Visual Prompt Discovery via Semantic Exploration

Jaechang Kim[1,2*], Yotaro Shimose[1], Zhao Wang[1], Kuang-Da Wang[1,3*], Jungseul Ok[2], and Shingo Takamatsu[1]

[1] Sony Group Corporation
[2] Pohang University of Science and Technology
[3] National Yang Ming Chiao Tung University
[*] Work done during an internship in Sony
jaechang@postech.ac.kr, Shingo.Takamatsu@sony.com

Abstract. Large Vision-Language Models (LVLMs) encounter significant challenges in image understanding and visual reasoning, leading to critical perception failures. Visual prompts, which incorporate image manipulation code, have shown promising potential in mitigating these issues. While visual prompts have emerged as a promising direction, previous methods for visual prompt generation have focused on tool selection rather than diagnosing and mitigating the root causes of LVLM perception failures. Because of the opacity and unpredictability of LVLMs, optimal visual prompts must be discovered through empirical experiments, which have relied on manual human trial-and-error.

In this work, we propose an automated semantic exploration framework for discovering task-wise visual prompts. Unlike previous methods, our approach enables diverse yet efficient exploration through agent-driven experiments, minimizing human intervention and avoiding the inefficiency of per-sample generation. We introduce a semantic exploration algorithm named SEVEX, which addresses two major challenges of visual prompt exploration: (1) the distraction caused by lengthy, low-level code and (2) the vast, unstructured search space of visual prompts. Specifically, our method leverages an abstract idea space as a search space, a novelty-guided selection algorithm, and a semantic feedback-driven ideation process to efficiently explore diverse visual prompts based on empirical results.

We evaluate SEVEX on the BlindTest and BLINK benchmarks, which are specifically designed to assess LVLM perception. Experimental results demonstrate that SEVEX significantly outperforms baseline methods in task accuracy, inference efficiency, exploration efficiency, and exploration stability. Notably, our framework discovers sophisticated and counter-intuitive visual strategies that go beyond conventional tool usage, offering a new paradigm for enhancing LVLM perception through automated, task-wise visual prompts.

Keywords: Large Vision Language Model · Visual prompt · Automated Prompt Engineering

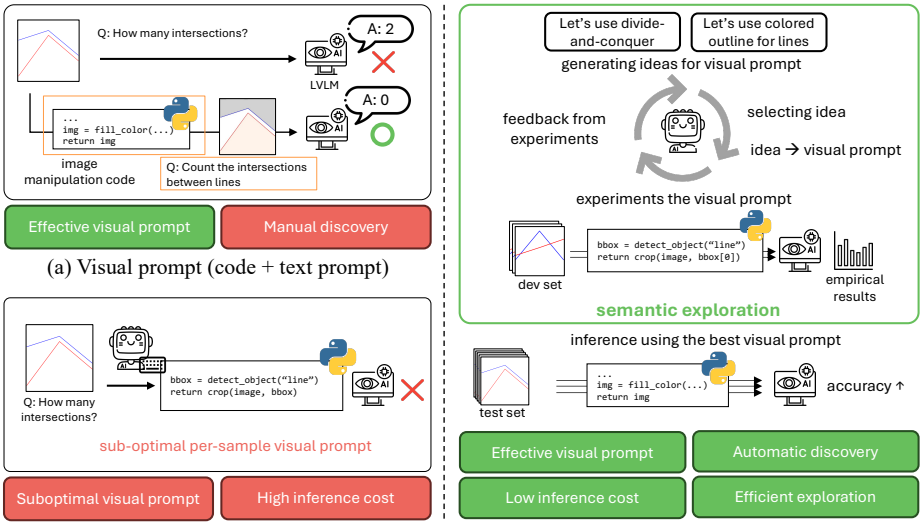


Fig. 1: Overview of visual prompting paradigms for LVLMs. Visual prompts are an effective approach for mitigating perception failure of LVLM. Rather than manual discovery and zero-shot generation, we propose a framework to explore task-wise visual prompts efficiently.

1 Introduction

Large Vision-Language Models (LVLMs) have demonstrated remarkable capabilities in complex reasoning and open-ended dialogue. However, LVLMs often struggle with fundamental visual perception such as identifying fine-grained attributes or understanding spatial relationships, leading to hallucinations or incorrect reasoning based on misperceived visual input [4, 13, 14]. The inherent opacity and unpredictability of these models make it difficult to anticipate such failures, necessitating specialized interventions to ground their vision more reliably.

One of the most promising directions to mitigate these failures is visual prompting [5–7, 16, 17, 21, 23]. In addition to traditional text-based prompts, visual prompting involves direct manipulation of the input image—often through programmatic code—to explicitly guide the model's attention to relevant features. By overlaying geometric cues [7, 16], highlighting specific regions [21], or applying external tools [6], visual prompts enable LVLMs to bypass the failure points and to focus on the most important part. This programmatic approach allows for a precise, reproducible way to enhance visual grounding, offering a powerful tool for correcting the perceptual blind spots of LVLMs.

Despite its potential, current research on visual prompt generation remains in its infancy, often focusing on tool usage rather than diagnosing and resolving fundamental perception failures. More importantly, discovering an effective visual

prompt for a specific task remains a heavily manual process of trial-and-error. Because LVLMs react to visual changes in highly sensitive and non-intuitive ways, human researchers must spend significant effort empirically testing various ideas to find the optimal strategy. This challenge is further compounded by the lack of transferability across different architectures; as we demonstrate in Section 4.6, a visual prompt optimized for one backbone model rarely yields consistent gains when applied to another. This implies that a visual prompt for every model needs to be discovered independently. This heavy dependency on human effort limits the scalability of visual prompting and underscores the urgent need for an automated, agent-driven discovery framework.

However, automating this discovery process is not straightforward; it presents two significant technical challenges. First, the distraction of low-level code: lengthy and complex image-manipulation scripts can introduce unintended noise, overwhelming the LVLM instead of aiding it. Second, the vast and unstructured search space: the infinite combinations of visual modifications make it nearly impossible for a naive agent to find an optimal solution.

To address these, we propose an automated semantic exploration framework for task-wise visual prompts. Instead of searching through raw code, our approach operates on a high-level idea space. By utilizing a novelty-guided selection algorithm and backpropagating actionable insights from sample-wise analysis, our framework efficiently explores diverse visual strategies and converges on effective, task-wise prompts with minimal human oversight.

We evaluate our framework on two benchmarks [4, 14] specifically designed to expose the visual perception failures of LVLMs. Our experiments demonstrate that the automatically discovered visual prompts significantly outperform existing baselines. Furthermore, our analysis reveals that optimal visual prompts are rarely transferable across different LVLM settings, a finding that strongly emphasizes the necessity of our automated discovery framework for model-specific optimization. These results validate the efficiency of semantic exploration and suggest a new paradigm for enhancing the reliability of vision-language models through automated discovery.

We illustrate the problem setting and our motivation in Figure 1 and summarize our contributions as follows:

- **Automated Discovery of Task-wise Visual Prompts:** We introduce an agent-driven framework that automatically discovers task-wise visual prompts, moving beyond manual engineering and sub-optimal zero-shot generation. Recognizing that effective visual prompts are highly model-specific and rarely transferable, our framework provides a scalable solution.
- **Semantic Exploration:** We propose SEmantic Visual prompt EXploration (SEVEX) to overcome the challenges of exploration in raw visual prompt space. SEVEX utilizes 1) an abstract idea space as a search space and 2) a novelty-guided node selection algorithm to enable efficient and diverse exploration, and 3) a semantic backpropagation that analyzes sample-wise results into actionable insights for future idea generation, enabling the diverse and efficient exploration.

– **Performance & Analysis:** We demonstrate the effectiveness of SEVEX on the BlindTest and BLINK benchmarks, achieving superior task accuracy, inference efficiency, and exploration stability. Notably, our framework discovers sophisticated, counter-intuitive visual strategies, not limited to conventional tool usage.

2 Related Work

2.1 Perception failures in LVLMs and visual prompting

Perception failures in LVLMs. Recent studies have increasingly revealed that LVLMs, despite their advanced reasoning capabilities, suffer from systematic *perception failures*. Benchmarks such as BlindTest [14], VLM Eye Exam [13], and Blink [4] demonstrate that these models often struggle with fundamental visual tasks, including fine-grained attribute identification and spatial relationship understanding [19, 24]. These failures lead to hallucinations where the model's reasoning is grounded on misperceived visual information.

Visual prompting strategies. We define a *visual prompt* as image-manipulation code paired with text prompts. Existing strategies for improving LVLM perception can be grouped into two paradigms, depending on their objective and generation timing: 1) *Visual tool-use:* Early methods [5, 6] generate Python programs at inference time to call external vision tools, such as segmentation or depth estimation models. While effective at outsourcing perception, they treat the LVLM mainly as a controller and do not address its intrinsic visual reasoning failures. Since tool-use is generated zero-shot without diagnosis, an initially misinterpreted problem can lead to mismatched tools, cascading errors, and high inference cost. Fine-tuning-based methods [2, 18] also focus on tool selection rather than understanding when LVLMs fail. 2) *Visual scaffolding for intrinsic perception:* In contrast, visual scaffolding directly modifies the input image to mitigate intrinsic failures such as spatial ambiguity and attribute binding. Methods such as Viser [7], BBVPE [16], Zoom-in [23], and Set-of-Mark [21] overlay geometric cues or markers on task-relevant regions. However, these methods remain largely heuristic, manually designed, or dependent on model fine-tuning.

Despite its promise, visual scaffolding remains underexplored as an automated, task- and model-specific prompting strategy. Existing methods are mostly static or require human experimentation, and our experiments show that LVLM sensitivity makes shared visual prompts ineffective across backbones. Our framework fills this gap with an agent-driven system that automatically discovers task-wise visual prompts through empirical experimentation, yielding model-specific prompts that mitigate intrinsic perception failures without per-sample tool generation.

2.2 Automated prompt engineering

From text to visual domain. Since the introduction of Chain-of-Thought (CoT) prompting [9], prompt engineering has become essential for maximizing

the performance of Large Language Models (LLMs). To overcome the limitations of manual trial-and-error, automated methods [20, 25] have been developed to optimize prompts based on empirical feedback from small datasets. In the visual domain, early optimization efforts primarily focused on vision-language representation models like CLIP, utilizing gradient-based soft prompt learning [3, 15]. However, these continuous optimization techniques are inapplicable to the discrete and non-differentiable nature of generating programmatic visual prompts for LVLMs, which require structured code and natural language instructions rather than continuous vector adjustments.

Complexity of programmatic visual prompts. A significant hurdle in automating visual prompt engineering is the inherent complexity and length of the prompts. Unlike text-only prompts, a programmatic visual prompt consists of a combination of formatted image-manipulation code and descriptive text. This hybrid structure results in significantly longer contexts, which poses a long-context distraction problem for LLMs [11]. The increased search space of potential code snippets and their sensitive interaction with the LVLM's perception make direct generation and optimization via raw text or code search extremely difficult. Direct search often leads to the agent being overwhelmed by low-level implementation details rather than focusing on the high-level logic of visual grounding.

Mitigating complexity via semantic exploration. To mitigate these issues, our framework shifts the paradigm from searching in the raw visual prompt to semantic exploration within an abstract idea space. Instead of forcing the agent to reason over every line of low-level manipulation code, our approach enables the agent to iterate on high-level conceptual strategies. By decoupling the semantic intent (the "Idea") from its implementation (the "Code"), our method reduces the cognitive load on the agent and enables a more efficient, novelty-guided search. This abstraction allows the agent to focus on diagnosing the core perception failures and discovering effective visual strategies, bridging the gap between automated text prompt engineering and the complex requirements of LVLM visual grounding.

3 Method

We propose SEmantic Visual prompt EXploration (SEVEX), which is designed to discover optimal, task-wise visual prompts for LVLMs. Unlike previous works that rely on manual trial-and-error or per-sample tool-use, our framework treats visual prompt discovery as an iterative search problem over a high-level idea space. The core of our approach is a dynamically expanding search tree $\mathcal{T}$ that enables an agent to hypothesize, implement, and explore visual prompts based on empirical feedback. We illustrate the overview of our algorithm in Figure 2.

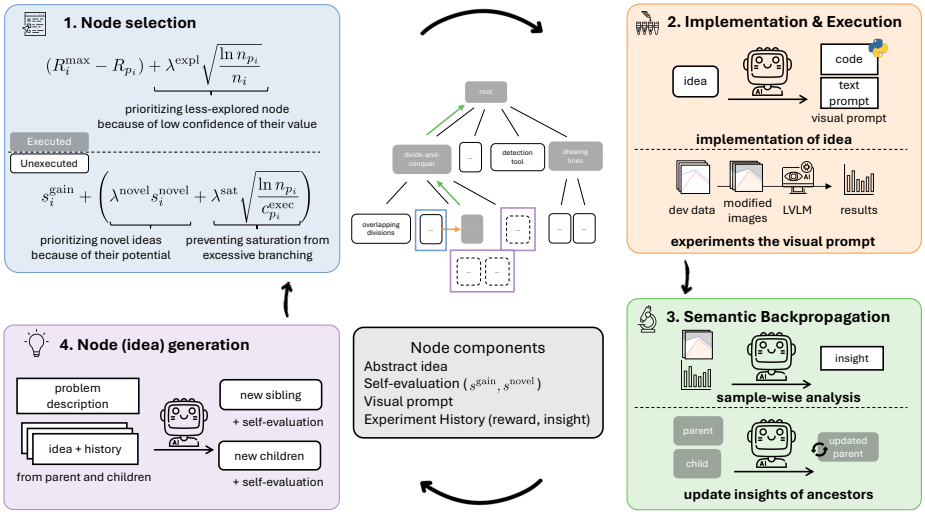


Fig. 2: Overview of SEVEX. SEVEX generates diverse ideas, selects a node with the highest potential, executes the idea, and back-propagates the insights for future idea generation.

3.1 Search space and dynamic tree structure

To overcome the vast and unstructured nature of visual prompts, we define the search space in a high-level domain. The search tree $\mathcal{T}$ is not pre-defined but grows dynamically as the agent generates new ideas. Every executed node has k unexecuted child nodes to ensure exploration of diverse nodes. Whenever an unexecuted node is executed, a new node with a new idea is generated. Each node N in the tree represents a distinct idea for a visual prompt and encapsulates the following components:

- **Abstract Idea (I):** A natural language idea describing a visual prompt.
- **Implementation (P):** The concrete instantiation of I, consisting of executable Python code utilizing a pre-defined set of visual tools and the corresponding textual prompt.
- **Self-Evaluation Estimates (S):** Heuristic scores $\{s^{\text{gain}}, s^{\text{novel}}\}$ evaluated by the agent to guide the search before execution. s^{gain} represents expected gain from using that idea, and s^{novel} represents the novelty of the idea compared to its siblings.
- **Experiment History (H):** Results of empirical experiments, including performance metrics and qualitative observations from the experiment and insights back-propagated from descendant nodes.

3.2 Exploration pipeline

The exploration follows a four-stage iterative loop: 1) **Selection**, where the most promising idea is chosen; 2) **Implementation and Execution**, where the

idea is translated into a visual prompt and tested on a development set; and 3) **Backpropagation**, where results are distilled into insights to guide the next generation of ideas; 4) **Expansion**, where new ideas are generated based on the insights from the previous executions.

3.3 Node selection via Novelty-guided UCT

We use the same motivation of Upper Confidence Bound (UCB) [1] algorithm and UCB for Trees (UCT) [8] algorithm, which employ node value and its potential for exploration. Previous algorithms execute every child node at least once and calculate the potential based on the confidence of node value. However, this approach is inefficient in our context because LLM can generate infinite number of new children from a parent node, and generated ideas can be similar to each other or include less promising ideas.

To overcome these challenges, we propose a Novelty-guided UCT (NUCT) which employs novelty to estimate the potential of an unexecuted node. NUCT differentiates the priority scoring mechanism based on a node's execution status, selecting between using empirical results and using expectations. In our tree structure, every executed node maintains k unexecuted child nodes to ensure continuous exploration. The novelty of an unexecuted node is evaluated with two terms: a novelty compared to its siblings, and the number of executed siblings, which corresponds to the saturation of the parent node. Specifically, the priority score P_i for a node i is determined as follows:

For executed nodes ($n_i > 0$): We follow the standard UCB formula with slight modification. Instead of the mean reward, we use the maximum reward achieved by the node or any of its descendants ($R_i^{\max}$) relative to its parent's reward (R_{p_i}):

$$P_i = (R_i^{\max} - R_{p_i}) + \lambda^{\text{expl}} \sqrt{\frac{\ln n_{p_i}}{n_i}} \tag{1}$$

, where n_i is the visitation count and p_i is the parent node of node i. This term encourages selecting the node with the best reward in the layer-wise comparison.

For unexecuted nodes ($n_i = 0$): Since empirical rewards are unavailable, we estimate the reward and its potential using the agent's self-evaluation and saturation of its parent node:

$$P_i = s_i^{\text{gain}} + \left(\lambda^{\text{novel}} s_i^{\text{novel}} + \lambda^{\text{sat}} \sqrt{\frac{\ln n_{p_i}}{c_{p_i}^{\text{exec}}}} \right) \tag{2}$$

, where s_i^{gain} and s_i^{novel} are normalized scores for the agent's estimate of gain from using the idea and how distinct the idea is from its siblings. $c_{p_i}^{exec}$ denotes the number of executed children of p_i, which is equivalent to the number of executed sibling nodes of node i. This saturation term (multiplied by λ^{sat}) penalizes over-explored branches, prioritizing deeper exploration of promising descendants rather than executing less-promising child ideas of a well-explored parent.

3.4 Implementation and empirical evaluation

Once a node is selected, the agent acts as an "Engineer" to instantiate the abstract idea (I) into a visual prompt (P). We provide the agent with a list of visual tools and documentation to use them. This visual prompt is executed on a development set, which is a small subset for exploration. The performance (e.g., accuracy) and intermediate images are saved, and numerical score is saved as the empirical reward R_i.

3.5 Semantic backpropagation with sample-wise analysis

Following execution, an Analyst agent performs a sample-wise failure analysis. Rather than simply propagating numerical rewards, we implement **Semantic Backpropagation**. The analyst diagnoses why a strategy succeeded or failed with executed results for development set. We provide predictions and ground truth for each sample, and a few representative images actually passed to the LVLM. To reduce token consumption during exploration, we use representative images for successful cases and failed cases instead of using all images. These raw analyses are distilled into **Actionable Insights**—high-level lessons about which visual components are effective for the task. These insights are back-propagated to the *Experiment History (H)* of all ancestor nodes. This ensures that the important lessons are propagated, preventing the agent from repeating ineffective visual manipulations in future branches.

3.6 Insight-driven idea generation

The cycle concludes with the expansion of $\mathcal{T}$. Guided by the updated H, the agent generates new child nodes. To ensure a balance between depth and breadth, for every executed node, the agent generates: sibling nodes to explore alternative conceptual directions at the same abstraction level, and child nodes to refine and specialize the current strategy based on backpropagated feedback. Newly created nodes are initialized with self-evaluation scores S, making them ready for the next Selection phase. Agent generates self-evaluation of expectation, novelty, and feasibility. Feasibility evaluates if the idea is feasible to implement with the given tools. If not, the idea is rejected and a new node is generated instead. Expectation scores and novelty scores are normalized and re-scaled. This closed-loop system allows the framework to autonomously navigate the complex landscape of visual prompting without human intervention.

4 Experiments

4.1 Experimental settings

Datasets. We evaluate our framework using the **BlindTest** [14] and **BLINK** [4] datasets, which are specifically designed to assess the perception failures of LVLMs. Following the experimental protocol of SketchPad [6], we select a set of tasks requiring fine-grained visual grounding and complex reasoning:

Table 1: Description of Visual tools used in our experiments.

Category	Tool Name	Description	Parameter(s)
basic	get_image_size	Returns image dimensions	image
	convert_image_grayscale	Converts image to grayscale	image
	crop	Crops a specified region of the image	image, coordinates
	overlay_images	Overlays two images	image1, image2, coordinates, opacity
drawing	draw_line	Draws a line on the image	image, coordinates, color
	draw_box	Draws a rectangle on the image	image, coordinates, color
	draw_filled_box	Draws a filled rectangle on the image	image, coordinates, color
external model	detect_objects	Object detection using GroundingDINO [12]	image, text query, threshold
	sliding_window_detection	Sliding window object detection	image, text query
	segment_and_mark	Semantic segmentation using SemanticSAM [10]	image, granularity, mark type
	estimate_depth	Depth estimation using DepthAnything [22]	image
LVLM	ask_to_LVLM	Sends a query to LVLM and returns response	images, text prompt

- **BlindTest:** Counting Line Intersection, The Circled Letter, Following Single-colored Paths (Subway Map), Counting Overlapping Shapes.
- **BLINK:** Jigsaw, Relative Depth, Spatial reasoning, Semantic Correlation, Visual Correlation.

For each task, we randomly sampled 30 images to construct a development set for exploration, reserving the remaining images for the test set. Utilizing a small-sized development set aims to demonstrate that SEVEX's semantic exploration is significantly more cost-efficient than alternative optimization methods such as fine-tuning. We describe the details of each task and the full list of development set samples in Appendix B.

Baseline methods. We compare SEVEX against the following visual prompting baselines:

- **Naive**: The standard LVLM inference where the LVLM receives the raw image and the task query without additional visual prompting.
- **SketchPad** [6]: A zero-shot test-time reasoning framework that dynamically generates image-annotation code during inference. To ensure a fair comparison, we add drawing tools to the original tool list to match the capabilities of SEVEX. SketchPad includes few-shot examples of tool usage and Blink tasks.
- **SketchPad+APE** [25]: An enhanced version of SketchPad where the text prompt is optimized using Automatic Prompt Engineering (APE). We applied iterative APE with resampling, using the original SketchPad prompt as the initial state. This baseline uses the same development set as SEVEX for a controlled comparison.

Implementation details. We utilize a comprehensive set of visual tools, which are listed in Table 1. We use Gemini-2.5-flash as our primary backbone LVLM. We disabled the 'thinking' feature during the benchmark inference, while enabled it for SEVEX (e.g., idea generation and implementation) to leverage the agent's reasoning capabilities. According to the documentation of Gemini, disabling

Table 2: Comparison of visual prompting methods. Bold and underlined texts denote the best and second-best methods for each setting, respectively. Inf. cost indicates the number of total tokens per-sample.

Task	Naive		SketchPad		SEVEX(ours)	
	accuracy	inf. cost	accuracy	inf. cost	accuracy	inf. cost
LineIntersections	73.0	981	33.3	16730	**90.5**	991
CircledLetter	78.8	1323	82.0	13902	**83.7**	1337
SubwayMap	60.0	1333	58.1	16482	**62.8**	1585
OverlappingShapes	50.4	2429	16.3	24833	**52.4**	2612
Average-BlindTest	65.6	1517	47.4	13987	**72.4**	1631
Jigsaw	75.8	869	70.8	14896	**95.8**	682
Depth	83.0	1349	**85.1**	14158	**85.1**	1457
Spatial	81.4	1525	85.8	15818	**86.7**	1555
SemanticCorr.	55.0	712	58.7	9477	**63.3**	1358
VisualCorr.	87.3	641	**90.9**	14291	89.4	801
Average-Blink	76.5	1019	78.3	13728	**84.1**	1171
Average	71.6	1240	64.6	15621	**78.9**	1375

Table 3: Comparison to visual prompt exploration baseline. Expl. cost indicates the number of total tokens per-iteration in exploration stage. Inf. cost indicates the number of total tokens per-sample in inference stage.

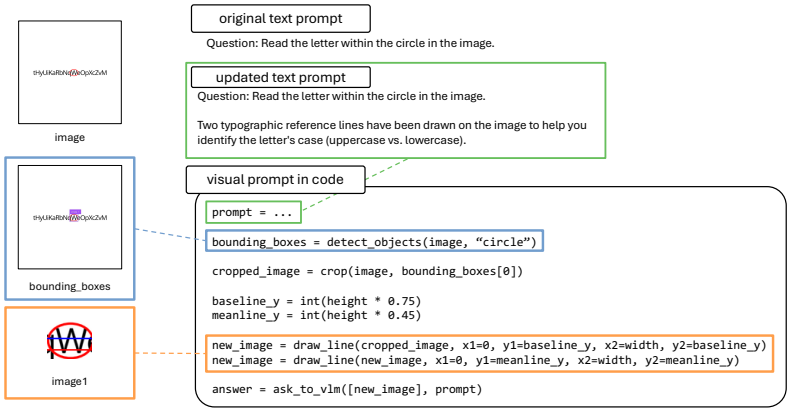
Task	SketchPad+APE				SEVEX(ours)			
	dev acc.	test acc.	expl. cost	inf. cost	dev acc.	test acc.	expl. cost	inf. cost
LineIntersections	30.0	16.4	797k	11005	93.3	90.5	39k	991
CircledLetter	93.3	73.7	980k	17647	85.6	83.7	79k	1337
SubwayMap	50.5	42.5	759k	21471	80.0	62.8	165k	1585
OverlappingShapes	33.3	22.9	818k	20672	56.7	52.4	86k	2612
Average-BlindTest	51.7	38.9	838k	17699	78.9	72.4	92k	1631
Jigsaw	76.7	66.4	761k	18176	96.7	95.8	75k	682
Depth	76.7	85.9	794k	18867	85.6	85.1	87k	1457
Spatial	90.0	76.2	589k	19074	91.1	86.7	83k	1555
SemanticCorr.	76.7	52.6	385k	12354	72.2	63.3	74k	1358
VisualCorr.	83.3	83.1	761k	11937	88.9	89.4	78k	801
Average-Blink	80.7	72.8	658k	16082	86.9	84.1	80k	1171
Average	67.8	57.7	738k	16800	83.3	78.9	85k	1375

thinking yields better results in image perception tasks like image segmentation. Semantic exploration was conducted for 50 iterations with the following hyperparameters: $\lambda^{\text{expl}} = 0.5$, $\lambda^{\text{novel}} = 0.15$, $\lambda^{\text{sat}} = 0.5$, and $k = 3$.

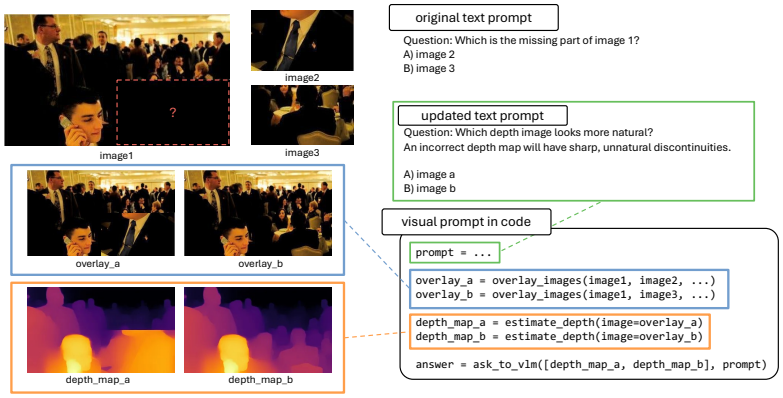
Evaluation metrics. The primary metrics are task accuracy and inference cost, where the cost is defined as the sum of input, output, and reasoning tokens. Unless otherwise specified, all reported accuracies are based on the test set.

4.2 Comparison to visual prompting methods

As shown in Table 2 and Table 3, SEVEX demonstrates superior performance across all critical dimensions: task accuracy, inference efficiency, exploration efficiency, and exploration stability compared to SketchPad and SketchPad+APE.



(a) An example visual prompt in CircledLetter. It crops the target region and draws lines for distinguishing uppercase and lowercase.



(b) An example visual prompt in Jigsaw. It overlays images and uses depth estimation model to judge naturalness of images, although it is not the intended usage of the tool.

Fig. 3: Results of SEVEX. The code and text prompts are simplified for better visualization.

Task performance. SEVEX outperforms existing baselines in seven out of nine tasks of BLINK and BlindTest. SEVEX achieves a superior average accuracy of 78.9%, significantly surpassing the Naive (71.6%) and SketchPad (64.6%) approaches. This performance gap is most pronounced in the BlindTest benchmark, where SEVEX attains an average accuracy of 72.4% compared to SketchPad's 47.4%. The failure of SketchPad in BlindTest can be attributed to its reliance on zero-shot tool use without an empirical understanding of the LVLM's specific perceptual behaviors. When an LVLM fails at a seemingly trivial task such as counting line intersections, zero-shot generation methods typically fail to detect or recover from such errors, whereas SEVEX identifies effective strategies through systematic experimentation.

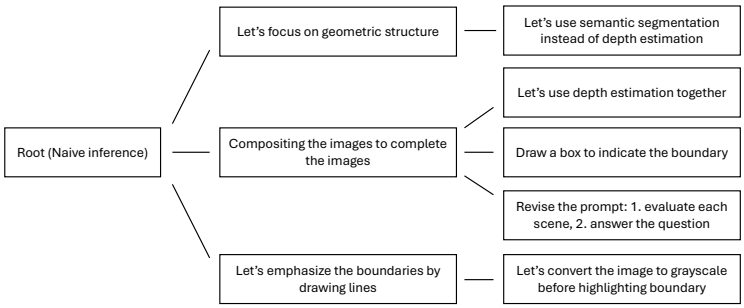


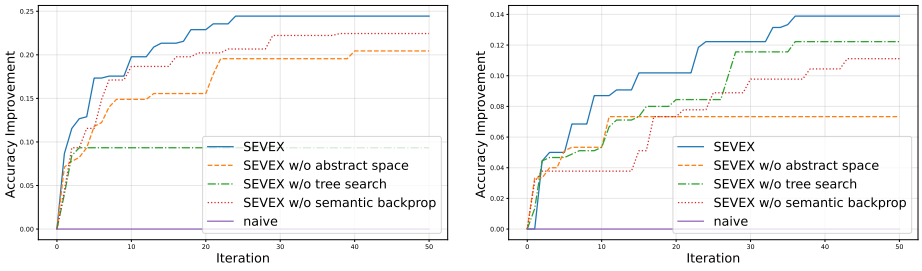
Fig. 4: An example of ideas in a search tree. It shows which kinds of ideas are explored for Jigsaw task. A small portion of nodes is visualized for simplicity.

Inference and exploration efficiency. SEVEX demonstrates remarkable inference efficiency by amortizing the cost of prompt generation over the exploration stage. Consequently, its inference cost is only 10.9% higher than the Naive method, while providing a 91.2% reduction in token consumption compared to SketchPad. Furthermore, SEVEX exhibits superior exploration efficiency; its average exploration cost is 11.5% of the cost required by SketchPad+APE. SEVEX incurs an upfront exploration cost equivalent to approximately 3,442 naive inferences or 273 SketchPad inferences. Consequently, for tasks requiring more than 273 inferences, SEVEX is a more cost-effective solution than SketchPad, while providing the highest accuracy. We provide a more detailed comparison to other exploration strategies in Section 4.4.

Stability of semantic exploration. Despite utilizing a small-sized development set, SEVEX exhibits a narrower generalization gap than SketchPad+APE. The significant generalization gap observed in SketchPad+APE is expected, as it tends to make subtle differences such as paraphrasing. In contrast, SEVEX maintains higher stability by anchoring its search in high-level ideas first, rather than focusing on the subtle implementation details. To further examine sensitivity to the choice of development set, we repeat the discovery on additional random development splits for CircledLetter; the discovered strategy remains stable on the held-out test set across splits (see Appendix C.2).

4.3 Qualitative analysis of visual prompt and exploration tree

We provide qualitative examples of visual prompts discovered by SEVEX in Figure 3. In the *CircledLetter* task, the agent utilizes object detection to localize the target region and renders precise typographic reference lines to assist the LVLM in distinguishing character cases, demonstrating the framework's ability to optimize precise parameters for visual tools. In the *Jigsaw* task, the agent discovers a counter-intuitive strategy: it overlays missing image components onto the query image and employs a depth estimation model to detect unnatural discontinuities.



(a) Jigsaw **(b)** CircledLetter

Fig. 5: Accuracy improvements over iterations. Each figure compares exploration algorithms over iterations. We report the accuracies in the development set averaged on five independent runs.

By repurposing tools in ways not originally intended, the discovery process proves that it is not limited by human bias. Figure 4 visualizes a representative exploration tree with its ideas. SEVEX first explores diverse conceptual directions at the root level before refining these concepts into concrete implementations through a hierarchical structure. This approach enables the framework to both explore a broad strategy space and refine implementation details based on empirical feedback.

4.4 Ablation study over exploration strategies

The contribution of individual exploration components is analyzed in Figure 5. We compare SEVEX against: (1) *Naive*, (2) *SEVEX w/o Tree Structure*, and (3) *SEVEX w/o Semantic Backpropagation*, (4) *SEVEX w/o Abstract Space*. The results indicate each component is essential for efficiency of SEVEX. Especially for SEVEX w/o tree structure, it could achieve good performance when the first direction is promising, but if the task is complicated and requires trials in diverse directions, it is less likely to find a good solution. Because of the resource consumption of each experiment, we only provide ablation studies for Jigsaw and CircledLetter tasks. To further isolate the contribution of the search itself from that of the external pretrained tools, we additionally evaluate SEVEX with a restricted toolbox containing only basic and drawing primitives; even without external model tools, SEVEX retains most of its gains (see Appendix C.1).

4.5 Generality across LVLM backbones

To verify that the gains of SEVEX are not specific to Gemini-2.5-flash, we run the full discovery process on two additional backbones, GPT-4o and Claude Sonnet 4.0, over four BlindTest tasks (Table 5). SEVEX improves the average accuracy over the Naive baseline on both GPT-4o (38.8% $\rightarrow$ 62.4%) and Claude Sonnet 4.0 (66.8% $\rightarrow$ 75.2%), confirming that the improvement persists across backbones with distinct perceptual behaviors. The gain is largest where the

Table 4: Non-transferability of visual prompts. We compare four visual prompts with three LVLM models. Accuracies are calculated in LineIntersection task.

	accuracy of Gemini-2.5-flash	accuracy of Claude-Sonnet-4	accuracy of GPT-4o
naive	73.0	66.3	8.2
visual prompt 1 (found with Gemini)	90.5	82.9	59.1
visual prompt 2 (found with Claude)	87.9	87.8	31.0
visual prompt 3 (found with GPT)	62.3	35.0	57.1

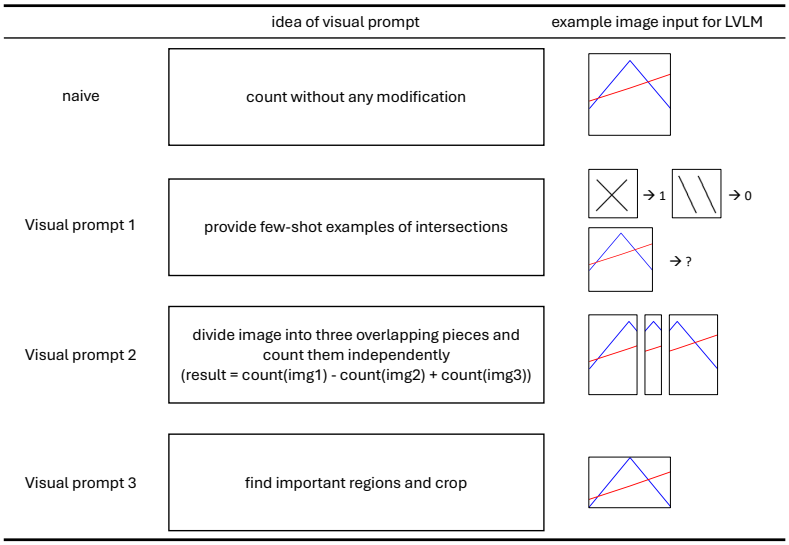


Fig. 6: Simplified visual prompts used in Table 4. Each visual prompt is result of SEVEX with different LVLM models. Visual prompts have different effects on different LVLM model, which means we cannot expect transferability of visual prompts.

backbone is weakest, such as GPT-4o on LineIntersections (8.2% → 57.1%). SEVEX is not uniformly the best on every task and backbone, e.g., SketchPad slightly outperforms SEVEX on Claude for CircledLetter (90.7% vs. 80.5%), but SEVEX consistently provides the strongest average. Crucially, the most effective prompt differs across backbones, which motivates the model-specific discovery analyzed next.

4.6 Non-transferability of visual prompt

Table 4 and Figure 6 demonstrate that the discovered visual prompts are rarely transferable across different LVLM backbones. For instance, the *visual prompt 2* that divides an image into three overlapping pieces yields greater improvements than *visual prompt 3* for Claude, whereas this tendency is reversed for GPT. These results underscore the necessity of discovering model-specific visual prompts tailored to the unique perceptual biases of each backbone. Given the laborious

Table 5: Generality across LVLM backbones. Test accuracy (%) on four BlindTest tasks for three backbones. The Gemini-2.5-flash columns are reproduced from Table 2; the GPT-4o and Claude Sonnet 4.0 columns are obtained by running the full SEVEX discovery on each backbone. The best result within each backbone is in **bold**.

	Gemini-2.5-flash			**GPT-4o**			**Claude Sonnet 4.0**		
Task	Naive	SketchPad	SEVEX	Naive	SketchPad	SEVEX	Naive	SketchPad	SEVEX
LineIntersections	73.0	33.3	**90.5**	8.2	19.2	**57.1**	66.3	63.6	**87.8**
CircledLetter	78.8	82.0	**83.7**	61.1	43.9	**85.4**	76.1	**90.7**	80.5
SubwayMap	60.0	58.1	**62.8**	49.6	19.6	**61.0**	59.7	37.4	**60.6**
OverlappingShapes	50.4	16.3	**52.4**	36.4	33.3	**46.2**	65.1	59.8	**72.0**
Average	65.6	47.4	**72.4**	38.8	29.0	**62.4**	66.8	62.9	**75.2**

nature of manual prompt engineering, this highlights the importance of our automated discovery framework.

5 Conclusion

In this work, we introduced SEVEX, an agent-driven framework designed to automatically discover task-wise visual prompts that mitigate the intrinsic perception failures of LVLMs. Motivated by the observation that LVLMs often struggle with fundamental visual reasoning despite their advanced linguistic capabilities, our approach shifts from manual trial-and-error or per-sample code generation to a structured search within a high-level, abstract idea space. By decoupling semantic intent from programmatic implementation, SEVEX effectively navigates the vast search space of visual modifications while avoiding long-context distraction.

Our experiments on the BlindTest and BLINK benchmarks demonstrate that SEVEX significantly outperforms existing baselines, including zero-shot tool-use frameworks and automated text prompt engineering. A critical takeaway from our analysis is the non-transferability of visual prompts; optimal visual cues discovered for one model frequently fail or even degrade performance when applied to another. This finding reinforces the necessity of an automated, model-specific discovery process to account for the unique perceptual biases of different backbones. Finally, we discuss broader implications and future research directions in Appendix D.

Acknowledgement

Jaechang Kim and Jungseul Ok were partly supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT)(RS-2024-00509258, Global AI Frontier Lab).

References

1. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine learning **47**(2), 235–256 (2002)
2. Bai, T., Hu, Z., Sun, F., Qiu, J., Jiang, Y., He, G., Zeng, B., He, C., Yuan, B., Zhang, W.: Multi-step visual reasoning with visual tokens scaling and verification. NeurIPS (2025), https://arxiv.org/abs/2506.07235
3. Chen, A., Yao, Y., Chen, P.Y., Zhang, Y., Liu, S.: Understanding and improving visual prompting: A label-mapping perspective. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 19133–19143 (2023)
4. Fu, X., Hu, Y., Li, B., Feng, Y., Wang, H., Lin, X., Roth, D., Smith, N.A., Ma, W.C., Krishna, R.: Blink: Multimodal large language models can see but not perceive. In: European Conference on Computer Vision. pp. 148–166. Springer (2024)
5. Gupta, T., Kembhavi, A.: Visual programming: Compositional visual reasoning without training. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 14953–14962 (2023)
6. Hu, Y., Shi, W., Fu, X., Roth, D., Ostendorf, M., Zettlemoyer, L., Smith, N.A., Krishna, R.: Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. Advances in Neural Information Processing Systems **37**, 139348–139379 (2024)
7. Izadi, A., Banayeeanzade, M.A., Askari, F., Rahimiakbar, A., Vahedi, M.M., Hasani, H., Baghshah, M.S.: Visual structures help visual reasoning: Addressing the binding problem in vlms. In: NeurIPS (2025)
8. Kocsis, L., Szepesvári, C.: Bandit based monte-carlo planning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) Machine Learning: ECML 2006. pp. 282–293. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
9. Kojima, T., Gu, S.S., Reid, M., Matsuo, Y., Iwasawa, Y.: Large language models are zero-shot reasoners. Advances in neural information processing systems **35**, 22199–22213 (2022)
10. Li, F., Zhang, H., Sun, P., Zou, X., Liu, S., Yang, J., Li, C., Zhang, L., Gao, J.: Semantic-sam: Segment and recognize anything at any granularity. ECCV (2024)
11. Liu, N.F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., Liang, P.: Lost in the middle: How language models use long contexts. Transactions of the association for computational linguistics **12**, 157–173 (2024)
12. Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Jiang, Q., Li, C., Yang, J., Su, H., Zhu, J., Zhang, L.: Grounding dino: Marrying dino with grounded pre-training for open-set object detection. ECCV (2024)
13. Nam, H.W., Moon, Y.B., Choi, W., Lee, H., Oh, T.H.: Vlm's eye examination: Instruct and inspect visual competency of vision language models. Transactions on Machine Learning Research **2025** (2025)
14. Rahmanzadehgervi, P., Bolton, L., Taesiri, M.R., Nguyen, A.T.: Vision language models are blind. In: Proceedings of the Asian Conference on Computer Vision. pp. 18–34 (2024)
15. Rezaei, R., Sabet, M.J., Gu, J., Rueckert, D., Torr, P., Khakzar, A.: Learning visual prompts for guiding the attention of vision transformers. arXiv preprint arXiv:2406.03303 (2024)
16. Woo, S., Zhou, K., Zhou, Y., Wang, S., Guan, S., Ding, H., Cheong, L.L.: Black-box visual prompt engineering for mitigating object hallucination in large vision language models. In: Chiruzzo, L., Ritter, A., Wang, L. (eds.) Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association

for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers). pp. 529–538. Association for Computational Linguistics, Albuquerque, New Mexico (Apr 2025). https://doi.org/10.18653/v1/2025.naacl-short.45, https://aclanthology.org/2025.naacl-short.45/

17. Wu, J., Zhang, Z., Xia, Y., Li, X., Xia, Z., Chang, A., Yu, T., Kim, S., Rossi, R.A., Zhang, R., et al.: Visual prompting in multimodal large language models: A survey. arXiv preprint arXiv:2409.15310 (2024)
18. Wu, M., Yang, J., Jiang, J., Li, M., Yan, K., Yu, H., Zhang, M., Zhai, C., Nahrstedt, K.: Vtool-r1: Vlms learn to think with images via reinforcement learning on multimodal tool use (2025), https://arxiv.org/abs/2505.19255
19. Xu, W., Lyu, D., Wang, W., Feng, J., Gao, C., Li, Y.: Defining and evaluating visual language models' basic spatial abilities: A perspective from psychometrics. In: Che, W., Nabende, J., Shutova, E., Pilehvar, M.T. (eds.) Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 11571–11590. Association for Computational Linguistics, Vienna, Austria (Jul 2025). https://doi.org/10.18653/v1/2025.acl-long.567, https://aclanthology.org/2025.acl-long.567/
20. Yang, C., Wang, X., Lu, Y., Liu, H., Le, Q.V., Zhou, D., Chen, X.: Large language models as optimizers. In: The Twelfth International Conference on Learning Representations (2024)
21. Yang, J., Zhang, H., Li, F., Zou, X., Li, C., Gao, J.: Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. arXiv preprint arXiv:2310.11441 (2023)
22. Yang, L., Kang, B., Huang, Z., Xu, X., Feng, J., Zhao, H.: Depth anything: Unleashing the power of large-scale unlabeled data. In: CVPR (2024)
23. Zhang, J., Khayatkhoei, M., Chhikara, P., Ilievski, F.: MLLMs know where to look: Training-free perception of small visual details with multimodal LLMs. In: The Thirteenth International Conference on Learning Representations (2025), https://arxiv.org/abs/2502.17422
24. Zhang, Z., Hu, F., Lee, J., Shi, F., Kordjamshidi, P., Chai, J., Ma, Z.: Do vision-language models represent space and how? evaluating spatial frame of reference under ambiguities (2025), https://arxiv.org/abs/2410.17385
25. Zhou, Y., Muresanu, A.I., Han, Z., Paster, K., Pitis, S., Chan, H., Ba, J.: Large language models are human-level prompt engineers. In: The Eleventh International Conference on Learning Representations (2023), https://openreview.net/forum?id=92gvk82DE-