e)** `regions`: reef centre, towers at both ends **(f)** `compose`: a submerged neon city

**Figure 2: Controls, all from one seed** (cyberpunk + underwater, 1024 × 512, 30 steps, identical in every other respect). **(a–b)** the circular ablation, both rolled by 50% so the join sits in the centre: without wrapping, two unrelated scenes are butted together mid-frame; with it, one continuous world. **(c–f)** the four fusion modes — the naive concatenation (c) drops the underwater theme outright and paints a night skyline above water, as the 77-token measurement predicts, while only `compose` (f) puts both themes in every pixel.

**Table 3:** Every submitted configuration and the seam ratio measured on each finished panorama (≈ 1 = a perfect loop). All 2048 × 512 except † (1536 × 768). Mean over all 28: **1.12**.

| Theme(s) | Seam | Theme(s) | Seam |
|---|---|---|---|
| cyberpunk | 1.16 | alien_jungle + | 1.29 |
| | 1.27 | desert_planet | 1.36 |
| | 0.87 | abandoned_park + | 1.11 |
| stained_glass† | 1.11 | toy_world† | 1.24 |
| | 1.14 | ink_wash_mtns + | 1.53 |
| cyberpunk + | 1.23 | dragon_spire | 1.27 |
| underwater | 1.34 | volcanic_lava + | 1.20 |
| | 0.96 | flooded_city | 0.88 |
| arctic_ice_cave + | 0.76 | giant_tree + | 1.00 |
| cozy_village | 1.47 | biopunk_city | 1.08 |
| | 0.89 | (2 separate runs) | 0.91 |
| fantasy + | 1.62 | | 1.12 |
| medieval | 1.19 | gothic_courtyard + | 0.92 |
| | 0.79 | abandoned_park | 0.61 |

visibly different scenes butted together, with it the join is unfindable; the largest gap measured was 14.63 → 1.36.

### 5.3 Control 2: what the four fusion modes do

Fig. 2(c–f) renders all four strategies from one seed, and the differences are systematic rather than cosmetic. `single_prompt` *loses the second theme entirely* — it paints a dry neon skyline reflected in water, with nothing submerged, exactly as the token measurement predicts. `blend` washes out into a midpoint that is neither city nor reef, because the average of two distant CLIP embeddings is rarely either concept. `regions` gives each theme its own territory: reef through the centre, city towers at both ends of the loop. `compose` spreads both themes over every pixel and was the only mode to produce genuinely hybrid architecture: coral-encrusted skyscrapers lit by their own neon. The cost matches the theory — $K+1$ passes per window instead of two — and over nine paired control runs `compose` took **1.53**× the single-branch modes (1.50–1.63×), which were within a second of each other.

### 5.4 Control 3: is the re-implementation correct?

`diffusers` ships its own (now deprecated) MultiDiffusion sampler, `StableDiffusionPanoramaPipeline`. Since mine is written from scratch, the notebook runs the reference on the same checkpoint, canvas, seed and step count as an independent check: over 12 pairs the seam ratios are statistically indistinguishable (reference $0.95 \pm 0.22$ versus mine $0.99 \pm 0.22$), i.e. the re-implementation behaves like the published method.

It also surfaced a real limitation of the reference: with `circular_padding=True` it scatters each finished window back using *canvas* row indices rather than window-local ones, so any canvas taller than one window raises a shape mismatch (and the pipeline is unmaintained past `diffusers` 0.33.1). Mine never addresses a window by canvas coordinates, which is why the 768-tall panoramas here work; the cross-check is pinned to 512 px for that reason.

### 5.5 Reproducibility

Two independent Colab sessions on different days, with the same seeds and settings, produced **byte-identical PNGs** (MD5 `f737a887...`, `16f0c451...`, `8a6e4445...`, seeds 1000–1002 of the cyberpunk run) — the practical payoff of the CPU noise generator: re-running the notebook returns *these* images, not merely similar ones.

### 5.6 What I learned, and what does not work

**Tokenisation, not attention, is what kills fused prompts.** I expected `single_prompt` to blur the two themes; instead it deletes one, and the token count says why — measuring the prompt beat staring at the image. Relatedly, **averaging is what buys global coherence**, at the price of slightly softer local contrast where windows pile up — which is why stride 16 (4× coverage) looked no worse to me than the reference's stride 8 (8×) while costing half as much.

**Limitations.** Composition is seed-sensitive — SD v1.5's model card flags multi-object compositionality as a weak point, and with distant concepts some seeds land on a genuine hybrid and others on a compromise that reads as neither, hence several seeds per run and a worst-of-both-themes ranking. High-contrast pairs (volcanic_lava + arctic_ice_cave) tend to cancel under `compose` and are better served by `regions`. The seam metric certifies the wrap, not semantic continuity: a world can loop perfectly and still change mood across the frame. Object-level fidelity stays SD v1.5's — small figures and animals in the busier worlds are often malformed under magnification, and no sampling schedule fixes that.

### 5.7 Rule compliance

SD v1.5 (Table 1) generated every submitted pixel — no other generative model, no fine-tuning, no source images (hence no `data.zip`), CLIP confined to scoring. The deliverable is one `.ipynb` that runs end to end on a Colab T4 with all cell outputs saved, and every visual is a static PNG.

## References


[1] O. Bar-Tal, L. Yariv, Y. Lipman, T. Dekel. *MultiDiffusion: Fusing Diffusion Paths for Controlled Image Generation.* ICML 2023. arXiv:2302.08113.

[2] N. Liu, S. Li, Y. Du, A. Torralba, J. B. Tenenbaum. *Compositional Visual Generation with Composable Diffusion Models.* ECCV 2022. arXiv:2206.01714.

[3] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer. *High-Resolution Image Synthesis with Latent Diffusion Models.* CVPR 2022. arXiv:2112.10752.

[4] J. Song, C. Meng, S. Ermon. *Denoising Diffusion Implicit Models.* ICLR 2021. arXiv:2010.02502.

[5] J. Ho, T. Salimans. *Classifier-Free Diffusion Guidance.* NeurIPS 2021 Workshop. arXiv:2207.12598.

[6] A. Radford et al. *Learning Transferable Visual Models From Natural Language Supervision.* ICML 2021. arXiv:2103.00020.

[7] P. von Platen et al. *Diffusers: State-of-the-art diffusion models.* https://github.com/huggingface/diffusers.