Lang-SVG: Hierarchical Image Vectorization with Language Priors

Xi Liu[1] **Chaoyi Zhou**[1] **Run Wang**[1] **Jiaang Li**[2]
Feng Luo[1] **Junxiang Huang**[3] **Siyu Huang**[1]*
[1]Clemson University [2]University of Copenhagen [3]eBay

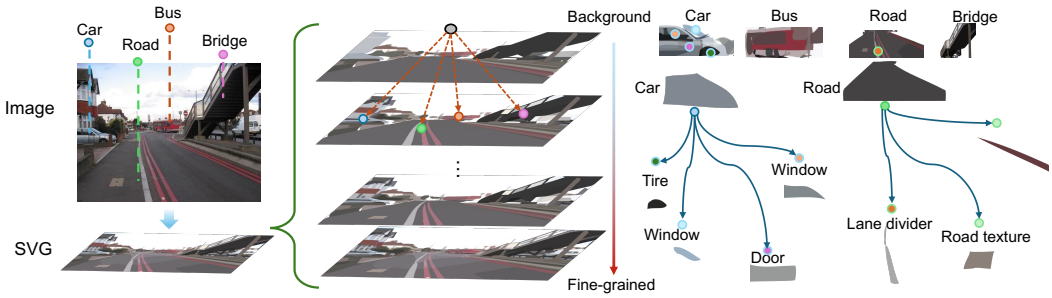

Figure 1: This work proposes **Lang-SVG**, a tree-structured SVG representation that vectorizes natural images into hierarchical object-part SVGs with language labels.

Abstract

Image vectorization reconstructs raster images as compact scalable vector graphics (SVG), providing a structured and editable code representation of images. However, most existing vectorization methods optimize for pixel-level rendering fidelity, often producing SVGs that lack alignment with human-perceived object-part hierarchies, making them difficult to manipulate. To address this, we propose Lang-SVG, which introduces an explicit tree structure to organize SVG primitives with semantic labels and parent-child relationships. Lang-SVG incorporates granularity-controllable and open-vocabulary segmentation priors to initialize object-part hierarchies and leverages multimodal large language models (MLLMs) with SVG tree context to assign semantic labels to unlabeled nodes. To reduce redundancy, we introduce depth-adaptive simplification to reduce shape complexity and Tree-Guided Pruning and Merging to eliminate redundant SVG nodes. L1-SAM-Prompting further recovers missing regions to improve reconstruction fidelity and structural completeness. Beyond reconstruction fidelity, we introduce a new evaluation protocol that measures structural quality and semantic part alignment. Experiments demonstrate that Lang-SVG achieves state-of-the-art performance in both rendering fidelity and structural quality. Project website: https://xiliu8006.github.io/Lang-SVG/.

1 Introduction

Image vectorization transforms raster images into compact scalable vector graphics (SVG), providing a structured, renderable, and editable code representation of visual content. As a text-based representation, SVG is increasingly becoming an interface between images and language models. Recent MLLMs learn to generate SVG code [1–4], benchmarks evaluate visual understanding through

*Corresponding author: Siyu Huang

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

image-to-SVG generation [5, 6], and code-based approaches [7–11] explore structured visual code for generation, manipulation, and reasoning. However, existing SVG datasets [1, 2, 12, 13] largely focus on icons and illustrations, offering limited coverage of complex natural images. Vectorizing natural images offers a much larger and more diverse source of SVG code beyond simple icons and illustrations, but representing their complex visual content as structured code remains challenging.

Recent differentiable SVG methods [14–17] have substantially advanced image vectorization in terms of rendering fidelity, efficiency, and compactness, with recent approaches [18–20, 4] further exploring layered and semantic representations to improve editability. However, these methods typically organize SVG primitives as flat layers without explicitly encoding object-part hierarchies. As a result, primitives belonging to the same object may be distributed across different layers, making it difficult to select and edit individual objects or their parts. Moreover, without hierarchical guidance, SVG optimization may allocate excessive primitives to redundant shapes or high-frequency textures rather than meaningful semantic regions. These limitations highlight the need for image vectorization that preserves semantic structure while maintaining high rendering fidelity.

We argue that this limitation partly stems from the lack of explicit semantic and structural constraints in existing image vectorization objectives. While pixel-level reconstruction losses effectively optimize SVG geometry, opacity, and color, they do not explicitly encourage primitives to form meaningful semantic structures. SGLIVE [19] incorporates SAM [21] to improve structural alignment, but SAM masks do not explicitly capture fine-grained semantic object-part hierarchies. LIVSS [18] further employs diffusion-based simplification to generate images at multiple semantic levels, followed by SAM to extract masks from coarse semantic level to fine-grained level. However, diffusion-based simplification may distort geometric details, causing shape boundaries to deviate from the image.

To address these issues, we propose **Lang-SVG**, a tree-structured SVG representation that organizes vector primitives with semantic labels and parent-child hierarchy, as illustrated in Fig. 1. Unlike flat SVG representations, our hierarchical structure explicitly models object-part relationships while preserving the flexibility of free-form vector primitives. We initialize its semantic hierarchy with open-vocabulary segmentation [22] and enrich it with coarse-to-fine masks from granularity-controllable models such as Semantic-SAM [23], avoiding the geometric distortions introduced by diffusion-based simplification. However, these masks often contain overlapping or redundant regions, making direct vectorization difficult under a limited control-point budget. We therefore introduce depth-adaptive simplification to allocate geometric complexity according to tree depth, and a Tree-Guided Pruning and Merging module to remove redundant regions while preserving semantically labeled nodes. We further introduce L1-SAM-Prompting, which uses reconstruction residuals to prompt SAM for coherent missing regions, progressively refining the tree beyond the initial segmentation masks. Our experiments demonstrate that the explicit tree structure improves both reconstruction fidelity and alignment with human-perceived image structures.

Beyond hierarchical image vectorization, Lang-SVG further supports SVG semantic labeling. While open-vocabulary segmentation provides initial semantic labels, finer regions introduced during tree refinement may remain unlabeled. To address this, we develop a tree-guided semantic tagging method that leverages multimodal large language models (MLLMs) with visual evidence and hierarchical SVG context, including parent, sibling, and child relationships. This enables semantic labeling beyond the initial segmentation models, supporting semantic selection and part-level manipulation.

Evaluating hierarchical SVG vectorization requires going beyond photometric and perceptual metrics, which cannot determine whether SVG primitives align with human-perceived object-part structures. We therefore introduce a new evaluation protocol that combines StructXLIP [24], a structure-aware vision-language model for measuring image-level structural similarity, with part/subpart recall to assess the alignment between SVG regions and human annotations at different IoU thresholds. On the ADE20K [25] dataset, Lang-SVG improves StructXLIP from 0.462 to 0.506 and average part recall from 28.8% to 48.5% at 576 control points, while achieving the best reconstruction performance among the compared methods. Furthermore, incorporating SVG tree context improves semantic tagging accuracy from 58.5% to 81.3% in our masked-label evaluation, demonstrating the value of hierarchical information for semantic reasoning.

Overall, our contributions are summarized as follows:

- To the best of our knowledge, this work is the first to construct hierarchical SVG trees with semantic labeling from raster images.

- We propose Lang-SVG, a novel hierarchical SVG representation with semantic labels and parent-child relationships. To handle the complex and redundant masks introduced by Semantic-SAM's fine-grained, multi-level segmentation, we introduce depth-adaptive simplification and tree-guided pruning and merging. We further introduce L1-SAM-Prompting to recover missing regions and tree-aware MLLM labeling for semantic tagging.
- We propose a new evaluation protocol for image vectorization that goes beyond reconstruction fidelity by measuring structural quality and semantic part alignment. The experiments show that Lang-SVG achieves state-of-the-art results under both reconstruction fidelity and structural quality.

2 Related Work

Image vectorization. Image vectorization transforms a raster image into a parametric representation, enabling resolution-independent rendering while making the image directly editable by humans [26–30]. DiffVG [14] was among the first works to make the rendering of SVG differentiable, reformulating the vectorization task as an end-to-end optimization problem. Bézier Splatting [15] improves rendering efficiency by introducing a splatting-based formulation inspired by 2D Gaussian representations [31], making vectorization more practical for high-resolution rendering. LIVE [20] introduces a layer-wise coarse-to-fine strategy that progressively adds primitives, improving the quality and topology of SVG representations. Subsequent works [17, 16] address the over-complexity of SVGs by simplifying vector representations and applying add-and-prune strategies to remove redundant primitives. Beyond compactness, SGLIVE [19], LIVSS [18] and SemLayer [32] incorporate segmentation priors to better align vectorized shapes with the structural information underlying natural image textures, leading to more semantically meaningful decompositions. AmodalSVG [33] goes further by peeling semantically complete raster layers with MLLM guidance and inpainting.

A recent line of work reformulates vectorization as an SVG code generation task with MLLMs. StarVector [1], OmniSVG [2], InternSVG [34] and IR4RL [10] perform image-to-SVG generation directly in the SVG code space, leveraging semantic visual understanding to predict richer primitives and more compact outputs. These methods indicate that vectorization is moving from pure reconstruction toward editability and semantic decomposition. However, most existing methods still represent SVGs as an ordered set of primitives, or first separate semantic regions in the raster domain and then vectorize them independently. In contrast, our proposed Lang-SVG representation explicitly encodes parent-child relations and semantics among SVG nodes, enabling applications such as grouping and semantic analysis of SVG primitives.

Open-vocabulary segmentation. Open-vocabulary segmentation provides a flexible way to extract semantic regions from images using language-level concepts, which is particularly useful for semantic vectorization. The SAM family [21, 35, 36] has enabled general-purpose promptable mask generation across diverse image domains, with recent variants further extending prompts from points and boxes to concept-level inputs. However, these semantic prompts are typically provided by users. Grounded SAM [22] addresses this limitation by coupling open-vocabulary detection or tagging with SAM-based segmentation: semantic candidates are first localized or generated automatically, and SAM is then applied to produce the corresponding masks. In this way, language-level concepts can be converted into raster-domain candidate masks with less manual prompting, providing useful guidance for downstream vectorization. Some works [37, 38] further target open-vocabulary part segmentation and object-part parsing. Another line of work [23, 39, 40] improves the granularity and semantic controllability of segmentation foundation models, enabling finer-grained masks. Our work leverages open-vocabulary segmentation and granularity-controllable SAM models to introduce semantically aligned hierarchical information into image vectorization. This hierarchical structure allows vectorized layers to better align with object-part semantics, and further supports tagging semantically meaningful regions that are not explicitly parsed by the segmentation model.

Multimodal large language model. Multimodal large language models (MLLMs) are typically pretrained via next-token prediction over heterogeneous inputs, including images, text, and videos. Early MLLMs, such as BLIP [41], Flamingo [42], and Idefics [43], introduce cross-attention mechanisms to facilitate interaction between visual and linguistic representations. Modern MLLMs largely follow the paradigm popularized by LLaVA [44], where visual tokens produced by pretrained vision encoders such as CLIP [45] are projected into the LLM backbone. This design simplifies multimodal

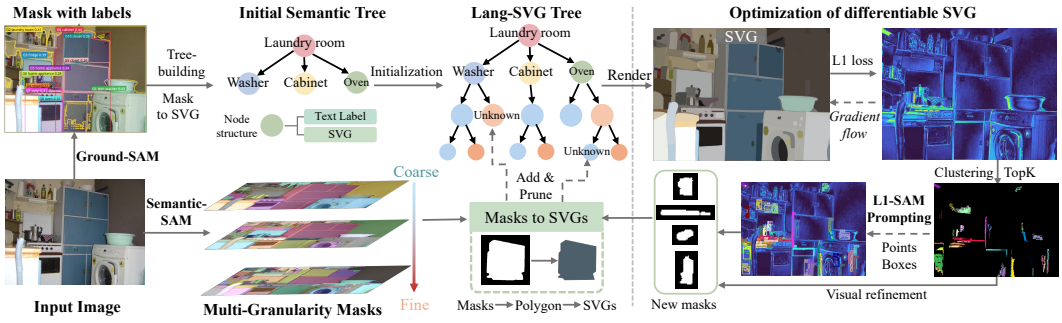

Figure 2: **Overview of Lang-SVG framework.** Given an input image, we initialize a hierarchical Lang-SVG tree from multi-granularity and open-vocabulary segmentation masks. The tree is iteratively refined through differentiable SVG optimization, L1-SAM-Prompting to recover missing regions, and Add & Prune operations to add new leaves and prune redundant ones.

fusion while retaining strong language modeling capabilities. Recent research shows that MLLMs can ground language in visual content, leveraging both visual and text context with implicit commonsense knowledge acquired during large-scale training [46–50]. We utilize MLLMs to predict unknown tags of SVG representations conditioned on visual inputs and structured semantic context derived from a tree-based representation.

3 Methodology

Given an input raster image $\mathbf{I} \in \mathbb{R}^{H\times W\times 3}$, we aim to construct a tree-structured SVG representation $\mathcal{T}$ that achieves high-fidelity rendering while preserving the semantic hierarchy and fine-grained structural details of the original image. We define

$$\mathcal{T} = (\mathcal{V}, \mathcal{E}, r), \qquad v_k = (y_k, s_k, \boldsymbol{m}_k). \tag{1}$$

Here, $\mathcal{V} = \{v_k\}_{k=1}^{K}$ is the set of nodes, $\mathcal{E}$ defines the parent–child relationships, and r is the root node. Each edge $(v_k, v_{k'}) \in \mathcal{E}$ indicates that $v_{k'}$ is a semantic constituent of v_k. Each node v_k stores a semantic tag y_k, a semantic mask $\boldsymbol{m}_k$, and a set of SVG primitives parameterized by s_k.

3.1 SVG Tree Construction via Granularity-Controllable Segmentation

To construct a tree-structured SVG representation that preserves semantic hierarchies and fine-grained structural details, we first obtain masks with semantic tags from open-vocabulary [22] or subpart-aware segmentation methods [38, 37], as shown in Fig. 2. These masks initialize a semantic tree with labeled object-part regions. We then enrich the tree with multi-granularity masks from granularity-controllable segmentation models, such as Semantic-SAM [23] and UnSAM [39]. These additional masks are initially assigned *unknown* semantic tags and serve as candidate nodes for refining the hierarchy. Formally, the multi-granularity masks are defined as

$$\{\mathcal{M}^{(\ell)}\}_{\ell=1}^{L}, \qquad \mathcal{M}^{(\ell)} = \{\boldsymbol{m}_i^{(\ell)}\}, \tag{2}$$

where each set $\mathcal{M}^{(\ell)}$ contains masks at a specific semantic granularity.

We construct the hierarchy by inserting candidate masks from coarse to fine based on spatial containment. A node v_k is considered a candidate parent of $v_{k'}$ if

$$\frac{|\boldsymbol{m}_{k'} \cap \boldsymbol{m}_k|}{|\boldsymbol{m}_{k'}|} \geq \tau. \tag{3}$$

Among valid parents, we select the node at the greatest tree depth, breaking ties by mask IoU. If no valid parent exists, the node is attached to the full-image root. This process organizes segmentation masks into explicit parent-child relationships rather than flat layers, providing a hierarchical structure for subsequent SVG initialization and optimization. Appendix A.2 provides the complete construction and update procedures.

3.2 Tree-Guided SVG Optimization

Tree-guided SVG initialization. For each node v_k, we convert its mask $\boldsymbol{m}_k$ into a polygon, which is then approximated by a set of Bézier curves to initialize the corresponding vector layer $\mathcal{L}_k$ as a closed region. A key challenge arises during this conversion: masks from coarser segmentation levels tend to have complex, irregular boundaries, yet due to alpha compositing, nodes closer to the tree root are more occluded and contribute less to the final rendered image. Fitting all masks with high-complexity polygons therefore introduces redundant primitives with little gain in reconstruction quality. To address this, we propose depth-adaptive simplification. Rather than using a fixed tolerance for all nodes, we adapt the approximation tolerance according to the node's depth in the tree: nodes closer to the root are fitted with higher tolerance, yielding simpler polygons, while leaf nodes use lower tolerance to preserve fine-grained boundary details. This allocates geometric complexity in proportion to visual contribution, achieving a better balance between reconstruction fidelity and primitive efficiency.

L1-SAM-Prompting. A common practice in image vectorization is to initialize SVG primitives from clusters derived from the L1 loss map between the current rendering and the target image. While this improves reconstruction metrics such as PSNR, it tends to allocate primitives toward fitting high-frequency textures rather than capturing structurally meaningful regions, resulting in insufficient representation of semantic structure. Although LIVSS [18] incorporates SAM to generate masks, it still relies on loss-driven clustering to meet path budgets and improve visual quality, since even with dense point prompts, SAM is not intended to produce fine-grained masks.

To address this, we propose L1-SAM-Prompting, which bridges loss-driven mask generation and SAM. Instead of directly using L1 clustering masks for SVG initialization, we extract their bounding boxes and mean points as prompts for SAM, producing masks that are more coherent and structurally complete, rather than dominated by frequency discrepancies between low-frequency SVG renderings and high-frequency natural images. We integrate this strategy as an intermediate step between two optimization stages: a *structure optimization* stage that fits segmentation-derived masks, and a *visual refinement* stage that refines appearance details. The L1-SAM-Prompting step generates supplementary masks that emphasize meaningful structures rather than noise from pixel-level loss, improving the balance between structural representation and visual fidelity.

Tree-guided pruning and merging. Multi-granularity segmentation produces a large number of masks that are often redundant and unstable. A common approach is to prune masks based on spatial overlap (e.g., IoU); however, IoU-based pruning is insufficient, as it fails to remove masks with different shapes but similar appearance and semantics. Rather than relying solely on geometric overlap, our representation enables appearance-aware pruning using the associated SVG layers. Specifically, we compare each node with its parent based on the color of their SVG layers: if an unlabeled child node exhibits high color similarity to its parent, we remove it; parent–child or sibling nodes within the same semantic branch can be merged when they exhibit similar color and sufficient spatial overlap. These branch constraints prevent similarly colored regions from different objects from being merged. Color and overlap therefore identify rendering redundancy within a branch rather than establish semantic identity. Nodes that already carry semantic tags are preserved and excluded from pruning or merging to retain semantic information.

3.3 Tree-Guided Semantic Labeling with Multimodal Large Language Models

Given a tree-structured SVG representation $\mathcal{T}$, our objective is to assign semantic labels to nodes with unknown tags. We focus on unlabeled nodes within subtrees rooted at semantically labeled nodes, allowing us to exploit hierarchical context. As shown in Fig. 4, we use Qwen3.5-27B [51] to predict the semantic label of each unlabeled node by jointly considering its local neighborhood and ancestral context within the tree.

For each unlabeled node v_k, we construct a structured prompt containing the full image, a crop extracted from the bounding box of the target SVG region, and its RGB color from SVG primitives. We further include parent and ancestor tags to define the semantic scope, along with labeled sibling and child tags and sibling bounding boxes to provide local relational and spatial cues for disambiguation. The prompt is formulated as a structured decision task, where the MLLM is asked to infer the semantic tag of the target node based on these visual and hierarchical cues. We consider both closed-set and

open-vocabulary settings. In the closed-set setting, the MLLM selects a label from a predefined candidate set, while in the open-vocabulary setting, it predicts a free-form tag. Ambiguous nodes are assigned a default *texture* or *unknown* tag to avoid unreliable semantic over-assignment. This process augments the SVG tree with node-level semantic labels, enabling vector primitives to be organized by both geometry and semantic identity. Table 5 shows that the tree structure provides useful context for semantic tag recovery.

4 Experiments

4.1 Experiment Setup

Datasets. Existing image vectorization methods [20, 19], support only reconstruction metrics and provide neither multi-level object-part hierarchies nor node-level semantic labels. Therefore, we conduct our main evaluation on ADE20K-Part234 [25], whose diverse natural scenes are annotated with object,and part masks and labels. These annotations allow us to test whether SVG regions align with human-annotated parts, whether parent-child relations and semantic tags are correct, and whether tree context helps recover missing tags. We uniformly sample 200 validation images, which contain 752 ground-truth part/subpart masks with 1624 annotated parts. To test generalization, we further use COCO-AOKVQA [52], whose image descriptions enable text-to-image semantic evaluation, and 40 complex SVG-style images selected from Iconfont [13].

Evaluation metrics. We first evaluate reconstruction fidelity using SSIM, PSNR, LPIPS, and CLIP similarity. However, these metrics do not fully capture the structural details of an image or its semantic object-part hierarchy. Therefore, we adopt StructXLIP [24], a structure-aware vision-language model trained with edge-based cues, to measure image-level structural similarity, and introduce part/subpart recall on ADE20K-Part234 to evaluate the alignment between SVG regions and ground-truth parts and subparts at different IoU thresholds. To evaluate semantic alignment, we compute text-to-image ($t2i$) CLIP and StructXLIP similarities between COCO-AOKVQA image descriptions and rendered SVGs.

To evaluate the correctness of SVG semantic hierarchies, we first match each ground-truth object to an SVG node with the same object label and introduce three metrics at a threshold t: *Child Containment* is the percentage of the matched nodes' direct children whose area lies inside the ground-truth object by more than t; *Tagging Reliability* is the percentage of same-label child groups whose area is covered by the ground-truth part with that label by more than t; and *Labeled Subpart Recall* is the percentage of ground-truth parts whose IoU with the same-label children exceeds t. We report $t \in \{0.3, 0.5, 0.7\}$ under the 128-path setting.

Evaluation mode. Since different vectorization methods use different SVG parameterizations, a fair comparison should control the total number of points. LIVE [20] and DiffVG [14] use a fixed representation, so the number of paths directly determines the total point budget. In contrast, LIVSS [18] and our method convert masks to SVG paths, and each path may contain a different number of points. Therefore, we report two protocols. *Same points* is our main protocol, where all methods are compared under the same total number of points. We also report *Same paths* for LIVSS [18] and our method, using the same number of SVG paths to ensure a fair comparison.

4.2 Comparison to Baselines

Quantitative and Qualitative results. We compare Lang-SVG with state-of-the-art differentiable SVG vectorization methods, including SGLIVE [19] and LIVSS [18], which also leverage segmentation priors. As shown in Tables 1 and 2, our method consistently outperforms existing methods in both reconstruction fidelity and structural quality across all point budgets. At 576 control points, Lang-SVG improves StructXLIP from the second-best 0.462 to 0.506, and average part recall from 28.8 to 48.5. The improvements also hold under the *Same paths* protocol. As shown in Fig. 3, our method better preserves coherent object regions and fine-grained structures, especially under limited primitive budgets, rather than fitting local textures or reconstruction residuals. These results demonstrate that Lang-SVG achieves better visual fidelity and object-part structural alignment with a limited primitive budget.

Table 1: Quantitative evaluation on ADE20K. Unmarked methods use the *Same points* protocol; † denotes *Same paths*. LIVE and DiffVG use nine control points per path and are reported once for both protocols. S-XLIP denotes StructXLIP.

Method	576 points / 64 curves					1152 points / 128 curves					2304 points / 256 curves				
	$SSIM^{\uparrow}$	$PSNR^{\uparrow}$	$LPIPS^{\downarrow}$	S-XLIP$^{\uparrow}$	$CLIP^{\uparrow}$	$SSIM^{\uparrow}$	$PSNR^{\uparrow}$	$LPIPS^{\downarrow}$	S-XLIP$^{\uparrow}$	$CLIP^{\uparrow}$	$SSIM^{\uparrow}$	$PSNR^{\uparrow}$	$LPIPS^{\downarrow}$	S-XLIP$^{\uparrow}$	$CLIP^{\uparrow}$
LIVE [20]	0.631	21.28	0.468	0.420	0.625	0.680	23.16	0.394	0.582	0.692	0.739	25.45	0.302	0.748	0.764
LIVSS [18]	0.598	19.50	0.490	0.379	0.609	0.643	21.48	0.421	0.531	0.664	0.707	24.12	0.336	0.695	0.734
DiffVG [14]	0.624	21.81	0.472	0.290	0.597	0.667	23.29	0.407	0.437	0.658	0.716	24.83	0.332	0.629	0.745
SGLIVE [19]	0.633	21.38	0.464	0.462	0.634	0.681	23.32	0.385	0.610	0.694	0.732	25.04	0.305	0.731	0.752
Ours	**0.638**	**21.88**	**0.436**	**0.506**	**0.651**	**0.694**	**24.07**	**0.360**	**0.655**	**0.717**	**0.746**	**25.90**	**0.291**	**0.775**	**0.777**
LIVSS† [18]	0.629	21.13	0.445	0.500	0.652	0.668	22.68	0.387	0.608	0.697	0.712	24.34	0.328	0.718	0.743
Ours†	**0.651**	**22.46**	**0.421**	**0.553**	**0.672**	**0.698**	**24.24**	**0.355**	**0.677**	**0.724**	**0.745**	**25.88**	**0.293**	**0.777**	**0.773**

Table 2: Part-mask recall of rendered SVG regions on ADE20K-Part234. We uniformly sample 200 images from the ADE20K [25] validation set. $R@t$ is the percentage of ground-truth masks whose best matching SVG region has IoU at least t. Avg.R is the arithmetic mean of the four displayed recalls, $R@.3$, $R@.5$, $R@.7$, and $R@.8$.

Protocol	Method	576 points / 64 curves					1152 points / 128 curves					2304 points / 256 curves				
		$R@.3^{\uparrow}$	$R@.5^{\uparrow}$	$R@.7^{\uparrow}$	$R@.8^{\uparrow}$	Avg.R$^{\uparrow}$	$R@.3^{\uparrow}$	$R@.5^{\uparrow}$	$R@.7^{\uparrow}$	$R@.8^{\uparrow}$	Avg.R$^{\uparrow}$	$R@.3^{\uparrow}$	$R@.5^{\uparrow}$	$R@.7^{\uparrow}$	$R@.8^{\uparrow}$	Avg.R$^{\uparrow}$
Same points	LIVE [20]	42.4	17.2	5.2	2.7	16.9	51.3	20.5	4.7	1.9	19.6	56.9	23.3	6.8	2.8	22.4
	DiffVG [14]	43.1	13.0	2.8	0.7	14.9	46.7	13.6	2.8	0.8	16.0	49.5	12.6	2.1	0.7	16.2
	LIVSS [18]	49.9	33.4	19.7	12.2	28.8	65.4	51.7	30.7	17.8	41.4	79.3	61.8	35.4	20.1	49.1
	SGLIVE [19]	42.8	18.4	6.2	3.5	17.7	50.5	21.9	6.2	2.9	20.4	57.4	23.4	5.6	2.7	22.3
	Ours	**75.1**	**60.5**	**37.2**	**21.1**	**48.5**	**85.1**	**70.7**	**44.0**	**26.3**	**56.5**	**94.3**	**79.3**	**51.3**	**31.8**	**64.2**
Same paths	LIVSS [18]	60.6	45.5	28.7	18.2	38.3	74.5	56.6	33.5	19.4	46.0	85.1	68.0	38.7	22.9	53.7
	Ours	**77.7**	**64.4**	**40.2**	**23.7**	**51.5**	**85.8**	**72.2**	**47.5**	**28.7**	**58.5**	**93.9**	**79.3**	**54.7**	**34.8**	**65.7**

Table 3: Quantitative evaluation on COCO-AOKVQA under the *Same points* protocol, including reconstruction and image/text semantic metrics.

Method	576 points					1152 points					2304 points				
	$PSNR^{\uparrow}$	S-XLIP$^{\uparrow}$	$CLIP^{\uparrow}$	S-XLIP$^{\uparrow}_{t2i}$	$CLIP^{\uparrow}_{t2i}$	$PSNR^{\uparrow}$	S-XLIP$^{\uparrow}$	$CLIP^{\uparrow}$	S-XLIP$^{\uparrow}_{t2i}$	$CLIP^{\uparrow}_{t2i}$	$PSNR^{\uparrow}$	S-XLIP$^{\uparrow}$	$CLIP^{\uparrow}$	S-XLIP$^{\uparrow}_{t2i}$	$CLIP^{\uparrow}_{t2i}$
LIVE [20]	19.76	0.371	0.607	0.234	0.238	21.46	0.534	0.675	0.291	0.261	23.41	0.692	0.752	0.331	0.276
LIVSS [18]	18.29	0.349	0.599	0.226	0.236	19.94	0.472	0.653	0.273	0.255	22.16	0.641	0.724	0.323	0.272
DiffVG [14]	20.35	0.271	0.591	0.188	0.228	21.63	0.387	0.642	0.233	0.249	22.95	0.559	0.717	0.294	0.270
Ours	**20.53**	**0.460**	**0.644**	**0.270**	**0.254**	**22.27**	**0.601**	**0.703**	**0.313**	**0.268**	**23.78**	**0.724**	**0.760**	**0.341**	**0.279**

Cross-dataset generalization. To further evaluate generalization, we test image vectorization on COCO-AOKVQA [52] and Iconfont [13]. COCO-AOKVQA provides scene descriptions, allowing us to evaluate semantic alignment between the text descriptions and rendered SVG images using text-to-image CLIP and StructXLIP scores. As shown in Table 3, our method achieves the best reconstruction and semantic scores across all point budgets. At 576 points, compared with DiffVG, the strongest PSNR baseline, PSNR improves modestly from 20.35 to 20.53, while text-to-image StructXLIP increases substantially from 0.188 to 0.270, indicating better preservation of semantic content beyond pixel-level reconstruction. On Iconfont SVG-style images, our method also achieves the highest PSNR, improving from 28.62 (SGLIVE) to 29.28. Additional results are provided in Appendix A.1.

4.3 Evaluation of Hierarchy and Semantic Quality

Table 4 evaluates the hierarchy and semantic quality of the generated SVG trees. Child Containment reaches 95.55% at threshold 0.5 and remains 86.65% at 0.7, demonstrating reliable parent-child relationships. At threshold 0.5, Tagging Reliability reaches 68.25%, while Labeled Subpart Recall reaches 47.06%, indicating that our method recovers meaningful semantic labels and object-part structures. Since semantic tags and subpart coverage depend partly on upstream segmentation, stronger segmentation models could further improve these results.

Table 4: Hierarchy and semantic quality on ADE20K. Results are percentages (higher is better).

Threshold	Child Containment	Tagging Reliability	Labeled Subpart Recall
0.3	98.13	82.56	65.98
0.5	95.55	68.25	47.06
0.7	86.65	46.62	27.00

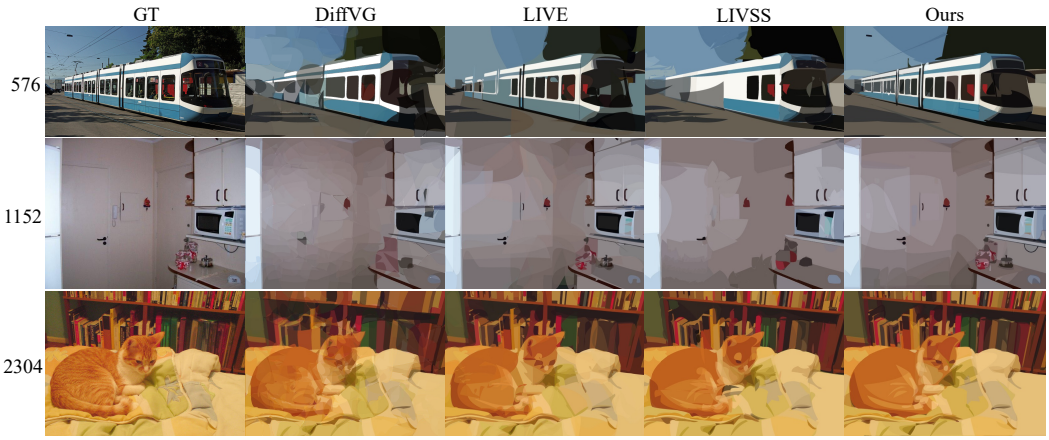

Figure 3: **Image vectorization comparison.** We compare our method with DiffVG [14], LIVE [20], and LIVSS [18]. Our SVGs achieve better reconstruction fidelity and semantic alignment with the ground-truth images, preserving object boundaries and fine-grained structural details.

Table 5: **Ablation of tree-guided semantic assignment.** We report exact accuracy for masked-label prediction on ADE20K-Part nodes. *Full Img.* denotes the original input image; *Crop Img.* denotes target-region crops; *Obj./Par.* denotes object and parent semantic labels; *Sib./Child* denotes sibling and child semantic labels; *Geo.* denotes parent, sibling, and child geometric layout; and *Color* denotes SVG region color statistics.

ID	Full Img.	Crop Img.	Obj./Par.	Sib./Child	Geo.	Color	Acc.
A0	✓	✓					58.5
A1	✓	✓	✓				67.7
A2	✓	✓	✓	✓			68.9
A3	✓	✓	✓	✓	✓		80.2
A4	✓	✓	✓	✓	✓	✓	**81.3**
Negative controls							
A5	✓		✓	✓	✓	✓	69.2
A6			✓	✓	✓	✓	60.7

4.4 Semantic SVG Tagging

To evaluate whether the hierarchical structure of Lang-SVG provides useful semantic information beyond reconstruction, we conduct a node-wise tag recovery task. Starting from ADE20K ground-truth masks and labels, we construct SVG trees following Sec. 3.1 and randomly hide up to 50% of the part and subpart tags in each scene, while preserving the masks, node locations, and tree topology. The hidden tags are used only for evaluation. Given the full image, target region, and different subsets of tree context, Qwen3.5-27B predicts the missing tags from the closed ADE20K label set. This setup isolates semantic tagging from segmentation quality and evaluates the contribution of hierarchical context to label recovery.

Table 5 shows that hierarchical context substantially improves tag prediction for subpart nodes. Visual evidence alone achieves 58.5% exact accuracy, which increases to 67.7% with object and parent identity and 68.9% with additional sibling and child labels. Including the geometric layout of parent, sibling, and child regions further improves accuracy to 80.2%, while adding region color statistics raises it to 81.3%. We further examine the contribution of visual inputs in A5 and A6 by removing localized crops or all image inputs. Even without these inputs, the structured context achieves accuracy close to the visual-only baseline, suggesting that the SVG tree itself provides substantial semantic information. These results demonstrate the effectiveness of hierarchical SVG context for node-level semantic tag recovery.

We further evaluate the full pipeline without ground-truth masks or labels. In this setting, Lang-SVG is initialized using open-vocabulary part segmentation [38], followed by tree-guided semantic tagging of unlabeled SVG nodes. As shown in Fig. 4, qualitative results on Pascal-Part [53] demonstrate that our method can recover meaningful semantic labels for regions not recognized by the pretrained

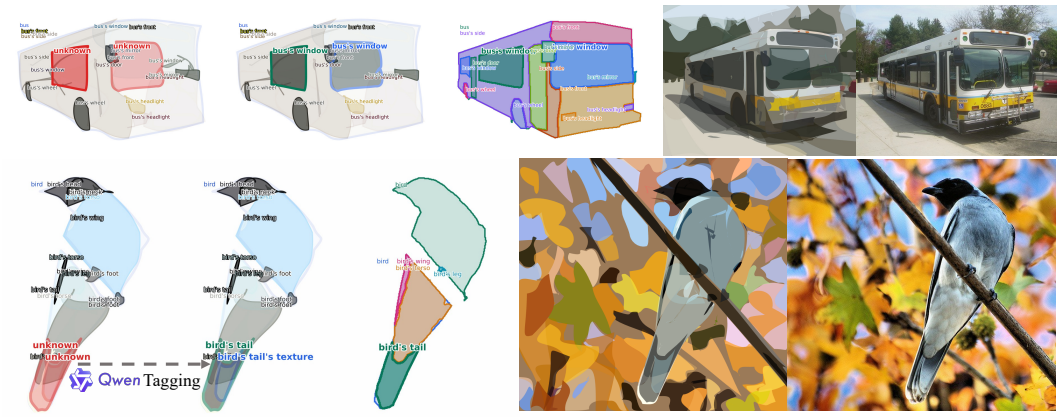

Figure 4: **Semantic SVG tagging.** We leverage the tree context and Qwen3.5-27B to predict the unknown tags of SVGs shown on the left. For clearer visualization, we only render SVG components whose semantic tags belong to the target object. The results show that Lang-SVG enables fine-grained semantic reasoning, where the tail region is predicted as *"bird tail"*, while a smaller sub-region is further recognized as the *"texture"* of the bird tail.

Table 6: Segmentation priors and vectorization design on ADE20K with 2304 points. The first two rows compare segmentation priors within the LIVSS pipeline; the last two rows compare LIVSS and Ours using the same Semantic-SAM and Grounded-SAM priors.

Method	**Segmentation priors**	$SSIM^{\uparrow}$	$PSNR^{\uparrow}$	$LPIPS^{\downarrow}$	S-XLIP$^{\uparrow}$	$CLIP^{\uparrow}$
LIVSS	SAM	0.707	24.12	0.336	0.695	0.734
LIVSS	Semantic-SAM + Grounded-SAM	0.707	24.23	0.338	0.708	0.735
Ours	Semantic-SAM + Grounded-SAM	**0.746**	**25.90**	**0.291**	**0.775**	**0.777**

segmentation model. By leveraging hierarchical relationships, spatial layout, and appearance cues, the MLLM can identify missing object-part labels and, in some cases, finer subpart or texture-level regions. Ambiguous nodes can be assigned *texture* or *unknown* tags to avoid unreliable semantic predictions.

4.5 Ablation Study

We conduct three ablation studies on ADE20K [25] under the 2304-point parameter budget to examine segmentation-prior replacement, tree-related designs, and the remaining modules.

Segmentation priors and vectorization design. Table 6 compares segmentation-prior replacement in LIVSS with our complete vectorization framework. Replacing SAM with Semantic-SAM and Grounded-SAM in the LIVSS pipeline increases PSNR from 24.12 to 24.23 and StructXLIP from 0.6950 to 0.7081. Using the same segmentation priors, Lang-SVG further improves PSNR to 25.90 and StructXLIP to 0.775. These results show that the performance gains cannot be explained by stronger segmentation priors alone, supporting the contribution of our vectorization design.

Tree structure and tree-guided operations. Table 7 evaluates the tree structure, depth-adaptive simplification, and tree-guided pruning and merging using the same segmentation priors. Tree-guided pruning and merging operate on parent–child and sibling relations in the SVG tree and therefore require the tree structure; disabling the tree necessarily disables tree pruning. To evaluate the tree's contribution with pruning disabled, we compare *w/o tree and tree pruning* with *w/o Tree Prune*. Enabling the tree increases PSNR from 24.29 to 25.71 and StructXLIP from 0.7054 to 0.7687. The *w/o depth simplification* variant retains both the tree and tree pruning, reducing PSNR from 25.90 to 24.84 and StructXLIP from 0.775 to 0.7140. Finally, enabling pruning and merging in the *w/o Tree Prune* configuration improves PSNR from 25.71 to 25.90 and StructXLIP from 0.7687 to 0.775. This indicates that redundant masks otherwise remain and compete for the same parameter budget.

L1-SAM-Prompting and segmentation backbone. As shown in Table 8, *w/o L1-SAM-Prompting* disables the step that uses the L1 loss map to generate SAM prompts for obtaining additional masks,

Table 7: Tree-related ablations on ADE20K with 2304 points using Semantic-SAM and Grounded-SAM priors. Depth denotes depth-adaptive simplification; Tree Pruning denotes tree-guided pruning and merging. Checkmarks indicate enabled components. Tree pruning requires the tree structure, so removing the tree also disables tree pruning. Ours is the same final model as in Table 8.

Method	Depth	Tree	Tree Pruning	$SSIM^{\uparrow}$	$PSNR^{\uparrow}$	$LPIPS^{\downarrow}$	S-XLIP$^{\uparrow}$	$CLIP^{\uparrow}$
w/o depth simplification	×	✓	✓	0.717	24.84	0.332	0.714	0.744
w/o tree and tree pruning	✓	×	×	0.708	24.29	0.337	0.705	0.737
w/o Tree Prune	✓	✓	×	0.743	25.71	0.296	0.769	0.770
Ours	✓	✓	✓	**0.746**	**25.90**	**0.291**	**0.775**	**0.777**

Table 8: Ablation of L1-SAM-Prompting and segmentation backbone on ADE20K with 2304 points.

Method	$SSIM^{\uparrow}$	$PSNR^{\uparrow}$	$LPIPS^{\downarrow}$	S-XLIP$^{\uparrow}$	$CLIP^{\uparrow}$
w/o L1-SAM-Prompting	0.740 (-0.006)	25.41 (-0.49)	0.298 (+0.007)	0.762 (-0.013)	0.770 (-0.007)
w/ UnSAMv2	0.741 (-0.005)	25.73 (-0.17)	0.297 (+0.006)	0.763 (-0.012)	0.769 (-0.008)
Ours	**0.746**	**25.90**	**0.291**	**0.775**	**0.777**

and instead directly uses the masks from L1 clustering. This variant drops StructXLIP from 0.775 to 0.762, because more SVG primitives are allocated to optimizing the pixel-level L1 residual rather than to masks that follow semantic structures. In the *w/ UnSAMv2* variant, we replace Semantic-SAM with UnSAMv2 [40] at granularity levels $[0.8, 0.6, 0.4]$, and also replace SAM in L1-SAM-Prompting with UnSAMv2 at granularity 0.1. Semantic-SAM performs better than UnSAMv2 in our framework. All three ablation tables use the same final *Ours* reference.

5 Conclusion

We present Lang-SVG, a tree-structured SVG representation for hierarchical image vectorization and semantic labeling. Lang-SVG organizes SVG primitives into semantic object-part hierarchies and incorporates depth-adaptive simplification, Tree-Guided Pruning and Merging, and L1-SAM-Prompting to reduce redundancy, recover missing regions, and improve rendering fidelity. We further introduce a tree-guided semantic tagging method that leverages hierarchical context to guide MLLMs in labeling SVG nodes. To evaluate these representations beyond reconstruction fidelity, we propose a new evaluation protocol that measures structural quality and semantic alignment. Experiments on ADE20K demonstrate that Lang-SVG achieves state-of-the-art performance in both rendering fidelity and structural quality, providing structured and semantically labeled SVG representations for editing and downstream tasks.

Limitations and Future Work. Our method relies on open-vocabulary segmentation models to provide initial labels and language labels, such that the parsing quality is still partially bounded by the accuracy and granularity of these upstream models [54]. Missing parts, ambiguous masks, or incomplete labels may be inherited by the SVG tree, especially for small subparts and texture-like regions. Although our tree-guided MLLM assignment can recover many unknown tags using structural context, it is not yet specifically trained for SVG semantic understanding. A promising future direction is to fine-tune MLLMs for SVG-aware labeling. By training on SVG trees, rendered vector regions, parent-child relations, and part-level annotations, such models could assign more accurate semantic tags to vector components, reduce dependence on open-vocabulary segmentation, and improve fine-grained parsing of complex natural images.

Acknowledgments

This research was supported in part by the eBay eRUPT program and the SC NASA EPSCoR Research Grant Program (No. 21-117-RGP-CU-011). This research used in part resources on the Palmetto Cluster at Clemson University under NSF awards MRI 1228312, II NEW 1405767, MRI 1725573, and MRI 2018069. JL is supported by DNRF grant P1. The views expressed in this article do not necessarily represent the views of any funders.

References

[1] Juan A. Rodriguez, Abhay Puri, Shubham Agarwal, Issam H. Laradji, Pau Rodriguez, Sai Rajeswar, David Vazquez, Christopher Pal, and Marco Pedersoli. Starvector: Generating scalable vector graphics code from images and text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16175–16186, June 2025.

[2] Yiying Yang, Wei Cheng, Sijin Chen, Xianfang Zeng, Jiaxu Zhang, Liao Wang, Gang Yu, Xingjun Ma, and Yu-Gang Jiang. Omnisvg: A unified scalable vector graphics generation model. *arXiv preprint arXiv:2504.06263*, 2025.

[3] Ronghuan Wu, Wanchao Su, and Jing Liao. Chat2svg: Vector graphics generation with large language models and image diffusion models. *arXiv preprint arXiv:2411.16602*, 2024.

[4] Jooyeol Yun and Jaegul Choo. Vector prism: Animating vector graphics by stratifying semantic structure. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17316–17325, 2026.

[5] Kevin Qinghong Lin, Yuhao Zheng, Hangyu Ran, Dantong Zhu, Dongxing Mao, Linjie Li, Philip Torr, and Alex Jinpeng Wang. Vcode: a multimodal coding benchmark with svg as symbolic visual representation. *arXiv preprint arXiv:2511.02778*, 2025.

[6] Siqi Chen, Xinyu Dong, Haolei Xu, Xingyu Wu, Fei Tang, Hang Zhang, Yuchen Yan, Linjuan Wu, Wenqi Zhang, Guiyang Hou, et al. Svgenius: Benchmarking llms in svg understanding, editing and generation. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 13289–13296, 2025.

[7] Yipeng Gao, Ziyao Zeng, Jinfa Huang, Yijiang Li, Haolin Xiong, Wang Qin, Xinlei Yu, Youheng Yao, Jie Zhang, Kun-Yu Lin, et al. 3dcodeverse: Exploring 3d worlds as code. *arXiv preprint arXiv:2610.22341*, 2026.

[8] Souraja Kundu, Aditya Gupta, Joel Julin, Yizhou Zhao, Liuyue Xie, and Laszlo A Jeni. Physimcode: A benchmark and evaluation method for physics video to code generation. *arXiv preprint arXiv:2610.15722*, 2026.

[9] Hanyang Wang, Yimo Cai, Weiliang Chen, Jiawei Chi, Haowen Sun, Qiyu Dai, Yi-Hsin Hung, Xingzhuo Guo, Jinshan Ren, Runmao Yao, et al. Code as worlds: Agentic discovery of executable world representations for physical reasoning. *arXiv preprint arXiv:2608.27549*, 2026.

[10] Omri Kaduri, Kate Feingold, Phillip Isola, and Tali Dekel. Reinforcement learning from intermediate renders for image-to-code generation. *arXiv preprint arXiv:2609.34587*, 2026.

[11] Junyan Ye, Jun He, Zilong Huang, Dongzhi Jiang, Xuan Yang, Rui Chen, and Weijia Li. Genclaw: Code-driven agentic image generation. *arXiv preprint arXiv:2605.30248*, 2026.

[12] Feiyu Wang, Zhiyuan Zhao, Yuandong Liu, Da Zhang, Junyu Gao, Hao Sun, and Xuelong Li. Svgen: Interpretable vector graphics generation with large language models. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 9608–9617, 2025.

[13] Alibaba Group. Iconfont. https://www.iconfont.cn/. Accessed: 2025-11-01.

[14] Tzu-Mao Li, Michal Lukáč, Gharbi Michaël, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 39(6):193:1–193:15, 2020.

[15] Xi Liu, Chaoyi Zhou, Nanxuan Zhao, and Siyu Huang. Bézier splatting for fast and differentiable vector graphics rendering. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

[16] Kaibo Zhao, Liang Bao, Yufei Li, Xu Su, Ke Zhang, and Xiaotian Qiao. Less is more: Efficient image vectorization with adaptive parameterization. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18166–18175, 2025.

[17] Or Hirschorn, Amir Jevnisek, and Shai Avidan. Optimize & reduce: A top-down approach for image vectorization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 2148–2156, 2024.

[18] Zhenyu Wang, Jianxi Huang, Zhida Sun, Yuanhao Gong, Daniel Cohen-Or, and Min Lu. Layered image vectorization via semantic simplification. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 7728–7738, 2025.

[19] Hengyu Zhou, Hui Zhang, and Bin Wang. Segmentation-guided layer-wise image vectorization with gradient fills. In *European Conference on Computer Vision*, pages 165–180. Springer, 2024.

[20] Xu Ma, Yuqian Zhou, Xingqian Xu, Bin Sun, Valerii Filev, Nikita Orlov, Yun Fu, and Humphrey Shi. Towards layer-wise image vectorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16314–16323, 2022.

[21] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023.

[22] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, et al. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*, 2024.

[23] Feng Li, Hao Zhang, Peize Sun, Xueyan Zou, Shilong Liu, Chunyuan Li, Jianwei Yang, Lei Zhang, and Jianfeng Gao. Segment and recognize anything at any granularity. In *European Conference on Computer Vision*, pages 467–484. Springer, 2024.

[24] Zanxi Ruan, Songqun Gao, Qiuyu Kong, Yiming Wang, and Marco Cristani. Structxlip: Enhancing vision-language models with multimodal structural cues. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[25] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127(3):302–321, 2019.

[26] Pradyumna Reddy, Michael Gharbi, Michal Lukac, and Niloy J Mitra. Im2vec: Synthesizing vector graphics without vector supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7342–7351, 2021.

[27] Raphael Gontijo Lopes, David Ha, Douglas Eck, and Jonathon Shlens. A learned representation for scalable vector graphics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7930–7939, 2019.

[28] Defu Cao, Zhaowen Wang, Jose Echevarria, and Yan Liu. Svgformer: Representation learning for continuous vector graphics using transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10093–10102, June 2023.

[29] Peiying Zhang, Nanxuan Zhao, and Jing Liao. Text-to-vector generation with neural path representation. *ACM Transactions on Graphics (TOG)*, 43(4):1–13, 2024.

[30] Teng Hu, Ran Yi, Baihong Qian, Jiangning Zhang, Paul L Rosin, and Yu-Kun Lai. Supersvg: Superpixel-based scalable vector graphics synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24892–24901, 2024.

[31] Xinjie Zhang, Xingtong Ge, Tongda Xu, Dailan He, Yan Wang, Hongwei Qin, Guo Lu, Jing Geng, and Jun Zhang. Gaussianimage: 1000 fps image representation and compression by 2d gaussian splatting. In *European Conference on Computer Vision*, 2024.

[32] Haiyang Xu, Ronghuan Wu, Li-Yi Wei, Nanxuan Zhao, Chenxi Liu, Cuong Nguyen, Zhuowen Tu, and Zhaowen Wang. Semlayer: Semantic-aware generative segmentation and layer construction for abstract icons. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 42082–42092, June 2026.

[33] Juncheng Hu, Ziteng Xue, Guotao Liang, Anran Qi, Buyu Li, Sheng Wang, Dong Xu, and Qian Yu. Amodalsvg: Amodal image vectorization via semantic layer peeling. *arXiv preprint arXiv:2604.10940*, 2026.

[34] Haomin Wang, Jinhui Yin, Qi Wei, Wenguang Zeng, Lixin Gu, Shenglong Ye, Zhangwei Gao, Yaohui Wang, Yanting Zhang, Yuanqi Li, Yanwen Guo, Wenhai Wang, Kai Chen, Yu Qiao, and Hongjie Zhang. Internsvg: Towards unified svg tasks with multimodal large language models. *arXiv preprint arXiv:2510.11341*, 2025.

[35] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollar, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *The Thirteenth International Conference on Learning Representations*, 2025.

[36] Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, RISHI HAZRA, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollar, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. SAM 3: Segment anything with concepts. In *The Fourteenth International Conference on Learning Representations*, 2026.

[37] Junyi Li, Junfeng Wu, Weizhi Zhao, Song Bai, and Xiang Bai. Partglee: A foundation model for recognizing and parsing any objects. In *European Conference on Computer Vision*, pages 475–494. Springer, 2024.

[38] Yang Miao, Jan-Nico Zaech, Xi Wang, Fabien Despinoy, Danda Pani Paudel, and Luc Van Gool. LangHOPS: Language grounded hierarchical open-vocabulary part segmentation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

[39] XuDong Wang, Jingfeng Yang, and Trevor Darrell. Segment anything without supervision. *Advances in Neural Information Processing Systems*, 37:138731–138755, 2024.

[40] Junwei Yu, Trevor Darrell, and XuDong Wang. Unsamv2: Self-supervised learning enables segment anything at any granularity. *arXiv preprint arXiv:2511.13714*, 2025.

[41] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR, 2022.

[42] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[43] Hugo Laurençon, Lucile Saulnier, Léo Tronchon, Stas Bekman, Amanpreet Singh, Anton Lozhkov, Thomas Wang, Siddharth Karamcheti, Alexander Rush, Douwe Kiela, et al. Obelics: An open web-scale filtered dataset of interleaved image-text documents. *Advances in Neural Information Processing Systems*, 36:71683–71702, 2023.

[44] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

[45] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[46] Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, et al. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 91–104, 2025.

[47] Penghao Wu and Saining Xie. V?: Guided visual search as a core mechanism in multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13084–13094, 2024.

[48] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[49] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[50] Jiaang Li, Yifei Yuan, Wenyan Li, Mohammad Aliannejadi, Daniel Hershcovich, Anders Søgaard, Ivan Vulić, Wenxuan Zhang, Paul Pu Liang, Yang Deng, and Serge Belongie. RAVENEA: A benchmark for multimodal retrieval-augmented visual culture understanding. In *The Fourteenth International Conference on Learning Representations*, 2026.

[51] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[52] Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. A-okvqa: A benchmark for visual question answering using world knowledge. In *European conference on computer vision*, pages 146–162. Springer, 2022.

[53] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1971–1978, 2014.

[54] Jiaang Li, Chengzu Li, Zhaochong An, Yifei Yuan, Xi Liu, Serge Belongie, and Vésteinn Snæbjarnarson. Seeing or knowing? visual context sensitivity in multimodal large language models. *arXiv preprint arXiv:2607.26326*, 2026.

A Appendix

This appendix provides an additional evaluation on SVG-style images, the hierarchical SVG construction algorithms, the ADE20K tagging system prompt, and further qualitative vectorization and semantic-tagging results.

A.1 Generalization to SVG-Style Images

We evaluate 40 complex SVG-style images selected from Iconfont [13] to complement the natural-image benchmarks. All methods use the same budget of 576 control points. Table 9 shows that Lang-SVG achieves the highest PSNR, SSIM, and CLIP similarity and the lowest LPIPS. Compared with the strongest PSNR baseline, SGLIVE, PSNR increases from 28.62 to 29.28. LIVE achieves the highest StructXLIP score, while our method scores 0.9549. These results demonstrate that the pipeline also applies to SVG-style content. This experiment measures reconstruction and perceptual quality; hierarchy and semantic-label quality are evaluated separately using ADE20K object–part annotations in Sec. 4.3.

Table 9: Vectorization of 40 complex SVG-style images selected from Iconfont [13] under the same 576-point budget. Best results are bold.

Method	$SSIM^{\uparrow}$	$PSNR^{\uparrow}$	$LPIPS^{\downarrow}$	S-XLIP$^{\uparrow}$	$CLIP^{\uparrow}$
DiffVG	0.9125	26.46	0.1274	0.8899	0.9397
LIVE	0.9349	28.19	0.0882	**0.9616**	0.9704
SGLIVE	0.9308	28.62	0.0815	0.9546	0.9728
LIVSS	0.9319	28.24	0.0945	0.9491	0.9696
Ours	**0.9384**	**29.28**	**0.0782**	0.9549	**0.9766**

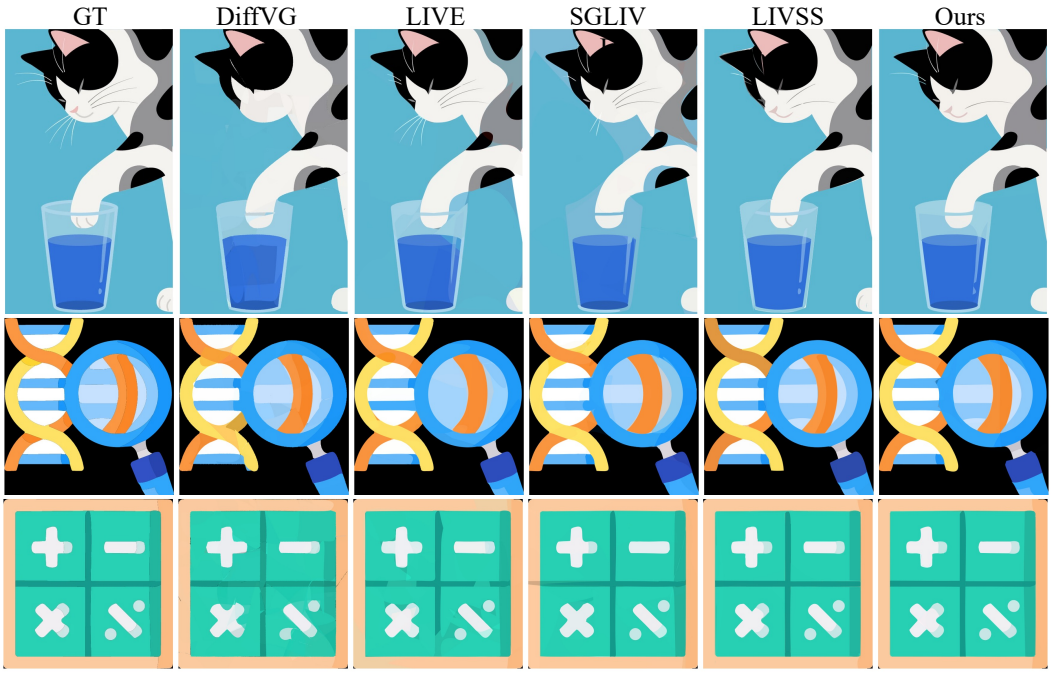

Figure 5: **SVG-style qualitative comparison.** We compare SVGs produced by DiffVG [14], LIVE [20], SGLIVE [19], LIVSS [18], and our method.

A.2 Algorithms for Hierarchical SVG Construction

Algorithms 1 and 2 detail the overall pipeline and the tree update procedure. Open-vocabulary masks initialize labeled semantic branches, while multi-granularity masks and reconstruction-driven

proposals add finer nodes. For each new mask, candidate parents are selected by spatial containment; the deepest candidate is preferred, with IoU used to break ties. The full-image root provides a fallback. This geometric assignment rule supports masks at different granularities, although imperfect segmentation and partially overlapping regions can still produce ambiguous relations. The hierarchy evaluation in Sec. 4.3 measures the resulting parent alignment and label consistency.

Algorithm 1 Overall pipeline for hierarchical SVG construction

Require: Image I; open-vocabulary masks $\mathcal{O}$ with labels and hierarchy $\mathcal{H}_{\mathcal{O}}$; multi-granularity masks $\mathcal{M}$; point budget B; thresholds $\tau_{\text{parent}}, \tau_{\text{prune}}$
Ensure: Semantic SVG tree $\mathcal{T}$ and flattened SVG S

Create root r with the full-image background mask
Initialize $\mathcal{T}$ from $\mathcal{O}$, preserving labels and $\mathcal{H}_{\mathcal{O}}$
Simplify and vectorize initial masks according to their tree depths
$\mathcal{T} \leftarrow \textsc{UpdateTree}(\mathcal{T}, \mathcal{M}, I, B, \tau_{\text{parent}}, \tau_{\text{prune}})$
while the point budget has not been reached **do**
    Compute the reconstruction residual between I and $\mathrm{Render}(\mathcal{T})$
    $\mathcal{M}_{\text{new}} \leftarrow$ generate masks using L1-SAM-Prompting
    if $\mathcal{M}_{\text{new}} = \emptyset$ **then**
        $\mathcal{M}_{\text{new}} \leftarrow$ generate masks from high-residual regions
    end if
    if $\mathcal{M}_{\text{new}} = \emptyset$ **then**
        break
    end if
    $\mathcal{T} \leftarrow \textsc{UpdateTree}(\mathcal{T}, \mathcal{M}_{\text{new}}, I, B, \tau_{\text{parent}}, \tau_{\text{prune}})$
    if no further update can be admitted under the point budget **then**
        break
    end if
end while
Label unknown nodes using the visual input, tree context, geometry, and color
Flatten $\mathcal{T}$ into painter's rendering order to obtain S
Store semantic labels and tree relations as SVG metadata
return $\mathcal{T}, S$

Algorithm 2 UPDATETREE: mask assignment, optimization, and pruning

Require: Current tree $\mathcal{T}$; candidate masks $\mathcal{Q}$; image I; point budget B; thresholds $\tau_{\text{parent}}, \tau_{\text{prune}}$
Ensure: Updated tree $\mathcal{T}$

function $\textsc{UpdateTree}(\mathcal{T}, \mathcal{Q}, I, B, \tau_{\text{parent}}, \tau_{\text{prune}})$
    for each mask $q \in \mathcal{Q}$, from coarse to fine **do**
        $\mathcal{P} \leftarrow \{v \in \mathcal{T} : |q \cap m_v|/|q| \geq \tau_{\text{parent}}\}$
        if $\mathcal{P} = \emptyset$ **then**
            $p \leftarrow r$
        else
            $d^* \leftarrow \max_{v \in \mathcal{P}} \mathrm{depth}(v)$
            $\mathcal{P}^* \leftarrow \{v \in \mathcal{P} : \mathrm{depth}(v) = d^*\}$
            $p \leftarrow \arg\max_{v \in \mathcal{P}^*} \mathrm{IoU}(q, m_v)$
        end if
        Attach q as a child of p, using *unknown* for any missing tag
        Simplify and vectorize q according to its tree depth
    end for
    Optimize SVG paths and colors against I
    Prune or merge redundant parent–child and sibling nodes within the same semantic branch using color, overlap, and τ_{prune}
    Preserve semantically labeled nodes and enforce the point budget B
    return $\mathcal{T}$
end function

A.3 ADE20K Tagging System Prompt

We provide the system prompt used for closed-set ADE20K semantic tag recovery below. The complete set of 234 candidate labels is supplied as `candidate_labels` in the user message. Only the final JSON output schema is omitted from this display.

```
You are an ADE20KPart234 scene-graph part classification evaluator.

You will receive:
- the original full image
- a crop around the containing object, when available
- a crop around the target bbox, when available
- metadata copied from a layered scene-graph JSON
- known object/part context and sibling layout from that JSON
- bbox layout metadata for the target, parent, siblings, and parent children
- a closed candidate list of valid part/subpart labels

The ground-truth target node tag has been hidden. Classify the semantic part or subpart
inside the target bbox using the image and the unmasked scene-graph context.

Rules:
- Choose exactly one label from candidate_labels.
- Use the target bbox metadata to locate the masked part/subpart inside the full image and object crop.
- Use the object crop for object-level context, not as the target region itself.
- Use the target bbox crop for local visual details.
- Use ancestor_tags, parent_tag, sibling_tags, child_tags, parent_layout,
  sibling_bboxes, and parent_child_layout from the JSON as context.
- candidate_labels are the only valid outputs. They are full hierarchical labels;
  reason about their leaf part names, but output the exact candidate label string.
- predicted_label must exactly match one candidate label string.
- Do not invent a new label outside the candidate list.
- Return exactly one JSON object.
- Do not output prose, markdown, or reasoning text.

Before choosing the label, silently run this parent-to-subpart checklist:
1. Identify the parent object/part from ancestor_tags, parent_tag, and parent_layout.
2. Generate plausible child parts for that parent, then eliminate candidates whose
   position, size, color/material, or sibling layout is inconsistent.
3. Compare the target bbox with every sibling bbox under the same parent. Existing
   known siblings are strong constraints; masked siblings may still provide layout
   evidence through their bbox and color but not their hidden tag.
4. Use bbox_average_rgb and visual color/material continuity as supporting evidence.
   Similar color/texture can indicate another segment of the same object; very
   different color may indicate clothing, shoe, material detail, or texture.
5. Avoid impossible duplicate anatomy or object parts unless the image and layout
   clearly support duplication. Choose "unknown" only if no candidate label is
   spatially and visually plausible, but prefer a valid candidate when evidence is
   sufficient.

Examples of the intended reasoning:
- If parent is person and known siblings already include left/right arm, a third
  arm candidate at the lower body is usually wrong; a lower bbox near feet is more
  likely leg, foot, shoe, or clothing depending on shape and color.
- If parent is person's leg and the target bbox is at the bottom end with a shoe-like
  color/shape, prefer the candidate whose leaf is foot or shoe over hand or arm.
- If parent is person's face/head, small upper symmetric boxes are likely eyes; a
  central box below eyes and above mouth is likely nose; a lower face box is more
  likely mouth/chin than eye.
- If parent is car/vehicle, lower left/right boxes are likely wheel/tire; boxes in
  the upper body are more likely window, door, mirror, or light.
- If one known sibling is a wheel on the lower left and the target is a similar
  colored/size lower-right box, the target is likely another wheel candidate.
```

A.4 Additional Qualitative Results

We compare the SVG outputs generated by DiffVG [14], LIVE [20], LIVSS [18], and our method under the same point-budget setting. These examples show that Lang-SVG better preserves the visual structure of natural images while maintaining high rendering fidelity. We also include more semantic SVG tagging examples, where our tree-guided semantic assignment predicts meaningful object, part, and subpart labels for SVG components.

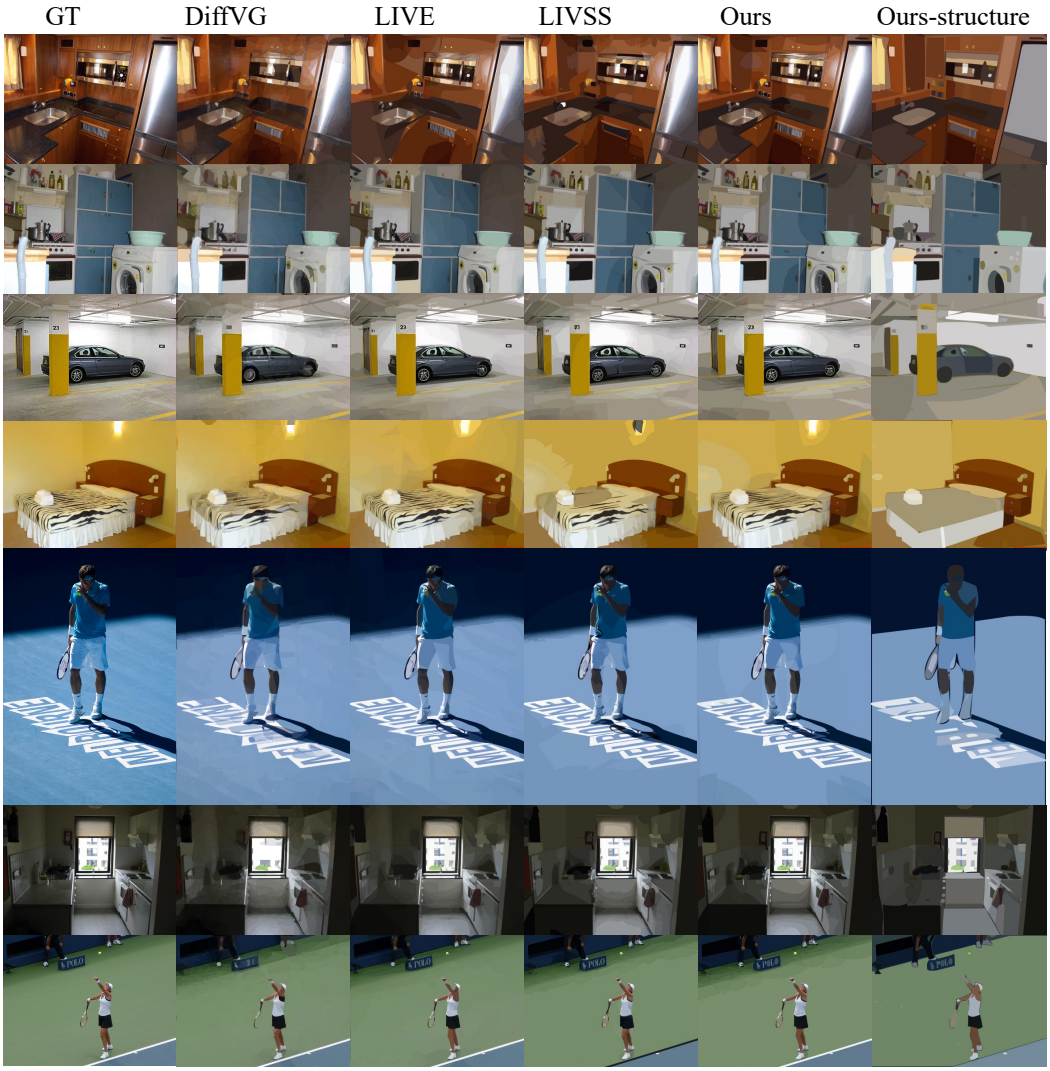

Figure 6: **More comparison results.** We compare SVGs produced by DiffVG [14], LIVE [20], LIVSS [18], and our method. Our SVGs better align with the target images and preserve more coherent visual structures.

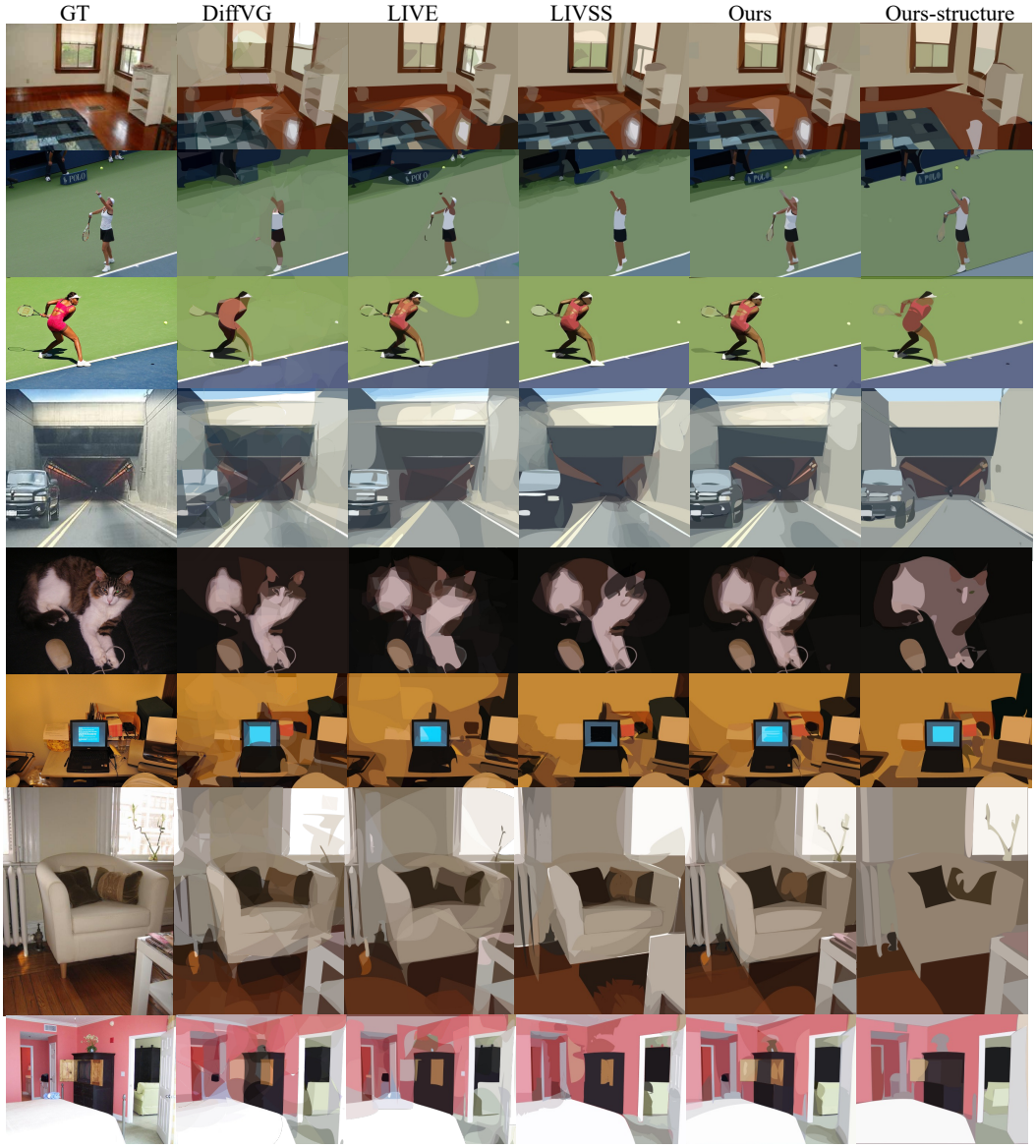

Figure 7: **More comparison results under a lower point budget.** We compare SVGs produced by DiffVG [14], LIVE [20], LIVSS [18], and our method. Our method produces SVGs with better visual fidelity and more coherent structure under the same point budget.

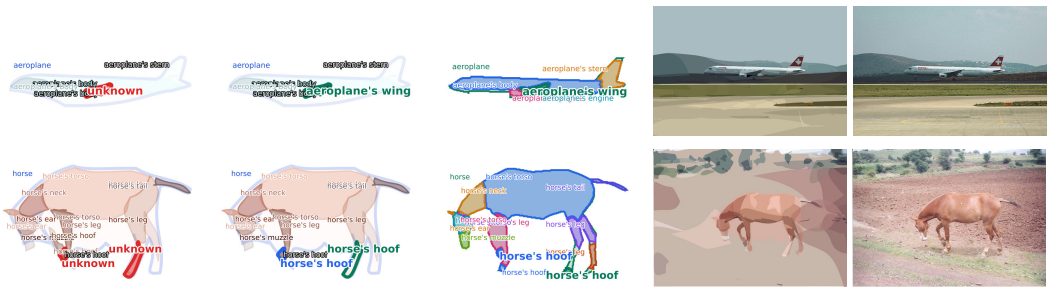

Figure 8: **More semantic SVG tagging results.** We show additional examples of semantic tags predicted for SVG components using our tree-guided semantic assignment.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the main contributions.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a "Limitations and Future Work" subsection.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper presents an empirical system with no theoretical proofs or lemmas.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The methodology is described in sufficient detail in Section 3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code will be released at https://xiliu8006.github.io/Lang-SVG/.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the datasets, point/path budgets, evaluation metrics, and baselines. Full hyperparameter details are provided in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Error bars are not reported. The evaluation is deterministic given fixed inputs and a fixed VLM (Qwen3.5-27B), so variance across runs is expected to be negligible.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Experimental settings are included inside Section 4.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research involves image vectorization using publicly available datasets and models. No sensitive data, human subjects, or dual-use concerns are present.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work focuses on image vectorization for structured editing. It does not directly enable harmful applications. Potential positive impacts include more accessible graphic design tools; no significant negative societal impacts are foreseen.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The method produces SVG representations of images and does not release a generative model with misuse potential. No safeguards beyond standard practice are necessary.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets (ADE20K, COCO, Pascal-Part, IconFont) and models (SAM, Grounded SAM, Semantic-SAM, UnSAM, Qwen3.5-27B) are properly cited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper does not release a new dataset. The evaluation protocol and code will be released upon acceptance; no new standalone asset requiring separate documentation is introduced.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: No crowdsourcing or human subject experiments were conducted. All evaluations are automatic using ground-truth annotations from existing datasets.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects or crowdsourcing.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes],

    Justification: Qwen3.5-27B is an important and original component of the core method, used for tree-guided semantic tagging of SVG primitives (Section 3.3). Its role, prompting strategy, and evaluation are described in detail.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.