CoSketcher: Collaborative and Iterative Sketch Generation with LLMs under Linguistic and Spatial Control

Liwen Mei[1†], Manhao Guan[1†], Yifan Zheng[1] and Dongliang Zhang[1‡]

[1]College of Computer Science and Technology, Zhejiang University, China

Abstract

Sketching serves as both a medium for visualizing ideas and a process for creative iteration. While early neural sketch generation methods rely on category-specific data and lack generalization and iteration capability, recent advances in Large Language Models (LLMs) have opened new possibilities for more flexible and semantically guided sketching. In this work, we present CoSketcher, a controllable and iterative sketch generation system that leverages the prior knowledge and textual reasoning abilities of LLMs to align with the creative iteration process of human sketching. CoSketcher introduces a novel XML-style sketch language that represents stroke-level information in structured format, enabling the LLM to plan and generate complex sketches under both linguistic and spatial control. The system supports visual appealing sketch construction, including skeleton-contour decomposition for volumetric shapes and layout-aware reasoning for object relationships. Through extensive evaluation, we demonstrate that our method generates expressive sketches across both in-distribution and out-of-distribution categories, while also supporting scene-level composition and controllable iteration. Our method establishes a new paradigm for controllable sketch generation using off-the-shelf LLMs, with broad implications for creative human-AI collaboration.

CCS Concepts

• ***Human-centered computing*** → ***Human computer interaction (HCI);***

1. Introduction

Sketching is a medium of ideas visualization and a process of creative iteration [Jon02, KS17]. From an early age, humans develop the ability to draw with pen or pencil, expressing their visual understanding of the world through strokes on paper [EHA12]. Artists, designers, and engineers often rely on sketching during the concept design phase to externalize, evaluate, and refine their ideas [LS24, MWLZ22] (see Figure 1). Through multiple rounds of addition, revision, and adjustment, complex sketches gradually take shape and better convey the creator's intent [KAP*22, LKM*25]. However, crafting sketches that effectively convey creative intent requires a certain level of skill. In principle, artificial intelligence systems capable of assisting in sketching can reduce users' cognitive load related to visual representation, allowing them to focus more on idea exploration [EHoHC*23].

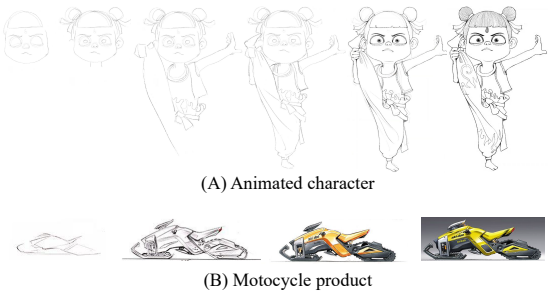
Early research on automatic sketch generation has mainly focused on data-driven deep learning methods [HE17, GGZP20, BDM*20, BKC*22]. These approaches typically generate object-level sketches in a single round, based on category labels. As a result, these data-driven methods typically face two major challenges: **(1) Limited generalization due to dataset dependency**. The generation quality and diversity are strongly constrained by the categories, abstraction levels, and styles present in the training data [GGZP20, GSH*19]. Consequently, these systems often struggle to handle unseen categories or generate sketches that differ significantly from training distributions. **(2) Lack of support for iteration in sketch generation**. These system are designed to produce sketches in one round and do not support incremental sketching or step-wise iteration. This contradicts the natural sketching behavior of humans, which involves visual reasoning, repeated adjustments, and continuous modification over time.

To overcome the generalization limitations of conventional data-driven methods, recent efforts have introduced large pretrained models with strong generalization abilities into the sketch generation process. These methods can be broadly categorized based on their output granularity: pixel-level generation and stroke-level generation. Pixel-level methods [RPG*21, SCS*22] excel at generating photorealistic images from text prompts, while stroke-level methods [XWZ*23] generate strokes represented as sequences of coordinate points. Although stroke-level sketches can be easily rasterized into pixel images, the inverse is not possible as pixel-level sketches lack sequential and structural information that is inherent to stroke-based representations [HE17]. Therefore, despite their

† Equal Contribution.

‡ Corresponding Author.

DOI: 10.2312/pg.20251298



(A) Animated character

(B) Motocycle product

Figure 1: *Examples of iterative concept sketching as a key process to externalize ideas and refine designs. Image sources: (A),(B).*

visual appeal, pixel-level approaches are less suitable for many sketch-based applications due to limited interpretability, incompatibility with stroke-based editing, and inefficiency in generation. For these reasons, we focus on stroke-level sketch generation. Research targeting at stroke-level generation has explored combining generation models with differentiable rendering to optimize parameterized curves on canvas [XWZ*23, WDQ*23]. These methods can produce visually plausible results beyond predefined categories or styles, thereby enhancing visual realism. However, their optimization processes often ignore the temporal and semantic dependencies across strokes, which are fundamental to human sketching. As a result, the generated outputs lack interpretability and diverge from the natural process of sequential, stroke-by-stroke construction [LKM*25]. This is intrinsically linked to the second challenge, the lack of support for iterative sketch generation.

To address the second challenge related to iteration, some studies have utilized sketch datasets in combination with traditional graphics techniques [LZC11] or generative adversarial networks [HXFM23, HPH*22, GZD*19, CCSY19] to provide intelligent drawing assistance. More recent several studies have integrated large language models such as T5 [RSR*20], and GPT-2 [RWC*19], enabling users to compose multi-object sketches using natural language instructions [HSHC20, VSZ*24]. The incorporation of multiple-round dialogues with these models introduces the possibility of supporting iterative interactions in sketch generation systems. However, due to the overly simplistic design of stroke grammars and interaction protocols, these methods often fail to effectively capture spatial relationships between objects or strokes, and are unable to implement accurate and precise refinement through language guidance alone. Consequently, this gives rise to another key challenge: (3) **Insufficient controllability for iterative sketch refinement**.

In this work, we present CoSketcher, a controllable stroke-level sketch generation and iterative editing system built on an off-the-shelf LLM. Leveraging its broad prior knowledge, the LLM interpret diverse sketch concepts, from abstract to concrete, from simple to complex. Moreover, the multi-round conversational nature of LLMs aligns naturally with the iterative creative process of human sketching. To support complex human-AI collaborative sketching and overcome the lack of fine-grained control, we design a customized XML-style sketch language that organizes stroke-level information into structured, interpretable textual representations. This structured format not only improves semantic clarity but also supports dual-mode control through both natural language and spatial handles. For example, when given a sketching task, the LLM first performs Chain-of-Thought (CoT) planning [WWS*22], which includes decomposing the task, ordering strokes, constructing high-level layouts, and designing low-level details. Based on this plan and the defined sketch language, the LLM generates each stroke as a semantic unit, defined by sampled keypoints and automatically labeled with compositional concepts. To enhance visual expressiveness and support controllable editing handles, strokes depicting volumetric structures such as limbs or pillars are first represented by skeletons, upon which corresponding contours are subsequently generated to form the final strokes. Once all strokes are produced, layout-level information is aggregated to explicitly infers the semantic relationships across strokes. The layout reasoning expressing in textual form can further enhance LLM's spatial understanding and inference capabilities.

Our sketch language greatly enhances the LLM's ability to generate detailed, structured, and expressive sketches, as demonstrated in our comparison of various methods for object-level sketch generation across both standard and novel categories beyond existing datasets. Moreover, for scene-level composition and iterative editing, the CoSketcher system outperforms in sketch-based image retrieval tasks by integrating natural language prompts with optional spatial handles. These comprehensive inputs enable the LLM to perform targeted edits with both spatial awareness and contextual understanding. To our knowledge, the CoSketcher is the first system to support human-like, controllable, and iterative stroke-level sketch generation using LLMs, offering a new paradigm for applying general AI systems in creative human-computer interaction.

2. Related Work

Sketch Generation Early approaches [Can86, NH05] relied on handcrafted edge detection operators to convert images into contour-like representations, leveraging edge maps as a proxy for human sketches. With the development of neural networks, data-driven methods became dominant, capturing both semantic and temporal patterns of human drawing [XHY*22]. The seminal model SketchRNN [HE17] simulates real-time sketching with RNNs. Various deep learning tools like GAN [WBZ21], Transformer [RBCP20, BKC*22], and Reinforcement Learning agents [LCL23, ZFW*18] are also introduced as backbones to enhance reasonable structures and fine details in generation. Despite these advances, the performance of data-driven methods remains constrained by the training data, including limitations in object categories, abstraction levels, and stylistic diversity. As a result, these models often fail to meet the demands of diverse sketch-based downstream tasks in general-purpose scenarios, such as sketch-based image retrieval [LLL*23], image generation [ERH*11], image editing [XF24, TFC*25], and 3D modeling [LLG*14]. For example, QuickDraw [HE17] the largest available sketch dataset with over 50M samples, includes only 345 object categories, most of which are simple and iconic with minimal shape detail. S. Ge et al. [GGZP20] collected 10,000 sketches through a crowd-sourced prompting strategy for enhancement in creativity. These efforts highlight that collecting sketches tailored for specific tasks remains time-consuming and labor-intensive.

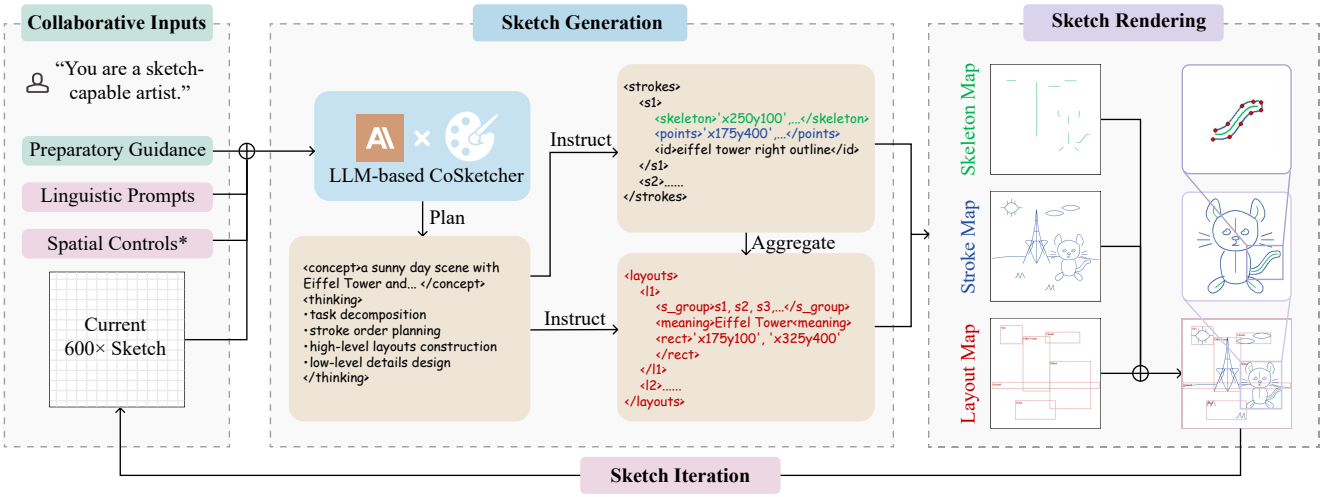


Figure 2: *Pipeline Overview. The LLM-based CoSketcher system receives various **collaborative inputs** and generate a XML-style string following the rules in a customized sketch language. For **sketch generation**, the LLM first interprets and plans the sketching task, then guides the generation of skeletons and strokes. Semantic relationships among strokes are aggregated into layout representations. For **sketch rendering**, our designed stroke representation enables rendering that is both friendly to the LLM and accessible to user. For **sketch iteration**, the resulting sketch can be reused as input for subsequent rounds of generation. (*) indicates the spatial control is optional.*

Recent works leverage large pretrained models for improved generalization. Pixel-level methods, such as Stable Diffusion [RBL*22, TFY*25, TJZ*25] and DALL-E [RPG*21], generate high-quality images but lack stroke structure and sequential information, making them less suitable for sketch-based applications. Stroke-level methods, in contrast, use LLMs to generate sketches as structured sequences of "visual sentences" from natural language prompts [ZWY*24]. While stroke-level sketches can be rasterized, pixel-level images cannot recover temporal or structural cues. Therefore, we focus on stroke-level generation and employ LLMs to model the sequential and semantic structure of sketches, enabling controllable and interpretable outputs.

Stroke Representation For stroke representation, traditional neural models often adopt a five-element format in which each point is encoded as $(\Delta x, \Delta y, p_1, p_2, p_3)$ [HE17], where the first two terms denote relative displacement and the latter three represent pen states as a one-hot vector. Although compact and suitable for sequence modeling, this format lacks spatial flexibility and semantic clarity, limiting its utility in fine-grained control and high-level reasoning. Neural representations have recently gained attention for their strong capabilities in feature extraction and compression. SketchBERT [LFXJ20] learns bidirectional representations using transformer-based encoders trained via self-supervised objectives, enabling the model to capture global structure across the entire sketch sequence. SketchFormer [RBCP20] extends this idea by encoding sketches as vector inputs into transformer layers to model structural dependencies among parts. SketchINR [BBC*24] takes a different route by modeling sketches as implicit neural functions, compressing the entire drawing into a compact latent code that supports smooth interpolation and abstraction-level control. However, such implicit and compressed formats are not well suited for LLMs, which excel at parsing explicit and structured textual inputs. LLMs benefit more from representations that are symbolic, segmented, and text-based structures that support explicit reasoning and token-level manipulation. Among structured formats, SVG is one of the most accessible to LLMs due to its widespread presence in web-based vector data. The SVG format, commonly found in the prior knowledge of LLMs, provides an expressive syntax for vector graphics, supporting primitives like <line>, <rect>, and <ellipse>, as well as <path> for Bézier curves. Stroke geometry is specified via attributes such as coordinates and control points, and simple semantic annotations can be included as comments. However, SVG was designed for colored graphics and visual styling, not for stroke-based sketching. Its verbosity and rendering-oriented design introduce ambiguity and cognitive overhead when used for semantic sketch generation with LLMs.

Thus, we propose a novel XML-style sketch language tailored for off-the-shelf LLMs. Our representation encodes strokes as structured textual sequences with both geometric attributes and semantic labels, balancing expressiveness and simplicity.

Multimodal Collaborative Control In generative tasks, user control has become increasingly fine-grained and personalized. Research has progressed from simple prompts to multimodal conditioning, enabling more precise and user-friendly interactions. For example, in image generation, ControlNet [ZRA23] adds spatial constraints through edge maps, depth maps, or style references. In 3D scene generation, bounding boxes and text prompts are used to control semantic regions and ensure smooth transitions [PW24]. These advances improve control accuracy and expand the potential of human-AI collaboration. In contrast, control in sequential sketch generation remains limited. Early models rely on autoregressive decoding [HE17] with weak conditioning on categories or partial sketches, lacking fine control over user intent. Later works like CLIPasso [VPB*22] and SketchINR [BBC*24] intro-

duce abstraction control or implicit representations, but are still constrained by the training dataset rather than real-time user input. Scones [HSHC20] explores interactive sketching by using GPT-2 to parse text instructions and mask-based models to control object layout and strokes, yet its generation space remains narrow. While LLMs have enabled sketch generation via code-like formats such as SVG or JSON, their visual reasoning is limited, and language prompts alone often lack precision. To address this, we regard the skeletons and layouts as explicit spatial handles to guide LLMs. These help bridge the gap between textual intent and visual structure, enhancing controllability through in-context reasoning.

3. Method

In this section, we present our method for controllable sketch generation through collaboration with an LLM, guided by a customized sketch language and enriched with both linguistic and spatial controls. The overall pipeline is illustrated in Figure 2. The backbone of our CoSketcher system is an off-the-shelf, frozen LLM, which interacts with users through four types of collaborative inputs: **(1) preparatory guidance**, including an introduction to the sketch language and dynamic instructions specifying interaction requirements (e.g., adding, moving, deleting, scaling, or editing); **(2) linguistic prompts** and **(3) spatial controls**, which jointly express user intent in textual and structural forms; **(4) the current sketch**, encoded in sketch language for the LLM and rendered as a visual canvas for user interaction, supporting iterative co-creation. Upon receiving these inputs, the LLM interprets the sketching task and formulates a generation plan, which guides the production of skeletons and strokes with semantic labels. The system then aggregates strokes based on semantic relationships to construct layout information. The resulting sketch is rendered using a structured stroke format that ensures both LLM compatibility and visual accessibility, and can be seamlessly reused as input for subsequent rounds of collaborative generation. Additionally, serving as human-in-the-loop interactive handles, the skeletons and layouts enables users to perform fine-grained, spatially precise editing through the LLM.

3.1. Stroke Representation

To represent detailed vectorized strokes, a stroke-level sketch $\mathbf{S}$ is typically modeled as an ordered sequence of m strokes $\mathbf{S} = \{S_i\}_{i=0}^{m}$, where each stroke S_i is a sequence of $n_i + 1$ point coordinates: $S_i = \{p_j\}_{j=0}^{n_i} = \{(x_j, y_j)\}_{j=0}^{n_i}$. Compared to pixel-level sketches, stroke-level vector graphics offer greater editability and tolerance, as their structured format enables the explicit integration of geometric, semantic, and other fine-grained information for each stroke. To visually render these strokes, it is necessary to convert the discrete point sequences into continuous visual forms. A straightforward method is to connect adjacent points with polylines. However, if the points are too sparsely distributed, polylines often result in rigid and jagged appearances that fail to capture the smooth and natural qualities of hand-drawn sketches. Conversely, recording a dense sequence of points imposes a substantial storage burden. Therefore, an effective stroke representation approach tends to balance compact storage with high-fidelity rendering.

To achieve this, we propose a new stroke representation approach. During storage, we retain only the keypoints at notable turning points (i.e., the red dots in Figure 3) to preserve the overall geometry structure of the stroke. During rendering, we apply cubic spline interpolation [ML98] to generate interpolation points (i.e., the yellow dots in Figure 3) to reconstruct a smooth, high-fidelity parametric stroke that passes through all stored keypoints.

Given a sequence of keypoints $\{(x_j, y_j)\}_{j=0}^{n_i}$, our goal is to draw a curve that passes through all keypoints. To achieve this, we divide keypoints into n_i segments and construct each child curve as a 2D parametric curve. Specially, for each child curve, an intermediate variable r is introduced to control the coordinates x and y, which are defined as follows:

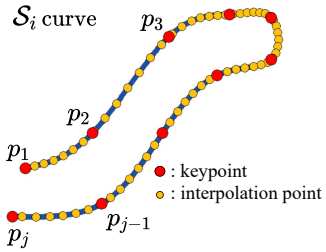


Figure 3: *Illustration of our stroke representation approach.*

$$\begin{cases} \mathbf{x}_j(r) = a_j^{(x)}(r-r_j)^3 + b_j^{(x)}(r-r_j)^2 + c_j^{(x)}(r-r_j) + d_j^{(x)} \\ \mathbf{y}_j(r) = a_j^{(y)}(r-r_j)^3 + b_j^{(y)}(r-r_j)^2 + c_j^{(y)}(r-r_j) + d_j^{(y)}, \end{cases} \tag{1}$$

where $\{a_j^{(\cdot)}, b_j^{(\cdot)}, c_j^{(\cdot)}, d_j^{(\cdot)}\}$ are the coefficients of the cubic spline function. To ensure the smoothness of the global curve, four spline constraints must be satisfied: (1) interpolation between keypoints: $\mathbf{x}(r_j) = x_j, \mathbf{x}(r_{j+1}) = x_{j+1}$; (2) first-order continuity: $\mathbf{x}'_j(r_{j+1}) = \mathbf{x}'_{j+1}(r_{j+1})$; (3) second-order continuity: $\mathbf{x}''_j(r_{j+1}) = \mathbf{x}''_{j+1}(r_{j+1})$; and (4) natural boundary conditions: $\mathbf{x}''(r_0) = 0, \mathbf{x}''(r_n) = 0$. The same set of constraints is applied independently to $y(r)$. We denote r as the relative position along the global curve and assume that the coordinates of each keypoint are evenly distributed on the r axis:

$$r_j = \frac{j}{n}, j = 0, 1, ..., n. \tag{2}$$

Based on the above constraints, we obtain two independent tridiagonal systems of equations with the coefficients of the cubic spline functions as unknowns, and employ Thomas algorithm [Lee11] to solve them. The resulting coefficients define the parametric form of each child curve.

3.2. Sketch Language

With the stroke representation approach established, we position LLM as a sketch-capable artist and provide it with a preparatory guidance to ensure it fully understands our custom-designed sketch language.

The preparatory guidance is mainly about the sketch language for its XML-style format and rules. This language defines how strokes, keypoints, and semantic attributes are organized to support controllable generation and semantic interpretation. A full answer generated from LLM about a vector sketch consist of four XML tags, encompassing the ideation process (<concept> and <thinking>), the stroke representation of the sketch (<strokes>), and the layout summary (<layouts>). To illustrate the meaning of each tag more clearly, we present an example of "Pikachu" described in our customized sketch language in Figure 4, in the top left of which each stroke is rendered in a distinct color.

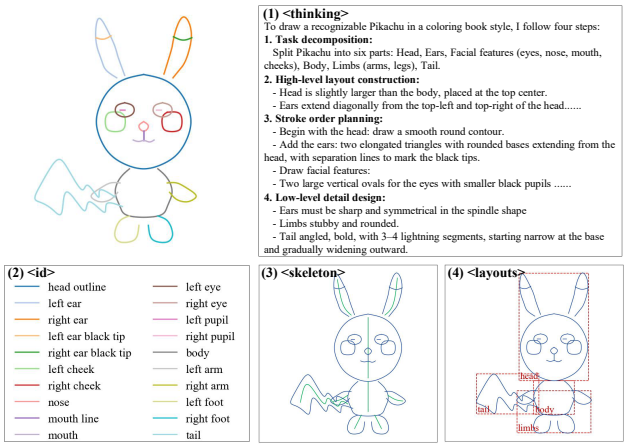


Figure 4: *An example of "Pikachu"described in custom-designed sketch language generated by the CoSketcher system. (1) A selected excerpt from the <thinking> tag illustrates four progressive steps: task decomposition, high-level layout construction, stroke order planning, and low-level detail design. (2) The <id> tag is visualized as a legend, showing the semantic label automatically assigned by the LLM to each stroke in different colors. (3) In the <skeleton> tag, green lines indicate optional skeletal structures that guide stroke shapes. (4) In the <layouts> tag, red bounding boxes represent aggregated layout elements, each annotated with a semantic label at the bottom-right corner of the box.*

- **<concept>** This tag is tasked to summarizes the user's drawing intent expressed in natural language and specifies the depicted concept in a concise and precise manner.
- **<thinking>** This tag represents a coarse-to-fine planning process, inspired by the Chain-of-Thought prompting strategy [CZS*24]. It includes task decomposition, high-level layout construction, stroke order planning, and low-level detail design (see Figure 4(1)).
- **<strokes>** This tag forms the core of the sketch language, which includes structured sketch information organized in stroke order. Each stroke S_i is stored as its own sub-tag <si>. Within each <si>, there are three key components: (1) <skeleton>: a list of coordinate strings such as "x100y350", defining the path of skeleton that optionally guides the contour in <points> (see the green lines in Figure 4(3)); (2) <points>: a list of keypoint coordinates in the same format as <skeleton>, representing the stroke's path. These keypoints correspond to salient turning points along the stroke and are used to reconstruct the final curve via cubic spline interpolation, as described in Section 3.1; (3) <id>: a short semantic identifier that indicates which part of the sketch the stroke depicts. As shown in the legend of Figure 4(2), strokes with different colors represent different semantic parts.
- **<layouts>** This tag aggregates related strokes into layout elements based on the <strokes> content, explicitly expressing strokes' relationships. The Layouts have flexible, prompt-controlled granularity, and each element <li> includes three components: (1) <s_group>: a list of stroke identifiers that belong to the group; (2) <meaning>: a semantic label specifying the functional role of the group (see the red labels in Figure 4(4)); (3) <rect>: the minimum bounding box that defines the spatial extent of the group (see the red dashed rectangles in Figure 4(4)).

Furthermore, basic stroke primitives, such as lines, triangles and rectangles, together with their corresponding representations, are provided to deepen the LLM's understanding of the relationship between the rendered cubic spline curves and the specified <points>. To ensure LLM's output can be directly parsed by our CoSketcher system, the preparatory guidance also offer two complete sketch examples and explicitly require the LLM to produce outputs in the specified structured format.

3.3. Controllable Generation and Editing

To address the ambiguity of natural language in fine-grained spatial control, we introduce two spatial control mechanisms based on <skeleton> and <layouts>. These mechanisms enable fine-grained and interpretable sketch control, and serve as operational handles to the structured tags defined in the sketch language.

The skeleton-based mechanism serves as a stroke-level control interface. The <skeleton> tag captures a stroke's spatial semantics including its direction, curvature, and volumetric structure, which go beyond what can be inferred from abstract symbolic prompts. Unlike simplified stick-figure or iconic sketching, the skeleton allows the LLM to generate detailed, shape-driven strokes with a stronger sense of form and depth. More importantly, the skeleton acts as a controllable handle for stroke editing. When users modify the skeleton rendered on the canvas, the CoSketcher system will note down the change in sketch language format and emphasize its meaning in the user prompt, delegating the low-level reconstruction of the <points> of current and relevant strokes to the LLM. This allows efficient, targeted shape control without requiring users to manually drag each keypoint in <points>. The concrete examples of contextual operations can be found in Section 4.4.

The layout-based mechanism, in contrast, operates at the object level. Users can control object-level behavior like repositioning and scaling by editing the layout attributes. For example, moving the minimal bounding box of a layout group in <rect> tag leads the LLM to reposition the associated strokes while preserving their relative arrangement and semantics. This mechanism provides a lightweight yet expressive means to control the global structure of the sketch, while ensuring that adjustments to a layout element affect only its associated strokes, without impacting strokes of other objects within the same bounding box.

Given the strong language capabilities of LLMs, the CoSketcher system imposes no rigid language constraints on user's inputs. Common sketching functionalities, such as stroke generation, incremental sketching, object repositioning and scaling, and detail refining, can all be executed via natural language prompts and explicit control handles. However, as sketch complexity increases, the number of strokes on the canvas can easily exceed 30, making it inefficient for the LLM to regenerate the entire sketch when only local modifications are required. To tackle this issue, the CoSketcher system introduces two additional output modes beyond the default full output: partial incremental output and partial update output. All modes follow the XML-style sketch language defined in Section 3.2. In the incremental or update modes, only the strokes or layout

elements that need to be added or modified are wrapped in <increment> or <update> tags, respectively. The indices of stroke (<si>) and layout (<li>) sub-tags must remain consistent with their counterparts on the canvas. These output rules can be dynamically specified within the user prompt following the natural language instruction. Experiments show that this strategy significantly enhances the system's ability to handle complex sketches, while mitigating limitations imposed by LLM output token constraints.

3.4. Collaborative Sketching

To support human-in-the-loop sketch generation, the CoSketcher system allows users to draw directly on the canvas, as well as interacting through natural language and explicit spatial controls introduced in Section 3.3. Since the hand-drawn strokes are dense polylines composed of raw point coordinates, they cannot be directly interpreted by the LLM. To bridge this gap, the CoSketcher system combines geometric processing techniques with the reasoning capabilities of LLMs to transform freehand stroke into structured format defined by our sketch language, consisting of <skeleton>, <points>, and <id>. To invert the sketch interpolation process, we apply the RDP algorithm [DP73] to extract a sparse <points> sequence suitable for cubic spline reconstruction.

Given a raw stroke $C_i = \{c_j\}_{j=0}^{k_i}$, we preserve its shape by recursively selecting a representative subset of ordered points S_i, controlled by an adaptive distance tolerance parameter ϵ. The resulting sequence is defined as:

$$S_i = \{c_{j'}\}_{j'=0}^{q_i} \subseteq C_i,\ q_i \ll k_i. \tag{3}$$

We first compute the diagonal length l_d of the minimum bounding rectangle of C_i, and the total length l_t of all segments in C_i. An adaptive distance tolerance parameter is then defined as:

$$\epsilon = \frac{l_d}{l_t} \cdot \epsilon_0, \tag{4}$$

where ϵ_0 is a hyperparameter denoting the maximum distance tolerance. The algorithm connects endpoints c_0 and c_{k_i} with a line segment L, then finds the intermediate point c_{j^*} with the maximum perpendicular distance d_{max}. If $d_{max} > \epsilon$, the segment is split at c_{j^*} and processed recursively. Otherwise, the segment is approximated by the straight line. Thus, dense points from freehand are converted into sparse keypoint sequence <points> by aggregating all retained points in order.

The extracted <points> are paired with an empty <skeleton> and a default <id> (e.g., stroke_i) to form a stroke element. The LLM then completes the <skeleton> based on the concept and inferred shape, and assigns the semantic <id>. This workflow enhances interpretability and significantly reduces manual annotation effort.

4. Experiments

The CoSketcher system supports various commercial LLMs as its backbone via API access to enable controllable sketch generation. For fairness, all methods involving LLMs in our experiments use Claude-Sonnet-3.7 [Ant25] as the backbone model. The canvas size is set to 600×600, and the distance tolerance hyperparameter for keypoint extraction is configured as $\epsilon_0 = 4.0$.

4.1. Object-Level Sketch Evaluation

In prior sketch datasets, both human-drawn sketches and those produced by earlier data-driven methods are typically constrained to a limited number of categories and costly to extend. To systematically evaluate the semantic expressiveness of generated sketches under both concepts included and exclude the standard category of QuickDraw dataset [HE17], we design two experimental settings:

- **ID Category Group**: Following prior work [VACOS23, XWZ*23], we randomly sample 50 In-Distribution(ID) categories from the 345 QuickDraw classes. For each category, 10 raster images are rendered from sketches generated by various methods, totaling 500 test sketches.
- **OOD Category Group**: To test generalization beyond regular distributions, we curate another set of 50 Out-of-Distribution (OOD) categories that fall outside the predefined category space. These include widely recognizable IP characters (e.g., *Mickey Mouse, Doraemon*), mechanical components(e.g., *screw, gear*), and well-known landmarks (e.g., *Big Ben, Golden Gate Bridge*).

In both settings of this experiment, each generation method is executed in a single, non-interactive round to produce sketches for these categories without refinement or user feedback.

For evaluation, we adopt a CLIP zero-shot classifier [RKH*21] to assess the semantic alignment between each generated sketch and the textual label of its target category. Specifically, we report both Top-1 and Top-5 scores under two metrics:

- **Accuracy(Acc.)**: the proportion of test sketches (out of 500) whose ground-truth category appears in the Top-1 or Top-5 predicted categories ranked by CLIP recognition scores.
- **Prediction(Pred.)**: the average CLIP score assigned to the GT category across all test samples, with zero score if the GT label is absent from the predictions.

Performance is compared across several generation approaches. SketchRNN [HE17] is a data-driven method trained on filtered human-drawn sketches with separate model parameters for each category. Due to its dependence on category-specific training data, SketchRNN is evaluate only on the ID category group. In contrast, LLM-based methods, such as SketchAgent [VSZ*24], and our proposed CoSketcher, are capable of generating sketches for both ID and OOD categories without additional training. To showcase the inherent generation capability of LLMs, we also demonstrate the SVG results from Claude-Sonnet-3.7, prompted with: "*Write a SVG string that draws a sketch of a <concept>. Use only black and clean strokes*", which we refer to as "LLM-SVG" throughout the paper. As a baseline of this experiment, human-drawn sketches are evaluated in both ID and OOD category settings.

Quantitative results are shown in Table 1. For the ID group only , SketchRNN performs best due to its supervised training on category-specific data, though its applicability is limited to predefined classes and does not extend to unseen concepts. However, as shown in the rightmost column of Table 1, data-driven methods require substantial training time and large amounts of annotated data for each new object category. Moreover, each model of SketchRNN is restricted to a single class, resulting in pool generation and limited practical utility. In contrast, although LLM-based

Table 1: *Comparison of sketch recognition performance across methods. The ID category group refers to in-distribution categories from the 345 standard categories in the QuickDraw dataset, while the OOD category group denotes out-of-distribution categories that fall outside the predefined categories.*

Method	ID Category Group				OOD Category				Runtime	
	Top-1 Acc.↑	Top-5 Acc.↑	Top-1 Pred.↑	Top-5 Pred.↑	Top-1 Acc.↑	Top-5 Acc.↑	Top-1 Pred.↑	Top-5 Pred.↑	Training	Inference
Data-driven sketch generation method limited in ID categories.										
SketchRNN	0.38	0.60	0.27	0.32	-	-	-	-	2h+	<1s
LLM-based sketch generation methods with zero-shot capability.										
LLM-SVG	0.32	0.53	0.18	0.20	0.29	**0.52**	0.15	0.19	-	12s
SketchAgent	0.26	0.45	0.11	0.13	0.21	0.41	0.09	0.11	-	15s
CoSketcher	**0.35**	**0.56**	**0.19**	**0.22**	**0.33**	**0.52**	**0.18**	**0.20**	-	18s
Human	0.36	0.57	0.23	0.26	0.30	0.53	0.19	0.20	-	20s

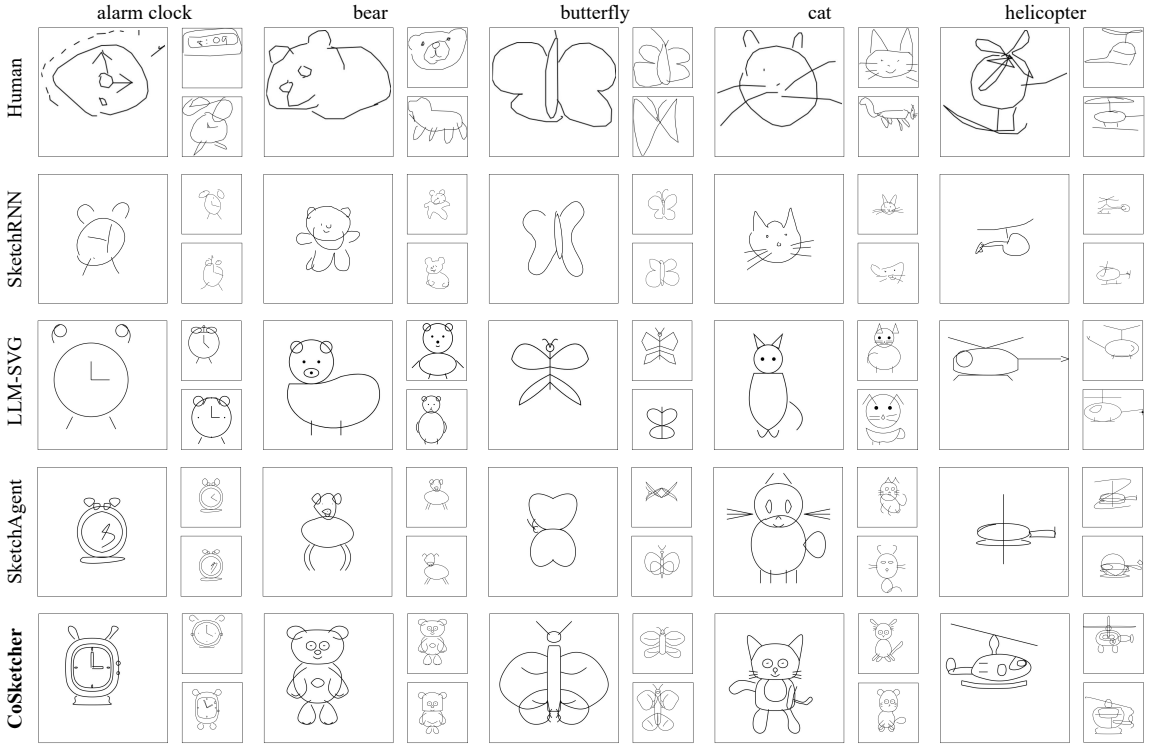


Figure 5: *Randomly selected sketches from the ID category group. (Three parallel subfigures with one enlarged for closer inspection)*

methods require longer inference time, they offer broader generalization capabilities and eliminate the need for additional training, as they do not rely on paired training data or handcrafted labels. Among these, CoSketcher consistently shows superior performance across both ID and OOD categories. While LLM-SVG yield reasonably high scores, visual inspection (Figure 5, 6) shows rigid and uniform shapes, lacking freehand variation. By leveraging a interpretable sketch language with intermediate control mechanisms such as skeleton and layout, CoSketcher generates sketches with clearer semantics and geometry, leading to stronger alignment with textual labels and improved CLIP recognition. Overall, CoSketcher delivers robust generalization and high-quality results across both standard and novel categories, confirming its effectiveness as a unified, controllable sketch generation framework.

4.2. Scene-Level Sketch Evaluation

Although CLIP's image encoder (ViT) is not specifically designed for sparse and stroke-based sketches, it is widely used as a general semantic evaluator. Following this convention, we employ a CLIP-based zero-shot classifier in Section 4.1 to assess object-level sketches. In contrast, for scene-level sketches involving multiple objects and more strokes, we adopt the zero-shot sketch-based image retrieval task to evaluate scene-level sketches involving multiple objects and a relatively large number of strokes. We introduce the ZSE-SBIR model [LLL*23] specifically tailored to sparse sketch inputs with a learnable tokenizer. Compared to CLIP, this tokenizer optimized for sparse and abstract sketches is more sensitive to subtle variations in strokes and details, making it better suited for evaluating the expressive richness of complex sketches. Retrieval accuracy thus serves as a stronger proxy for whether gen-

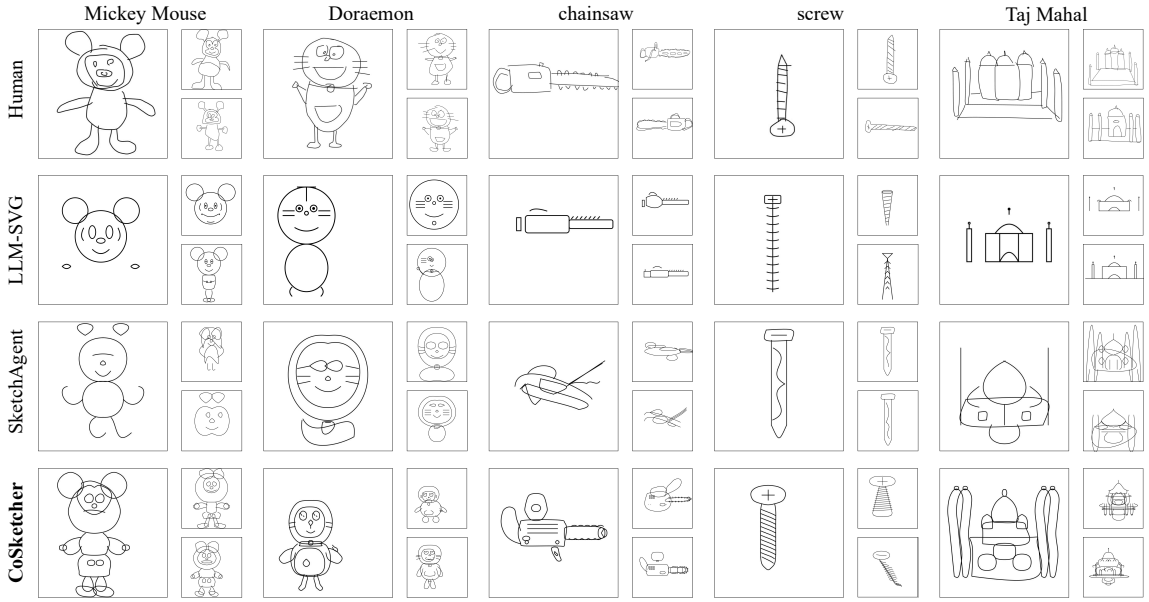


Figure 6: *Randomly selected sketches from the OOD category group. (Three parallel subfigures with one enlarged for closer inspection)*

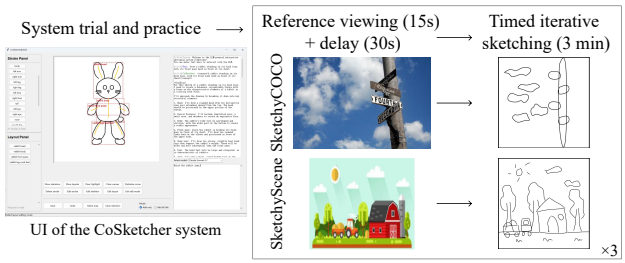


Figure 7: *Sketch collection procedure. Participants observe a reference image, undergo a memory delay, and complete a timed sketch using one of the tested systems. Additionally, the leftmost image shows the User Interface of our system.*

erated sketches capture user intent and scene semantics. We include human-drawn sketches and evaluate three interactive systems: Scones [HSHC20], SketchAgent [VSZ*24], and our CoSketcher. As in Section 4.1, an LLM-SVG group is added to illustrate LLMs' inherent sketch generation ability via direct SVG outputs.

The sketch collection procedure is shown in Figure 7. Participants first practiced briefly, then viewed a reference image for 15 seconds, followed by a 30-second delay with no image. This view-delay design simulates the early stage of creative ideation, where ideas are initially vague and gradually take shape during the sketching process. They then completed the sketch within 3 minutes using the assigned system. To resolve bias by task familiarity, we use a Latin square design to balance the influence from the presentation order of methods across participants. Each participant is asked to complete six scene-level sketches per method, with references randomly selected from realistic-style SketchyCOCO dataset [GLX*20] and cartoon-style SketchyScene dataset [ZYD*18].

For quantitative evaluation, each collected sketch is used as the query input in the retrieval, with its original reference image serving as the ground truth. We randomly select 50 images from the each of the two datasets for the separate retrieval task, and adopt the Top-1 and Top-5 Accuracy, and the ZSE-SBIR Score [LLL*23] as evaluation metrics. Quantitative results are summarized in Table 2, and representative examples are shown in Figure 8. In both the SketchyCOCO and SketchyScene retrieval settings, sketches generated by our method achieve the best performance among other interactive systems and are closest to human-drawn sketches, demonstrating stronger semantic expressiveness at the scene level. Qualitative results in Figure 8 further show that our sketches exhibit greater visual detail and structural coherence, aligning more naturally with human-drawn examples. The Scones method suffers from poor generalization, largely due to the limited category space and data-driven dependency of its composition proposer and object generator. Examples show that Scones often fails to generate appropriate objects when the target category falls outside its predefined composition. For example, in *SketchyCOCO-1*(seen in Figure 8), the target object "Big Ben" is incorrectly rendered as unrelated objects such as "lightning" or "bouncy." Additionally, Scones lacks the ability to generate more than one instance per object category, which is a critical requirement for scene-level sketches. This limitation is evident in *SketchyCOCO-2* (five giraffes), *SketchyScene-1* (three birds), and *SketchyScene-2* (two men). While the LLM-SVG group achieves relatively competitive performance in terms of retrieval accuracy, the generated sketches reveal critical limitations. Specifically, the outputs are highly repetitive and lack the randomness and natural variation typical of human sketches, suggesting that its retrieval performance partially attributed to learned token regularities rather than a true understanding of sketch composition. The performance of SketchAgent also drops significantly in scene-level tasks. Although it maintains reasonable interpretability at the object level, its ability to handle layout and visual detail is notably weaker in complex settings. Furthermore, the coordinate-time-based parametric stroke representation used by SketchAgent

Table 2: *Comparison of sketch-based image retrieval performance for scene-level sketches generated by different methods.*

Method	SketchyCOCO			SketchyScene		
	Top-1 Acc.↑	Top-5 Acc.↑	ZSE-SBIR Score↑	Top-1 Acc.↑	Top-5 Acc.↑	ZSE-SBIR Score↑
Scones	0.02	0.12	0.73	0.03	0.10	0.38
LLM-SVG	0.22	0.43	0.84	0.17	0.32	0.60
SketchAgent	0.07	0.15	0.77	0.05	0.13	0.43
CoSketcher	**0.38**	**0.47**	**0.87**	**0.22**	**0.42**	**0.66**
Human	0.32	0.48	0.88	0.23	0.43	0.74

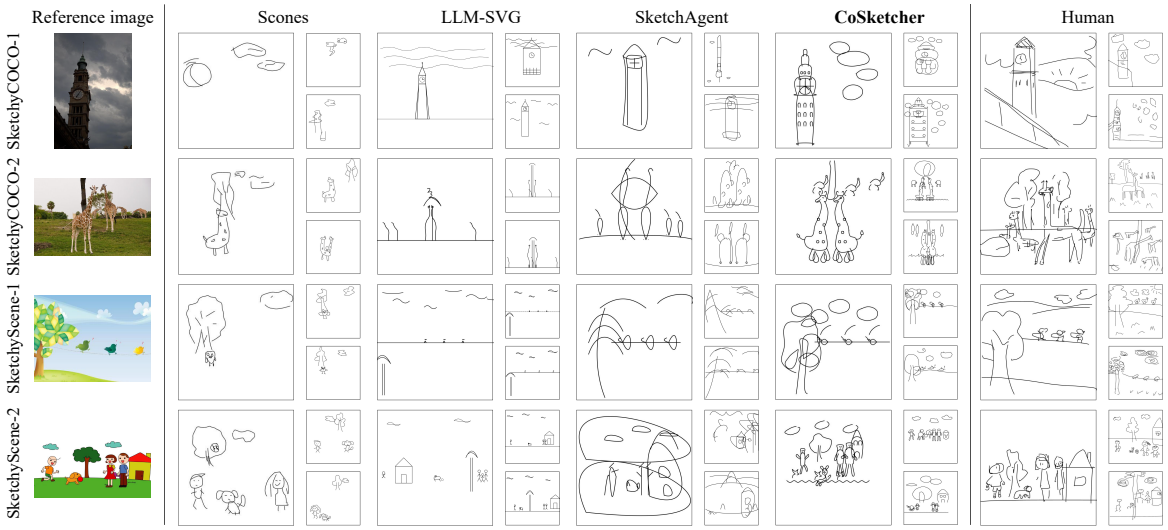


Figure 8: *Randomly selected sketches from the scene-based image retrieval experiment. (Three parallel subfigures with one enlarged for closer inspection)*

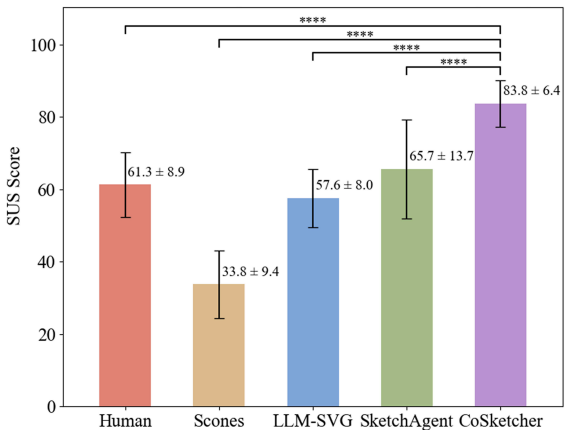


Figure 9: *Statistical comparison of SUS scores across systems.*

tends to produce overly smooth and homogeneous curves. As observed in the visualizations, these strokes often appear as simple arcs or loops, which may hinder the LLM's ability to capture precise object shapes and spatial relations.

4.3. User Study

In the scene sketch reproduction experiment, participants used each designated system to create sketches and then completed the SUS (System Usability Scale) questionnaire as a post-task evaluation (see Supplementary Material for the full questionnaire). The SUS evaluation results, presented as a bar chart in Figure 9, show that the proposed CoSketcher achieved the highest SUS score and significantly outperformed all other sketching tools, demonstrating clear advantages in usability and interaction experience.

Beyond the 10 standard SUS items assessing system usability, we added two questions on the system's impact on creativity; details and results are also provided in Supplementary Material.

Participants were also interviewed after experiments. Many

Table 3: *Ablation study. We systematically remove each component and replace multiple LLMs as the backbone in our pipeline.*

Component	ID Category Group			
	Top-1 Acc.↑	Top-5 Acc.↑	Top-1 Pred.↑	Top-5 Pred.↑
CoSketcher(full, Claude-Sonnet-3.7)	**0.35**	**0.56**	**0.19**	**0.22**
w/o <thinking>	0.33	0.52	0.16	0.18
w/o <skeleton>	0.30	0.49	0.17	0.21
w/o stroke representation examples	0.27	0.44	0.19	0.22
w/o sketch examples	0.27	0.43	0.10	0.14
CoSketcher(full, Claude-Sonnet-3.7)	**0.35**	0.56	**0.19**	**0.22**
CoSketcher(full, ChatGPT-4o)	0.34	**0.57**	**0.19**	**0.22**
CoSketcher(full, Gemini-2.5-flash)	0.33	0.50	0.17	0.15
CoSketcher(full, DeepSeek-R1)	0.25	0.42	0.11	0.15

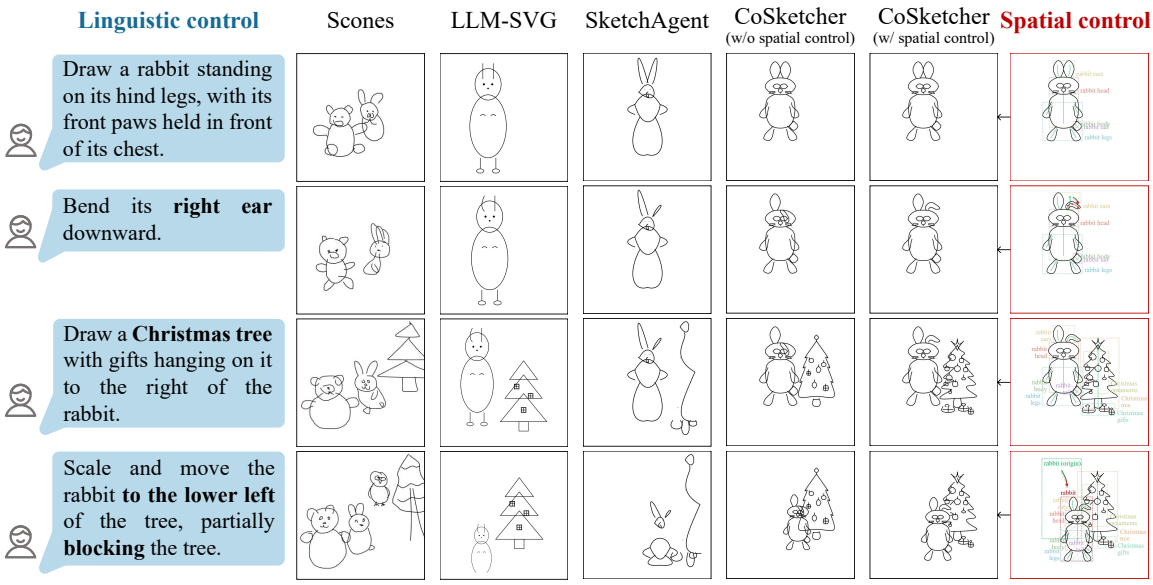


Figure 10: *Comparison of collaborative and iterative sketching by different methods.*

praised the CoSketcher's spatial control, noting its intuitive handling of a common LLM limitation, where linguistic analysis often fails to yield precise spatial outputs. One participant remarked, *"CoSketcher allowed me to focus more on composing imagined scenes rather than rendering visual details."* These responses suggests our system effectively supports creative thinking and ideation.

4.4. Collaborative Sketching

Furthermore, we demonstrate the CoSketcher's effectiveness in supporting interactive sketch iteration. The system generates complex, recognizable sketches with rich details from natural language prompts and spatial controls, both converted into textual inputs for the LLM. We focus on four representative types of interactive editing tasks: initial generation, incremental sketching, object repositioning, and fine-grained redrawing. For all methods under comparison, we use the same predefined textual instructions. In addition, the CoSketcher system includes two experimental conditions: with and without spatial control. The spatial control is implemented through skeletons and layouts, which are provided alongside linguistic inputs at the corresponding steps.

As shown in Figure 10, CoSketcher consistently generates more recognizable details than the others. Skeletons give strokes volumetric shapes, whereas LLM-SVG and SketchAgent produce abstract limb strokes with weaker appeal in the rabbit sketching task. Scones, on the other hand, misinterprets the instruction and generates unintended objects such as a bear and an owl. In the fine-grained redrawing task of *"bending the right ear downward"*, CoSketcher with spatial control performs a precise adjustment, while SketchAgent only shrinks the ear, LLM-SVG bends the wrong one, and CoSketcher without control renders it fully drooped. Regarding spatial understanding, most methods correctly respond to repositioning instructions like *"move to the lower left"*, while Scones fails to do so.

4.5. Ablation and Discussion

As discussed in Section 4.4, the spatial control mechanism plays a key role in supporting iterative interactions within the system. In the following, we further evaluate the contribution of the relevant components of our method to the quality of sketch generation. Specifically, we conduct an ablation study by systematically removing individual tags defined in Section 3.2 and measuring object-level ID category sketch recognition, as described in Section 4.1. We assess the effects of removing the <thinking> and <skeleton> tags from the sketch language, as well as omitting the stroke representation of basic primitives and the complete sketch examples included in the preparatory guidance.

As shown in upper section of Table 3, the full CoSketcher pipeline achieves the highest performance, underscoring the importance of each individual component. In particular, excluding the <thinking> and <skeleton> tags leads to a substantial drop in performance, highlighting their critical role in constructing detailed and coherent sketches. Additionally, removing the sketch examples from the preparatory guidance hinders the LLM's ability to understand the sketch language in the style of human-drawn sketches.

To further assess the generalizability of our method, we replace Claude-Sonnet-3.7 with a range of widely used off-the-shelf LLMs, including ChatGPT-4o, Gemini-2.5-flash, and DeepSeek-R1. As shown in the lower section of Table 3, our pipeline maintains robust performance across different model backbones, demonstrating strong adaptability. Though we observe that different LLMs exhibit varying levels of spatial reasoning and language grounding capabilities, our pipeline achieves consistently high performance when combined with most off-the-shelf LLMs. Although DeepSeek-R1 performs relatively worse, it still generates semantically coherent and structurally valid sketches. This confirms that our design is not tightly coupled with any specific LLM and demonstrates strong generalizability across different model backbones. These findings confirm that our method can adapt to a variety of LLMs with differ-

ent reasoning capacities, and also suggest that its sketch generation performance will continue to improve as more capable language models become available.

5. Limitations and Future Work

While CoSketcher shows strong generalization and controllability, there are still several limitations. First, compared to data-driven models trained on specific categories, our sketches may appear less detailed for in-distribution objects, prioritizing semantic flexibility over low-level polish. Second, the inference speed of CoSketcher is slower, primarily due to the response latency introduced by the use of commercial, off-the-shelf LLMs. We believe the response latency of off-the-shelf LLMs can be effectively reduced through standard acceleration techniques, such as low-bit quantization and pruning. In addition, switching to lightweight LLMs that support local deployment provides a feasible path toward faster inference. For failure cases, the XML-style sketch language generated by the LLM may contain parsing errors, such as invalid tag nesting or out-of-range coordinate values. These issues can often be mitigated through post-processing or corrected by prompting the LLM for self-repair. In future work, we aim to combine sketch-specific visual modules to improve drawing quality, explore lightweight LLM alternatives for faster inference, and enhance the system's error-handling mechanisms.

6. Conclusion

In this work, we introduce CoSketcher, a controllable and iterative sketch generation system that leverages off-the-shelf LLMs to address long-standing challenges in generalization, iteration, controllability, and automatic semantic annotation at stroke level. By designing a structured XML-style sketch language and providing detailed preparatory guidance, CoSketcher enables LLMs to plan, generate, and refine complex sketches under both linguistic and spatial controls. Our system supports detailed and expressive stroke generation, handles unseen categories, and facilitates complex sketching in a human-like iterative manner. These results highlight the potential of general-purpose LLMs as a powerful engine for creative, structured, and collaborative sketching, paving the way for more flexible and semantically aligned human-AI co-creation.

Acknowledgments

This work was supported by the National Key Research and Development Program of China [grant No.2022YFB3303100].

References

[Ant25] ANTHROPIC: Claude, 2025. Accessed: 2025-05-18. URL: https://www.anthropic.com/. 6

[BBC*24] BANDYOPADHYAY H., BHUNIA A. K., CHOWDHURY P. N., SAIN A., XIANG T., HOSPEDALES T., SONG Y.-Z.: Sketchinr: A first look into sketches as implicit neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), pp. 12565–12574. 3

[BDM*20] BHUNIA A. K., DAS A., MUHAMMAD U. R., YANG Y., HOSPEDALES T. M., XIANG T., GRYADITSKAYA Y., SONG Y.-Z.: Pixelor: A competitive sketching ai agent. so you think you can sketch? *ACM Transactions on Graphics (TOG) 39*, 6 (2020), 1–15. 1

[BKC*22] BHUNIA A. K., KHAN S., CHOLAKKAL H., ANWER R. M., KHAN F. S., LAAKSONEN J., FELSBERG M.: Doodleformer: Creative sketch drawing with transformers. In *European Conference on Computer Vision* (2022), Springer, pp. 338–355. 1, 2

[Can86] CANNY J.: A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, 6 (1986), 679–698. 2

[CCSY19] CHOI J., CHO H., SONG J., YOON S. M.: Sketchhelper: Real-time stroke guidance for freehand sketch retrieval. *IEEE Transactions on Multimedia 21*, 8 (2019), 2083–2092. 2

[CZS*24] CHEN Z., ZHOU Q., SHEN Y., HONG Y., SUN Z., GUTFREUND D., GAN C.: Visual chain-of-thought prompting for knowledge-based visual reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2024), vol. 38, pp. 1254–1262. 5

[DP73] DOUGLAS D. H., PEUCKER T. K.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization 10*, 2 (1973), 112–122. 6

[EHA12] EITZ M., HAYS J., ALEXA M.: How do humans sketch objects? *ACM Transactions on graphics (TOG) 31*, 4 (2012), 1–10. 1

[EHoHC*23] EPSTEIN Z., HERTZMANN A., OF HUMAN CREATIVITY I., AKTEN M., FARID H., FJELD J., FRANK M. R., GROH M., HERMAN L., LEACH N., ET AL.: Art and the science of generative ai. *Science 380*, 6650 (2023), 1110–1111. 1

[ERH*11] EITZ M., RICHTER R., HILDEBRAND K., BOUBEKEUR T., ALEXA M.: Photosketcher: interactive sketch-based image synthesis. *IEEE Computer Graphics and Applications 31*, 6 (2011), 56–66. 2

[GGZP20] GE S., GOSWAMI V., ZITNICK C. L., PARIKH D.: Creative sketch generation. *arXiv preprint arXiv:2011.10039* (2020). 1, 2

[GLX*20] GAO C., LIU Q., XU Q., WANG L., LIU J., ZOU C.: Sketchycoco: Image generation from freehand scene sketches. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), pp. 5174–5183. 8

[GSH*19] GRYADITSKAYA Y., SYPESTEYN M., HOFTIJZER J. W., PONT S. C., DURAND F., BOUSSEAU A.: Opensketch: a richly-annotated dataset of product design sketches. *ACM Trans. Graph. 38*, 6 (2019), 232–1. 1

[GZD*19] GHOSH A., ZHANG R., DOKANIA P. K., WANG O., EFROS A. A., TORR P. H., SHECHTMAN E.: Interactive sketch & fill: Multiclass sketch-to-image translation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 1171–1180. 2

[HE17] HA D., ECK D.: A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477* (2017). 1, 2, 3, 6

[HPH*22] HUANG Z., PENG Y., HIBINO T., ZHAO C., XIE H., FUKUSATO T., MIYATA K.: dualface: Two-stage drawing guidance for freehand portrait sketching. *Computational Visual Media 8*, 1 (2022), 63–77. 2

[HSHC20] HUANG F., SCHOOP E., HA D., CANNY J.: Scones: towards conversational authoring of sketches. In *Proceedings of the 25th International Conference on Intelligent User Interfaces* (2020), pp. 313–323. 2, 4, 8

[HXFM23] HUANG Z., XIE H., FUKUSATO T., MIYATA K.: Aniface-drawing: Anime portrait exploration during your sketching. In *ACM SIGGRAPH 2023 conference proceedings* (2023), pp. 1–11. 2

[Jon02] JONSON B.: Sketching now. *International Journal of Art & Design Education 21*, 3 (2002), 246–253. 1

[KAP*22] KO H.-K., AN S., PARK G., KIM S. K., KIM D., KIM B., JO J., SEO J.: We-toon: A communication support system between writers and artists in collaborative webtoon sketch revision. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology* (2022), pp. 1–14. 1

[KS17] KELLEY T. R., SUNG E.: Sketching by design: Teaching sketching to young learners. *International Journal of Technology and Design Education 27* (2017), 363–386. 1

[LCL23] LIN Z., CHEN Y., LIU Z.: Sketch rl: Interactive sketch generation for long-horizon tasks via vision-based skill predictor. *IEEE Robotics and Automation Letters 9*, 1 (2023), 867–874. 2

[Lee11] LEE W.: Tridiagonal matrices: Thomas algorithm. *MS6021, Scientific Computation, University of Limerick 1054* (2011). 4

[LFXJ20] LIN H., FU Y., XUE X., JIANG Y.-G.: Sketch-bert: Learning sketch bidirectional encoder representation from transformers by self-supervised learning of sketch gestalt. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 6758–6767. 3

[LKM*25] LIN D. C.-E., KANG H. B., MARTELARO N., KITTUR A., CHEN Y.-Y., HONG M. K.: Inkspire: Supporting design exploration with generative ai through analogical sketching. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (2025), pp. 1–18. 1, 2

[LLG*14] LI B., LU Y., GODIL A., SCHRECK T., BUSTOS B., FERREIRA A., FURUYA T., FONSECA M. J., JOHAN H., MATSUDA T., ET AL.: A comparison of methods for sketch-based 3d shape retrieval. *Computer Vision and Image Understanding 119* (2014), 57–80. 2

[LLL*23] LIN F., LI M., LI D., HOSPEDALES T., SONG Y.-Z., QI Y.: Zero-shot everything sketch-based image retrieval, and in explainable style. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2023), pp. 23349–23358. 2, 7, 8

[LS24] LEWIS M., STURDEE M.: The future of sketching. In *Sketching in Human Computer Interaction: A Practical Guide to Sketching Theory and Application*. Springer, 2024, pp. 263–278. 1

[LZC11] LEE Y. J., ZITNICK C. L., COHEN M. F.: Shadowdraw: real-time user guidance for freehand drawing. *ACM Transactions on Graphics (ToG) 30*, 4 (2011), 1–10. 2

[ML98] MCKINLEY S., LEVINE M.: Cubic spline interpolation. *College of the Redwoods 45*, 1 (1998), 1049–1060. 4

[MWLZ22] MA J., WANG J., LI J., ZHANG D.: Real-time skeletonization for sketch-based modeling. *Computers & Graphics 102* (2022), 56–66. 1

[NH05] NAIR V., HINTON G. E.: Inferring motor programs from images of handwritten digits. *Advances in neural information processing systems 18* (2005). 2

[PW24] PO R., WETZSTEIN G.: Compositional 3d scene generation using locally conditioned diffusion. In *2024 International Conference on 3D Vision (3DV)* (2024), IEEE, pp. 651–663. 3

[RBCP20] RIBEIRO L. S. F., BUI T., COLLOMOSSE J., PONTI M.: Sketchformer: Transformer-based representation for sketched structure. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), pp. 14153–14162. 2, 3

[RBL*22] ROMBACH R., BLATTMANN A., LORENZ D., ESSER P., OMMER B.: High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2022), pp. 10684–10695. 3

[RKH*21] RADFORD A., KIM J. W., HALLACY C., RAMESH A., GOH G., AGARWAL S., SASTRY G., ASKELL A., MISHKIN P., CLARK J., ET AL.: Learning transferable visual models from natural language supervision. In *International conference on machine learning* (2021), PmLR, pp. 8748–8763. 6

[RPG*21] RAMESH A., PAVLOV M., GOH G., GRAY S., VOSS C., RADFORD A., CHEN M., SUTSKEVER I.: Zero-shot text-to-image generation. In *International conference on machine learning* (2021), Pmlr, pp. 8821–8831. 1, 3

[RSR*20] RAFFEL C., SHAZEER N., ROBERTS A., LEE K., NARANG S., MATENA M., ZHOU Y., LI W., LIU P. J.: Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research 21*, 140 (2020), 1–67. 2

[RWC*19] RADFORD A., WU J., CHILD R., LUAN D., AMODEI D., SUTSKEVER I., ET AL.: Language models are unsupervised multitask learners. *OpenAI blog 1*, 8 (2019), 9. 2

[SCS*22] SAHARIA C., CHAN W., SAXENA S., LI L., WHANG J., DENTON E. L., GHASEMIPOUR K., GONTIJO LOPES R., KARAGOL AYAN B., SALIMANS T., ET AL.: Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems 35* (2022), 36479–36494. 1

[TFC*25] TU J., FENG Q., CHEN C., DONG J., ZHAO H., ZHANG C., QIAN H.: Ce-sdwv: Effective and efficient concept erasure for text-to-image diffusion models via a semantic-driven word vocabulary. *arXiv preprint arXiv:2501.15562* (2025). 2

[TFY*25] TU J., FU H., YANG F., ZHAO H., ZHANG C., QIAN H.: Texttoucher: Fine-grained text-to-touch generation. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2025), vol. 39, pp. 7455–7463. 3

[TJZ*25] TU J., JI W., ZHAO H., ZHANG C., ZIMMERMANN R., QIAN H.: Driveditfit: Fine-tuning diffusion transformers for autonomous driving data generation. *ACM Transactions on Multimedia Computing, Communications and Applications 21*, 3 (2025), 1–29. 3

[VACOS23] VINKER Y., ALALUF Y., COHEN-OR D., SHAMIR A.: Clipascene: Scene sketching with different types and levels of abstraction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), pp. 4146–4156. 6

[VPB*22] VINKER Y., PAJOUHESHGAR E., BO J. Y., BACHMANN R. C., BERMANO A. H., COHEN-OR D., ZAMIR A., SHAMIR A.: Clipasso: Semantically-aware object sketching. *ACM Transactions on Graphics (TOG) 41*, 4 (2022), 1–11. 3

[VSZ*24] VINKER Y., SHAHAM T. R., ZHENG K., ZHAO A., FAN J. E., TORRALBA A.: Sketchagent: Language-driven sequential sketch generation. *arXiv preprint arXiv:2411.17673* (2024). 2, 6, 8

[WBZ21] WANG S.-Y., BAU D., ZHU J.-Y.: Sketch your own gan. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 14050–14060. 2

[WDQ*23] WANG Q., DENG H., QI Y., LI D., SONG Y.-Z.: Sketchknitter: Vectorized sketch generation with diffusion models. In *The Eleventh International Conference on Learning Representations* (2023). 2

[WWS*22] WEI J., WANG X., SCHUURMANS D., BOSMA M., XIA F., CHI E., LE Q. V., ZHOU D., ET AL.: Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems 35* (2022), 24824–24837. 2

[XF24] XIAO C., FU H.: Customsketching: Sketch concept extraction for sketch-based image synthesis and editing. In *Computer Graphics Forum* (2024), vol. 43, Wiley Online Library, p. e15247. 2

[XHY*22] XU P., HOSPEDALES T. M., YIN Q., SONG Y.-Z., XIANG T., WANG L.: Deep learning for free-hand sketch: A survey. *IEEE transactions on pattern analysis and machine intelligence 45*, 1 (2022), 285–312. 2

[XWZ*23] XING X., WANG C., ZHOU H., ZHANG J., YU Q., XU D.: Diffsketcher: Text guided vector sketch synthesis through latent diffusion models. *Advances in Neural Information Processing Systems 36* (2023), 15869–15889. 1, 2, 6

[ZFW*18] ZHOU T., FANG C., WANG Z., YANG J., KIM B., CHEN Z., BRANDT J., TERZOPOULOS D.: Learning to sketch with deep q networks and demonstrated strokes. *arXiv preprint arXiv:1810.05977* (2018). 2

[ZRA23] ZHANG L., RAO A., AGRAWALA M.: Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision* (2023), pp. 3836–3847. 3

[ZWY*24] ZHOU Y., WANG J., YANG J., NI P., LU G., FANG H., LI Z., YU H., HUANG K.: Cross-modal pixel-and-stroke representation aligning networks for free-hand sketch recognition. *Expert Systems with Applications 240* (2024), 122505. 3

[ZYD*18] ZOU C., YU Q., DU R., MO H., SONG Y.-Z., XIANG T., GAO C., CHEN B., ZHANG H.: Sketchyscene: Richly-annotated scene sketches. In *Proceedings of the european conference on computer vision (ECCV)* (2018), pp. 421–436. 8