SketchDirector: Complex Sketching via Agentic Layout Reasoning and Detail Refinement

Yiqi Liang and Xiaodong Cun*

GVC Lab, Great Bay University
https://yiki77.github.io/SketchDirector/

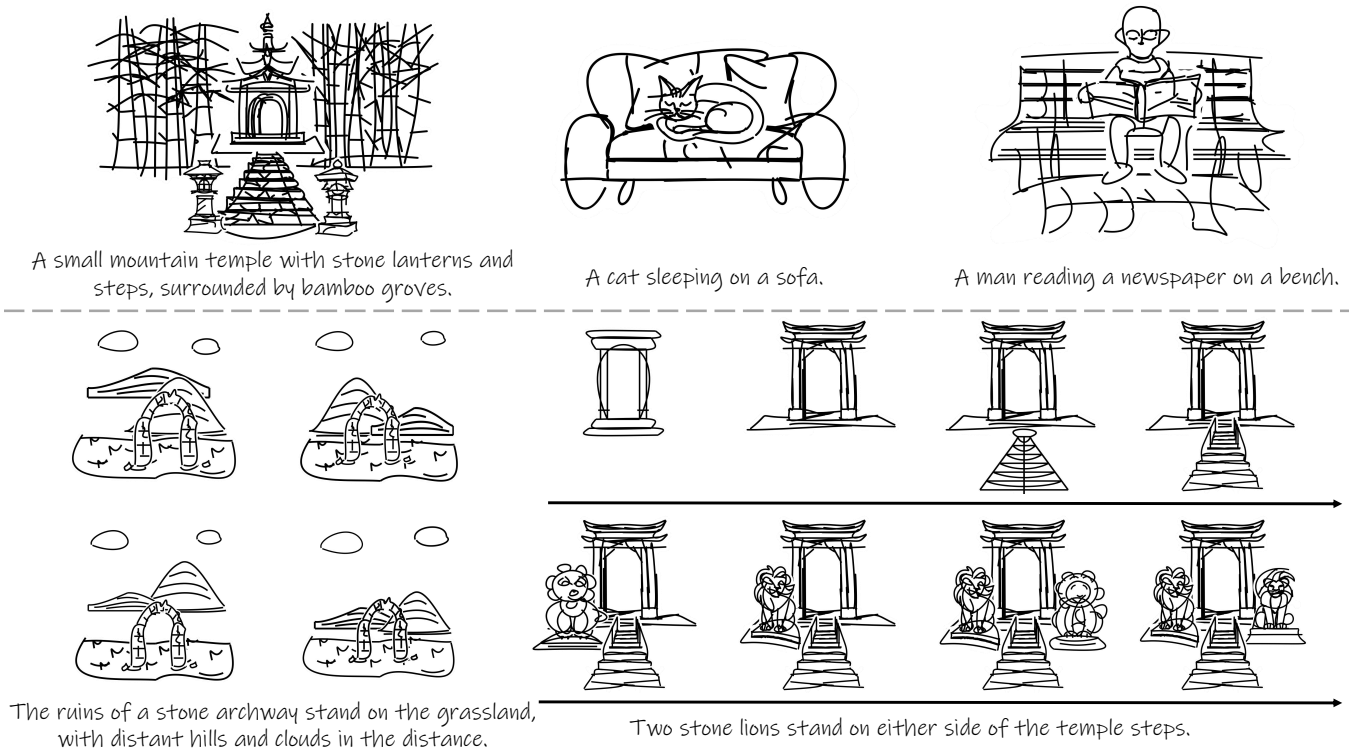


Figure 1: **Visual results of SketchDirector**. The top row shows scene sketches depicting diverse concepts. The bottom-left row shows rich layout variations for the same prompt, highlighting the effectiveness of agentic reasoning for automatic scene composition. The bottom-right row shows the progressive sketch refinement process, which enhances sketch details while preserving sketch structure and spatial layout.

Abstract

Existing sketch generation methods often rely on image guidance and are largely limited to individual objects or simplified scenes, leaving text-guided generation of multi-concept scene sketches underexplored. We present SketchDirector, an agentic framework that mimics the human sketching workflow for automatic multi-concept scene sketching. Given a scene-level text prompt, SketchDirector first employs a large language model (LLM) to decompose the scene into multiple draft assets, where each is represented by a set of Bézier curves. Then, an agentic reasoning loop iteratively plans and executes their spatial arrangement, guided by self-evolving memory. Finally, a detail refinement module leverages diffusion priors to enhance sketch quality while preserving geometric structure. To support future research and applications in multi-concept scene sketch generation, we further introduce the *MCSketch8K* dataset, comprising editable vector assets, compositional layouts, and object-level manipulation annotations. Extensive experiments show that SketchDirector outperforms state-of-the-art baselines in semantic and compositional quality while effectively preserving overall contour organization during refinement. The project page is: https://yiki77.github.io/SketchDirector/.

*Corresponding author.

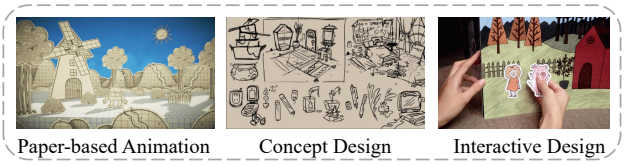
1 Introduction

Sketching is a concise visual representation that effectively captures the intentions of human beings. As its scope expands from individual objects to multiple concepts, scene sketching has shown great potential in applications such as animation, proof of concept, and interactive design, as in Fig. 2 (A). However, multi-concept scene sketching requires not only understanding individual objects but also organizing their visual attributes and spatial relationships into a coherent scene. Therefore, generating semantically aligned scene sketches with reasonable layouts and coherent details remains a big challenge.

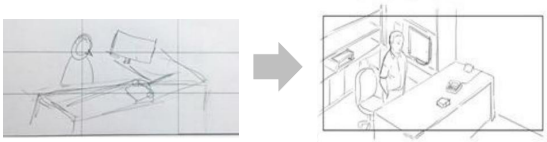
Previous automatic scene sketching generation methods (Li et al. 2019a,b; Chan, Durand, and Isola 2022; Vinker et al. 2023; Liang et al. 2025b) primarily operate at the holistic scene level, treating scene sketches as an indivisible entity without object representations, thereby hindering local editing and object reuse. Recent studies (Wu, Wang, and Yang 2025; Zhang et al. 2026; Zhan et al. 2025) have explored object-aware generation, but objects are often treated as predefined components or auxiliary structures. Consequently, these methods lack dynamic control, layout planning, and iterative optimization for individual objects, which limits their potential as visual components.

In contrast, real-world sketch creation is inherently iterative: artists continually revise, group, and rearrange individual elements to improve the overall composition and presentation. They also adapt the level of refinement to suit their creative goals, producing sketches that range from rough drafts to highly detailed renderings, as illustrated in Fig. 2 (B). Digital vector sketches are particularly well suited to such flexible and iterative workflows because they are naturally represented as collections of Bézier curves, enabling efficient programmatic manipulation.

We thus propose *SketchDirector*, an automatic agentic framework (Vinker et al. 2025; Du et al. 2026; Liu et al. 2025; Liang et al. 2025a) for text-guided multi-concept scene sketch generation by mimicking the pipeline of human creators. Specifically, given a textual description, *SketchDirector* first employs a Large Language Model (LLM) based method to decompose the scene and generate each individual object as draft asset, each represented as a set of Bézier curves. To enable layout control between multiple asset, we introduce an operating action library comprising a set of executable operations, *e.g.,* `move()`, `scale()`, to manipulate object attributes, enabling flexible adjustment of spatial relationships among objects. Building upon this library, we propose an agentic reasoning loop, which formulates layout manipulation as an iterative operations and execution process. To achieve reliable closed-loop reasoning, rather than predicting the entire scene layout at once, *SketchDirector* progressively organizes draft assets through an iterative *Plan–Execute–Verify–Correct* process. At each step, *SketchDirector* generates and executes object-level operations, updates the scene representation, and evaluates the resulting composition through geometric checks and multimodal verification. When the layout fails verification, *SketchDirector* revises its decisions and corrects the layout using a hierarchical self-evolving memory that leverages short-term failure reflections and accumulated long-term experiences.



(A) Examples of scene sketch used in various applications.



(B) Examples of different levels of sketch refinement.

Figure 2: **Motivation of multi-concept scene sketch generation.** (A) Widely used in animation, concept art, and design. (B) Artists adjust sketch refinement from draft to detailed sketch preserving structure for different goals.

After that, we introduce an additional detail refinement step to address the challenges of structural consistency in scene sketch distillation from the diffusion prior. Unlike traditional methods that treat Bézier curves as unstructured point sets, we treat object curve segments as the optimization primitives. Then, by jointly optimizing local objects and the global canvas using pre-scene distillation, our method is able to generate scene sketches with varying levels of object refinement while maintaining consistency with the textual description. Finally, with the lack of available object-aware and editable scene sketch datasets, we construct a multi-concept scene sketch dataset called *MCSketch8K* to facilitate future research and applications in scene sketch generation. The dataset comprises 1,000 scene sketches with over 4,000 disentangled vector object assets across 800 object categories. Each scene is annotated with compositional layouts and object-level manipulation operations, providing structured supervision for object decomposition, spatial arrangement reasoning, and controllable scene editing.

Our key contributions are summarized as:

- We propose *SketchDirector*, an automatic framework for text-guided multi-concept scene sketch generation with flexible object-level control.
- *SketchDirector* contains two key techniques to assist in scene sketch generation: *i) agentic reasoning*, which formulates generation as a closed-loop process; *ii) detail refinement*, which performs Bézier curve parametrization and progressively joint local-global vector optimization to refine scene sketches.
- We construct *MCSketch8K*, a multi-concept scene sketch dataset comprising vector object assets, compositional scene layouts, and object-level manipulation annotations.

2 Related Work

Scene Sketch Generation. Scene sketch generation aims to translate visual or textual descriptions into coherent sketches. Early methods formulated this task as image-to-sketch translation (Li et al. 2019a,b; Chan, Durand, and Isola 2022),

converting photographs into rasterized contour drawings via supervised learning or adversarial training. Later studies increasingly adopt vector representations for scalable and editable sketch generation. Image-conditioned methods such as Clipascene (Vinker et al. 2023) and MROSS (Liang et al. 2025b) produce vector scene sketches from a reference image, while text-guided methods (Frans, Soros, and Witkowski 2022; Jain, Xie, and Abbeel 2023; Xing et al. 2023, 2024, 2025) further enable open-domain vector sketch generation from textual descriptions. In particular, VectorFusion (Jain, Xie, and Abbeel 2023) distills text-to-image diffusion priors into SVGs through score distillation, DiffSketcher (Xing et al. 2023) directly optimizes Bézier curves with diffusion-based supervision, and SVGDreamer (Xing et al. 2024) improves text-to-SVG generation through semantic-driven vectorization and particle-based score distillation. Despite these advances, existing methods optimize the scene holistically, leaving objects implicitly encoded and limiting editing to strokes rather than objects.

Multi-concept Sketch Generation. Recent studies explore structured scene generation by introducing different forms of concept decomposition and controllability. For instance, SketchTriplet (Wu, Wang, and Yang 2025) assembles individual objects to construct scene datasets, although the objects mainly serve as fixed composition units without supporting layout planning or per-concept editing. SketchAny (Zhang et al. 2026) separates foreground and background to improve structural coherence, yet provides limited control over individual concept instances. CoProSketch (Zhan et al. 2025) represents objects with unsigned distance fields to enable progressive scene generation and improved editability, while generation is still performed at the scene level rather than through independent concept generation. Collectively, these methods highlight the value of concept-aware representations, yet objects remain passive elements rather than autonomous units for planning, interaction, and iterative refinement.

Agents for Sketch Generation. The rapid development of multimodal large language models (MLLMs) has introduced reasoning, planning, and iterative interaction into sketch generation. For instance, SketchAgent (Vinker et al. 2025) introduces a language-driven sequential sketch generation framework, where an LLM-based agent can generate sequences of vector sketches and supports conversational sketch refinement. Building upon this, Du et al. (Du et al. 2026) further explore controllable vector sketch generation by training a multimodal agent with part-level annotations and process-level feedback, enabling interpretable and locally editable sketch synthesis. In addition, AutoSketch (Chin et al. 2025) investigates style-aware vector sketch completion, leveraging VLM-based iterative adjustment to preserve the abstraction level and stroke characteristics of input sketches during completion. Despite promising results, existing methods lack high-level scene reasoning and multi-object coordination. SketchDirector addresses this with an agentic MLLM framework for hierarchical planning, object-level control, and iterative refinement.

3 SketchDirector

The goal of *SketchDirector* is to generate editable multi-concept scene sketches from textual descriptions. Each sketch is represented as a collection of SVG assets parameterized by black Bézier curves, enabling object-level editability and structural preservation. Given a scene description, we first decompose it into individual concepts and generate corresponding vector draft assets (Sec. 3.1). To support executable scene composition, we define an operating action library consisting of primitive operations that manipulate object attributes and update the scene state (Sec. 3.2). Building upon these, an agentic layout reasoning step progressively composes the scene through a closed loop, where agents translate spatial intentions into executable editing programs and iteratively plan, execute, verify, and correct the scene composition (Sec. 3.3). Finally, a detail refinement step further enhances sketch details through Bézier curve contour resampling and progressive joint optimization, while preserving the structure of draft assets and spatial layout of the canvas (Sec. 3.4). A high-level illustration is shown in Fig. 3.

3.1 Assets Generation

Given a textual description T, we first employ a Large Language Model (LLM) to parse the scene description and extract a set of objects $\mathcal{O} = \{o_1, o_2, ..., o_n\}$. Each object corresponds to an independently drawable visual entity (such as stones, a lighthouse, etc.), while abstract or environmental elements (e.g., mist) are excluded since they cannot be naturally and visibly represented as sketch assets. To support subsequent draft generation and layout arrangement, each object is formulated as $o_i = (\mathrm{id}_i, d_i, z_i)$, where id_i denotes a unique object identifier, d_i is a simple description sentence of the object, and z_i is its relative depth order.

After obtaining the object set $\mathcal{O}$, we further generate a simple draft for each object. Specifically, we employ SketchAgent (Vinker et al. 2025) as the draft generator. Given the description d_i, we predict an ordered sequence of sampled points on a canvas to describe the object outline. Then, by solving a least-squares optimization problem, these sampled points are fitted to a vector draft composed of cubic Bézier curves, producing object draft g_i. Each draft is synthesized on a separate canvas and added to its corresponding object $o_i = (\mathrm{id}_i, d_i, z_i, g_i)$. To prevent interference from other objects, we further inject a vectorized white background mask derived from the object's contour, enabling subsequent reasoning to assemble the scene in a collage-like manner.

3.2 Operating Action Library

To support interpretable and executable scene composition, we define a compact set of object-level operation codes that serve as the action space of the reasoning agent. Given the current scene state, the agent produces a sequence of executable actions to progressively arrange and revise the retrieved sketch assets. An action is selected from six primitive operations:

`move`(o, x, y) moves object o to canvas coordinate (x, y).
`scale`(o, s) rescales object o by a relative factor s.
`rotate`(o, θ) rotates object o by angle θ around its center.

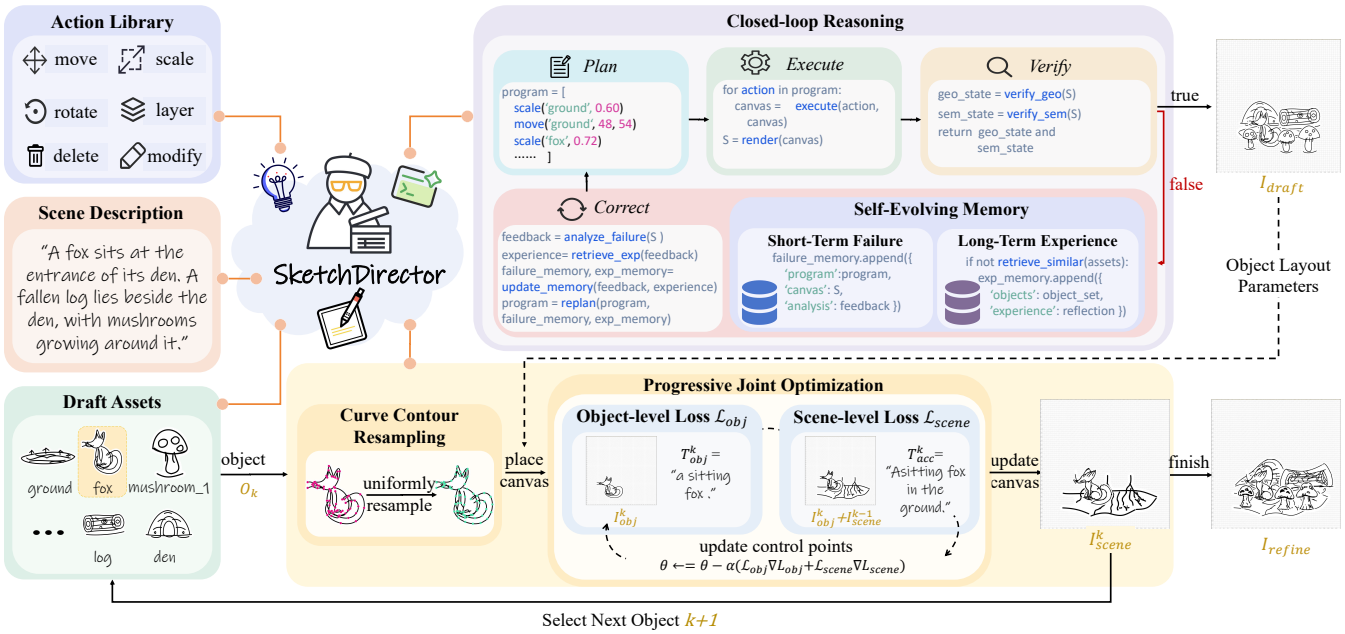


Figure 3: ***SketchDirector* pipeline**. Given a scene description, *SketchDirector* first generates object-level draft assets. The agent then progressively composes the scene through executable operations from the action library, maintaining scene programs and spatial asset attributes. A closed-loop process iteratively adjusts the layout, supported by self-evolving memory that incorporates short-term failure feedback and long-term experiences. Finally, detail refinement employs curve contour resampling and progressive joint optimization to enhance sketch details while preserving the established scene layout and object structures.

`layer`(o, l) changes the depth order of object o by assigning it to layer l.
`delete`(o) removes object o from the scene composition.
`modify`(o, d) regenerates object o based on description d.

Note that a single layout revision may require composing multiple primitives. For example, placing a large object behind another may involve a `scale`, followed by a `move` and a `layer` operation. By keeping these primitives minimal and composable, the agent can express diverse layout manipulations with a compact action vocabulary, while each individual operation remains explicit and easy to verify.

3.3 Agentic Layout Reasoning

After generating draft assets, directly assembling drafts often produces implausible spatial relationships (*e.g.*, inappropriate scales, incorrect occlusions). Therefore, *SketchDirector* formulates scene sketching as an agentic layout reasoning process that employs a closed-loop *Plan–Execute–Verify–Correct* process to iteratively improve scene composition. At each iteration, the agent plans object-level programs, executes them, evaluates the resulting layout, and revises its decisions when necessary, as illustrated in Figs. 4.

Planning and Execution. Planning aims to translate high-level layout intentions into an executable program. We encode layout decisions as a sequence of functional tools rather than directly predicting object attributes, including `move()`, `scale()`, `rotate()`, `layer()`, `delete()` and `modify()`, enabling flexible and interpretable arrangement. To facilitate reliable spatial reasoning, the canvas is discretized into an $R \times R$ regular grid like previous works (Sun et al. 2025; Vinker et al. 2025). The agent infers the grid location of each object, which is deterministically mapped back to the corresponding canvas position during execution. The agent generates programs by jointly reasoning over geometric and visual clues, where object attributes, grid positions, and bounding boxes provide spatial constraints, while rendered canvases and layout histories provide global context. The generated operations are sequentially executed to update object attributes and scene states, producing an explicit and traceable composition trajectory, as illustrated in Fig. 5. This traceability allows the agent to identify problematic operations during subsequent verification and correction.

Multimodal Verification. To ensure the layout from the agent meets spatial constraints and textual descriptions, we propose a hybrid verification with geometric and visual-semantic check. Geometric verification checks basic spatial validity by ensuring that all draft assets lie within the canvas boundaries and their sizes avoid extreme scaling anomalies. Visual-semantic verification complements geometric checks by assessing high-level semantic consistency. The details are described below.

Geometric Verification. We evaluate the geometric plausibility of the generated layout from two aspects: boundary constraints and object scales. Specifically, after executing the predicted layout program, each object is represented by its transformed bounding box on the canvas. For boundary verification, we check whether the bounding box of each object is fully contained within the canvas region. Any object exceeding the canvas boundary is recorded as a boundary violation with a violation count I_{bound}. The boundary verification is considered successful only when no violation is detected:



Figure 4: **Example agentic reasoning trajectory in *SketchDirector*.** A real log illustrating how agents interleave layout planning, scene state updates, geometric-semantic verification, memory evolution, and correction.

$$P_{\text{bound}} = \text{True} \quad \text{if} \quad I_{\text{bound}} = 0.$$

For size verification, we examine the scale factor of each object to avoid extreme scaling artifacts. Objects with scale factors outside the predefined range $[0.1, 1.5]$ are recorded as size violations I_{size}. The size verification passes when:

$$P_{\text{size}} = \text{True} \quad \text{if} \quad I_{\text{size}} = 0.$$

The geometric verification is satisfied only when both boundary and size constraints are fulfilled.

Visual-Semantic Verification. We employ a Vision-Language Model (VLM) critic with a dual-image input consisting of the rendered scene canvas and the object collage. The VLM produces scores $S_i \in [0, 1]$ from five perspectives: (1) S_{sem}: semantic alignment between the generated scene and the target concept, evaluating whether the main objects and their intended meanings are correctly represented; (2) S_{comp}: spatial composition consistency, evaluating whether object placement, spacing, overlap, depth ordering, and overall arrangement form a coherent scene; (3) S_{compl}: object completeness, evaluating whether all required objects are present and recognizable according to the provided object collage; (4) S_{size}: size consistency, evaluating whether object scales and relative proportions are reasonable and physically plausible within the scene; (5) S_{aes}: visual aesthetics,

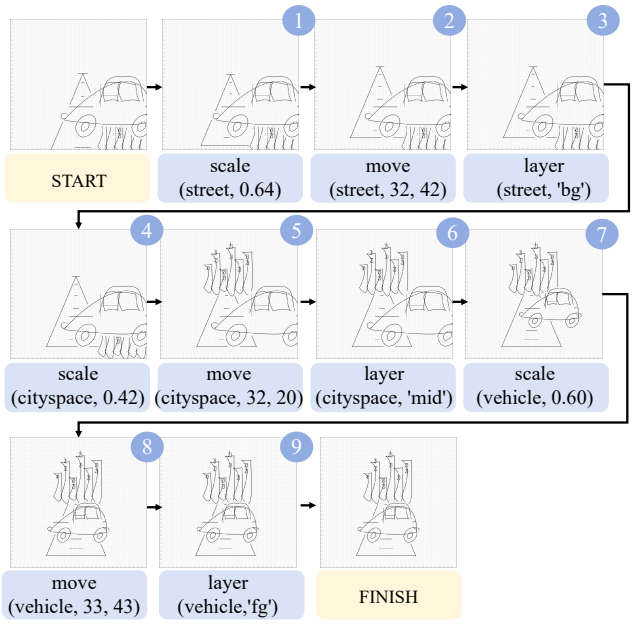


Figure 5: **A success example of planning and execution in *SketchDirector*.** Each operation updates both the executable program and the represented scene state, yielding a traceable trajectory from the initial drafts to the final layout.

evaluating the overall visual quality, including composition harmony, clarity, and visual appeal. The overall evaluation score is computed as a weighted aggregation:

$$S_{\text{visual}} = \sum \lambda_i S_i, \quad \sum \lambda_i = 1, \tag{1}$$

where $i \in \{\text{sem}, \text{comp}, \text{compl}, \text{size}, \text{aes}\}$.

The semantic verification is considered successful when:

$$P_{\text{visual}} = \text{True} \quad \text{if} \quad S_{\text{visual}} \geq \tau_{\text{visual}}.$$

Finally, the overall verification decision is determined by combining geometric and semantic assessments:

$$P_{\text{overall}} = P_{\text{bound}} \wedge P_{\text{size}} \wedge P_{\text{visual}}. \tag{2}$$

Only layouts satisfying all verification criteria are accepted. Otherwise, the detected violations and verification feedback are recorded and sent to the self-evolving correction module. the agent then uses these failure signals together with previous execution trajectories to generate a revised layout program in the next planning iteration.

Self-Evolving Correction. Different from contrastive experience for object-level sketch generation (Xiao et al. 2026), inspired by human learning, we propose a hierarchical self-evolving mechanism that Short-Term Failure Memory converts failure trajectories into corrective feedback and Long-Term Experience Memory transfers successful cross-scene trajectories into reusable experiences, which are incrementally stored and retrieved to improve subsequent layout planning and program generation.

The Short-Term Failure Memory. The Short-Term Failure Memory maintains a trajectory-level record for each failed attempt. Each entry contains the generated program,

intermediate rendering results, object-level attributes, verification feedback, and the corresponding failure category. These records are discarded after successful correction and are only used within the current scene generation process.

The Long-Term Experience Memory. The Long-Term Experience Memory stores successful layout experiences indexed by the corresponding object set. For a new scene, relevant experiences are retrieved by measuring the Jaccard similarity between the current object set and historical object sets (Park et al. 2023; Shinn et al. 2023):

$$\mathrm{Sim}(\mathcal{S}_{\mathrm{new}}, \mathcal{S}_{\mathrm{mem}}) = \frac{|\mathcal{S}_{\mathrm{new}} \cap \mathcal{S}_{\mathrm{mem}}|}{|\mathcal{S}_{\mathrm{new}} \cup \mathcal{S}_{\mathrm{mem}}|}. \tag{3}$$

For example, suppose the current scene contains $\mathcal{S}_{\mathrm{new}} = \{\texttt{tree}, \texttt{squirrel}, \texttt{bird}\}$, while a stored experience is indexed by $\mathcal{S}_{\mathrm{mem}} = \{\texttt{tree}, \texttt{squirrel}, \texttt{monkey}\}$. Their intersection is $\{\texttt{tree}, \texttt{squirrel}\}$ with cardinality 2, and their union is $\{\texttt{tree}, \texttt{squirrel}, \texttt{bird}, \texttt{monkey}\}$ with cardinality 4. Therefore,$\mathrm{Sim} = \frac{2}{4} = 0.50$ Since the similarity reaches the predefined threshold τ_{sim}, this experience is retrieved and provided to the agent as a reference for subsequent layout planning.

3.4 Detail Refinement

After reasoning, the generated sketches achieve the desired spatial arrangement but contain coarse details. To improve quality while preserving object structure and layout, we introduce a detail refinement using curve contour resampling to convert object contours into Bézier primitives and progressive joint optimization to refine local objects and the global canvas, as shown in Fig. 3.

Curve Contour Resampling. To obtain a fixed number of optimization primitives, we resample object contours into N_{target} cubic Bézier segments. Unlike point-based Bézier resampling method (Vinker et al. 2022), our approach directly subdivides the existing vector contours without reconstructing their geometry. Specifically, we convert foreground SVG paths into cubic Bézier curves, excluding background masks and degenerate segments. Given valid paths containing N_{orig} curves in total, we assume $N_{\mathrm{target}} \geq N_{\mathrm{orig}}$ and assign at least one segment to each original curve. The remaining budget is proportionally allocated first across paths and then across curves according to their arc lengths, using the largest-remainder method to ensure exactly N_{target} segments. Each original curve is subdivided into its allocated number of cubic Bézier segments using the de Casteljau algorithm, which preserves the original contour geometry. The resulting ordered sequence of primitives is defined as

$$\begin{aligned} \mathcal{B} &= \bigoplus_{k=1}^{K} \bigoplus_{j=1}^{n_k} \left(B_{k,j}^{(1)}, \ldots, B_{k,j}^{(N_{k,j})} \right), \\ |\mathcal{B}| &= \sum_{k=1}^{K} \sum_{j=1}^{n_k} N_{k,j} = N_{\mathrm{target}}. \end{aligned} \tag{4}$$

where $\bigoplus$ denotes ordered concatenation, preserving the original path and curve ordering. During subsequent refinement, the number and ordering of primitives remain fixed, while their control points are independently optimized. It provides a faithful starting point that retains object shapes and structural details, as illustrated in Fig. 11.

Progressive Joint Optimization. Based on the resampled representation, each object draft is expressed as a fixed-topology set of Bézier primitives parameterized by control points θ_i and rendered as $I_i = R(\theta_i)$, where $R(\cdot)$ denotes the differentiable vector renderer (Li et al. 2020). During optimization, only the control-point geometry is updated, while the curve ordering inherited from the draft remains fixed.

During local optimization, we apply Rectified-Flow Distillation Sampling (RFDS (Yang et al. 2025)) for supervision:

$$\mathcal{L}_{obj}^{i} = \mathcal{L}_{RFDS}(I_{obj}^{i}, T_{\mathrm{obj}}^{i}), \tag{5}$$

where T_{obj}^{i} corresponds to d_i, is the description of the object.

During global optimization, each object is incrementally composed with the canvas $I_{scene}^{i} = \mathcal{C}(I_{scene}^{i-1}, I_i, \ell_i)$ using layout attributes ℓ_i, where $\mathcal{C}(\cdot)$ denotes differentiable composition. The scene is optimized with a global RFDS loss:

$$\mathcal{L}_{\mathrm{scene}}^{i} = \mathcal{L}_{\mathrm{RFDS}} \left(I_{\mathrm{scene}}^{i}, T_{\mathrm{acc}}^{i} \right), \tag{6}$$

where T_{acc}^{i} describes all objects incorporated into the accumulated canvas and their semantic and spatial relationships at the i-th optimization stage. Consequently, the text condition is progressively expanded as new objects are optimized.

At optimization round for each object, only the control points θ_i of the current object are optimized. The objective combines both object and global supervision:

$$\mathcal{L}^{i} = \lambda_{obj}\mathcal{L}_{obj}^{i} + \lambda_{scene}\mathcal{L}_{scene}^{i}, \tag{7}$$

where λ_{obj} and λ_{scene} balance object-level semantic refinement and scene-level structural alignment.

4 Experiment

4.1 Experimental Setup

Implementation Details. We deploy *SketchDirector* on an NVIDIA GeForce RTX 4090 GPU. We employ OpenAI GPT-5.5 (OpenAI 2026) as the default agent and pydiffvg (Li et al. 2020) as the differentiable vector renderer. All stroke primitives are parameterized as cubic Bézier curves defined by 4 control points. The canvas is discretely configured as a 64×64 grid with each cell spanning 12×12 pixels, resulting in an overall rendered resolution of 780×780 pixels.

Benchmark. To evaluate the performance of scene sketch generation, we sample different multi-concept text as prompts spanning diverse themes and categories from *MCSketch8K*, to comprehensively assess sketch quality.

Baselines. We compare *SketchDirector* against five baselines for scene sketch generation, categorized into LLM-based and optimization-based methods. *LLM-based methods* include SketchAgent (Vinker et al. 2025) and OpenAI GPT-5.5 (OpenAI 2026); *Optimization-based methods* include SVGDreamer (Xing et al. 2024), VectorFusion (Jain, Xie, and Abbeel 2023) and DiffSketcher (Xing et al. 2023).

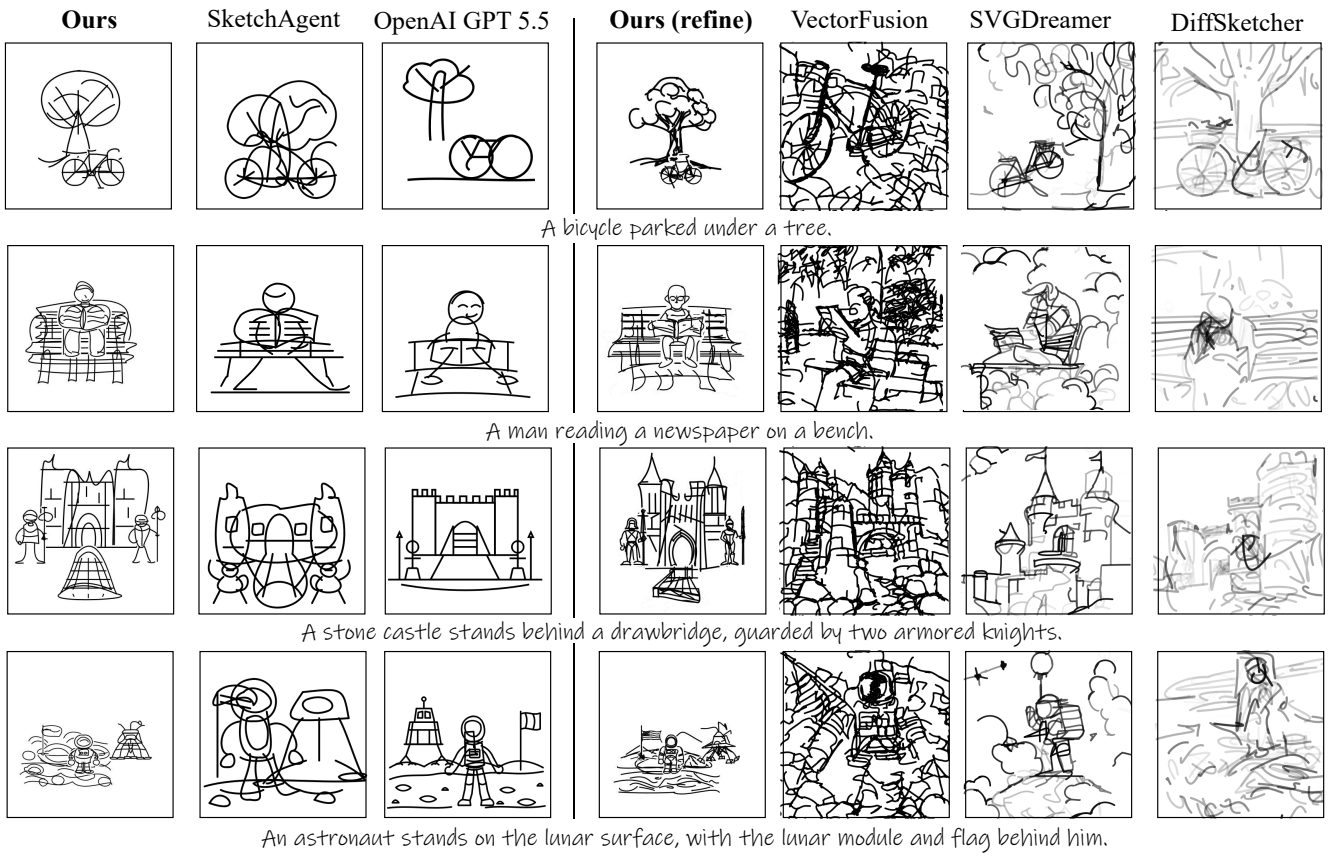


Figure 6: **Qualitative comparison with existing methods.** Compared with *LLM-based methods*, our sketch generation produces scene sketches with more explicit object organization, coherent layouts, and better visual-semantic alignment. Moreover, compared with *optimization-based methods*, our refinement improves sketch details while faithfully preserving the object layouts and spatial relationships in the draft.

Metrics. To assess the semantic and compositional quality of scene sketches, we adopt a multi-VLMs assessment to reduce evaluator bias. We employ Qwen3.7-Plus (Cloud 2026), Claude-Opus-4-7 (Anthropic 2026), Gemini3.1-Pro (DeepMind 2026), and OpenAI GPT-5.5 (OpenAI 2026) as judges. Each VLM independently evaluates five dimensions: Semantic Alignment (Sem.), Spatial Composition (Compo.), Object Completeness (Compl.), Visual Aesthetics (Aes.), and Size Consistency (Size). Scores are averaged across the four VLMs and all test samples for quantitative comparison.

4.2 Experiment Results

We evaluate *SketchDirector* through multi-VLMs assessment and user study to comprehensively measure quantitative performance and human preferences. we sample 30 multi-concept scene descriptions from *MCSketch8K* for evaluation. Tab. 1 shows the results between *SketchDirector* and five baseline methods based on VLM assessment. *SketchDirector* achieves the highest performance across all five metrics. Compared with *LLM-based methods*, *SketchDirector* further improves performance, showing the effectiveness of agentic reasoning. Compared with *optimization-based methods*, *SketchDirector*, OpenAI GPT-5.5, and SketchAgent obtain higher scores due to the capabilities of LLMs, indicating the importance of semantic understanding and high-level scene reasoning. The refinement enhances most metrics, demonstrating the effectiveness of detail refinement.

Method	Sem. ↑	Compo. ↑	Compl. ↑	Aes. ↑	Size ↑
SketchAgent	0.742	0.598	0.589	0.714	0.693
OpenAI GPT-5.5	0.911	0.730	0.811	0.813	0.820
VectorFusion	0.466	0.252	0.338	0.417	0.412
SVGDreamer	0.690	0.546	0.620	0.650	0.695
DiffSketcher	0.567	0.374	0.381	0.567	0.537
SketchDirector	0.916	0.772	0.821	0.836	0.821
SketchDirector (refine)	**0.923**	**0.788**	**0.843**	**0.869**	**0.842**

Table 1: **Comparison with baseline methods on multi-concept scene sketching.** *SketchDirector* achieves the highest performance across all five metrics.

We also conduct a user study with 68 participants recruited from the university community. We sample 15 diverse scene prompts from the *MCSketch8K* and generate sketches using *SketchDirector* and each baseline. For fair comparison, i.e., to isolate the effect of refinement, we consider

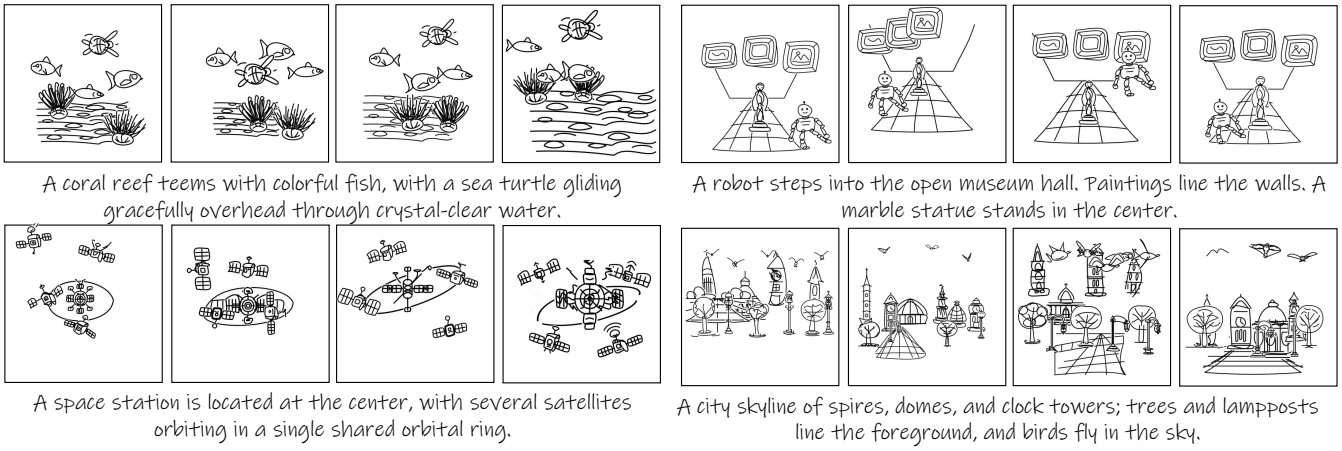


Figure 7: **Scene sketch diversity generated by SketchDirector.** Our object-level reasoning enables flexible scene layouts from textual descriptions, by producing diverse spatial arrangements (the top row) and object configurations under similar semantic descriptions (the bottom row).

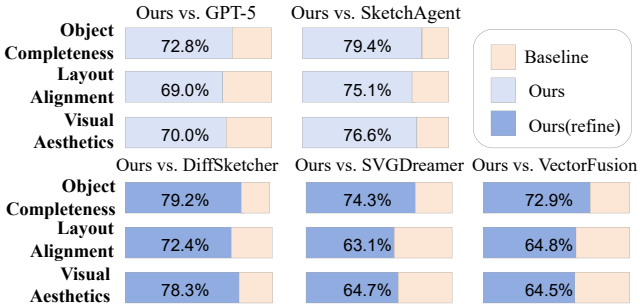


Figure 8: **User study results.** *SketchDirector* preference over baseline methods across three criterias.

two settings. Specifically, *SketchDirector* without refinement is compared with *LLM-based methods*, while *SketchDirector* with structural refinement (Ours (refine)) is compared with *optimization-based methods*. For each pair, participants viewed a description and two randomly ordered sketches and answered three questions: (1) (Object Completeness) *Which sketch better represents the objects in the text?* (2) (Layout alignment) *Which sketch has better layout alignment?* (3) (Visual Aesthetic) *Which sketch looks more aesthetically pleasing?* Overall, we collect 5100 responses per criterion. Fig. 8 reports *SketchDirector* preference rates for the three questions. Across both settings and all baselines, participants consistently favored our results.

Fig. 6 shows the qualitative comparison with baseline methods. *LLM-based methods* show strong semantic understanding and object-level depiction, but may struggle with object disentanglement and visual quality. SketchAgent (Vinker et al. 2025) generates clean sketches through sequential drawing actions, but often causes object overlap in complex multi-concept scenes, while OpenAI GPT-5.5 (OpenAI 2026) generates recognizable structures with rigid line styles. In contrast, *SketchDirector* generates sketches with clearer object boundaries and better visual presentation. *Optimization-based methods* often generate sketches with fragmented strokes and ambiguous structures, due to the lack of explicit object-level representations. VectorFusion (Jain, Xie, and Abbeel 2023) and SVGDreamer (Xing et al. 2024) tend to produce dense and irregular paths, while DiffSketcher (Xing et al. 2023) captures coarse semantics but often results in incomplete structures or limited details. Moreover, Fig. 7 shows that *SketchDirector* can generate diverse scene layouts from the same prompt.

4.3 Ablation Studies.

To systematically evaluate the individual contributions of key components in *SketchDirector*, we conduct ablation studies on another 30 multi-concept scene prompts from *MCSketch8K*. Specifically, we investigate the impact of different VLM backbones on asset generation and layout reasoning, the contributions of individual components within the agentic reasoning loop, and the effectiveness of contour resampling and progressive joint optimization in detail refinement.

Impact of VLM Backbones. To evaluate the generalizability and robustness of *SketchDirector* across different foundational models,we evaluate four state-of-the-art backbones: OpenAI GPT-5.5 (OpenAI 2026), Gemini-3.1-pro (DeepMind 2026), Claude-Opus-4-7 (Anthropic 2026), and Qwen3.7-Plus (Cloud 2026). Figure 9 and Table 2 show that that different backbones exhibit varying capabilities in spatial organization, visual-semantic alignment, and reasoning efficiency. OpenAI GPT-5.5 achieves the highest scores across all verification metrics and requires the fewest reasoning iterations, demonstrating stronger layout planning and scene composition capabilities. Gemini-3.1-Pro and Claude-Opus-4-7 achieve relatively strong semantic alignment but require more iterations to refine their layouts. Their generated sketches generally capture the main scene concepts, yet exhibit fewer visual details . In contrast, Qwen3.7-Plus obtains lower verification scores and requires substantially more reasoning iterations. Overall, these results demonstrate that *SketchDirector* can operate with different backbones, while stronger reasoning capabilities contribute to more accurate scene composition and efficient convergence.

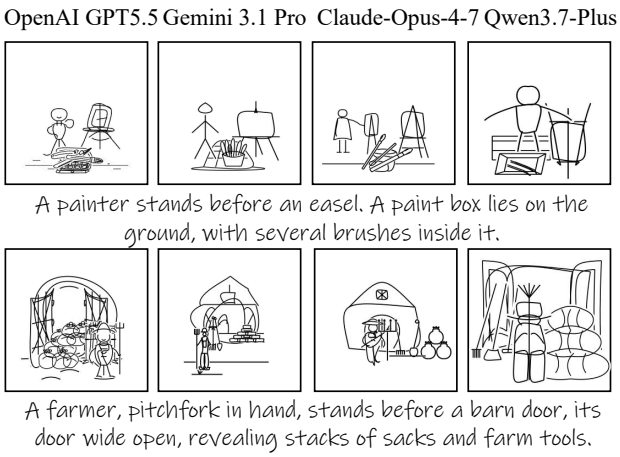


Figure 9: **Qualitative comparison of different VLM backbones.** Visual results highlight differences in object-level sketch fidelity, scene composition, and visual details across backbone models.

Method	Sem.↑	Comp.↑	Compl.↑	Aes.↑	Size↑	Iter.↓	Ver.↑
Gemini-3.1-pro	0.858	0.723	0.723	0.819	0.707	5.332	0.535
Claude-Opus-4-7	0.879	0.681	0.791	0.739	0.721	5.843	0.477
Qwen3.7-Plus	0.629	0.477	0.526	0.552	0.507	10.743	0.436
OpenAI GPT-5.5	**0.926**	**0.795**	**0.811**	**0.871**	**0.785**	**1.656**	**0.803**

Table 2: **Quantitative comparison of different VLM backbones.** Scores and reasoning iterations evaluate the layout reasoning quality and efficiency of *SketchDirector* across backbone models.

Impact of Agentic Layout Reasoning. Figure 10 and Table 3 show that removing any component leads to performance degradation, confirming that all modules in agentic reasoning are essential for high-quality scene sketch generation. Removing planning (*w/o Plan*) leads to severe layout degradation. Without executable scene programs, the model struggles to organize object attributes, resulting in frequent object overlaps, unreasonable spatial arrangements, and boundary violations. These demonstrate that converting high-level semantic intentions into structured layout operations is essential for reliable scene composition. Disabling geometric verification (*w/o Geo. Ver.*) mainly affects physical plausibility. Although the generated objects remain semantically relevant, their positions and sizes may violate basic spatial constraints, such as objects extending beyond the canvas or unrealistic relative scales between objects. This indicates that geometric constraints provide necessary low-level guidance for maintaining valid scene structures. Without visual-semantic verification (*w/o Sem. Ver.*), the agent lacks high-level visual feedback during layout refinement. Consequently, generated scenes may contain incorrect object relationships, missing semantic interactions, or unnatural occlusion orders. The comparison highlights the importance of multimodal evaluation in aligning generated layouts with the intended scene description. Furthermore, memory modules improve the agent's ability to learn from previous reasoning

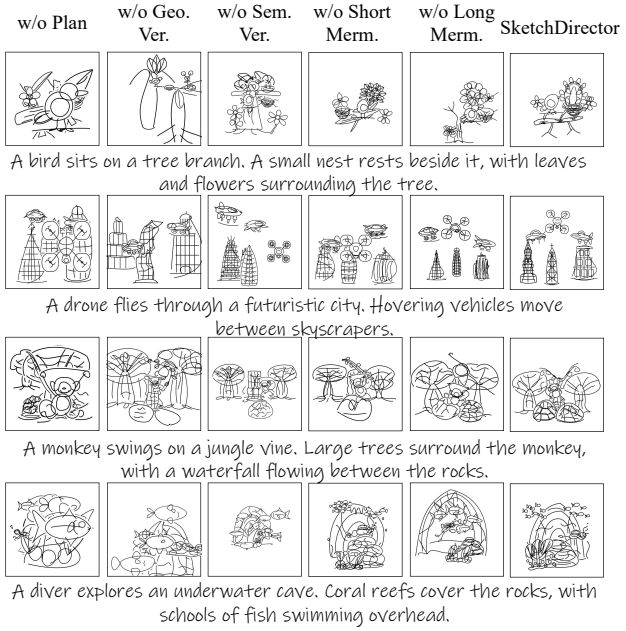


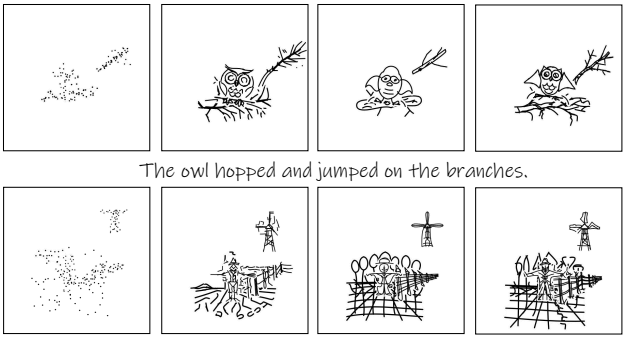
Figure 10: **Qualitative ablation of agentic layout reasoning.** Visual results illustrate how individual components affect scene composition and object-level spatial coherence.

Method	Sem.↑	Comp.↑	Compl.↑	Aes.↑	Size↑	Iter.↓	Ver.↑
w/o Plan	0.676	0.503	0.571	0.65	0.468	12.375	0.357
w/o Geo. Ver.	0.698	0.550	0.570	0.660	0.589	1.857	0.633
w/o Sem. Ver.	0.603	0.512	0.645	0.558	0.547	1.438	0.468
w/o Short Mem.	0.737	0.658	0.629	0.654	0.681	2.875	0.635
w/o Long Mem.	0.781	0.679	0.697	0.762	0.729	1.600	0.650
SketchDirector	**0.839**	**0.765**	**0.732**	**0.768**	**0.742**	**1.375**	**0.873**

Table 3: **Quantitative ablation of agentic layout reasoning.** Performance comparisons across metrics and reasoning iterations demonstrate the contributions of individual components to layout quality and reasoning efficiency.

trajectories. Removing short-term memory (*w/o Short Mem.*) does not cause significant visual layout degradation, but leads to repeated exploration of similar unsuccessful layouts, increasing the number of reflection iterations. Removing long-term memory (*w/o Long Mem.*) also preserves comparable visual structures in many cases, yet prevents the agent from retrieving reusable layout experiences from related scenes, resulting in less effective layout composition. Overall, these qualitative examples verify that each component contributes complementary capabilities, enabling more reliable and controllable scene reasoning.

Impact of Detail Refinement. Fig. 11 provides additional comparisons between point-based and curve-based sampling. Point-based sampling uniformly samples sparse control points from object regions without considering the underlying contours. During optimization, these points may drift independently, causing geometric distortions and loss of characteristic object shapes. In contrast, our curve contour resampling method derives Bézier primitives directly from object contours. By preserving the original contour topology, the curves provide a more stable optimization space. Fig. 12



(A) Point-based **(B) Curve-based**

Figure 11: **Comparison of point-based and curve-based contour resampling method.** Curve-based contour resampling preserves object shapes and structural details, providing a more stable initialization for subsequent refinement.

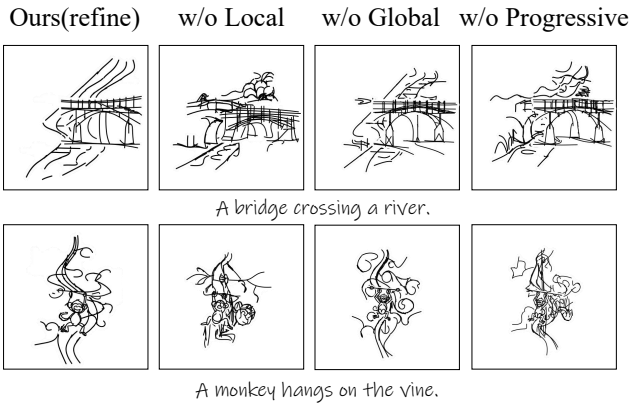
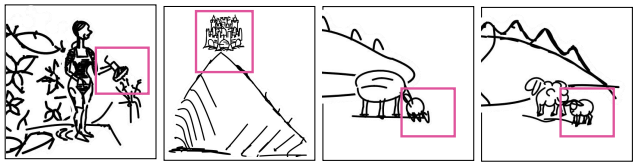


Figure 12: **Qualitative ablation of progressive joint optimization.** Removing local supervision, global supervision, or progressive optimization leads to degraded object fidelity and scene-level coherence.

presents additional comparisons of different optimization strategies. Removing local supervision (*w/o Local*) optimizes objects only according to global objectives, allowing unrelated semantic information to interfere with individual object. This often results in distorted local structures. Removing global supervision (*w/o Global*) focuses only on individual object while ignoring interactions among objects. Although individual objects may remain recognizable, the resulting scenes exhibit weaker spatial coherence, such as unnatural object relationships. Without progressive optimization (*w/o Progressive*), local and global objectives are optimized simultaneously from the beginning. The strong coupling between object-level refinement and scene-level consistency introduces optimization conflicts, causing unstable updates and reduced object fidelity. By progressively incorporating objects, our method enables stable local refinement first and gradually improves global scene integration.



(A) Complex Interactions **(B) Object Hallucination**

Figure 13: **Limitations of *SketchDirector*.** (A) Challenges in modeling complex physical and semantic interactions. (B) Object hallucination during refinement.

4.4 Limitations.

Although *SketchDirector* effectively generates multi-concept scene sketches, several limitations remain. First, it struggles with complex physical and semantic relations, such as interactions between a woman and a kettle or a castle located on a hill (Fig. 13 (A)). Stronger relational and interaction modeling may improve complex scene understanding. Second, refinement may hallucinate redundant instances, leading to object inconsistency, such as duplicating a sheep (Fig. 13 (B)). Instance-aware constraints could help preserve object identity and count. Third, multimodal verification and iterative optimization incur additional computational costs, particularly when multiple refinement rounds are required. More efficient verification and optimization strategies will be explored in future work.

5 Conclusion

We present *SketchDirector*, an agentic framework that emulates the human sketching workflow. Designed for complex scene sketch generation, *SketchDirector* converts textual descriptions into editable vector assets and executable layout operations. It consists of three stages: concept representation for generating object-level draft assets, agentic reasoning with iterative*Plan–Execute–Verify–Correct* and self-evolving memory for reliable spatial composition, and detail refinement for progressively optimizing Bézier-based vectors while preserving the original sketch structure. We further introduce *MCSketch8K*, a benchmark containing editable vector assets and composition annotations for multi-concept scene sketch generation. Extensive experiments demonstrate that *SketchDirector* outperforms existing methods in semantic and compositional quality while effectively preserving structural integrity during refinement.

References

Anthropic. 2026. Claude Opus 4.7. https://www.anthropic.com/news/claude-opus-4-7. Large language model.

Chan, C.; Durand, F.; and Isola, P. 2022. Learning to generate line drawings that convey geometry and semantics. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 7915–7925.

Chin, H.-Y.; Shen, I.-C.; Chiu, Y.-T.; Shamir, A.; and Chen, B.-Y. 2025. AutoSketch: VLM-assisted Style-Aware Vector Sketch Completion. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, 1–11.

Cloud, A. 2026. Qwen3.7-Plus. https://qwen.ai/blog?id=qwen3.7-plus. Large language model.

DeepMind, G. 2026. Gemini 3.1 Pro. https://deepmind.google/models/gemini/pro/. Large language model.

Du, X.; Xu, R.; Yunis, D.; Vinker, Y.; and Shakhnarovich, G. 2026. Teaching an Agent to Sketch One Part at a Time. *arXiv preprint arXiv:2603.19500*.

Frans, K.; Soros, L.; and Witkowski, O. 2022. Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. *Advances in Neural Information Processing Systems*, 35: 5207–5218.

Jain, A.; Xie, A.; and Abbeel, P. 2023. Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1911–1920.

Li, M.; Lin, Z.; Mech, R.; Yumer, E.; and Ramanan, D. 2019a. Photo-sketching: Inferring contour drawings from images. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1403–1412. IEEE.

Li, T.-M.; Lukáč, M.; Gharbi, M.; and Ragan-Kelley, J. 2020. Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics (TOG)*, 39(6): 1–15.

Li, Y.; Fang, C.; Hertzmann, A.; Shechtman, E.; and Yang, M.-H. 2019b. Im2pencil: Controllable pencil illustration from photographs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1525–1534.

Liang, G.; Hu, J.; Xing, X.; Zhang, J.; and Yu, Q. 2025a. Multi-object sketch animation with grouping and motion trajectory priors. In *Proceedings of the 33rd ACM International Conference on Multimedia*, 9237–9246.

Liang, Y.; Ying, L.; Long, D.; and Li, R. 2025b. MROSS: Multi-Round Region-based Optimization for Scene Sketching. In *2025 IEEE International Conference on Multimedia and Expo (ICME)*, 1–6. IEEE.

Liu, J.; Xin, Z.; Fu, Y.; Zhao, R.; Lan, B.; and Li, X. 2025. Multi-Object Sketch Animation by Scene Decomposition and Motion Planning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 11537–11546.

OpenAI. 2026. GPT-5.5. https://openai.com/index/introducing-gpt-5-5/. Large language model.

Park, J. S.; O'Brien, J.; Cai, C. J.; Morris, M. R.; Liang, P.; and Bernstein, M. S. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, 1–22.

Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; and Yao, S. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36: 8634–8652.

Sun, F.-Y.; Liu, W.; Gu, S.; Lim, D.; Bhat, G.; Tombari, F.; Li, M.; Haber, N.; and Wu, J. 2025. Layoutvlm: Differentiable optimization of 3d layout via vision-language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 29469–29478.

Vinker, Y.; Alaluf, Y.; Cohen-Or, D.; and Shamir, A. 2023. Clipascene: Scene sketching with different types and levels of abstraction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4146–4156.

Vinker, Y.; Pajouheshgar, E.; Bo, J. Y.; Bachmann, R. C.; Bermano, A. H.; Cohen-Or, D.; Zamir, A.; and Shamir, A. 2022. Clipasso: Semantically-aware object sketching. *ACM Transactions on Graphics (TOG)*, 41(4): 1–11.

Vinker, Y.; Shaham, T. R.; Zheng, K.; Zhao, A.; E Fan, J.; and Torralba, A. 2025. Sketchagent: Language-driven sequential sketch generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 23355–23368.

Wu, Z.; Wang, Q.; and Yang, J. 2025. SketchTriplet: Self-Supervised Scenarized Sketch–Text–Image Triplet Generation. *IEEE Internet of Things Journal*, 12(9): 13021–13032.

Xiao, H.; Xiao, X.; Wang, Y.; Zhang, Y.; and Qi, Y. 2026. 3DrawAgent: Teaching LLM to Draw in 3D with Early Contrastive Experience. *arXiv preprint arXiv:2604.08042*.

Xing, X.; Wang, C.; Zhou, H.; Zhang, J.; Yu, Q.; and Xu, D. 2023. Diffsketcher: Text guided vector sketch synthesis through latent diffusion models. *Advances in Neural Information Processing Systems*, 36: 15869–15889.

Xing, X.; Yu, Q.; Wang, C.; Zhou, H.; Zhang, J.; and Xu, D. 2025. Svgdreamer++: Advancing editability and diversity in text-guided svg generation. *IEEE transactions on pattern analysis and machine intelligence*.

Xing, X.; Zhou, H.; Wang, C.; Zhang, J.; Xu, D.; and Yu, Q. 2024. Svgdreamer: Text guided svg generation with diffusion model. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4546–4555.

Yang, X.; Chen, C.; Liu, F.; Lin, G.; et al. 2025. Text-to-image rectified flow as plug-and-play priors. In *International Conference on Learning Representations*, volume 2025, 13896–13920.

Zhan, R.; Li, Y.; He, Y.; Chen, S.; Shen, Y.; Chen, X.; Dong, Z.; Huang, Z.; and Zhang, G. 2025. CoProSketch: Controllable and Progressive Sketch Generation with Diffusion Model. In *Proceedings of the 33rd ACM International Conference on Multimedia*, 10142–10151.

Zhang, X.; Liu, Z.; Liu, J.; Zhang, Y.; and Li, X. 2026. SketchAny: A Text-Guided Diffusion Model for Vector-Based Scene Sketch Generation. *Available at SSRN 6138582*.