Self-correcting LLM-controlled Diffusion Models

Supplementary Material

A. More Discussions on Image Editing Capabilities

In Fig. 6 of the main paper, we showcased an example of object replacement contrasting our SLD method with previous approaches. In this section, we provide more visual examples in Fig. 2 that highlight the differences between SLD and prior diffusion-based image editing methods. While InstructPix2Pix [1] is confined to pixel-aligned changes (*e.g.*, style changes), DiffEdit [2] often fails with precise object-level edits. Our SLD framework excels in these detailed, fine-grained editing tasks.

Furthermore, our exploration reveals that even a modest set of four primitive latent operations within the SLD framework is remarkably effective at addressing a broad spectrum of editing applications. Beyond managing static objects, as depicted in Fig. 1, SLD adeptly processes actions (verbs) linked to objects through an attribute change operator. This operation can be easily extended to handle global style variations. Additionally, the deletion operator provides a means to eliminate undesirable/harmful content from images. These above examples further illustrate the wide-ranging utility of our method.

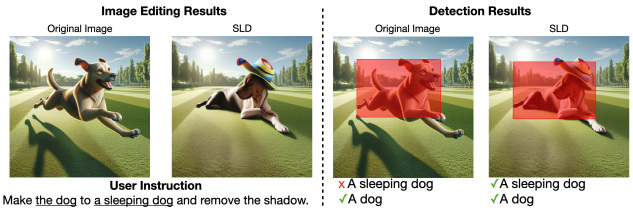

Figure 1. In addition to static objects with attributes, SLD can also perform image editing on objects with action.

B. Comparison between LMMs and Detector-LLM Combinations

In our main paper, we introduce a self-correction framework that utilizes both LLMs and object detectors for image assessment, followed by the provision of correction suggestions. With the rapid evolution of Large Multimodal Models (LMMs) such as GPT-4V [9] and LLaVA [4, 5], we have investigated the feasibility of using an LMM to conduct image assessment. In this setup, the LMM evaluates images generated by the open-loop generator alongside user prompts, aiming to provide precise, object-level editing recommendations.

However, GPT-4V's analysis of the "princess and dwarfs" image from DALL-E 3 (refer to Fig. 1 in our paper) reveals inaccuracies, as the model miscounts the characters, identifying seven dwarfs rather than the actual five, and struggles to define precise bounding box coordinates. In contrast, as highlighted in Fig. 3, the latest generation open-vocabulary detector, OWL-ViT v2, demonstrates remarkable proficiency in detecting minute objects (seagulls in the sky), which is vital for the correction process. This limitation underscores our current approach, which combines detectors with LLMs for more accurate assessment and editing suggestions. Nevertheless, the potential of integrating advanced LMMs for streamlined image generation and editing remains a compelling and promising direction for future research and development.

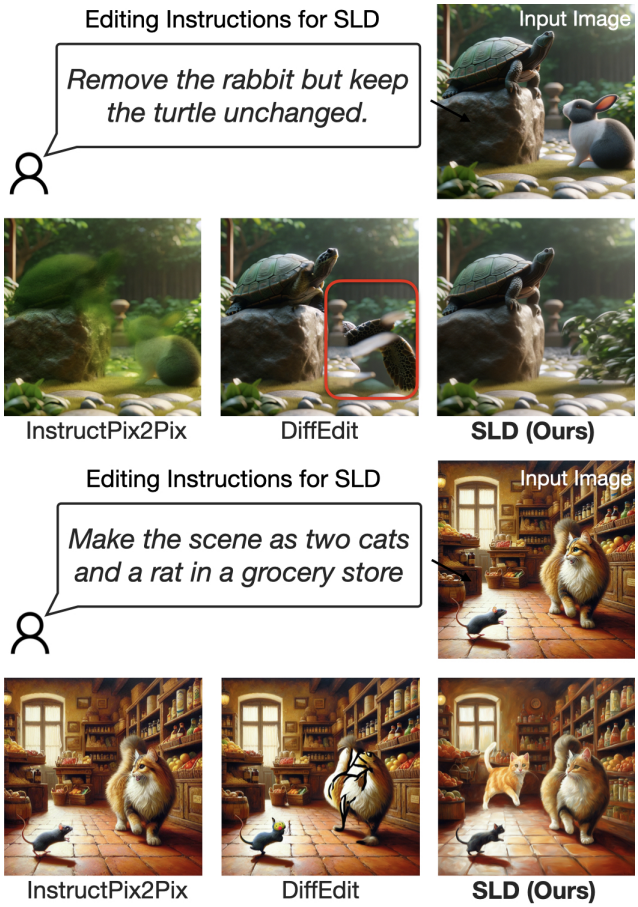

Figure 2. We demonstrate that current diffusion-based image editing methods, such as InstructPix2Pix and DiffEdit, face challenges in fundamental operations like object deletion (top example) and addition (bottom example). In contrast, our proposed SLD pipeline can easily handle these tasks. As highlighted in the red box, despite DiffEdit's ability to identify the object for removal, it falls short of generating a plausible background.

Method	Accuracy				
	Negation	Numeracy	Attribute	Spatial	Average
DALL-E 3 [8]	25%	38%	74%	71%	52.0%
+ 1-round SLD (OWL-ViT v1)	50%	51%	71%	82%	63.5% **(+ 11.5)**
+ 1-round SLD (OWL-ViT v2)	90%	61%	**80%**	83%	78.5% **(+ 26.5)**
LMD+ [3]	**100%**	82%	49%	86%	79.3%
+ 1-round SLD (OWL-ViT v1)	**100%**	85%	59%	89%	83.3% **(+ 4.0)**
+ 1-round SLD (OWL-ViT v2)	**100%**	**98%**	63%	**92%**	**88.3%** **(+ 9.0)**

Table 1. Our method can be applied to various image generation methods and improves the generation accuracy by a large margin.

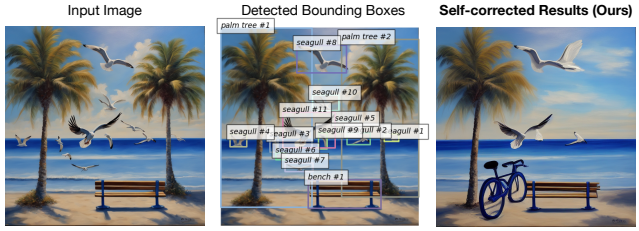

Figure 3. Leveraging the advanced localization abilities of OWL-ViT v2 open-vocabulary detectors, we accurately identify all seagulls in the image, enabling selective removal to align with the user's prompt.

C. Comprehensive Image Generation Results

In the main paper, we demonstrate the significant self-correction performance gain achieved by integrating open-vocabulary detectors, the SAM segmentation module, and LLMs into our SLD pipeline. To illustrate the adaptability of our SLD model, we explore various combinations with external models in this appendix, showing its robustness in diverse settings. Additionally, we delve into the efficacy of LLMs in layout correction, further evidencing the multi-faceted strengths of our approach.

Tab. 1 presents a comparison between the results obtained using OWL-ViT v1 [7] and OWL-ViT v2 [6] detectors within our framework. The results indicate that SLD consistently enhances overall accuracy compared to the baseline text-to-image generators, irrespective of the detector used. The marginal reduction in performance gains when substituting the v2 detector can be attributed to two main factors: **1)** the relatively inferior detection capabilities of the alternate detector employed in the SLD, and **2)** the variations in object recognition between the two detectors. This situation parallels real-world experiences, where individual perceptions and recognition of objects or attributes can vary significantly. For instance, a bowl perceived as distinctly blue by one might be seen as less blue by another. These perceptual variances contribute to the marginally lower attribute binding scores of SLD compared to the original DALL-E 3 results. Despite these discrepancies, the overall accuracy of our method, especially in areas such as numeracy and spatial reasoning, confirms the effectiveness of SLD, *regardless of the specific detector used in our self-correction pipeline*.

Tab. 2 ablated the choices of different LLMs and segmentation refinement modules in our pipeline. The result shows that SLD is robust to different model choices. In Tab. 3, we assess the LLM controller's success rate in correcting images from baseline LMD+ to see if the LLM has the superpower of outputting correct instruction and even bounding box coordinates purely through in-context learning. We use the same setting as Tab. 1 in the main paper, except that we evaluate bounding boxes provided to/corrected by the LLM, without the actual editing. While the baseline has a 20.7% error rate, the corrected boxes only have 1.0% error rate, indicating that the LLM corrected almost all incorrect bounding boxes in the benchmark, which suggests that accuracies of latent operations rather than LLM's abilities are the bottleneck.

Method | Accuracy →	Negation	Numeracy	Attribute	Spatial	Average
SLD (**default**, GPT-4, SAM-base)	100%	98%	63%	92%	**88.3%**
SLD (GPT-4→GPT-3.5)	100%	95%	61%	89%	86.3%
SLD (SAM-base→SAM-huge)	100%	96%	62%	95%	**88.3%**

Table 2. Ablation studies on different LLMs and SAM modules in the SLD pipeline.

Method	**Accuracy (success rate)**: evaluating the input/output bounding boxes of our LLM self-correction controller				
	Negation	Numeracy	Attribute	Spatial	Average
Before self-correction	**100%**	82%	49%	86%	79.3%
After self-correction	**100%**	**100%**	**98%**	**98%**	**99.0%** **(+19.7)**

Table 3. The LLM controller prompted with several in-context examples can correct most incorrect bounding boxes.

D. Our Prompts and Instructions to the LLM

As outlined in the main paper, we leverage two LLMs to steer the self-correction process: we employ one LLM parser to identify key objects from user prompts and another LLM controller to propose bounding box adjustments. The specific prompts for both the parser and the controller are detailed in Tab. 4 and Tab. 5, respectively. We also provide in-context examples for the LLM controller, tailored for self-correcting generation and image editing, in Tab. 6 and Tab. 7, respectively.

In crafting our LLM prompts, we emphasized clarity in defining the roles and guidelines for the LLM, drawing inspiration from prior work [11]. Notably, we discovered that GPT-4 possesses the capability to manipulate bounding box coordinates, a task that involves mathematical reasoning. We achieved improved results by guiding the model to employ chain-of-thought reasoning [10], where the model explicates its reasoning process during generation. This approach yielded more accurate suggestions compared to instances where the model's reasoning was not explicitly stated.

```
# Your Role: Excellent Parser

## Objective: Analyze scene descriptions to identify objects and their attributes.

## Process Steps
1. Read the user prompt (scene description).
2. Identify all objects mentioned with quantities.
3. Extract attributes of each object (color, size, material, etc.).
4. If the description mentions objects that shouldn't be in the image, take note at the negation part.
5. Explain your understanding (reasoning) and then format your result (answer / negation) as shown in the examples.
6. Importance of Extracting Attributes: Attributes provide specific details about the objects. This helps differentiate between similar objects and gives a clearer understanding of the scene.

## Examples

- Example 1
    User prompt: A brown horse is beneath a black dog. Another orange cat is beneath a brown horse.
    Reasoning: The description talks about three objects: a brown horse, a black dog, and an orange cat. We report the color attribute thoroughly. No specified negation terms.
    Objects: [('horse', ['brown']), ('dog', ['black']), ('cat', ['orange'])]
    Negation:

- Example 2
    User prompt: There's a white car and a yellow airplane in a garage. They're in front of two dogs and behind a cat. The car is small. Another yellow car is outside the garage.
    Reasoning: The scene has two cars, one airplane, two dogs, and a cat. The car and airplane have colors. The first car also has a size. No specified negation terms.
    Objects: [('car', ['white and small', 'yellow']), ('airplane', ['yellow']), ('dog', [None, None]), ('cat', [None])]
    Negation:

- Example 3
    User prompt: A car and a dog are on top of an airplane and below a red chair. There's another dog sitting on the mentioned chair.
    Reasoning: Four objects are described: one car, airplane, two dog, and a chair. The chair is red color. No specified negation terms.
    Objects: [('car', [None]), ('airplane', [None]), ('dog', [None, None]), ('chair', ['red'])]
    Negation:

- Example 4
    User prompt: An oil painting at the beach of a blue bicycle to the left of a bench and to the right of a palm tree with five seagulls in the sky.
    Reasoning: Here, there are five seagulls, one blue bicycle, one palm tree, and one bench. No specified negation terms.
    Objects: [('bicycle', ['blue']), ('palm tree', [None]), ('seagull', [None, None, None, None, None]), ('bench', [None])]
    Negation:

- Example 5
    User prompt: A realistic photo of a scene without backpacks.
    Reasoning: The description clearly states no backpacks, so this must be acknowledged. The user provides the negative prompt of backpacks.
    Objects: [('backpacks', [None])]
    Negation: backpacks

Your Current Task: Follow the steps closely and accurately identify objects based on the given prompt. Ensure adherence to the above output format.

User prompt: {the input user prompt}
Reasoning:
```

Table 4. Our full prompt for the LLM parser.

```
# Your Role: Expert Bounding Box Adjuster

## Objective: Manipulate bounding boxes in square images according to the user prompt while
    maintaining visual accuracy.

## Bounding Box Specifications and Manipulations
1. Image Coordinates: Define square images with top-left at [0, 0] and bottom-right at [1, 1].
2. Box Format: [Top-left x, Top-left y, Width, Height]
3. Operations: Include addition, deletion, repositioning, and attribute modification.

## Key Guidelines
1. Alignment: Follow the user's prompt, keeping the specified object count and attributes.
2. Boundary Adherence: Keep bounding box coordinates within [0, 1].
3. Minimal Modifications: Change bounding boxes only if they don't match the user's prompt.
4. Overlap Reduction: Minimize intersections in new boxes and remove the smallest, least overlapping
    objects.

## Process Steps
1. Interpret prompts: Read and understand the user's prompt.
2. Implement Changes: Review and adjust current bounding boxes to meet user specifications.
3. Explain Adjustments: Justify the reasons behind each alteration and ensure every adjustment abides
    by the key guidelines.
4. Output the Result: Present the reasoning first, followed by the updated objects section, which
    should include a list of bounding boxes in Python format.

## Examples

{In-context examples for self-correcting image generation or image editing}

Your Task: Carefully follow the provided guidelines and steps to adjust bounding boxes in accordance
    with the user's prompt. Ensure adherence to the above output format.

User prompt: {the input user prompt}
Current Objects: {a list of detected key objects}
Reasoning:
```

Table 5. Our full prompt for the LLM controller.

```
# Examples

- Example 1
    User prompt: A realistic image of landscape scene depicting a green car parking on the left of a blue truck, with a red air balloon and a bird in the sky
    Current Objects: [('green car #1', [0.027, 0.365, 0.275, 0.207]), ('blue truck #1', [0.350, 0.368, 0.272, 0.208]), ('red air balloon #1', [0.086, 0.010, 0.189, 0.176])]
    Reasoning: To add a bird in the sky as per the prompt, ensuring all coordinates and dimensions remain within [0, 1].
    Updated Objects: [('green car #1', [0.027, 0.365, 0.275, 0.207]), ('blue truck #1', [0.350, 0.369, 0.272, 0.208]), ('red air balloon #1', [0.086, 0.010, 0.189, 0.176]), ('bird #1', [0.385, 0.054, 0.186, 0.130])]

- Example 2
    User prompt: A realistic image of landscape scene depicting a green car parking on the right of a blue truck, with a red air balloon and a bird in the sky
    Current Output Objects: [('green car #1', [0.027, 0.365, 0.275, 0.207]), ('blue truck #1', [0.350, 0.369, 0.272, 0.208]), ('red air balloon #1', [0.086, 0.010, 0.189, 0.176])]
    Reasoning: The relative positions of the green car and blue truck do not match the prompt. Swap positions of the green car and blue truck to match the prompt, while keeping all coordinates and dimensions within [0, 1].
    Updated Objects:  [('green car #1', [0.350, 0.369, 0.275, 0.207]), ('blue truck #1', [0.027, 0.365, 0.272, 0.208]), ('red air balloon #1', [0.086, 0.010, 0.189, 0.176]), ('bird #1', [0.485, 0.054, 0.186, 0.130])]

- Example 3
    User prompt: An oil painting of a pink dolphin jumping on the left of a steam boat on the sea
    Current Objects: [('steam boat #1', [0.302, 0.293, 0.335, 0.194]), ('pink dolphin #1', [0.027, 0.324, 0.246, 0.160]), ('blue dolphin #1', [0.158, 0.454, 0.376, 0.290])]
    Reasoning: The prompt mentions only one dolphin, but two are present. Thus, remove one dolphin to match the prompt, ensuring all coordinates and dimensions stay within [0, 1].
    Updated Objects: [('steam boat #1', [0.302, 0.293, 0.335, 0.194]), ('pink dolphin #1', [0.027, 0.324, 0.246, 0.160])]

- Example 4
    User prompt: An oil painting of a pink dolphin jumping on the left of a steam boat on the sea
    Current Objects: [('steam boat #1', [0.302, 0.293, 0.335, 0.194]), ('dolphin #1', [0.027, 0.324, 0.246, 0.160])]
    Reasoning: The prompt specifies a pink dolphin, but there's only a generic one. The attribute needs to be changed.
    Updated Objects: [('steam boat #1', [0.302, 0.293, 0.335, 0.194]), ('pink dolphin #1', [0.027, 0.324, 0.246, 0.160])]

- Example 5
    User prompt: A realistic photo of a scene with a brown bowl on the right and a gray dog on the left
    Current Objects: [('gray dog #1', [0.186, 0.592, 0.449, 0.408]), ('brown bowl #1', [0.376, 0.194, 0.624, 0.502])]
    Reasoning: The leftmost coordinate (0.186) of the gray dog's bounding box is positioned to the left of the leftmost coordinate (0.376) of the brown bowl, while the rightmost coordinate (0.186 + 0.449) of the bounding box has not extended beyond the rightmost coordinate of the bowl. Thus, the image aligns with the user's prompt, requiring no further modifications.
    Updated Objects: [('gray dog #1', [0.186, 0.592, 0.449, 0.408]), ('brown bowl #1', [0.376, 0.194, 0.624, 0.502])]
```

Table 6. Self-correction in-context examples for the LLM controller.

```
## Examples

- Example 1
    User prompt: Move the green car to the right and make the blue truck larger in the image.
    Current Objects: [('green car #1', [0.027, 0.365, 0.275, 0.207]), ('blue truck #1', [0.350, 0.368, 0.272, 0.208])]
    Reasoning: To move the green car rightward, its x-coordinate needs to be increased from 0.027. The dimensions (height and width) of the blue truck must be enlarged. While adjusting bounding boxes, ensure they do not overlap excessively. All other elements remain unchanged.
    Updated Objects: [('green car #1', [0.327, 0.365, 0.275, 0.207]), ('blue truck #1', [0.350, 0.369, 0.472, 0.408])]

- Example 2
    User prompt: Swap the positions of a green car and a blue truck in this landscape scene with an air balloon.
    Current Output Objects: [('green car #1', [0.350, 0.369, 0.275, 0.207]), ('blue truck #1', [0.027, 0.365, 0.272, 0.208]), ('red air balloon #1', [0.086, 0.010, 0.189, 0.176])]
    Reasoning: Exchange locations of the car and truck to align the bottom right part; other objects remain unchanged.
    Updated Objects:  [('green car #1', [0.027, 0.365, 0.275, 0.207]), ('blue truck #1', [0.350, 0.364, 0.272, 0.208]), ('red air balloon #1', [0.086, 0.010, 0.189, 0.176])]

- Example 3
    User prompt: Change the color of the dolphin from blue to pink in this oil painting of a dolphin and a steamboat.
    Current Objects: [('steam boat #1', [0.302, 0.293, 0.335, 0.194]), ('blue dolphin #1', [0.027, 0.324, 0.246, 0.160])]
    Reasoning: Alter only the dolphin's color from blue to pink, without modifying other elements.
    Updated Objects: [('steam boat #1', [0.302, 0.293, 0.335, 0.194]), ('pink dolphin #1', [0.027, 0.324, 0.246, 0.160])]

- Example 4
    User prompt: Remove the leftmost bowl in this photo with two bowls and a dog.
    Current Objects: [('dog #1', [0.186, 0.592, 0.449, 0.408]), ('bowl #1', [0.376, 0.194, 0.324, 0.324]), ('bowl #2', [0.676, 0.494, 0.324, 0.324])]
    Reasoning: There are two bowls in the image and bowl #1 is identified as the leftmost one because its x coordinates (0.376) is smaller than that of bowl #2 (0.676).Thus, eliminate bowl #1 without modifying any remaining instances.
    Updated Objects: [('dog #1', [0.186, 0.592, 0.449, 0.408]), ('bowl #2', [0.676, 0.494, 0.324, 0.324])]

- Example 5
    User prompt: Add a pink bowl between two existing bowls in this photo.
    Current Objects: [('bowl #1', [0.076, 0.494, 0.324, 0.324]), ('bowl #2', [0.676, 0.494, 0.324, 0.324])]
    Reasoning: There are two bowls in the existing image. To add a pink bowl between the two, the x coordinates should be placed between 0.076 and 0.676 and the y coordinates should be between 0.494 and 0.494. When adding the object, be sure to prevent overlapping between existing objects and make sure the [top-left x-coordinate, top-left y-coordinate, top-left x-coordinate+box width, top-left y-coordinate+box height] lie between 0 and 1.
    Updated Objects: [('bowl #1', [0.076, 0.494, 0.324, 0.324]), ('bowl #2', [0.676, 0.494, 0.324, 0.324]), ('bowl #3', [0.376, 0.494, 0.324, 0.324])]
```

Table 7. Image editing in-context examples for the LLM controller.

References

[1] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023. 1

[2] Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. Diffedit: Diffusion-based semantic image editing with mask guidance. In *The Eleventh International Conference on Learning Representations*, 2023. 1

[3] Long Lian, Boyi Li, Adam Yala, and Trevor Darrell. Llm-grounded diffusion: Enhancing prompt understanding of text-to-image diffusion models with large language models. *arXiv preprint arXiv:2305.13655*, 2023. 2

[4] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning, 2023. 1

[5] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023. 1

[6] Neil Houlsby Matthias Minderer, Alexey Gritsenko. Scaling open-vocabulary object detection. *NeurIPS*, 2023. 2

[7] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, Xiao Wang, Xiaohua Zhai, Thomas Kipf, and Neil Houlsby. Simple open-vocabulary object detection. page 728–755, 2022. 2

[8] OpenAI. Dall·e 3 system card, 2023. 2

[9] OpenAI. Gpt-4v(ision) system card, 2023. 1

[10] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022. 3

[11] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C Schmidt. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*, 2023. 3