# Ranni: Taming Text-to-Image Diffusion for Accurate Instruction Following

Supplementary Material

## A. Illustration of the Complete Workflow

In this section, we provide a complete example of the workflow, including painting and editing instructions. Fig. A1 illustrates the conversation process for requesting LLM to create and manipulate the semantic panel, with step-by-step instructions. Visualized internal results and images are included for better understanding. Based on the explicit design of the semantic panel, `Ranni` presents a fully-automatic pipeline for assigning and manipulating images using a conversational approach.

## B. Dataset Construction

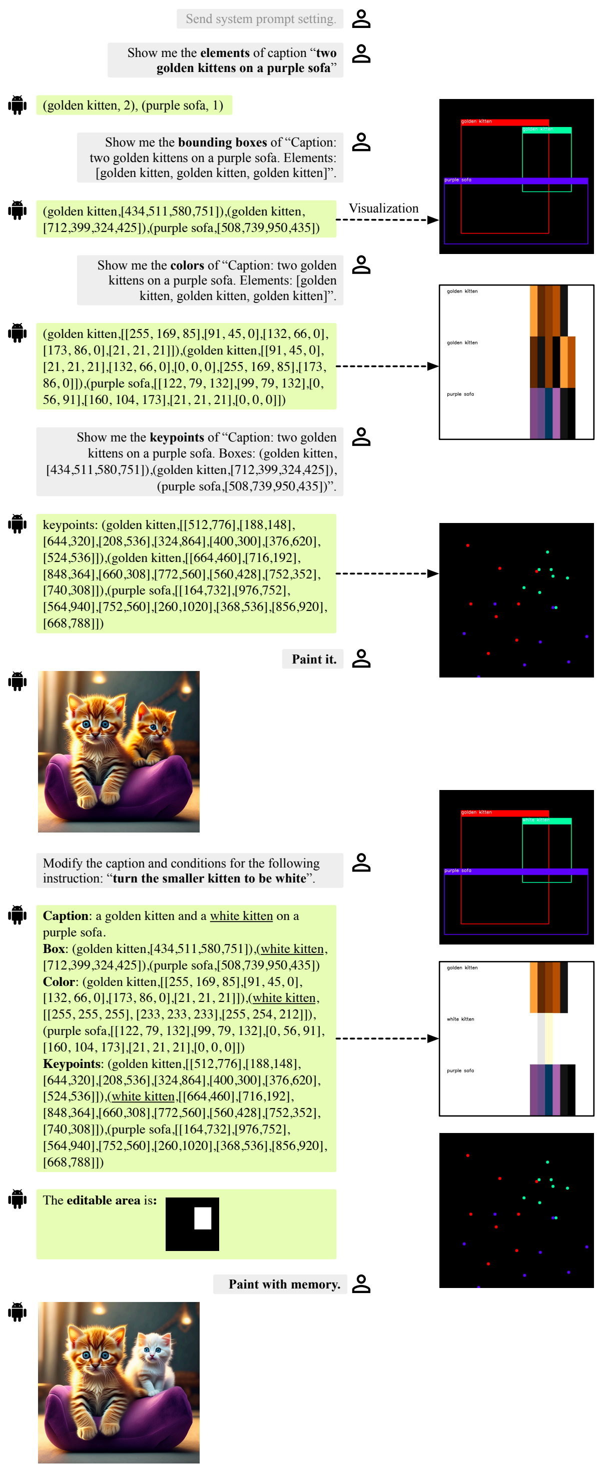
In this section, we present the details of our data preparation pipeline, including the attribute extraction and dataset augmentation process. Furthermore, we showcase visualizations of samples from various parts of the semantic panel dataset.

### B.1. Attribute Extraction

**Description and Box.** Given an image with a complete caption, we use Grounding DINO [2] to detect all the visible object boxes along with their corresponding descriptions in the caption. After the inference of Grounding DINO, we filter out redundant boxes that have general or meaningless descriptions, *e.g.* "an image", "objects", *etc.* We also remove boxes that have the same description as another box, with an IoU larger than 0.9.

**Colors.** For each object, we use SAM [1] mask to extract all its pixels. We first construct a color palette with CIELab [5] color space, which consists of 11 hue values, 5 saturation values and 5 light values. We calculate the color index of each pixel by searching for its nearest RGB value in the palette. Then we count the frequency of all color indices for the object, filter out indices with frequencies smaller than 5%, and pick the top-6 indices as the color representation. The final output of color attribute is a set of high frequent indices.

**Keypoints.** We use the farthest point sampling (FPS) algorithm [3] to sample keypoints within the SAM mask. Specifically, we define the candidate set as all the pixel coordinates $(x, y)$ inside the mask area. We start by randomly selecting a point and adding it to the sampled set. Then, for each iteration, we choose a point from the candidate set that has the farthest distance towards the sampled set. The distance from a point to a point set is determined by its distance to the closest point in that set. We stop sampling when the size of the sampled set exceeds 8 or



Send system prompt setting.

Show me the **elements** of caption "**two golden kittens on a purple sofa**"

(golden kitten, 2), (purple sofa, 1)

Show me the **bounding boxes** of "Caption: two golden kittens on a purple sofa. Elements: [golden kitten, golden kitten, golden kitten]".

(golden kitten,[434,511,580,751]),(golden kitten,[712,399,324,425]),(purple sofa,[508,739,950,435])

Show me the **colors** of "Caption: two golden kittens on a purple sofa. Elements: [golden kitten, golden kitten, golden kitten]".

(golden kitten,[[255, 169, 85],[91, 45, 0],[132, 66, 0],[173, 86, 0],[21, 21, 21]]),(golden kitten,[[91, 45, 0],[21, 21, 21],[132, 66, 0],[0, 0, 0],[255, 169, 85],[173, 86, 0]]),(purple sofa,[[122, 79, 132],[99, 79, 132],[0, 56, 91],[160, 104, 173],[21, 21, 21],[0, 0, 0]])

Show me the **keypoints** of "Caption: two golden kittens on a purple sofa. Boxes: (golden kitten,[434,511,580,751]),(golden kitten,[712,399,324,425]),(purple sofa,[508,739,950,435])".

keypoints: (golden kitten,[[512,776],[188,148],[644,320],[208,536],[324,864],[400,300],[376,620],[524,536]]),(golden kitten,[[664,460],[716,192],[848,364],[660,308],[772,560],[560,428],[752,352],[740,308]]),(purple sofa,[[164,732],[976,752],[564,940],[752,560],[260,1020],[368,536],[856,920],[668,788]])

**Paint it.**

Modify the caption and conditions for the following instruction: "**turn the smaller kitten to be white**".

**Caption**: a golden kitten and a white kitten on a purple sofa.
**Box**: (golden kitten,[434,511,580,751]),(white kitten,[712,399,324,425]),(purple sofa,[508,739,950,435])
**Color**: (golden kitten,[[255, 169, 85],[91, 45, 0],[132, 66, 0],[173, 86, 0],[21, 21, 21]]),(white kitten,[[255, 255, 255], [233, 233, 233],[255, 254, 212]]),(purple sofa,[[122, 79, 132],[99, 79, 132],[0, 56, 91],[160, 104, 173],[21, 21, 21],[0, 0, 0]])
**Keypoints**: (golden kitten,[[512,776],[188,148],[644,320],[208,536],[324,864],[400,300],[376,620],[524,536]]),(white kitten,[[664,460],[716,192],[848,364],[660,308],[772,560],[560,428],[752,352],[740,308]]),(purple sofa,[[164,732],[976,752],[564,940],[752,560],[260,1020],[368,536],[856,920],[668,788]])

The **editable area** is:

**Paint with memory.**

Figure A1. Examples of **a complete generation process** by `Ranni` with painting and editing instructions.

when the farthest distance is smaller than $0.1$ (the distance is normalized to the range $[0, 1]$).

In Fig. A2, we show visualizations of samples with all extracted attributes. Since all the attributes in semantic panel can be automatically extracted for existing image-text pairs, we can easily scale up the dataset and enable efficient training of `Ranni`.

## B.2. Dataset Augmentation

**Generate Synthesised Captions.** We use Llava [8] to generate captions for dataset augmentation. Llava is a visual question answering model that provides answers to questions based on the content within a given image. First, we ask it to pick out images with only one object by asking: *"Is there only one element or object in the image?"* Images with the answer *"No"* are kept because their raw captions usually overlook the details of some objects. Next, we request captions for these images with a limited length: "Analyze the image in less than twenty words, focusing on all objects in this image."

**Generate Pseudo Data.** The pseudo samples are generated by creating random prompts and arranging random visual elements for them according to specific rules. Firstly, we generate random prompts from a pool of diverse objects, and assign varying colors and numbers to each prompt. In cases involving spatial relationships, we randomly specify the relative positions of the objects. For the spatial arrangement of their bounding boxes, we create a large set of prompts and randomly assign positions. We then select appropriate samples based on the criterion of maximizing the separation of elements. This effectively prevents the issue of object concentration in pseudo data. The spatial arrangement is specifically designed for prompts with spatial relationships, such as *"on the left of"*.

## C. Implementation Details of Text-to-Panel

In this section, we show the details of LLM-based text-to-panel generation in `Ranni`. This process is conducted step-by-step for different attributes in panel. For all the attribute generation, we carefully search for a system prompt to leverage the zero-shot ability of LLM. All the system prompt templates are shown in Fig. A3.

### C.1. Description Generation

The description generation is the first step, which finds out all the elements to be appeared in the image. The task is a pure language-based problem, without requiring knowledge on visual space. Therefore, we directly leverage the zero-shot ability of LLM for this task. As the system prompt shown in Fig. A3, we define a specific output format on this task. Instead of a raw set of element descriptions, we request LLM to generate each unique element with its number, *e.g., "(cat, 3)"*. We empirically observe that such a strategy results in better performance of description prediction, especially for objects with larger number of amount. Furthermore, we also request the LLM to ignore style descriptions and all invisible objects. We find it works well to ignore the *unwanted* objects, such as *"a sky without cloud"*.

### C.2. Box Generation

It is more challenging to generate bounding boxes of predicted elements in the above process. The region information of bounding boxes gets closer to the image modality, requiring more knowledge on spatial distribution. First, we design the system prompt to teach the LLM understanding the coordinate system of image, *i.e.* the x and y axis, with values increasing from left-to-right and top-to-bottom, respectively. For the output format of bounding boxes, we find it useful to define it as $[x_c, y_c, w, h]$, where $(x_c, y_c)$ is the center point of the box, and $(w, h)$ is the width and height of box. Different from the most commonly used $[x_1, y_1, x_2, y_2]$ indicating the top-left and bottom-right of the box, our used format is more friendly to LLM. The size of box is fixed when moving to different position, which helps LLM to learn the relationship between object description and its size.

### C.3. Color Generation

We define the color representation as discrete indices in a 156-colored palette. Such a discrete representation is necessary to relax the range of output. In practical, we test different strategies for color prediction: (1) Use the name of each color in the palette. It can relax the color prediction as a easier task of language model, but restrict the size of palette for accurate representation. (2) Use the color indices, and predict the index list. This strategy is hard to learn for LLM, without any knowledge on the color indices and their relationship. (3) Use the color indices, and predict the RGB value list. This strategy helps LLM to understand the colors and relationships. We choose the final strategy in our method.

### C.4. Keypoints Generation

The keypoint generation is the most difficult task of LLM in `Ranni`. We find that it is hard to prompt LLM with a good initial prediction, by carefully setting system prompt. Thus, we focus on helping LLM to output in a correct format, and learn the ability of keypoint prediction in the fine-tune stage. As shown in Fig. A3, we also provide the predicted box of each object, which restrict the distribution of keypoints to be inside the box.

On an NVIDIA A100 GPU, the averaged runtime is $6.75 \pm 1.65$s for text-to-panel with Llama2-13B.

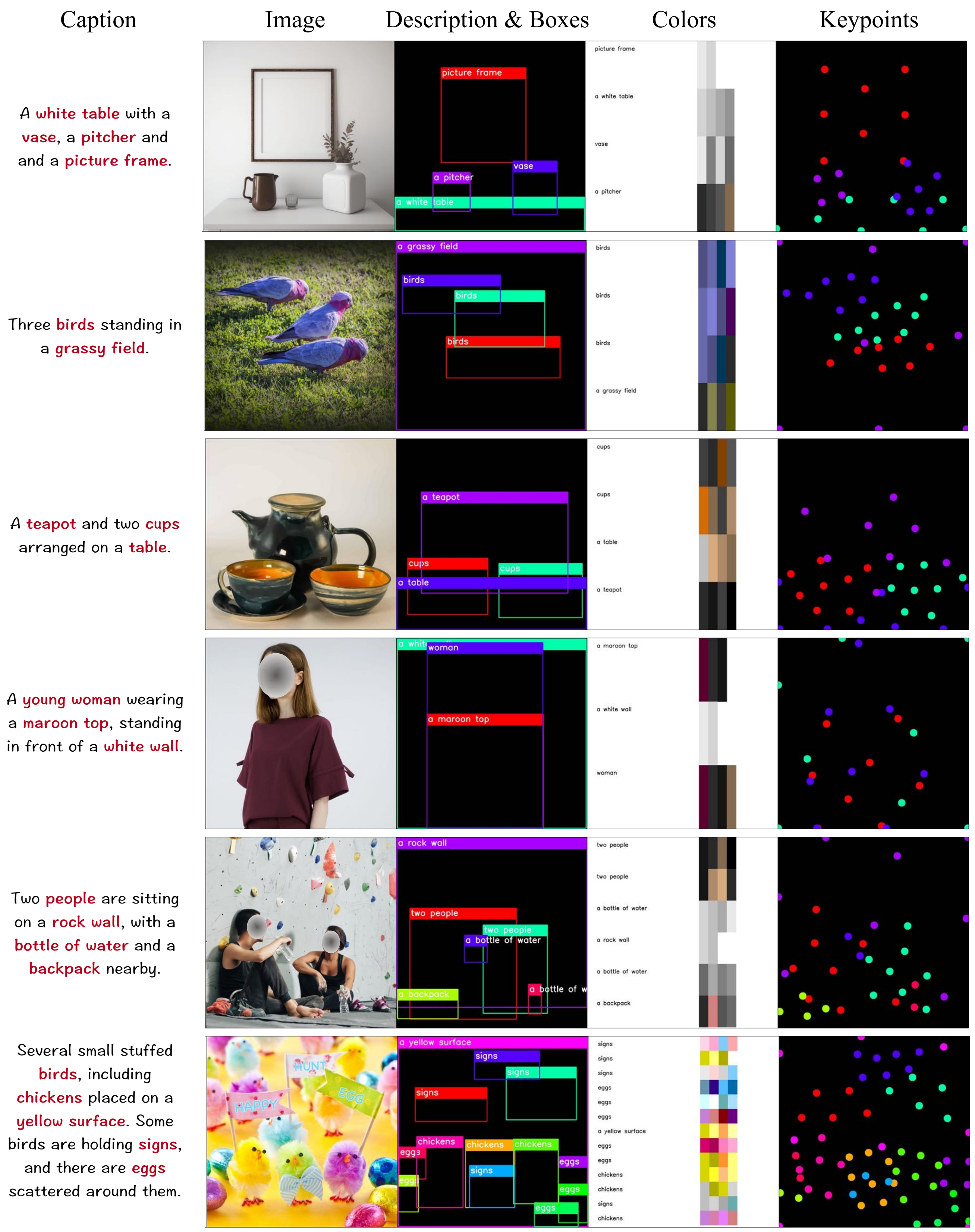

Caption
Image
Description & Boxes
Colors
Keypoints
A white table with a vase, a pitcher and and a picture frame.
picture frame
vase
a pitcher
a white table
Three birds standing in a grassy field.
a grassy field
birds
Two people are sitting on a rock wall, with a bottle of water and a backpack nearby.
A teapot and two cups arranged on a table.
a teapot
cups
a table
A young woman wearing a maroon top, standing in front of a white wall.
woman
a maroon top
a white wall
a rock wall
two people
a bottle of water
a backpack
Several small stuffed birds, including chickens placed on a yellow surface. Some birds are holding signs, and there are eggs scattered around them.
a yellow surface
signs
chickens
eggs
HAPPY
HUNT
EGG

Figure A2. **Visualization of samples** in the semantic panel dataset, with all the extracted attributes based on the orignal text-image pair.

**Description Generation**

I will provide you a caption of image, please imagine the image and generate text description of all elements that should be contained in the image. Also show the number of each element. Only generate noun phrases indicating visible objects in the image. Include their description words, e.g. a white cat. For example:

Caption: Two dogs and three cats playing on the grass, 4K image, best quality
Elements: (dog, 2), (cat, 3), (grass, 1)

Caption: Draw an image of a basket of green apples on the wooden table, in style of oil painting
Elements: (green apple, 6), (wooden table, 1)

Now show me the elements of caption "{}" in the above format. Answer shortly. Directly answer the elements. Do not repeat the caption.

---

**Box Generation**

I will provide you a caption of an image and all elements contained in it. Your task is to imagine the image and generate the bounding boxes for the provided element. The image is 1024 in width and 1024 in height, with a x-axis from left to right, and a y-axis from top to bottom. Then coordinate [x,y] is [0,0] for top-left, [1024,0] for top-right, [0,1024] for bottom-left and [1024,1024] for bottom-right. Each bouding box should be in the format of (element name, [x coordinate of element center, y coordinate of element center, width of element, height of element]).
1. For the coordinate, elements on left have smaller x, while elements on top have smaller y. Refer the caption and relations among elements for reasonable positions.
2. For the width and height, refer to the element description to generate reasonable size. Also refer to the element position and image size to avoid overlap with image boundary.
For example:

Caption: A white cat on the right of a black dog playing on the grass
Elements: [a white cat, a black dog, the grass]
Boxes:[(a white cat, [710,558,414,477]), (a black dog, [287,462,390,691]), (the grass,[512,731,1024,586])]

Caption: Two red apples lie on a green plate
Elements: [a red apple, a red apple, a green plate]
Boxes:[(a red apple, [403,668,300,300]), (a red apple, [630,628,300,300]), (a green plate, [506,816,738,72])]

Now show me the boxes of "Caption: {}, Elements: {}". Answer shortly. Directly answer the boxes. Do not repeat the caption and elements.

---

**Color Generation**

I will provide you a caption of an image and all elements contained in it. Your task is to imagine the image and generate the main colors for the provided element. For each element, generate a list of at most 6 colors in format of [R,G,B]. For example:
Caption: A white cat on the right of a black dog playing on the grass
Elements: [a white cat, a black dog, the grass]
Colors: [(a white cat, [[255,255,255], [128,128,128]]), (a black dog, [[0,0,0], [169,169,169]]), (the grass,[[0,255,0], [128,128,0]])]

Now show me the colors of "Caption: {}, Elements: {}"

---

**Keypoints Generation**

I will provide you a caption of an image and all elements in it with bounding boxes. Your task is to imagine the image and generate the keypoints for the provided element. The image is 1024 in width and 1024 in height, with a x-axis from left to right, and a y-axis from top to bottom. Each bouding box is presented in the format of (element name, [x1, y1, x2, y2]), where [x1,y1] is the top-left and [x2, y2] is the bottom-right of the box. For each element, generate a list of at most 8 keypoints' coordinates like [[x,y], [x,y], ...]. Noted that all the keypoints should be inside of the corresponding element bounding box.

Now show me the keypoints of "Caption: {}, Boxes: {}"

---

**Editing**

I will provide you caption of an image and all bounding boxes in it. Your task is to edit the caption and bounding box (bbox) following my instructions. The image is 1024 in width and 1024 in height, with a x-axis from left to right, and a y-axis from top to bottom. Then coordinate [x,y] is [0,0] for top-left, [1024,0] for top-right, [0,1024] for bottom-left and [1024,1024] for bottom-right. Each bouding box should be in the format of (element name, [x coordinate of element center, y coordinate of element center, width of element, height of element]). I will ask you to add, delete, move, resize or change labels of elements. For adding element, append its bbox to existing boxes. For deleting element, find the specified one and remove it. For moving or resizing element, find the element's bbox and change its position or size. For changing element, change its element name. Noted elements on top have smaller y coordinate, and elements on left have smaller x coordinate.

The caption is “{}”, and the original object bbox information is “{}”.
Please adjust the caption and bbox for the instruction “{}”

Figure A3. **System prompts** for all the LLM-based tasks in `Ranni`. We leave “{}” in red for positions of depended conditions.

## D. Implementation Details of Panel-to-Image

In this section, we present the details of panel-to-image, a controllable image synthesis process based on predicted semantic panel. As we have mentioned, the controlling strategy for panel-to-image contains two parts, *i.e.* panel conditioning and attention restriction.

### D.1. Panel Conditioning

We first encode each attribute in the semantic panel into a comprehensive condition:

(1) For the text description, we get its CLIP text embedding [4] individually. We use the same CLIP weights as the main text-to-image model, but take the global sentence embedding instead of word embedding.

(2) For the bounding box, we draw a binary mask in the same shape of image latent. The coordinates of box are resized into the latent space, *i.e.,* $1/8$ of original value. Then we set all positions inside the box as value $1$ in the mask.

(3) For the colors, we have got a list of color indices. Then we set a binary vector in size of 156 (same as the palette size), and set 1 for the given color indices. The vector is then mapped to a feature vector with learnable linear projection.

(4) For the keypoints, we draw a binary mask same as box. For each point, we draw a circle with radius of 6, and set 1 inside the circle.

All the conditions are then mapped by learnable convolutions into the same channel. To merge the the conditions in different shape, we further repeat the 1D conditions (text and color) into the same shape of image, and multiply it with the binary mask of bounding box. Finally, we sum all conditions up, and average it over all objects.

Except for the training strategy in main paper, we also study the strategy based on ControlNet [7], which digests the condition with a copied bypass encoder. Since the final condition map is a feature map in same shape of image latent, we can easily train a ControlNet for this task. In practical, we find comparable performance for the two different strategy, and choose the previous one for efficiency. To accompany `Ranni` with existing models, *e.g.,* Stable Diffusion [6], it would be better to choose ControlNet as a plug-in module for the base model.

### D.2. Attention Restriction

In the above process, we use sentence embedding for semantic description. When the phrase of description comes longer, *e.g.* "a red metal apple", the generated image may loss some semantics. To address this issue, we further introduce a controlling strategy for better alignment. It works via rectifying the cross-attention layer of diffusion model.

The present diffusion model involves cross-attention between $N_I$ image patches and $N_T$ words of input prompt. Given the generated semantic panel, we have already known the exact correspondence between patches and words. Then our rectifying is to restrict the attention map to follow such correspondence. We generate a attention mask $M \in \mathbb{R}^{N_I \times N_T}$ for such rectifying. For each object, we first locate the index range $[i_s, i_e]$ as its text description in the whole prompting text, then locate $[j_s, j_e]$ as the related image patches inside the bounding box. Then the attention mask is set as $M[i_s : i_e, j_s, j_e] = 1$, otherwise 0. We apply attention mask for all the cross-attention layers in the diffusion model.

The cross-attention rectifying significantly improves the alignment of semantics. But it can not restrict the object to be located inside the box. Thus, we combine it with the training-based panel conditioning together, and achieve better controlled generation.

On an NVIDIA A100 GPU, the averaged runtime is $19.28 \pm 0.19$s for panel-to-image with our pre-trained 3B UNet and 50 diffusion steps.

## E. Failure Case Analysis

We show failure cases in Fig. A4, including semantic confusion, wrong spatial relationship and missing objects. The text-to-panel stage might generate results with wrong or highly-overlapped positions in such cases, leading to failed images in final generation. As a preview, the users can refresh or adjust the elements before generating images for remedy. It is also observed that the panel-to-image generation is not strictly controlled by the panel, and shows some robustness to rectify improper layout from the first stage.

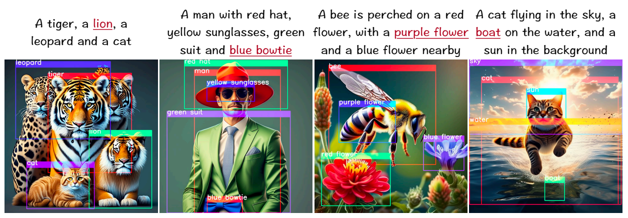

A tiger, a lion, a leopard and a cat
A man with red hat, yellow sunglasses, green suit and blue bowtie
A bee is perched on a red flower, with a purple flower and a blue flower nearby
A cat flying in the sky, a boat on the water, and a sun in the background

Figure A4. Failure cases of `Ranni`.

## F. More Results

In this section, we show more results of `Ranni`. Fig. A5 shows editing samples with more detailed re-colorization by setting the color attribute. Fig. A6 shows shape-editing samples by re-arranging the keypoints, where we set a main direction to reshape the object. Fig. A7 shows more editing results of the six unit operations. Fig. A8 shows more samples generated by `Ranni`.

## References

[1] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloé Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross B. Girshick. Segment anything. *ArXiv*, abs/2304.02643, 2023. 1

[2] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding DINO: marrying DINO with grounded pre-training for open-set object detection. *ArXiv*, abs/2303.05499, 2023. 1

[3] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Adv. Neural Inform. Process. Syst.*, pages 5099–5108, 2017. 1

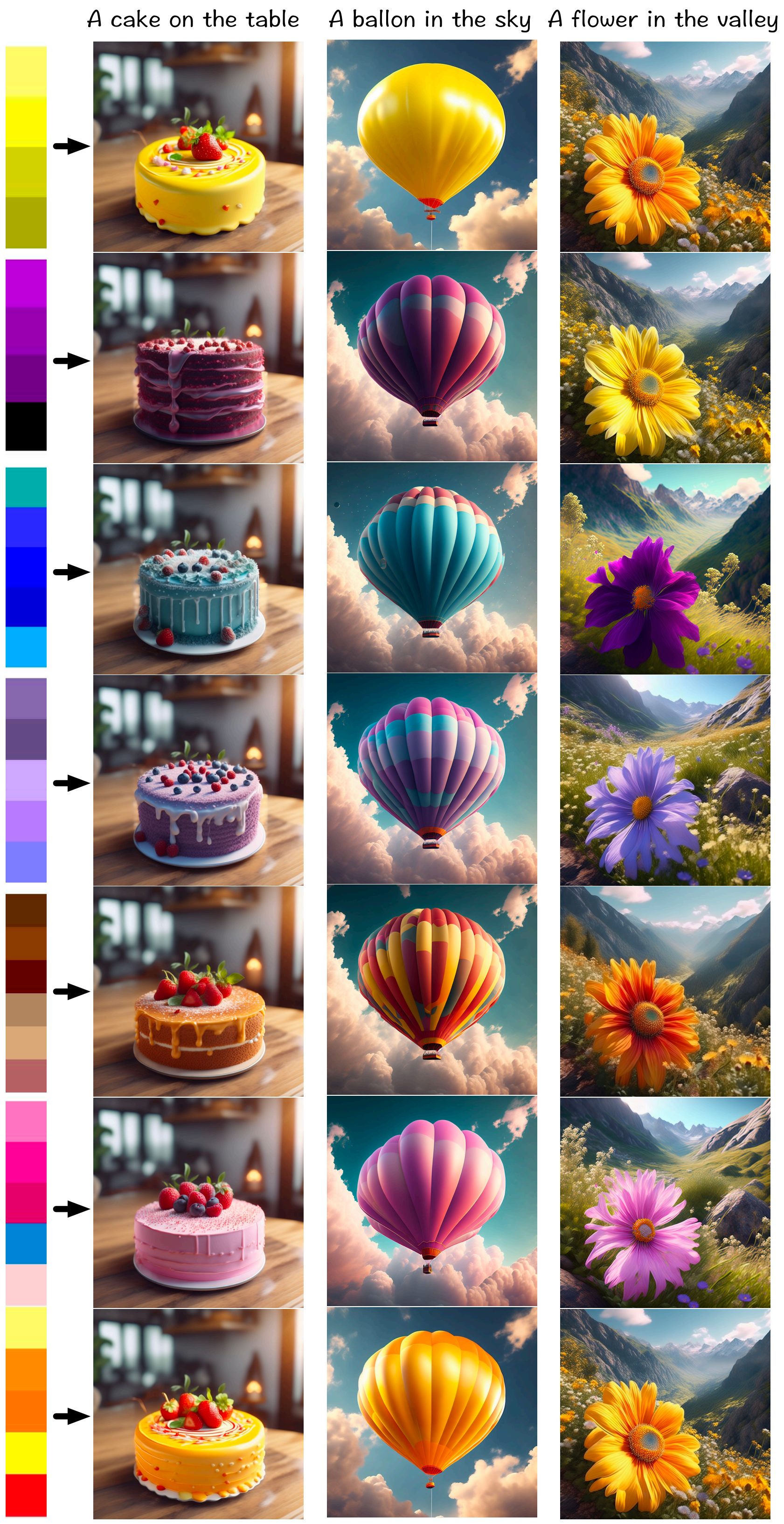


Figure A5. Examples of **color editing**.

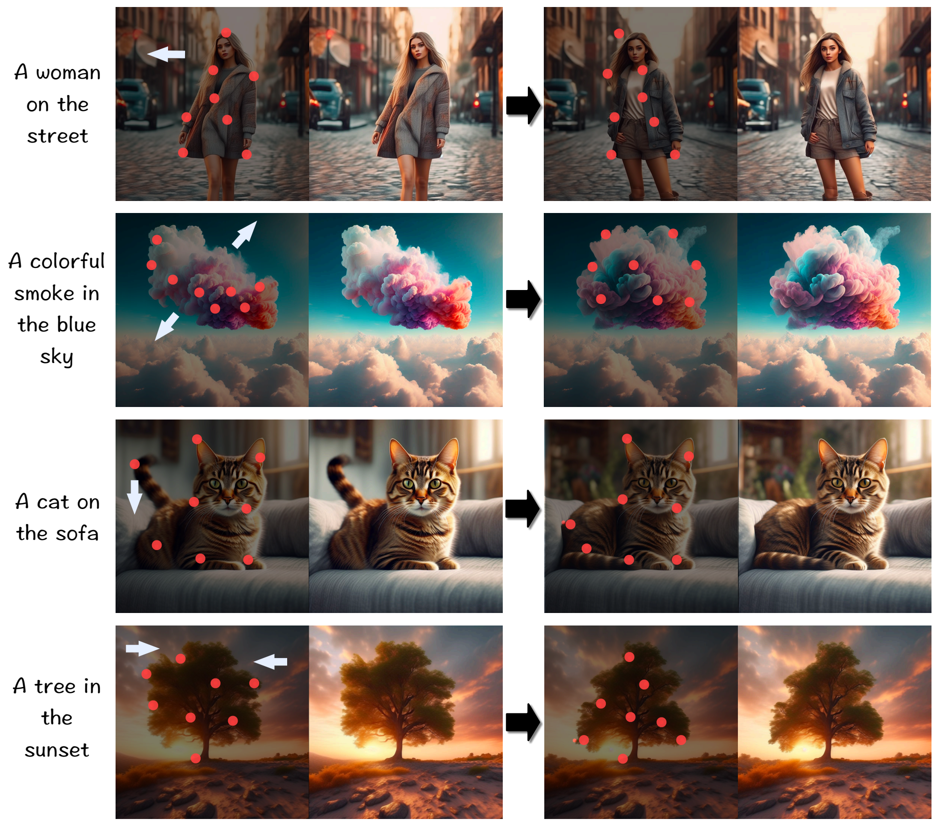


Figure A6. Examples of **shape editing**. The blue arrows indicate the direction of keypoints moving.

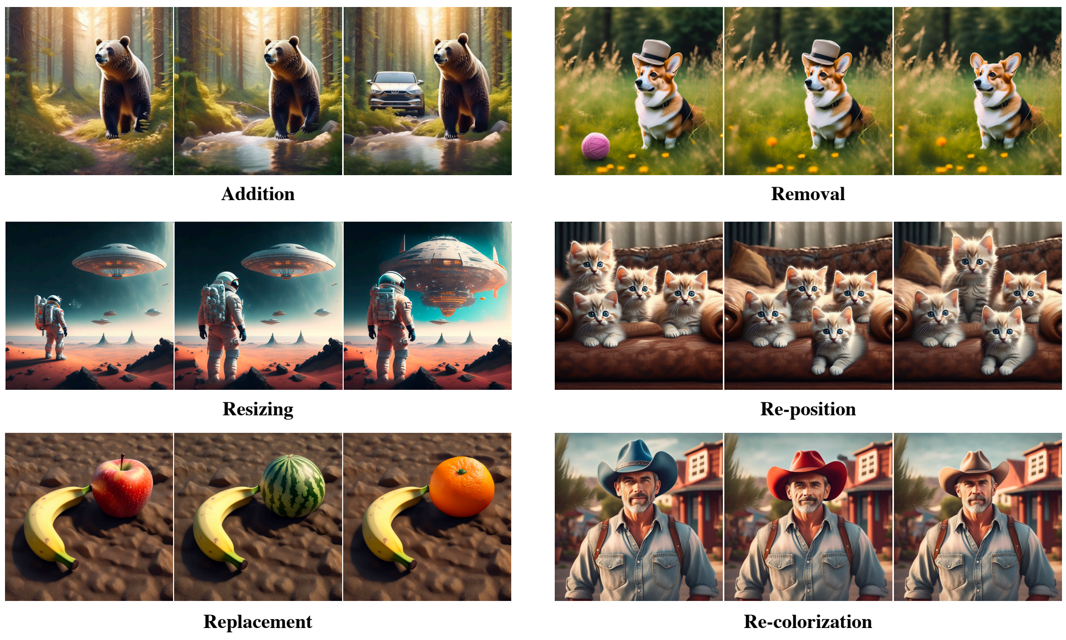
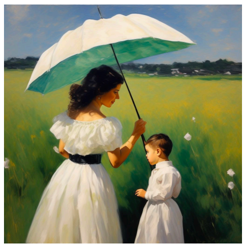
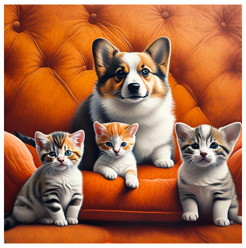
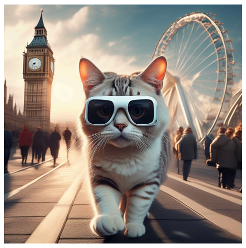


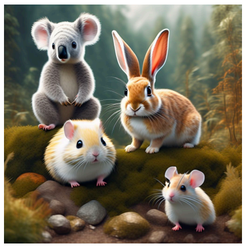
Figure A7. More editing cases of unit operations.

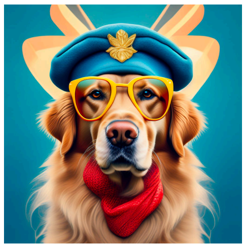
[4] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision.

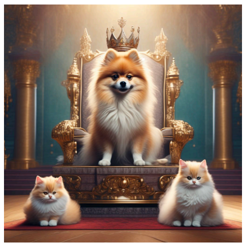

Monet's oil painting of a woman in white dress putting up a green umbrella, with her son nearby. They are standing on the grass under blue sky.

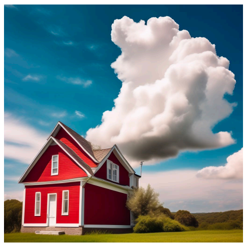

Three kittens and a Corgi on the orange sofa.

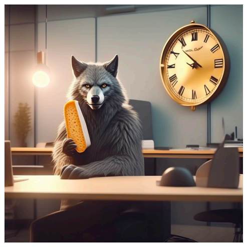

A giant cat wearing a white VR glasses, walking in London street. In the background, the Big Ben is on left while the London's eye is on right.

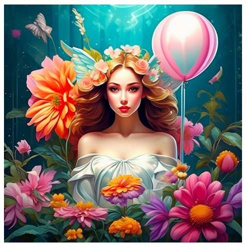

A rabbit, a rat, a hamster, and a koala.

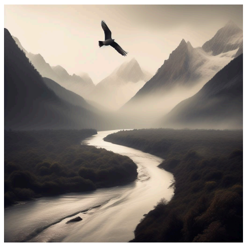

Golden retriever wearing a blue beret, a yellow sunglasses and a red scarf.

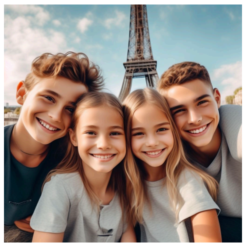

A Pomeranian wearing a crown sat on the king's throne, and two cats stood on either side of the throne.

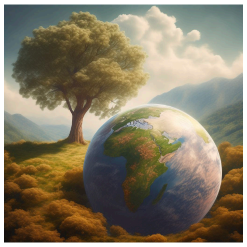

The white cloud on right of a red house.

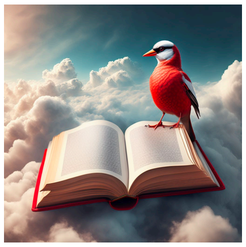

Wolfman sitting in an office cubicle, holding a bread in front of the desk. There is a golden clock hang on the wall behind.

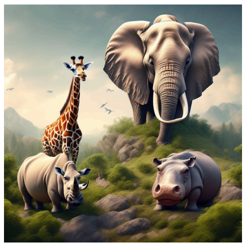

The flower is on left of the angel and the ballon is on right.


A winding river on the bottom, a flying bird on the top, In the distance were black and white peaks shrouded in fog.


Selfie of 2 boys and 2 girls with the Eiffle tower in the background.


The earth is on bottom of the tree.


A red book flying on the sky. A bird stands on the right, with a lot of clouds around.


An elephant, a giraffe, a rhinon, and a hippo.

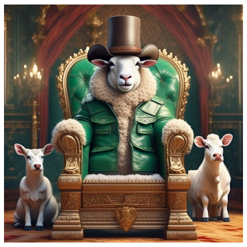

A sheep wearing a cowboy hat and green leather jacket sat on the king's throne. A cow and a deer stood on either side of the throne.

Figure A8. More samples generated by `Ranni`.

In *Int. Conf. Mach. Learn.*, pages 8748–8763, 2021. 4

[5] Sergeyk. Rayleigh: Search image collections by multiple color palettes or by image color similarity., 2016. 1

[6] stability.ai. Stable Diffusion 2.0 Release, 2022. 5

[7] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Int. Conf. Comput. Vis.*, 2023. 5

[8] Yanzhe Zhang, Ruiyi Zhang, Jiuxiang Gu, Yufan Zhou, Nedim Lipka, Diyi Yang, and Tong Sun. Llavar: Enhanced visual instruction tuning for text-rich image understanding. *ArXiv*, abs/2306.17107, 2023. 2