Adaptively-Realistic Image Generation from Stroke and Sketch with Diffusion Model *Supplementary Materials*

Shin-I Cheng*[1], Yu-Jie Chen*[1], Wei-Chen Chiu[1], Hung-Yu Tseng[2], and Hsin-Ying Lee[3]

[1]National Chiao Tung University, Taiwan, [2]Meta, [3]Snap Inc.

1. Dataset and Preprocessing Details

1.1. Dataset.

We consider three datasets: AFHQ [2], Landscapes [8], and Oxford Flower [7] in our experiments. AFHQ dataset has three domains: 5153 training and 500 testing images of“cat”, 4739 training and 500 testing images of “dog”, and 4738 training and 500 testing images of “wildlife” (e.g. tiger, lion, wolf, etc). Oxford Flower dataset contains 8189 images (split to 7189 images for training and 1000 for testing), and Landscapes (LHQ) dataset has 90000 images (split to 81000 images for training, and 9000 for testing).

1.2. Generating Sketch-stroke Pairs from Given Datasets.

For each image in the aforementioned three datasets, we prepare the corresponding sketch and stroke images for training and testing in our proposed task.

Sketch Generation. We utilize the official pretrained model of Photo-sketching [5], a GAN-based network extracting the contour drawings from the given image, to obtain the paired sketch data for images of the datasets used in our experiments. Noting that we perform additional foreground extraction using GrabCut algorithm in OpenCV specifically for the Oxford Flower images before applying the sketch generation, as we find that the images in this dataset usually have many leaves behind the main flowers which would cause some distraction for capturing outlines of the main object.

Stroke Generation. We do image-to-painting translation via applying two methods, Stylized Neural Painting [10] and Paint Transformer [6], in order to generate the paired stroke data for images of the datasets used in our experiments. Both Stylized Neural Painting and Paint Transformer are state-of-the-art image-to-painting frameworks and able to produce a sequence of meaningful and faithful stroke prediction. Considering that practical stroke images manually created by human users usually contain only coarse features, we generate stroke data by randomly selecting the intermediate canvas produced during the progressive procedure of Stylized Neural Painting or Paint Transformer. We apply Stylized Neural Painting on the AFHQ dataset and Paint Transformer on the datasets with more images (i.e. Oxford Flower and Landscapes) since Paint Transformer is less time-consuming.

1.3. Generating Sketch-stroke Pairs from Custom Input Images.

Given the custom input images $\mathbf{c}_{\text{comb}}$ which are the synthetic ones (e.g. the input images shown in Figure.7 of our main manuscript), as they are different from the real images (e.g. what we have in the AFHQ, Landscapes, and Oxford Flower datasets), we perform the following operations (different from the way that we described in the previous subsection 1.2) in order to extract black-white sketches $\mathbf{c}_{\text{sketch}}$ and colored stroke images $\mathbf{c}_{\text{stroke}}$. Firstly, we adopt the GrabCut algorithm in OpenCV to extract and concentrate on the foreground/main object (where the subordinate parts/fragments around the image border, which are likely to be background, are removed). Next, we utilize Canny algorithm to detect the edges, followed by finding the contour information upon the edges via [9] (findContours function in OpenCV), to obtain the black-white sketch $\mathbf{c}_{\text{sketch}}$. As for the colored strokes $\mathbf{c}_{\text{stroke}}$, they are generated by making the contour pixels in white upon the input image $\mathbf{c}_{\text{comb}}$.

For the two applications (i.e. multi-conditioned local editing and region-sensitive stroke-to-image) unleashed by our proposed DiSS, as users would provide their input $\mathbf{c}_{\text{comb}}$ via directly drawing on the top of the original image, we adopt another procedure to extract the sketch $\mathbf{c}_{\text{sketch}}$ and stroke $\mathbf{c}_{\text{stroke}}$: Firstly, we apply a thresholding operation (grayscale value > 50 to white; otherwise black) on the input to obtain the black-white sketches $\mathbf{c}_{\text{sketch}}$; then, we extract the colored strokes $\mathbf{c}_{\text{stroke}}$ by replacing the black pixels with white color, which is achieved by the bitwise AND operation on the input image $\mathbf{c}_{\text{comb}}$ and a binary mask (thresholding on the saturation value in which those pixels with saturation > 0 are labelled as 1, otherwise 0).

2. Implementation Details

We implement the models with Pytorch. The implementation details of DiSS are provided in Section 2.1, in which we describe the network architecture, the settings of hyper-parameters, how we design the computation for the downsampling size $\mathbf{N}$ used in realism control, and the algorithm of the adaptively-realistic image generation from stroke and sketch (cf. the first paragraph in Sec. 4.1 of our main manuscript). For the two applications that our DiSS unleashes, i.e. multi-conditioned local editing and region-sensitive stroke-to-image, we explain the details and provide their algorithms in Section 2.2.

2.1. DiSS

Network Architecture. We modify the UNet model in [3] to realize the posterior prediction in our sketch- and stroke-guided diffusion model (i.e. $\hat{\epsilon}_\theta(x_t, t, \mathbf{c}_{\text{sketch}}, \mathbf{c}_{\text{stroke}})$ in Sec. 3.2 of our main manuscript). The basic UNet model is constructed with a sequence of residual layers and downsampling convolutions as encoder, followed by a sequence of residual layers and the corresponding upsampling convolutions as decoder, with skip connections linking the intermediate layers with the same spatial size. After firstly being proposed in [4], the UNet model for diffusions is further improved by [3] with higher sampling quality. We then modify it by extending the input channel from 3 to 7, which allows the concatenation between the input image and the additional two conditions, i.e. sketches (1-channel) and strokes (3-channel).

Settings of Hyper-parameters. We apply the same settings of hyper-parameters among the three datasets, as shown below:

	AFHQ Cat 512×512 Oxford Flower 512×512 Landscapes-HQ 512×512
Diffusion steps	1000
Noise schedule	linear
Channels	128
Depth	3
Channels multiple	0.5, 1, 1, 2, 2, 4, 4
Heads channels	64
Attention resolution	32, 16, 8
BigGANup/downsample	yes
Dropout	0.0
Batchsize	2
Learning rate	1e-4

Table 1: **The settings of hyper-parameters among the three datasets.** We apply the same settings of hyper-parameters for the three datasets on our diffusion models.

Realism Control. The basic concept of our realism control is mainly inherited from ILVR [1], which utilizes a low-pass filter[1] to operate a downsampling procedure on the given reference $\mathbf{c}_{\text{comb}}$ (from the original image size $\mathbf{m} \times \mathbf{m}$ to size $\mathbf{N} \times \mathbf{N}$ and upsampling back to the original size). The key difference between the realism control of DiSS and ILVR is that the realism control of DiSS permits the downsampling to an arbitrary size $\mathbf{N}$ (from Equation 10 in the main manuscript), while ILVR performs downsampling to $\mathbf{m}/2^s$ (s is a non-negative integer) in which it only allows specific sizes.

[1] https://github.com/assafshocher/ResizeRight

To enable a continuous realism scale $s_{\text{realism}} \sim [0.0, 1.0]$, the higher (respectively lower) values of s_{realism} should be corresponding to a smaller (respectively larger) downsampling size $\mathbf{N}$ in order to realize a trade-off between the realism and the consistency for the synthesized output image. The computation $\mathbf{N}$ with respect to the corresponding realism scale s_{realism} is provided in Equation 10 of the main manuscript, in which we assume an affine relation between $\mathbf{N}$ and s_{realism}, i.e. $\mathbf{N} = s_{\text{realism}} \times a + b$ where a and b are scalars. In the following we provide the detailed explanation on how such computation is derived. Basically, when $s_{\text{realism}} = 1.0$ (the most realistic and the least consistent to $\mathbf{c}_{\text{comb}}$), we would like to force the transformed size $\mathbf{N} = 1$, which passes the least information of the reference image $\mathbf{c}_{\text{comb}}$ during the filtering; On the other hand, we make the transformed size $\mathbf{N} = \mathbf{m}$ (size remains the same, passes the most information of the reference image $\mathbf{c}_{\text{comb}}$ during the filtering) when $s_{\text{realism}} = 0.0$ (the least realistic and the most consistent to $\mathbf{c}_{\text{comb}}$). In order to fulfil such purpose, we hence set $a = -(\mathbf{m} - 1)$ and $b = \mathbf{m}$ in which the formulation becomes:

$$\mathbf{N} = -s_{\text{realism}}(\mathbf{m}/1 - 1) + (\mathbf{m}/1). \tag{1}$$

However, in practice, we discover that when applying $s_{\text{realism}} = [0.0, 0.8]$, the results are highly consistent to $\mathbf{c}_{\text{comb}}$. Consequently, we substitute the divisor in the formulation above with 8 to achieve adaptively-photorealistic translation with $s_{\text{realism}} = [0.0, 1.0]$. Furthermore, we append a constant term k to adapt on different datasets, $k = 0$ for the object-level dataset (AFHQ, flowers) and $k = 16$ for the scene-level dataset (landscapes). The final formulation hence becomes:

$$\mathbf{N} = -s_{\text{realism}}(\mathbf{m}/8 - 1) + (\mathbf{m}/8) + k \tag{2}$$

Algorithm. The detailed algorithm for realizing the overall "adaptively-realistic image generation from stroke and sketch", what our DiSS does, is presented in Algorithm 1.

Algorithm 1 DiSS

```
1: **Input:** Input custom image $\mathbf{c}_{\text{comb}}$
2: **Output:** Generated image $x = x_0$
3: Extract $\mathbf{c}_{\text{comb}} \rightarrow \mathbf{c}_{\text{sketch}}, \mathbf{c}_{\text{stroke}}$
4: Sample $x_T \sim \mathcal{N}(0, \mathbf{I})$
5: **For** $t = T$,...,1 **do**
6:     $\tilde{x}_{t-1} \sim \hat{p}_\theta(\tilde{x}_{t-1}|x_t, \mathbf{c}_{\text{sketch}}, \mathbf{c}_{\text{stroke}})$
7:     $\mathbf{c}_{\text{comb}_{t-1}} \sim q(\mathbf{c}_{\text{comb}_{t-1}}|\mathbf{c}_{\text{comb}_0})$     $\{\mathbf{c}_{\text{comb}_0} = \mathbf{c}_{\text{comb}}\}$
8:     $x_{t-1} \leftarrow \tilde{x}_{t-1} - LP_{\mathbf{N}}(\tilde{x}_{t-1}) + LP_{\mathbf{N}}(\mathbf{c}_{\text{comb}_{t-1}})$
9: **End for**
10: **Return** $x_0$
```

2.2. Applications

As described in the main manuscript, our proposed DiSS without any model retraining is able to unleash two applications, i.e. multi-conditioned local editing and region-sensitive stroke-to-image, in which here we provide their detailed descriptions in the following as well as their algorithms (Algorithm 2 and 3 respectively).

- **Multi-conditioned Local Editing.** Our proposed method can naturally realise both sketch and stroke local editing simultaneously on an existing image. As illustrated in Algorithm 2, we can handle an real image with guided local sketch and stroke drew on top of it as input, via extracting the corresponding sketch part and stroke part followed by performing sketch and stroke modifications with our proposed two-directional classifier-free guidance and realism control. We achieve this application mainly owing to the three-dimensional control technique that enables both sketch and stroke guidance while the realism control keeps the unmodified region of images and enhance the edited parts at the same time.

- **Region-sensitive stroke-to-image.** DiSS enables the partial color-conditioning on the specified regions and provides variations on the white colored regions. Algorithm 3 shows that after performing the two-directional classifier-free guidance, we apply an additional step before latent variable refinement at each time step to enforce the consistency of partial stroke. Specifically, we mask the non-colored regions to allow the variations and append the noisy colored information to the current latent image (line 9). Also note that here we use the stroke condition only, instead of the combination of sketch and stroke, to refine the current step image to further strengthen the partial stroke guidance.

3. User-Friendly Interface of DiSS

We provide a user-friendly webpage that enables user to create their own drawing and performs all DiSS applications (three dimensional control image generation, multi-conditioned local editing, and region-sensitive stroke-to-image). We present the screenshots of running our user interface in Figure 1, which sequentially demonstrates the initial state (cf. top subfigure), processing state after user creating a drawing, and the final generated result. Our interface allows the users to draw on their own (an example of this is provided in Figure 2) or simply upload an image. By clicking "Generate" button, the loading icon will show up and DiSS starts processing the input image (cf. the subfigure in the middle). The results will show on the webpage when the generation is done, the screenshot is shown in the bottom subfigure.

4. Additional Results

More qualitative results of two applications enabled by our proposed DiSS, i.e. multi-conditioned local editing and region-sensitive stroke-to-image, are shown in Figure 3. Also, more qualitative results demonstrating the trade-off between realism and consistency, together with the corresponding LPIPS score, are provided in Figure 4. Lastly, we provide another set of qualitative results of using different stroke and sketch scales on Lanscapes dataset in Figure 5.

References

[1] Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon. ILVR: Conditioning method for denoising diffusion probabilistic models. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[2] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. StarGAN v2: Diverse image synthesis for multiple domains. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[3] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[4] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[5] Mengtian Li, Zhe Lin, Radomir Mech, Ersin Yumer, and Deva Ramanan. Photo-sketching: Inferring contour drawings from images. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[2] https://freesvg.org/1497040842
[3] https://freesvg.org/1528308068-65465
[4] https://freesvg.org/dior-2
[5] https://freesvg.org/funny-cat-head-vector-illustration
[6] https://www.nicepng.com/maxp/u2q8e6e6y3t4r5i1/
[7] https://freesvg.org/1532149926
[8] https://free-vectors.net/nature/river-vector
[9] https://free-vectors.net/nature/summer-mountain-landscape-vector

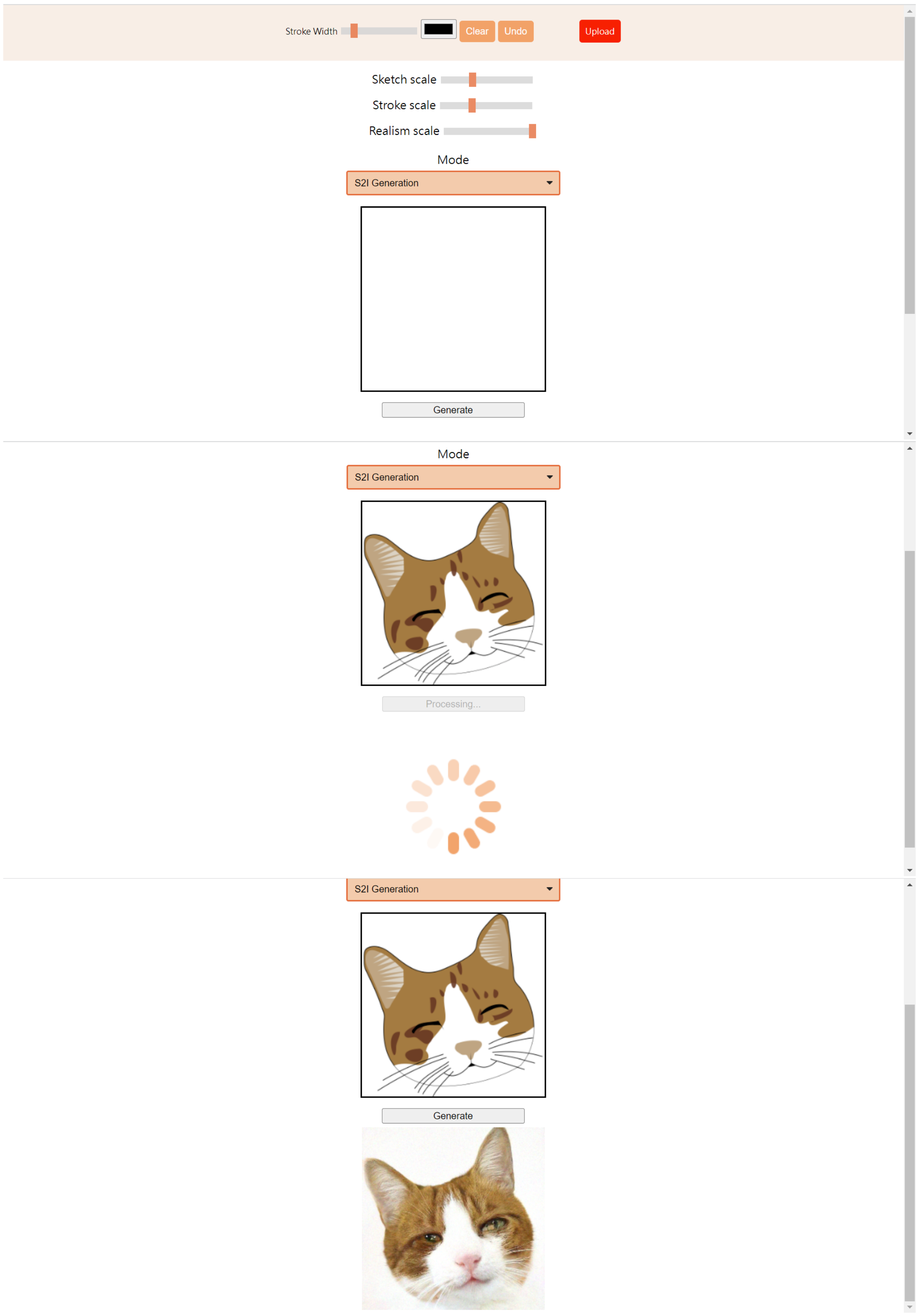


Figure 1: **DiSS User Interface.** We build a user interface enabling all the applications of DiSS. The screenshots from top to bottom respectively show the initial state, the processing state after creating a drawing, and the final generation result.

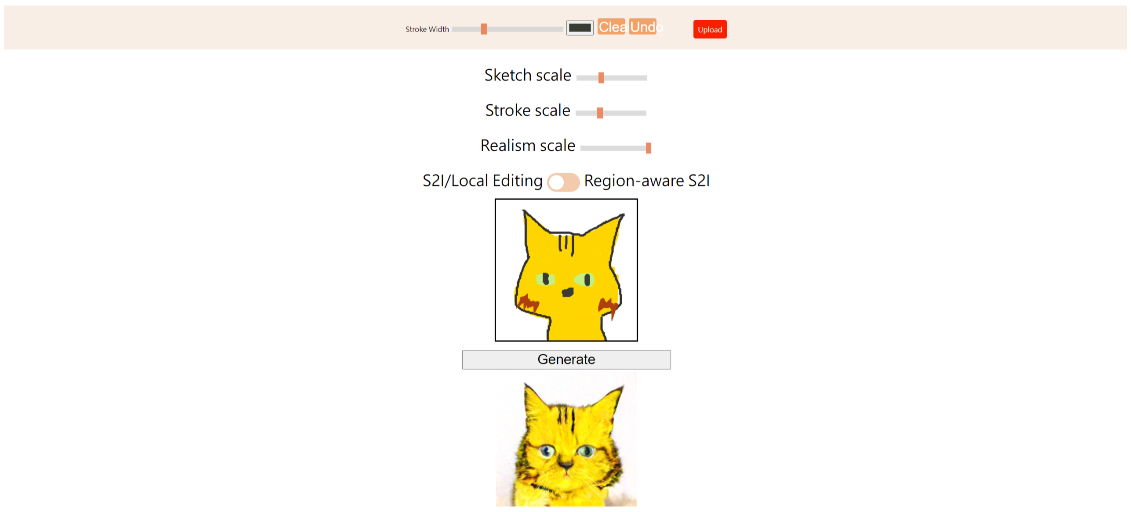


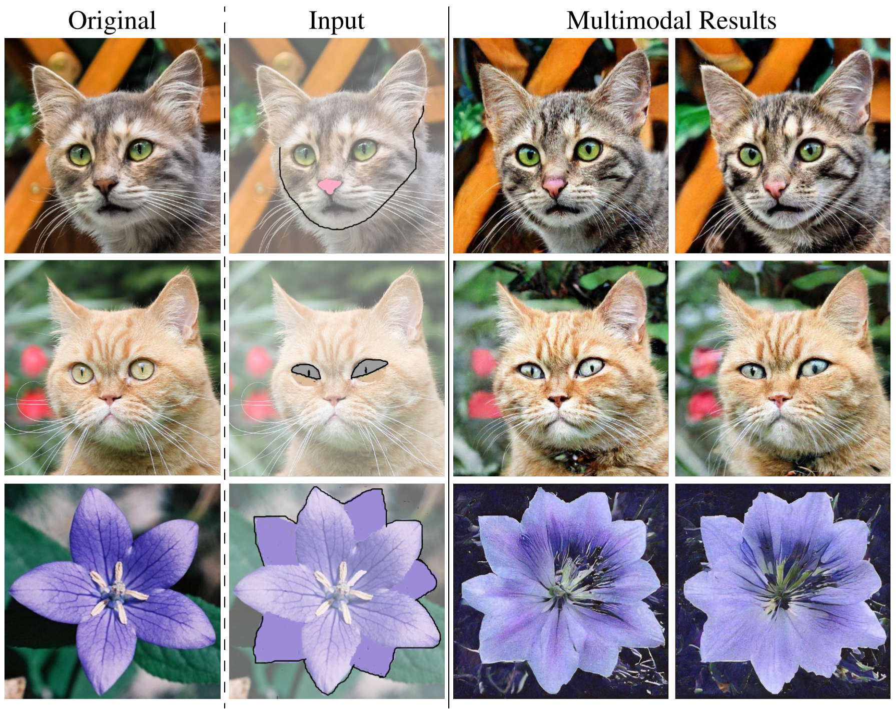
Figure 2: **DiSS User Interface with Hand-Drawing.** We play DiSS on the user interface by drawing on our own.

Algorithm 2 Application: Multi-conditioned Local Editing.

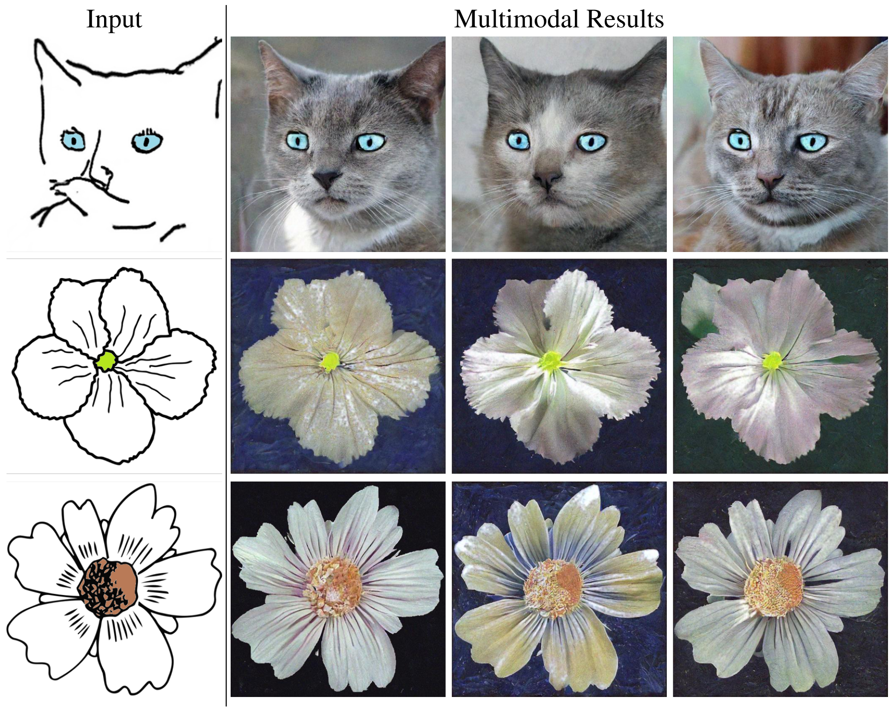
```
1: Input: Original image with hand-drawing editing, $\acute{x}$
2: Output: Edited image $x_{\text{edited}} = x_{\text{edited}0}$
3: Extract $\acute{x} \rightarrow \mathbf{c}_{\text{sketch}}$: local sketch editing, $\mathbf{c}_{\text{stroke}}$: stroke editing + the original image
4: $R$: range of the timestep refinement
5: Sample $x_{\text{edited}T} \sim \mathcal{N}(0, \mathbf{I})$
6: For $t = T$,...,1 do
7:     $\tilde{x}_{\text{edited}_{t-1}} \sim \hat{p}_\theta(\tilde{x}_{\text{edited}_{t-1}} | x_{\text{edited}_t}, \mathbf{c}_{\text{sketch}}, \mathbf{c}_{\text{stroke}})$
8:     If $t > R$ do
9:         $\mathbf{c}_{\text{comb}_{t-1}} \sim q(\mathbf{c}_{\text{comb}_{t-1}} | \mathbf{c}_{\text{comb}_0})$    $\{\mathbf{c}_{\text{comb}_0} = \mathbf{c}_{\text{comb}}\}$
10:        $x_{\text{edited}_{t-1}} \leftarrow \tilde{x}_{\text{edited}_{t-1}} - LP_{\mathbf{N}}(\tilde{x}_{\text{edited}_{t-1}}) + LP_{\text{N}}(\mathbf{c}_{\text{comb}_{t-1}})$
11: End for
12: Return $x_{\text{edited}0}$
```

Algorithm 3 Application: Region-sensitive Stroke-To-Image.

```
1: Input: Guided condition $\mathbf{c}_{\text{comb}}$ with sketch and partial colored stroke
2: Output: Region-sensitive stroke-to-image $x = x_0$
3: Extract $\mathbf{c}_{\text{comb}} \rightarrow \mathbf{c}_{\text{sketch}}, \mathbf{c}_{\text{stroke}}$ (Section 1.3)
4: $R$: range of the timestep refinement
5: mask: 0 on colored region of $\mathbf{c}_{\text{stroke}}$; otherwise 1
6: Sample $x_T \sim \mathcal{N}(0, \mathbf{I})$
7: For $t = T$,...,1 do
8:     $\tilde{x}_{t-1} \sim \hat{p}_\theta(\tilde{x}_{t-1} | x_t, \mathbf{c}_{\text{sketch}}, \mathbf{c}_{\text{stroke}})$
9:     If $t > R$ do
10:        $\mathbf{c}_{\text{stroke}t-1} \sim q(\mathbf{c}_{\text{stroke}t-1} | \mathbf{c}_{\text{stroke}_0})$ $\{\mathbf{c}_{\text{stroke}_0} = \mathbf{c}_{\text{stroke}}\}$
11:        $\tilde{x}_{t-1} \leftarrow \text{mask} \times \tilde{x}_{t-1} + (1 - \text{mask}) \times \mathbf{c}_{\text{stroke}t-1}$    { append $\mathbf{c}_{\text{stroke}_{t-1}}$ on $\tilde{x}_{t-1}$}
12:        $x_{t-1} \leftarrow \tilde{x}_{t-1} - LP_{\mathbf{N}}(\tilde{x}_{t-1}) + LP_{\text{N}}(\mathbf{c}_{\text{stroke}_{t-1}})$
13: End for
14: Return $x_0$
```

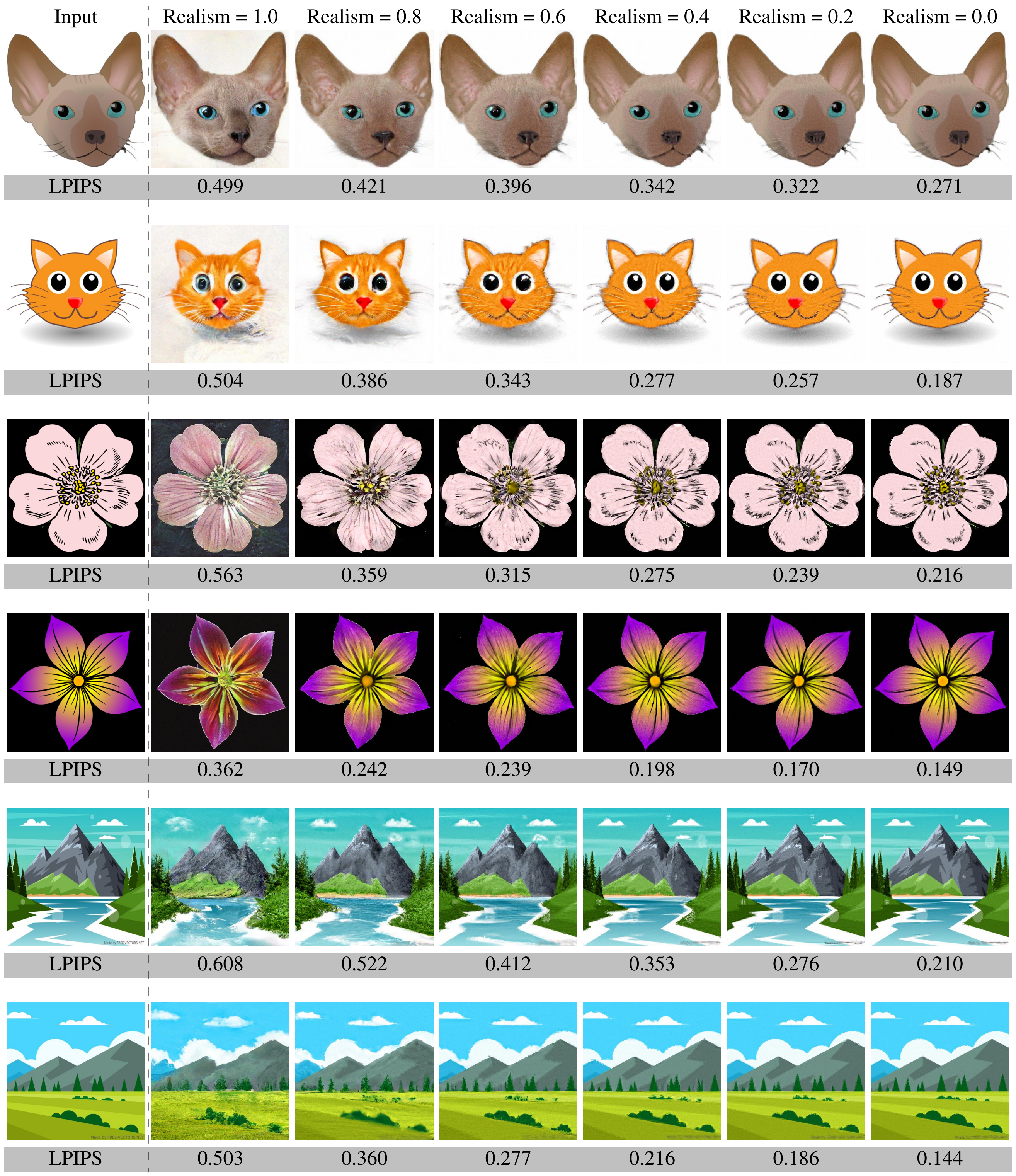


(a) Multi-conditioned Local Editing.



(b) Region-sensitive Stroke-to-Image.

Figure 3: **More Applications results.** (a) By drawing the new contour or color on an existing image, the proposed model enables the mask-free image editing. (b) With the partial colored stoke as the input, the proposed method synthesizes more diverse contents in the non-colored region. Here we use cats and flowers[23]as examples.



Input	Realism = 1.0	Realism = 0.8	Realism = 0.6	Realism = 0.4	Realism = 0.2	Realism = 0.0
LPIPS	0.499	0.421	0.396	0.342	0.322	0.271
LPIPS	0.504	0.386	0.343	0.277	0.257	0.187
LPIPS	0.563	0.359	0.315	0.275	0.239	0.216
LPIPS	0.362	0.242	0.239	0.198	0.170	0.149
LPIPS	0.608	0.522	0.412	0.353	0.276	0.210
LPIPS	0.503	0.360	0.277	0.216	0.186	0.144

Figure 4: **Trade-off between realism and consistency to image guidance.** We demonstrate the trade-off between the image realism and the correspondence to the input guidance, where the realism scale is varied from low (0.0, *right*) to high (1.0, *left*). We also show the LPIPS scores between the generated image and the input guidance. Both the object-level and scene-level input guidance images[4][5][6][7][8][9] are used in this experiment.

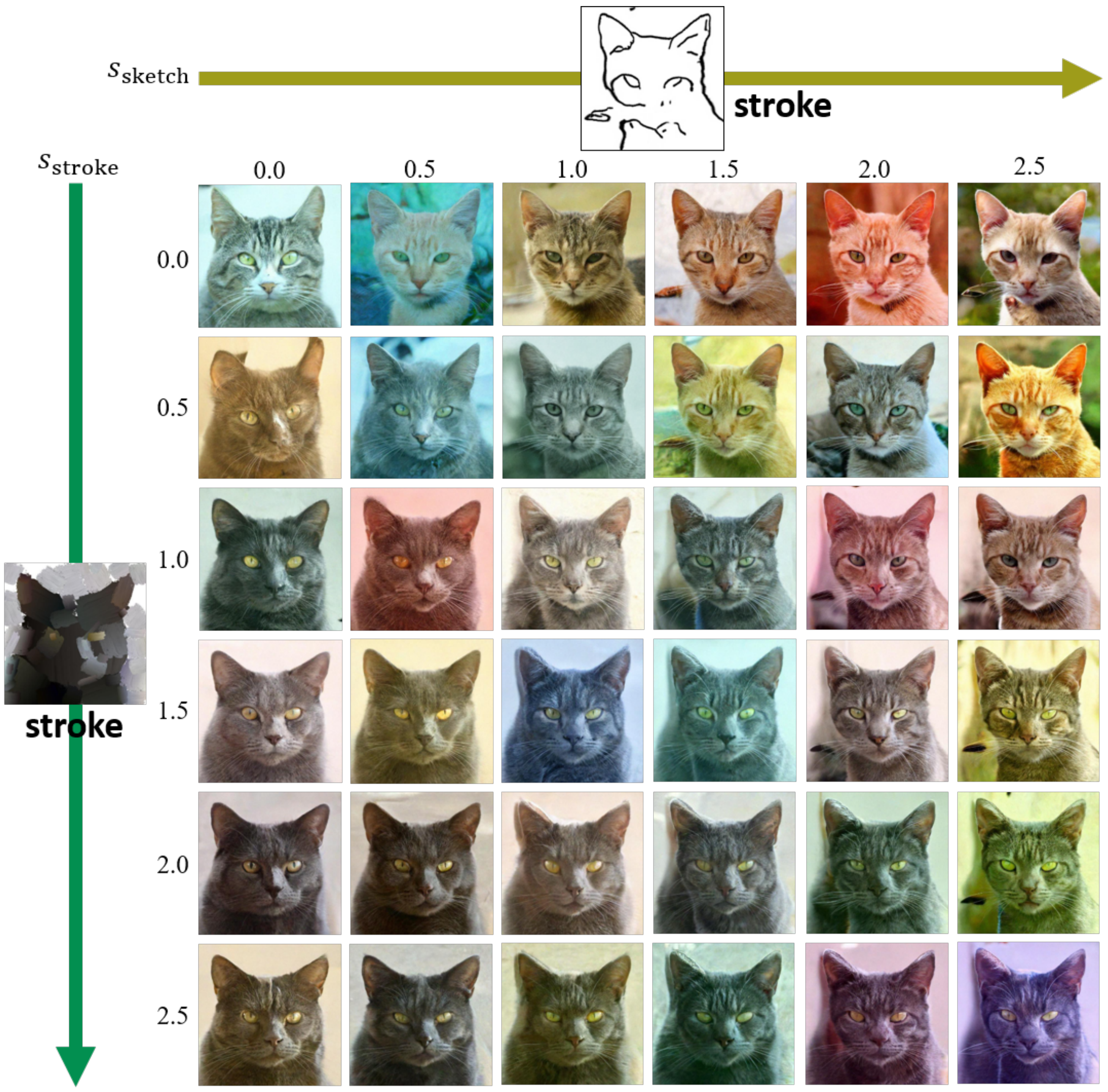


Figure 5: **Qualitative results on AFHQ cat dataset of using different stroke and sketch scales.** The top-left corner show the results generated without guidance. Stronger scale values lead to results which are more consistent to the input guidance.

[6] Songhua Liu, Tianwei Lin, Dongliang He, Fu Li, Ruifeng Deng, Xin Li, Errui Ding, and Hao Wang. Paint transformer: Feed forward neural painting with stroke prediction. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[7] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, 2008.

[8] Ivan Skorokhodov, Grigorii Sotnikov, and Mohamed Elhoseiny. Aligning latent and image spaces to connect the unconnectable. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[9] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 1985.

[10] Zhengxia Zou, Tianyang Shi, Shuang Qiu, Yi Yuan, and Zhenwei Shi. Stylized neural painting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.