SemanticDraw: Towards Real-Time Interactive Content Creation from Image Diffusion Models

Supplementary Material

Abstract

This is a supplementary material to the CVPR 2025 submission "SemanticDraw: Towards Real-Time Interactive Content Creation from Image Diffusion Models." Section S1 shows implementation details of our acceleration methods. In Section S2, additional visual results are shown. Finally, we provide our demo application as we have promised in our main manuscript. Our formulation introduces new controllable hyperparameters that users may interact in order to create images that respect their intentions. Section S3 demonstrates how our new tool can be used in image content creation.

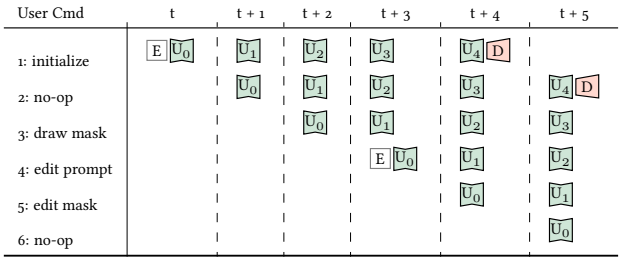


Figure S1. Example execution process of *Multi-Prompt Stream Batch* pipeline of SEMANTICDRAW. By aggregating latents at different timesteps a single batch, we can maximize throughput by hiding the latency.

S1. Implementation Details

We begin by providing additional implementation details.

S1.1. Acceleration-Compatible Regional Controls

Algorithm S1 compares between the the baseline MultiDiffusion [1] and our stabilized sampling from multiple regionally assigned text prompts introduced in Section 3.2 of the main manuscript. As we have discussed in Section 3 of the main manuscript, improper placing of the aggregation step and strong interference of its bootstrapping strategy limit the ability to generate visually pleasing images under modern fast inference algorithms [2, 4–6, 9, 13]. Therefore, we instead focus on changing the bootstrapping stage of line 9-13 and the diffusion update stage of line 14-15 of Algorithm S1 in order to establish compatibility to accelerating diffusion samplers.

The resulting Algorithm S2 developed in Section 3.2 of the main manuscript achieves this. The differences between our approach from the baseline inference algorithm are marked with blue. First, in line 10, we change the bootstrapping background color to white. Having extremely low number of sampling steps (4-5), this bootstrapping background is easily leaked through the final image as seen in Figure 3 of the main manuscript. We notice that white backgrounds are common in public image datasets on which the diffusion models are trained. Therefore, changing random background images into white backgrounds alleviate this leakage problem.

Diffusion models have a strong tendency to generate objects at the center of the frame. This positional bias makes generation from small, off-centered masks difficult especially in the accelerated sampling, where the final structure of generated images are determined at the first two inference steps. By masking with off-centered masks, the objects under generation are unnaturally cut, leading to defective generations. Lines 13-14 of Algorithm S2 are our *mask-centering* stage for bootstrapping to alleviate this problem. In the first few steps of generation, for each mask-designated object, intermediate latents are masked then shifted to the center of the object bounding box. This operation enforces the denoising network to focus on each foreground object located at the center of the screen. Lines 17-19 of Algorithm S2 undo this centering operation done in lines 13-14. The separately estimated foreground objects are aggregated into the single scene by shifting them back to their original absolute positions.

Finally, a single reverse diffusion step in line 14 of Algorithm S1 is split into the denoising part in line 16 of Algorithm S2 and the noise addition part in line 24 of Algorithm S2. As we have discussed with visual example in Figure 3c in the main manuscript, this simple augmentation of the original MultiDiffusion [1] stabilizes the algorithm to work with fast inference techniques such as LCM-LoRA [5, 6], SDXL-Lightning [4], Hyper-SD [9], and Flash Diffusion [2]. Also refer to panorama generation in Figure S7 where this wrongly placed aggregation after STEP operation causing extremely blurry generation under accelerating schedulers [5, 6]. The readers may also consult our submitted code for the implementation of Algorithm S2.

S1.2. Streaming Pipeline Execution

Extending Figure 4b of the main manuscript, Figure S1 elaborates on the pipelined execution from our *multi-prompt stream batch* architecture for near real-time generation from multiple regionally assigned text prompts. We have empiri-

Algorithm S1: Baseline [1].

Input: a diffusion model $\boldsymbol{\epsilon}_\theta$, a latent autoencoder $(\texttt{enc}, \texttt{dec})$, prompt embeddings $\boldsymbol{y}_{1:p}$, masks $\boldsymbol{w}_{1:p}$, timesteps $\boldsymbol{t} = t_{1:n}$, the output size (H', W'), the tile size (H, W), an inference algorithm STEP, a noise schedule α, the number of bootstrapping steps n_{bstrap}.

Output: An image $\boldsymbol{I}$ of designated size $(8H', 8W')$ generated from multiple text-mask pairs.

$\boldsymbol{x}'_{t_n} \sim \mathcal{N}(0,1)^{H' \times W' \times D}$ // sample the initial latent

$\{\mathcal{T}_1, \dots, \mathcal{T}_m\} \subset \{(h_\text{t}, h_\text{b}, w_\text{l}, w_\text{r}) : 0 \le h_\text{t} < h_\text{b} \le H', 0 \le w_\text{l} < w_\text{r} \le W'\}$ // get a set of overlapping tiles

for $i \leftarrow n$ **to** 1 **do**

  $\tilde{\boldsymbol{x}} \leftarrow \boldsymbol{0} \in \mathbb{R}^{H' \times W' \times D}$ // placeholder for the next step latent

  $\tilde{\boldsymbol{w}} \leftarrow \boldsymbol{0} \in \mathbb{R}^{H' \times W'}$ // placeholder for the next step mask weights

  for $j \leftarrow 1$ **to** m **do**

    $\bar{\boldsymbol{x}}_{1:p} \leftarrow \texttt{repeat}(\texttt{crop}(\boldsymbol{x}_{t_i}, \mathcal{T}_j), p)$ // get a cropped intermediate latent tile

    $\bar{\boldsymbol{w}}_{1:p} \leftarrow \texttt{crop}(\boldsymbol{w}_{1:p}, \mathcal{T}_j)$ // get cropped mask tiles

    if $i \le n_{bstrap}$ **then**

      $\boldsymbol{x}_\text{bg} \leftarrow \texttt{enc}(c\boldsymbol{1})$, where $c \sim \mathcal{U}(0,1)^3$ // get a uniform color background

      $\boldsymbol{x}_\text{bg} \leftarrow \sqrt{\alpha(t_i)}\boldsymbol{x}_\text{bg} \sqrt{1-\alpha(t_i)}\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0,1)^{H \times W \times D}$ // add noise to the background for mixing

      $\bar{\boldsymbol{x}}_{1:p} \leftarrow \bar{\boldsymbol{w}}_{1:p} \odot \bar{\boldsymbol{x}}_{1:p} + (\boldsymbol{1} - \bar{\boldsymbol{w}}_{1:p}) \odot \boldsymbol{x}_\text{bg}$ // bootstrap by treating as multiple single-instance images

    end

    $\bar{\boldsymbol{x}}_{1:p} \leftarrow \text{STEP}(\bar{\boldsymbol{x}}_{1:p}, \boldsymbol{y}_{1:p}, i; \boldsymbol{\epsilon}_\theta, \alpha, \boldsymbol{t})$ // prompt-wise batched diffusion update

    $\tilde{\boldsymbol{x}}[\mathcal{T}_j] \leftarrow \tilde{\boldsymbol{x}}[\mathcal{T}_j] + \sum_{k=1}^{p} \bar{\boldsymbol{w}}_k \odot \bar{\boldsymbol{x}}_k$ // aggregation by averaging

    $\tilde{\boldsymbol{w}}[\mathcal{T}_j] \leftarrow \tilde{\boldsymbol{w}}[\mathcal{T}_j] + \sum_{k=1}^{p} \bar{\boldsymbol{w}}_k$ // total weights for normalization

  end

  $\boldsymbol{x}_{t_{i-1}} \leftarrow \tilde{\boldsymbol{x}} \odot \tilde{\boldsymbol{w}}^{-1}$ // reverse diffusion step

end

$\boldsymbol{I} \leftarrow \texttt{dec}(\boldsymbol{x}_{t_1})$ // decode latents to get an image

Algorithm S2: SEMANTICDRAW pipeline of Section 3.2.

Input: a diffusion model $\boldsymbol{\epsilon}_\theta$, a latent autoencoder $(\texttt{enc}, \texttt{dec})$, prompt embeddings $\boldsymbol{y}_{1:p}$, quantized masks $\boldsymbol{w}_{1:p}^{(t_{1:n})}$, timesteps $\boldsymbol{t} = t_{1:n}$, the output size (H', W'), a noise schedule α and η, the tile size (H, W), an inference algorithm STEPEXCEPTNOISE, the number of bootstrapping steps n_{bstrap}.

Output: An image $\boldsymbol{I}$ of designated size $(8H', 8W')$ generated from multiple text-mask pairs.

$\boldsymbol{x}'_{t_n} \sim \mathcal{N}(0,1)^{H' \times W' \times D}$

$\{\mathcal{T}_1, \dots, \mathcal{T}_m\} \subset \{(h_\text{t}, h_\text{b}, w_\text{l}, w_\text{r}) : 0 \le h_\text{t} < h_\text{b} \le H', 0 \le w_\text{l} < w_\text{r} \le W'\}$

for $i \leftarrow n$ **to** 1 **do**

  $\tilde{\boldsymbol{x}} \leftarrow \boldsymbol{0} \in \mathbb{R}^{H' \times W' \times D}$

  $\tilde{\boldsymbol{w}} \leftarrow \boldsymbol{0} \in \mathbb{R}^{H' \times W'}$

  for $j \leftarrow 1$ **to** m **do**

    $\bar{\boldsymbol{x}}_{1:p} \leftarrow \texttt{repeat}(\texttt{crop}(\boldsymbol{x}_{t_i}, \mathcal{T}_j), p)$

    $\bar{\boldsymbol{w}}_{1:p}^{(t_i)} \leftarrow \texttt{crop}(\boldsymbol{w}_{1:p}^{(t_i)}, \mathcal{T}_j)$ // use different quantized masks for each timestep

    if $i \le n_{bstrap}$ **then**

      $\boldsymbol{x}_\text{bg} \leftarrow \texttt{enc}(\boldsymbol{1})$ // get a white color background

      $\boldsymbol{x}_\text{bg} \leftarrow \sqrt{\alpha(t_i)}\boldsymbol{x}_\text{bg} \sqrt{1-\alpha(t_i)}\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0,1)^{H \times W \times D}$

      $\bar{\boldsymbol{x}}_{1:p} \leftarrow \bar{\boldsymbol{w}}_{1:p} \odot \bar{\boldsymbol{x}}_{1:p} + (\boldsymbol{1} - \bar{\boldsymbol{w}}_{1:p}) \odot \boldsymbol{x}_\text{bg}$

      $\boldsymbol{u}_{1:p} \leftarrow \texttt{get\_bounding\_box\_centers}(\bar{\boldsymbol{w}}_{1:p}) \in \mathbb{R}^{p \times 2}$ // get the bounding box center of each mask

      $\bar{\boldsymbol{x}}_{1:p} \leftarrow \texttt{roll\_by\_coordinates}(\bar{\boldsymbol{x}}_{1:p}, \boldsymbol{u}_{1:p})$ // center foregrounds to their mask centers

    end

    $\bar{\boldsymbol{x}}_{1:p} \leftarrow \text{STEPEXCEPTNOISE}(\bar{\boldsymbol{x}}_{1:p}, \boldsymbol{y}_{1:p}, i; \boldsymbol{\epsilon}_\theta, \alpha, \boldsymbol{t})$ // pre-averaging

    if $i \le n_{bstrap}$ **then**

      $\bar{\boldsymbol{x}}_{1:p} \leftarrow \texttt{roll\_by\_coordinates}(\bar{\boldsymbol{x}}_{1:p}, -\boldsymbol{u}_{1:p})$ // restore from centering

    end

    $\tilde{\boldsymbol{x}}[\mathcal{T}_j] \leftarrow \tilde{\boldsymbol{x}}[\mathcal{T}_j] + \sum_{k=1}^{p} \bar{\boldsymbol{w}}_k \odot \bar{\boldsymbol{x}}_k$

    $\tilde{\boldsymbol{w}}[\mathcal{T}_j] \leftarrow \tilde{\boldsymbol{w}}[\mathcal{T}_j] + \sum_{k=1}^{p} \bar{\boldsymbol{w}}_k$

  end

  $\boldsymbol{x}_{t_{i-1}} \leftarrow \tilde{\boldsymbol{x}} \odot \tilde{\boldsymbol{w}}^{-1}$

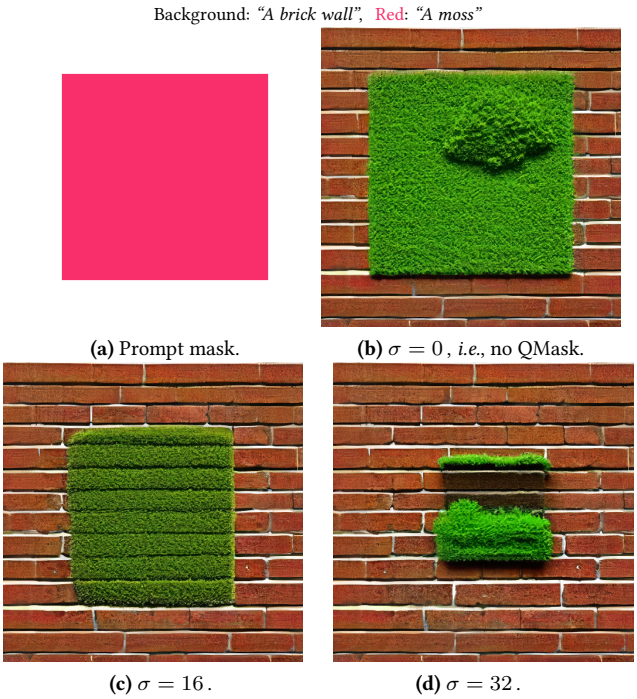
  $\boldsymbol{x}_{t_{i-1}} \leftarrow \boldsymbol{x}_{t_{i-1}} + \eta_{t_{i-1}}\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0,1)^{H \times W \times D}$ // post-addition of noise

end

$\boldsymbol{I} \leftarrow \texttt{dec}(\boldsymbol{x}_{t_1})$

(a) Masks (b) 1 step (c) **2 steps** (d) **3 steps** (e) 4 steps

Figure S2. The number of centering steps effectively trades off centered-bias against overall harmony. Composition, harmony, and mask-obedience are achieved in the sweet spot of 2-3 steps.



(a) Prompt mask. **(b)** $\sigma = 0$, *i.e.*, no QMask.

(c) $\sigma = 16$. **(d)** $\sigma = 32$.

Figure S3. Effect of the standard deviation in mask smoothing.

cally found that the text and image encoders for popular diffusion models take significantly longer latency than the denoising network. Assuming that users change text prompts and background images less frequently than they change the areas occupied by each semantic masks, such latency can be hidden under the high-throughput streaming generation of images. Moreover, mask processing takes almost negligible latency compared to image generation or text encoding. In other words, drawing with semantic masks of pre-encoded text prompts do not affect the generation speed, allowing users to almost seamlessly interact with the generation pipeline by friendly drawing interface. This user interface of our drawing-based interactive content creation is the same as commercial drawing software with brush tools. The only difference is that our brush tools apply semantic masks instead of colors or patterns. This similarity opens up a novel application for diffusion models, *i.e.*, SEMANTICDRAW.

S1.3. Controlling Fidelity-Harmony Trade-off

As we have mentioned in Section 3.2, accelerated samplers involving 5 or few steps like in our case rely heavily on the first few steps of inference in determining the structure of the image. Many diffusion models are trained using *natural* images that place their objects of interest at the center of the canvas. This makes these diffusion models generate all of their prompt-guided objects at the center of the canvas. Cropping by masks occasionally leads to destruction of such objects. Mask-centering bootstrapping is devised in order to alleviate this problem. However, applying bootstrapping from beginning to the end causes another problem of disharmony in the overall image with multiple region-based prompts. This can be seen in Figure S2e, where the girls' upper part of head is unnaturally cut. This problem also caused by the acceleration. Unlike gradual generation over tens of inference steps, in our accelerated scenario, the later inference steps are responsible for both high quality texture generation *and* boundary creation. Those quickly generated model-generated boundaries do not align well with the user-given mask inputs, creating unnatural cuts after merging with other prompt-guided subsections of the creation. We, therefore, provide a simple control handle that trades off mask-fidelity against overall harmony: the number of mask-centering steps in the bootstrapping stage. The effect of this control handle can be seen in Figure S2. We have empirically found that 1-3 steps work best, and we have used 2 steps throughout this work.

S1.4. Mask Quantization

To increase harmonization within a created image, we have introduced mask quantization as our final piece of the puzzle in Section 3.2 of the main manuscript. Mask quantization allows smooth masks with controllable smoothness that resemble soft brush tools in common drawing software. Therefore, this stage not only increases image fidelity but also enhances user experience in our SEMANTICDRAW application. This section explains additional technical details

Image prompt (row, column): Background (1, 1): *"Clear deep blue sky"*, Green (1, 2): *"Summer mountains"*, Red (1, 3): *"The Sun"*, Pale Blue (2, 1): *"The Moon"*, Light Orange (2, 2): *"A giant waterfall"*, Purple (2, 3): *"A giant waterfall"*, Blue (3, 1): *"Clean deep blue lake"*, Orange (3, 2): *"A large tree"*, Light Green (3, 3): *"A large tree"*

Figure S4. Mask overlay images of the generation result in Figure 2 of the main manuscript. Generation by our SEMANTICDRAW not only achieves high speed of convergence, but also high mask fidelity in the large-size region-based text-to-image synthesis, compared to the baseline MultiDiffusion [1]. Each cell shows how each mask (including the background one) maps to each generated region of the image, as described in the label below. Note that we have *not* provided any additional color or structural control other than our *semantic palette*, which is simply pairs of text prompts and binary masks.

of mask quantization.

As Figure 5 of the main manuscript shows, the mask smoothing is an optional preprocessing procedure before generation. Once users provide a set of masks corresponding to a set of text prompts they want to draw, the binary masks are smoothened with a low-pass filter such as Gaussian blurs. In order to perform masking with these continuous masks for discrete denoising steps of the accelerated schedulers [2, 4–6, 9], we create a set of binary masks from each of the continuous masks by thresholding with the noise levels predefined by the diffusion scheduler. For example, Figure 5 of the main manuscript shows five noise levels actually used in generating the results in the main manuscript and throughout this Supplementary Material. The resulting set of binary masks have monotonically increasing sizes as the corresponding noise levels become lower. Note that we can interpret a noise level of each generating step as a magnitude of uncertainty during the reverse diffusion process. Since the boundary of an object is fuzzier than the center of the object of prescribed masked region, the more uncertain boundary regions can be sampled only during the few latest steps where detailed textures dominant over structural development. Therefore, a natural way of applying these binary masks is in the order of increasing size. By applying each generated binary mask at the timestep with corresponding noise level, we effectively enlarge the size of the mask of a foreground text prompt as we proceed on the generative denoising steps.

The blurring and quantization of the binary masks have a nice interpretation of a *rough sketch*. In many cases where users prescribe masks to query for multi-object generation, the exact boundary locations for the best visual construction of an image are not known *a priori*. In other words, human creation of arts almost always starts with rough sketches. We can increase or decrease the standard deviation of the blur to control the roughness of the sketch, *i.e.*, the certainty of our designation on the boundary. This additional control knob is effective in creating AI-driven arts which inherently exploits high randomness in practice. For reference, Figure S3 shows the effect of increasing the blurriness at the mask proprocessing step. As the standard deviation of the mask blur increases from 0 to 32, the moss, the content of the mask, gradually shrinks and semantically blurred with the brick wall, the background content. As our supplementary code show, this *semantic mixing effect* of mask blurring and quantization is helpful to harmonize contents in generative editing tasks, *i.e.*, inpainting, where background images are predefined and not fully masked out during generation.

S2. More Results

In this section, we provide additional visual comparison results between baseline MultiDiffusion [1], a simple application of acceleration modules [5, 6] to the baseline, and our stabilized Algorithm S2. We show that our algorithm is capable of generating large-scale images from multiple regional prompts with a single commercial off-the-shelf

(a) Masks (b) LRDiff (45s) (c) (b)+LCM (1s) (d) **Ours** (1s)

Figure S5. Qualitative comparison between LRDiff+LCM and ours. Background prompt: "Iron Man and Hulk stand amidst the ruins, engaged in a fierce battle with each other." Left box prompt: "Iron-man" Right box prompt: "Hulk"

graphics card, *e.g.*, an RTX 2080 Ti GPU.

S2.1. Region-Based Text-to-Image Generation

We show additional region-based text-to-image generation results in Figure S6. In addition to Figure 6 of the main manuscript, the generated samples show that our method is able to accelerate region-based text-to-image generation consistently by $\times 10$ without compromising the generation quality. Moreover, Figure 2 of the main manuscript has shown that the benefits from our acceleration method for arbitrary-sized generation and region-based controls are indeed simultaneously enjoyable. Our acceleration method enables publicly available Stable Diffusion v1.5 [10] to generate a 1920×768 scene from eight hand-drawn masks in 59 seconds, which is $\times 52.5$ faster than the baseline [1] taking more than 51 minutes to converge into a low-fidelity image. Figure S4 shows mask fidelity of this generation. We can visualize that even if the generated image has larger dimension than the dimensions the model has been trained for, *i.e.*, 768×768, the mask fidelity is achieved under this accelerated generation. Locations and sizes of the Sun and the Moon match to the provided masks in near perfection; whereas mountains and waterfalls are harmonized within the overall image, without violating region boundaries. This shows that the flexibility and the speed of our generation paradigm, SEMANTICDRAW, is also capable of professional usage.

Furthermore, we have found that more recent methods such as LRDiff [8] also suffers the same instability problem when accelerated. Figure S5 shows one example. In this qualitative results, our method not only achieves faster generation speed ($\times 45$), but also enjoys better mask fidelity and perceptual quality. This further validates the significance of our strategy in professional interactive content creation.

Regarding that professional image creation process using diffusion models typically involves a multitude of resampling trials with different seeds, the original baseline model's convergence speed of one image per hour severely limits the applicability of the algorithm. In contrast, our acceleration method enables the same large-size region-based text-to-image synthesis to be done under a minute, making this technology practical to industrial usage.

S2.2. Panorama Generation

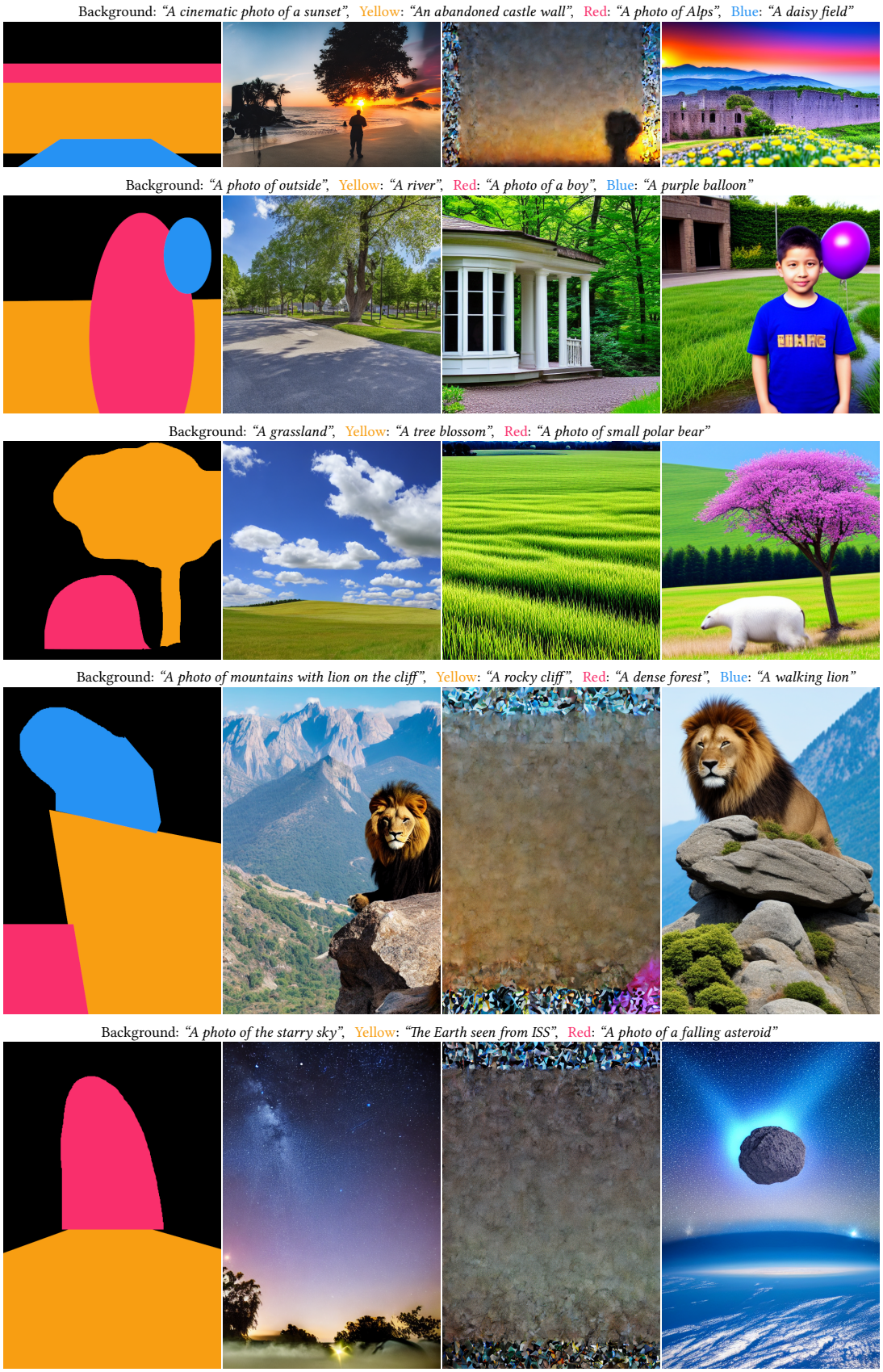
We can also visually compare arbitrary-sized image creation with panorama image generation task. As briefly mentioned in Section S1, comparing with this task reveals the problem of incompatibility between accelerating schedulers and current region-based multiple text-to-image synthesis pipelines. Figure S7 shows the results of large-scale panorama image generation using our method, where we generate 512×4608 images from a single text prompt. Naïvely applying acceleration to existing solution leads to blurry unrealistic generation, enforcing users to resort to more conventional diffusion schedulers that take long time to generate [12]. Instead, our method is compatible to accelerated samplers [2, 4–6, 9], showing $\times 13$ faster generation of images with sizes much larger than the resolutions of 512×512 or 768×768, for which the diffusion model [10] is trained. Combining results from Section S2.1 and S2.2 our Algorithm S2 significantly broadens the usability of diffusion models for professional content creators. This leads to the last section of this Supplementary Material, the description of our submitted demo application.

S3. Sample Application

This last section elaborates on the design and the example usage of our demo application of SEMANTICDRAW, introduced in Section 5 of the main manuscript. Starting from the basic description of user interface in Section S3.1, we discuss the expected usage of the app in Section S3.2. Our discussion mainly focuses on how real-time interactive content creation is achieved from accelerated region-based text-to-image generation algorithm we have provided.

S3.1. User Interface

As illustrated in Figure S8b, user interactions are classified into two groups, *i.e.*, the slow processes and the fast pro-



(a) Prompt (b) MD, 50 steps (c) MD+LCM, 5 steps (d) **Ours, 5 steps**

Figure S6. Additional region-based text-to-image synthesis results. Our method accelerates MultiDiffusion [1] up to $\times 10$ while preserving or even boosting mask fidelity.

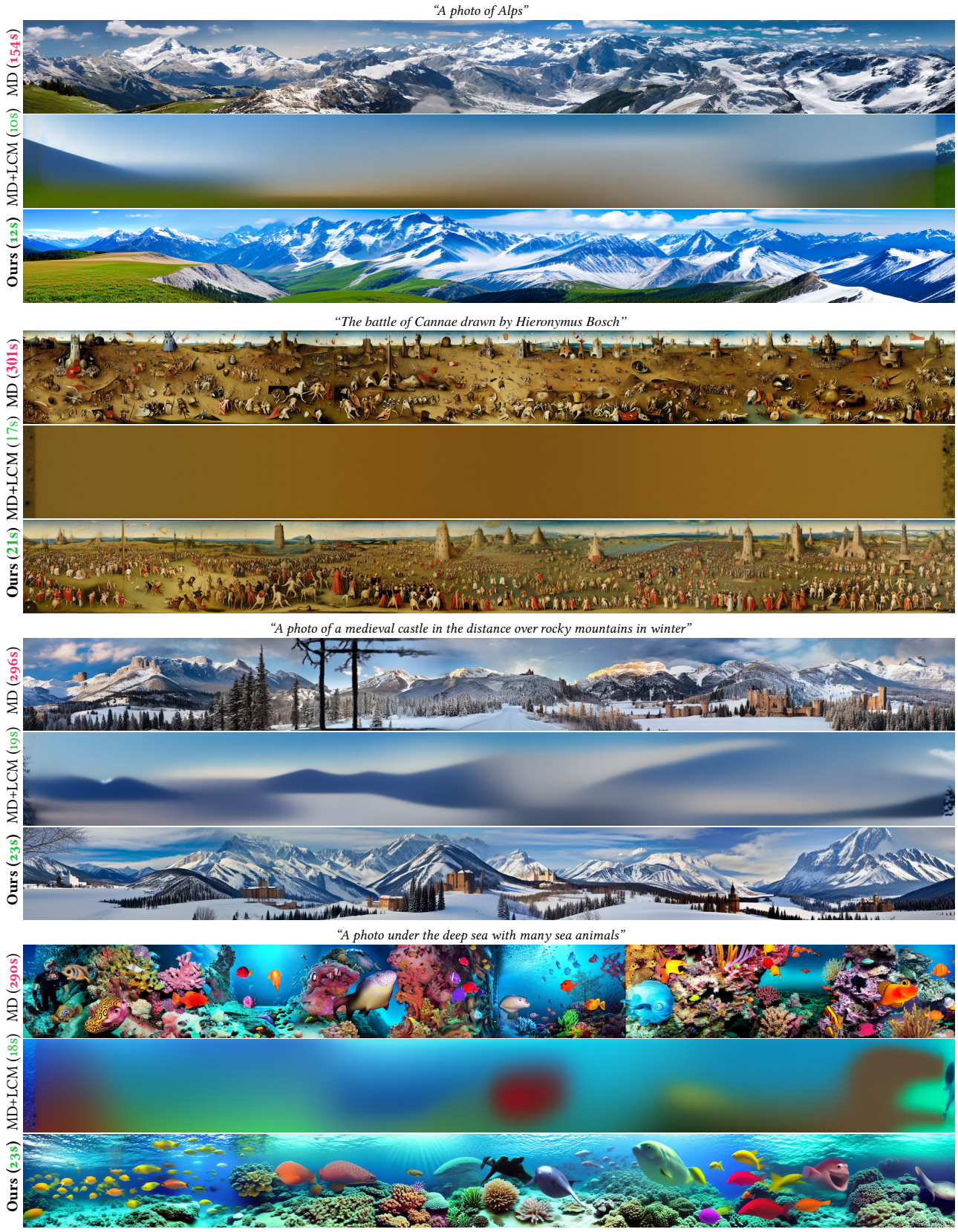
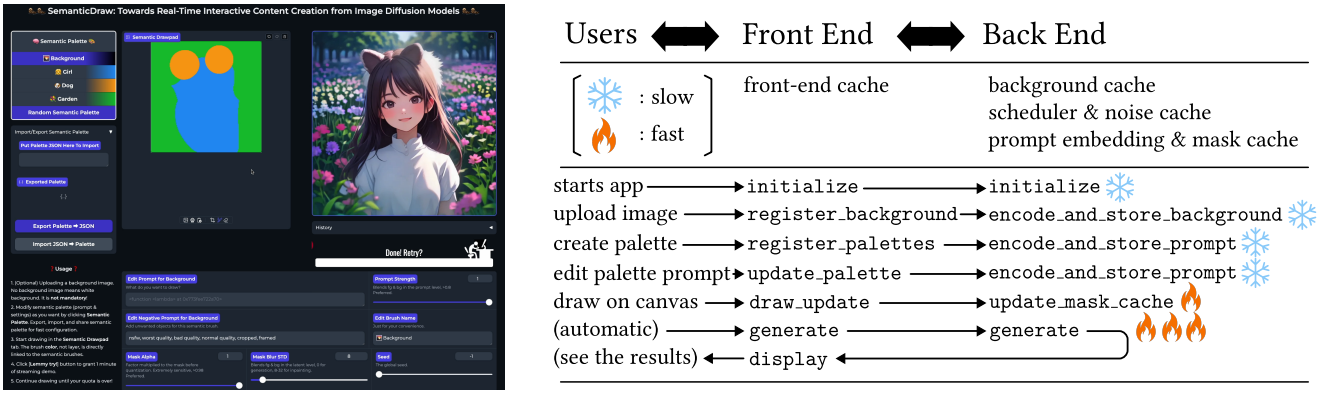


Figure S7. Additional panorama generation results. The images of size 512×4608 are sampled with 50 steps for MD and 4 steps for MD+LCM and Ours. Our SEMANTICDRAW can synthesize high-resolution images in seconds. We achieve $\times 13$ improvement in inference latency.



(a) Screenshot of the application.

(b) Application design schematics.

Figure S8. Sample application demonstrating *semantic palette* enabled by our SEMANTICDRAW algorithm. After registering prompts and optional background image, the users can create images in real-time by drawing with text prompts.

cesses, based on the latency of response from the model. Due to the high overhead of text encoder and image encoder, the processes involving these modules are classified as slow processes. However, operations such as preprocessing or saving of mask tensors and sampling of the U-Net for a single step take less than a second. These processes are, therefore, classified as fast ones. SEMANTICDRAW, our suggested paradigm of image generation, comes from the observation that, if a user first registers text prompts, image generation from user-drawn regions can be done *in real-time*.

The user interface of Figure S9 is designed based on the philosophy to maximize user interactions of the fast type and to hide the latency of the slow type. Figure S9 and Table S1 summarize the components of our user interface. The interface is divided into four compartments: the (a) *semantic palette*, which is a palette of registered text prompts (no. 1-2), the (b) drawing screen (no. 3-5), the (c) streaming display and controls (no. 6-7), and the (d) control panel for the additional controls (no 8-13). The (a) *semantic palette* manages the number of *semantic brushes* to be used in the generation, which will be further explained below. Users are expected to interact with the application mainly through (b) drawing screen, where users can upload backgrounds and draw on the screen with selected semantic brush. Then, by turning (c) streaming interface on, the users can receive generated images based on the drawn regional text prompts in real-time. The attributes of semantic brushes are modified through (d) control panel.

Types of the transaction data between application and user are in twofold: a (1) background and a (2) list of text prompt-mask pairs, named *semantic brushes*. The user can register these two types of data to control the generation stream. Each semantic brush consists of two part: (1) text prompt, which can be edited in the (d) control panel after clicking on the brush in (a) *semantic palette*, a set of available text prompts to draw with, and (2) mask, which can be edited by selecting the corresponding color brush at drawing tools (no. 5), and drawing on the drawing pad (no. 3) with a brush of any color. Note that in the released version of our code, the color of semantic brush does not affect generation results. Its color only separates a semantic brush from another for the users to discern.

As the interface of the (d) control panel implies, our reformulation of MultiDiffusion [1] provides additional hyperparameters that can be utilized for professional creators to control their creation processes. The mask alpha (no. 11) and the mask blur std (no. 12) determine preprocessing attributes of selected semantic brush. Before the mask is quantized into predefined noise levels of scheduler, as elaborated in Section S1.4, mask is first multiplied by mask alpha and goes through an isotropic Gaussian blur with a specified standard deviation. That is, given a mask $\boldsymbol{w}$, a mask alpha a, and the noise level scheduling function $\beta(t) = \sqrt{1-\alpha(t)}$, the resulting quantized mask $\boldsymbol{w}_{1:p}^{(t_i)}$ is:

$$\boldsymbol{w}_{1:p}^{(t_i)} = \mathbb{1}[a\boldsymbol{w} > \beta(t_i)]\,, \tag{S1}$$

where $\mathbb{1}[\cdot]$ is an indicator function taking the inequality as a binary operator to make a boolean mask tensor $\boldsymbol{w}_{1:p}^{(t_i)}$ at time t_i. The default noise levels $\beta(t)$ of the acceleration modules [2, 4–6, 9] are close to one, as shown in Figure 5 of the main manuscript. This makes mask alpha extremely sensitive. By changing its value only slightly, *e.g.*, down to 0.98, the corresponding prompt already skips first two sampling steps. This quickly degenerates the content of the prompt, and therefore, the mask alpha (no. 11) should be used in care. The effect of mask blur std (no. 12) is shown in Figure S3, and will not be further elaborated in this section. The seed of the system can be tuned by seed control (no. 13). Nonetheless, controlling pseudo-random generator will rarely be needed since the application generates

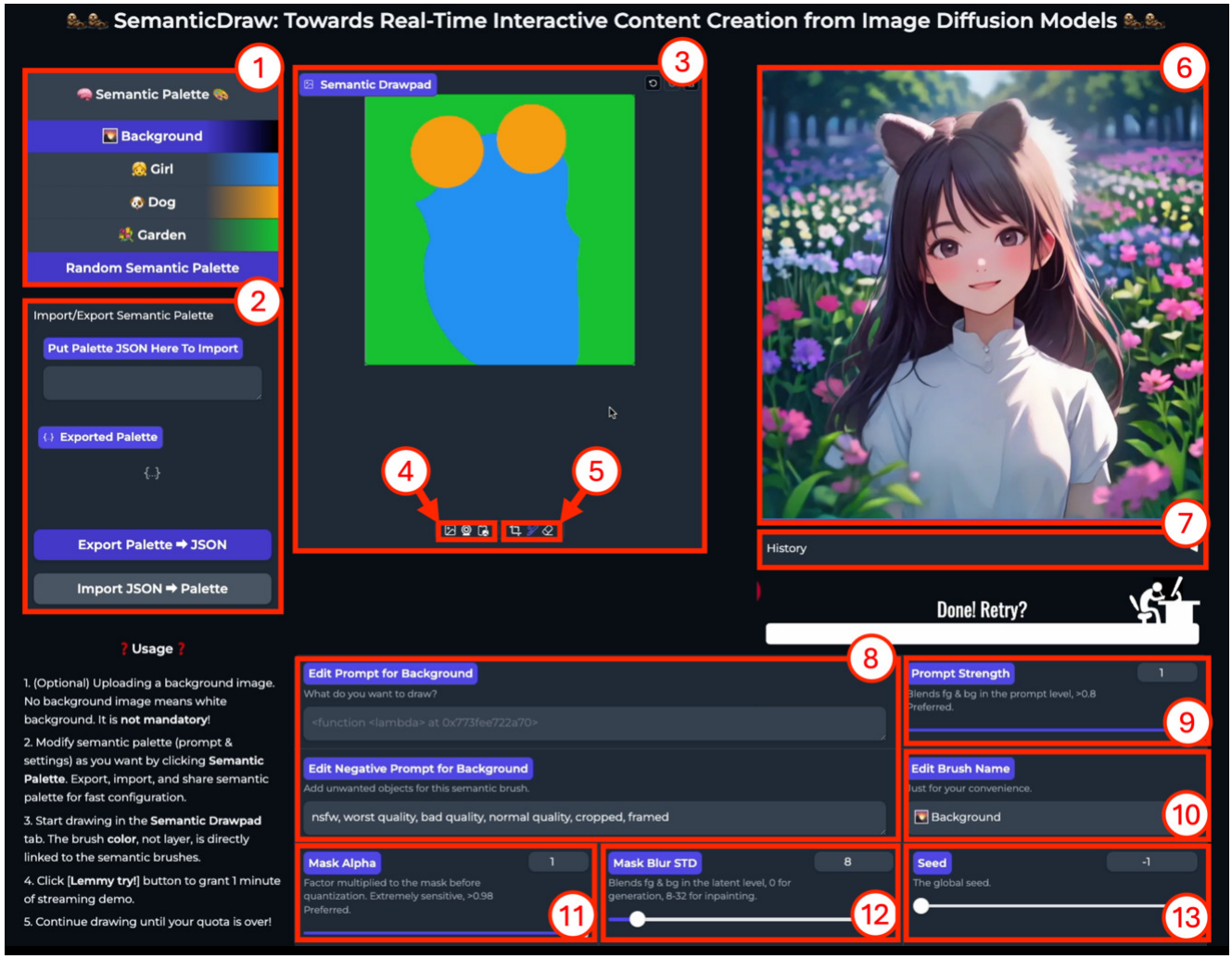
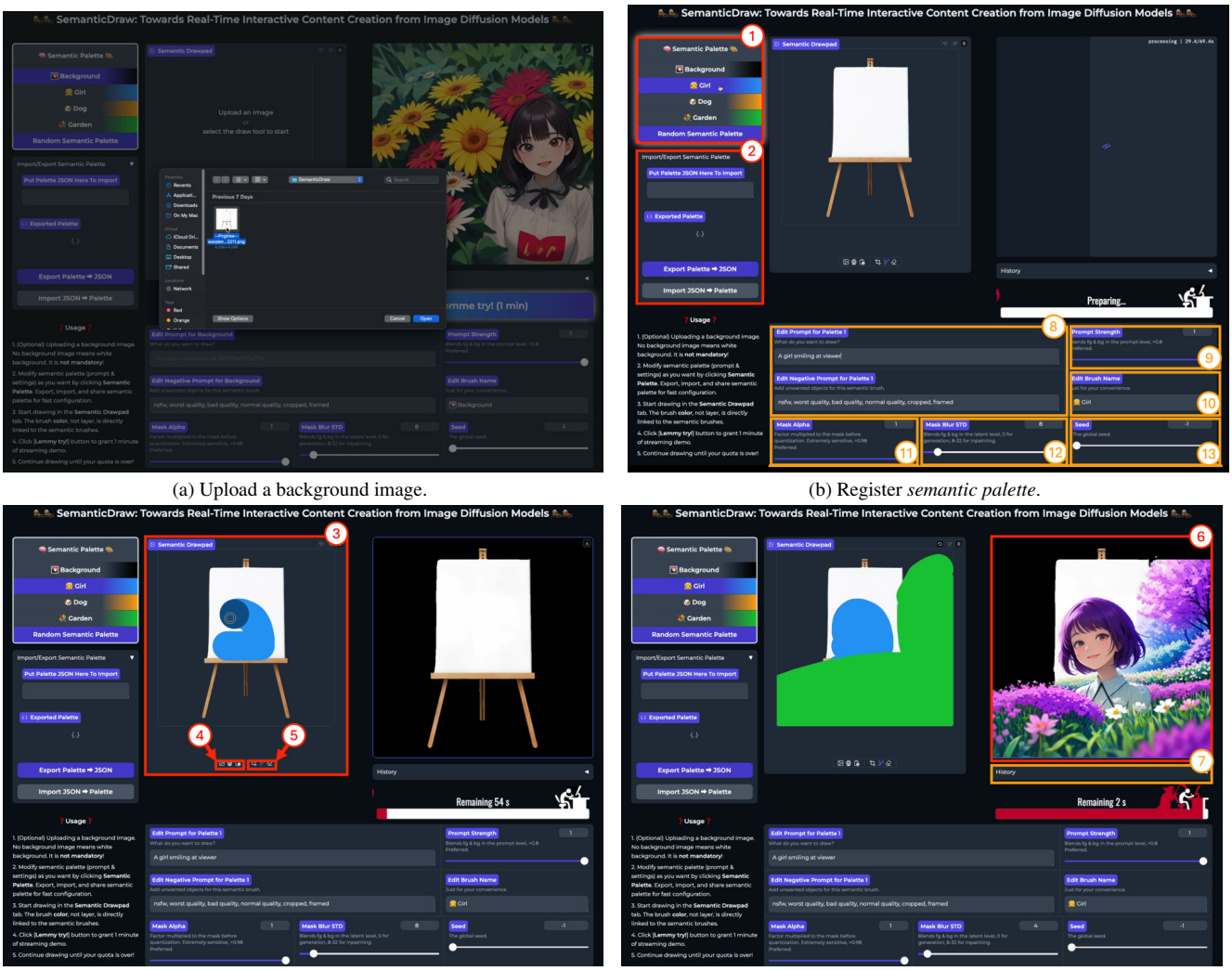


Figure S9. Screenshot of our supplementary demo application. Details of the numbered components are elaborated in Table S1.

Table S1. Description of each numbered component in the SEMANTICDRAW demo application of Figure S9.

No.	Component Name	Description
1	*Semantic palette*	Create and manage text prompt-mask pairs.
2	Import/export semantic palette	Easy management of text prompt sets to draw.
3	Main drawing pad	User draws at each semantic layers with brush tool.
4	Background image upload	User uploads background image to start drawing.
5	Drawing tools	Using brush and erasers to interactively edit the prompt masks.
6	Display	Generated images are streamed through this component.
7	History	Generated images are logged for later reuse.
8	Prompt edit	User can interactively change the positive/negative prompts at need.
9	Prompt strength control	Prompt embedding mix ratio between the current & the background. Helps content blending.
10	Brush name edit	Adds convenience by changing the name of the brush. Does not affect the generation.
11	Mask alpha control	Changes the mask alpha value before quantization. Recommended: > 0.95 .
12	Mask blur std. dev. control	Changes the standard deviation of the quantized mask of the current semantic brush.
13	Seed control	Changes the seed of the application.



(a) Upload a background image.

(b) Register *semantic palette*.

(c) Draw with semantic brushes.

(d) Play the stream and interact.

Figure S10. Illustrated usage guide of our demo application of SEMANTICDRAW.

images in an infinite stream. The prompt edit (no. 8) is the main control of semantic brush. The users can change text prompt even when generation is on stream. It takes exactly the total number of inference steps, *i.e.*, 5 steps, for a change in prompts to take effect. Further, we provide prompt strength (no. 9) as an alternative to highly sensitive mask alpha (no. 11) to control the saliency of the target prompt. Although modifying the alpha channel provides good intuition for graphics designer being already familiar to alpha blending, the noise levels of consistency model [2, 4–6, 9, 13] make the mask alpha value not aligned well with our intuition in alpha blending. Prompt strength is a mix ratio between the embeddings of the foreground text prompt of given semantic brush and background text prompt. We empirically find that changing the prompt strengths gives smoother control to the foreground-background blending strength than mask alpha. However, whereas the mask alpha can be applied locally, the prompt strength only globally takes effect. Therefore, we believe that the two controls are complementary to one another.

Finally, we provide seed-fixing option that enables incremental generation for drawing-like experience. The difference between simple streaming generation and streaming generation with our seed-fixing option is elaborated in Figure S11. By not only caching the prompt embeddings during streaming but also sharing noise tensors within a stream of generation, we can simply switch into incremental editing in our application. Therefore, with the seed-fixing option, we can maintain strong consistency across entire stream of generation, which we may call a *session* of content creation. This enables content creators to switch from random ideation to detailed editing and vice versa, greatly

Figure S11. Sequential generation of frames from real-time drawing of masks. Top row: Original without seed-fixing. Bottom row: Increased determinism with seed-fixing option. A row of images comes sequentially from a single stream of generation given the same sequence of interactive controls (from left to right).

increasing the practicality of the application. Both options are available in our official code.

S3.2. Basic Usage

We provide the simplest procedure of creating images from SEMANTICDRAW pipeline. Screenshots in Figure S10 illustrate the four-step process.

1. Start the Application. After installing the required packages, the user can open the application with the following command prompt:

```
python app.py --model
    "KBlueLeaf/kohaku-v2.1" --height 512
    --width 512
```

The application front-end is web-based and can be opened with any web browser through `localhost:8000`. We currently support various baseline architecture including Stable Diffusion 1.5 [10], Stable Diffusion XL [7], and Stable Diffusion 3 [11] checkpoints for `--model` option. For SD1.5, we support latent consistency model (LCM) [5, 6] and Hyper-SD [9], for SDXL, we support SDXL-Lightning [4], and for SD3, we support Flash Diffusion [2] for acceleration of the generation process. The height and the width of canvas should be predefined at the startup of the application.

2. Upload Background Image. See Figure S10a. The first interaction with the application is to upload any image as background by clicking the background image upload (no. 4) panel. The uploaded background image will be resized to match the canvas. After uploading the image, the background prompt of the uploaded image is automatically generated for the user by pre-trained BLIP-2 model [3]. The background prompt is used to blend between foreground and background in prompt-level globally, as elaborated in Section S3.1. The interpolation takes place when a foreground text prompt embedding is assigned with a prompt strength less than one. User is always able to change the background prompt like other prompts in the *semantic palette*.

3. Type in Text Prompts. See Figure S10b. The next step is to create and manage semantic brushes by interacting with the *semantic palette* (no. 1). A minimal required modification is text prompt assignment through the prompt edit (no. 8) panel. The user can additionally modify other options in the control panel marked as yellow in Figure S10b.

4. Draw. See Figure S10c. A user may start drawing by selecting a brush in drawing tools (no. 5) toolbar that matches the user-specified text prompt in the previous step. Grab a brush, draw, and submit the drawn masks. After initiating the content creation, the images are streamed through the display (no. 6) in real-time from dynamically changing user inputs. The past generations are saved in history (no. 7).

References

[1] Omer Bar-Tal, Lior Yariv, Yaron Lipman, and Tali Dekel. MultiDiffusion: Fusing diffusion paths for controlled image generation. In *ICML*, 2023. 1, 2, 4, 5, 6, 8

[2] Clement Chadebec, Onur Tasar, Eyal Benaroche, and Benjamin Aubin. Flash Diffusion: Accelerating any conditional diffusion model for few steps image generation. In *AAAI*, 2025. 1, 4, 5, 8, 10, 11

[3] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, 2023. 11

[4] Shanchuan Lin, Anran Wang, and Xiao Yang. SDXL-Lightning: Progressive adversarial diffusion distillation. *arXiv preprint arXiv:2402.13929*, 2024. 1, 4, 5, 8, 10, 11

[5] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-

resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023. 1, 4, 11

[6] Simian Luo, Yiqin Tan, Suraj Patil, Daniel Gu, Patrick von Platen, Apolinário Passos, Longbo Huang, Jian Li, and Hang Zhao. LCM-LoRA: A universal stable-diffusion acceleration module. *arXiv preprint arXiv:2311.05556*, 2023. 1, 4, 5, 8, 10, 11

[7] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *ICLR*, 2024. 11

[8] Zipeng Qi, Guoxi Huang, Chenyang Liu, and Fei Ye. Layered Rendering Diffusion Model for Controllable Zero-Shot Image Synthesis. In *ECCV*, 2024. 5

[9] Yuxi Ren, Xin Xia, Yanzuo Lu, Jiacheng Zhang, Jie Wu, Pan Xie, XING WANG, and Xuefeng Xiao. Hyper-SD: Trajectory segmented consistency model for efficient image synthesis. In *NeurIPS*, 2024. 1, 4, 5, 8, 10, 11

[10] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 5, 11

[11] Axel Sauer, Frederic Boesel, Tim Dockhorn, Andreas Blattmann, Patrick Esser, and Robin Rombach. Fast high-resolution image synthesis with latent adversarial diffusion distillation. In *SIGGRAPH Asia*, 2024. 11

[12] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2020. 5

[13] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *ICML*, 2023. 1, 10