Week 9
Structured Vector Graphics Editing With VLMs

Matthew Perlman

James Beetham, Dr. Amrit Bedi, Dr. Mubarak Shah

Progress This Week – REU/PG26 Poster

UMassAmherst | Manning College of Information & Computer Sciences

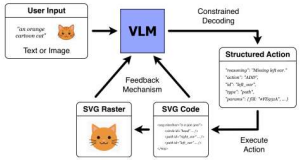
Evaluating Constrained Iterative Refinement for SVG Generation with Off-the-Shelf VLMs

Matthew Perlman[1], James Beetham[2], Niels da Vitoria Lobo[2], Amrit Singh Bedi[2], Mubarak Shah[2]
[1] University of Massachusetts Amherst, [2] University of Central Florida

UCF Institute of Artificial Intelligence, University of Central Florida

ABSTRACT

SVGs power much of the modern visual ecosystem, yet the recent wave of generative models focus almost entirely on rasterized images. We explore whether inference-time scaffolding can unlock SVG generation capabilities in off-the-shelf vision-language models (VLMs). We introduce a constrained iterative refinement framework that combines visual feedback, structured editing, and grammar-constrained decoding to address the challenges many VLMs face when generating SVGs. Across multiple VLMs and generation settings, we find that constrained decoding improves the rate of successful generation, while iterative refinement reveals a deficit in visual reasoning and self-correction. Our results highlight both the promise and current limitations of using inference-time scaffolding to adapt general-purpose VLMs for vector graphics generation.



Our Constrained Iterative Refinement Harness.

INTRODUCTION

Scalable Vector Graphics (SVGs) are widely used for web, interface, and illustration content because they are resolution-independent, compact, and directly editable. Despite the practical benefits of SVGs, generating them remains challenging for general-purpose vision-language models (VLMs): the model must jointly produce valid code, reason about spatial relationships, and translate precise visual details into a new medium. Existing SVG generation methods address these challenges through specialized architectures or task-specific training, decoupling them from the rapidly improving ecosystem of general-purpose multimodal models.

We investigate whether inference-time scaffolding is sufficient to make off-the-shelf VLMs effective SVG generators. In particular, we systematically evaluate iterative visual refinement, three levels of editing structure, and grammar-constrained decoding for both text-to-SVG and image-to-SVG generation across multiple model scales. Our study characterizes when these mechanisms improve syntactic reliability, how they affect visual quality, and how well current VLMs can effectively translate rendered feedback into changes to the underlying vector representation.

SYSTEM OVERVIEW

We introduce a harness that equips off-the-shelf VLMs with two complementary, inference-time mechanisms: iterative refinement based on rendered visual feedback and grammar-constrained decoding that restricts outputs to appropriate SVG and JSON syntax. Together, these mechanisms target the weaknesses of general-purpose VLMs, allowing the model to iteratively refine and correct its output while remaining bounded to a syntactically valid generation space.

QUANTITATIVE RESULTS

We evaluate our framework on MMSVG-Bench across multiple model scales (Qwen3.5 9B and 27B), with and without grammar-constrained decoding, and across all four generation pipelines. Metrics are computed over successfully compiled images, then averaged equally across the two models.

		Text→SVG				Image→SVG			
Decoding	**Pipeline**	**Compile↑**	**CLIP↑**	**Aesthetic.↑**	**#Tokens**	**Compile↑**	**DINO↑**	**LPIPS↓**	**#Tokens**
Unconstrained	Single Turn	**0.837**	**0.255**	**4.655**	1.4k	0.630	**0.704**	0.396	1.7k
	Unstructured	0.475	0.246	4.560	4.8k	0.113	**0.704**	**0.372**	10.7k
	Semi-Structured	0.180	0.205	4.415	7.9k	0.500	0.617	0.418	8.8k
	Structured	0.785	0.226	4.480	3.3k	**0.650**	0.633	0.413	8.1k
Constrained	Single Turn	**0.830**	0.220	**4.530**	0.6k	**0.830**	0.411	0.579	0.4k
	Unstructured	0.599	**0.242**	4.485	5.2k	0.647	**0.617**	0.430	5.1k
	Semi-Structured	0.512	0.206	4.445	2.7k	0.375	0.464	0.506	3.2k
	Structured	0.790	0.218	4.440	2.4k	0.695	0.603	**0.420**	3.3k

Results averaged across Qwen3.5 9B and 27B on MMSVG-Bench.

QUALITATIVE RESULTS

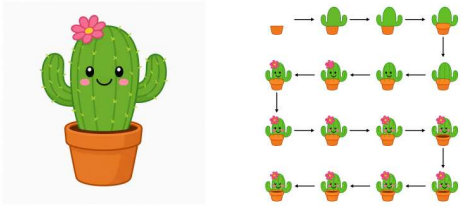
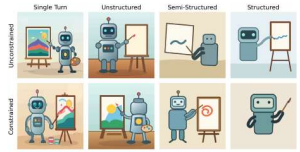


Image-to-SVG generation trajectory produced by GPT-5.6 with unconstrained structured actions. Left: reference image. Right: intermediate SVG renders across 16 iterative refinement steps.



Images generated by GPT-5.6 using each pipeline with the Text-to-SVG prompt "A robot painting".

DISCUSSION

Constrained Decoding Can Improve Compilation. We find that constrained decoding substantially improves compilation rates for most pipelines, a result consistent with our initial hypothesis.

Constrained Decoding Hurts Visual Quality. Across visual metrics and generation settings, constrained decoding consistently lowered image quality. We hypothesize that enforcing our limited grammar steers token generation toward unlikely and visually flawed SVGs.

Small Models Struggle With Visual Feedback. We find that for the models tested, iterative refinement has an inconsistent effect on visual quality, and often lowers compilation rates by introducing more opportunities to fail. Based on empirical results, we hypothesize these small models struggle to connect visual inconsistencies with the underlying SVG code.

FUTURE WORK

Several promising directions remain for improving inference-time scaffolding. Richer action spaces, alternative visual feedback mechanisms, stronger constrained decoding strategies, and prompt meta-optimization all have the possibility of improving generation quality and success rate. We also believe different evaluation metrics may better capture image quality and prompt adherence.

Beyond extending our harness, a deeper mechanistic analysis of these models' failure to incorporate visual feedback may inform improved general-purpose VLM spatial-reasoning capabilities.

ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation through the University of Central Florida Computer Vision REU Program. We thank the UCF Institute for Artificial Intelligence for their computing resources and guidance throughout this project.

Progress This Week – REU Report Drafted



Evaluating Constrained Iterative Refinement for SVG Generation with Off-the-Shelf VLMs

Matthew Perlman[1], James Boetham[2], Niels Da Vitoria Lobo[2], Amrit Bedi[2], Mubarak Shah[2]

[1]University of Massachusetts Amherst, [2]University of Central Florida

July 23, 2026

Abstract

Scalable Vector Graphics (SVGs) are widely used for web interfaces, illustrations, and other visual content due to their resolution independence, compact representation, and direct editability. Despite these advantages, recent advances in generative AI have largely focused on raster image generation rather than structured vector representations. Existing SVG generation approaches typically rely on specialized architectures or task-specific training, limiting their ability to directly benefit from improvements in general-purpose vision-language models (VLMs). In this work, we investigate whether off-the-shelf VLMs can leverage their existing reasoning capabilities in self-the-shelf VLMs without additional fine-tuning. We introduce Constrained Iterative Refinement, a framework that augments general-purpose VLMs with two complementary mechanisms: iterative visual feedback and grammar-constrained decoding. Iterative refinement enables models to revise generated SVGs using rendered visual feedback, while constrained decoding restricts outputs to valid SVG and structured action formats. We evaluate these techniques across text-to-SVG and image-to-SVG generation using Qwen3-5-9B and Qwen3-5-27B on MMSVG-Bench. We find that constrained decoding substantially improves generation reliability, while iterative refinement exposes limitations in current VLM visual reasoning and self-correction abilities. These results highlight both the promise of inference-time adaptation for structured graphics generation and the remaining challenges in enabling general-purpose multimodal models to effectively reason over editable visual representations.

1 Introduction

Scalable Vector Graphics (SVGs) are a fundamental representation for modern visual content, powering web graphics, user interfaces, icons, and illustrations. Unlike raster images, SVGs describe images through structured programs consisting of geometric primitives, paths, and attributes. This representation provides several practical benefits, including resolution independence, compact storage, and direct human editability.

Recent advances in generative AI have produced increasingly capable image generation systems, yet most approaches remain focused on rasterized outputs. Generating SVGs introduces additional challenges beyond visual synthesis: a model must produce syntactically valid code, reason about spatial relationships between objects, select appropriate geometric primitives, and translate visual intent into an executable representation. These requirements make SVG generation a challenging test of both visual understanding and structured code generation.

Existing SVG generation approaches have made significant progress through specialized architectures and targeted training on large SVG datasets. However, these systems are typically optimized for a specific task and require additional training infrastructure. Meanwhile, general-purpose VLMs continue to improve rapidly through increasingly diverse multimodal pretraining. This motivates a different direction: rather than training dedicated SVG generators, can inference-time techniques adapt existing VLMs to structured graphics generation?

This question follows a broader trend in code generation, where inference-time strategies such as constrained decoding, iterative refinement, and tool-based feedback have significantly improved the reliability of general-purpose models. These techniques allow models to operate within structured environments without modifying model parameters, providing a potentially scalable alternative as foundation models continue to improve.

In this work, we explore whether similar ideas can enable off-the-shelf VLMs to generate SVGs. We introduce Constrained Iterative Refinement, an inference-time framework that combines two complementary mechanisms:

- **Iterative visual refinement**: models repeatedly inspect their rendered SVG output and apply targeted edits.
- **Grammar-constrained decoding**: models generate SVG programs and structured actions under explicit syntactic constraints.

We evaluate these techniques across multiple generation pipelines and model scales, analyzing their effects on compilation reliability, visual similarity, and generation efficiency. Beyond improving immediate generation performance, our results provide insight into current VLM limitations, particularly their difficulty translating visual feedback into precise code modifications.



Our contributions are:

- We introduce an inference-time framework for adapting general-purpose VLMs to SVG generation without task-specific fine-tuning.
- We systematically evaluate iterative refinement and constrained decoding across text-to-SVG and image-to-SVG generation.
- We analyze the strengths and limitations of current VLMs for structured visual generation and identify directions for improving inference-time adaptation.

2 Related Work

2.1 SVG Generation

Recent work has explored several approaches for automatically generating SVG representations from text and images. A common strategy is to train specialized models directly on SVG datasets, allowing models to learn the relationship between visual concepts and vector representations.

VectorFusion [7] explores converting raster image generation capabilities into vector generation by optimizing SVG parameters using diffusion model guidance. Rather than directly predicting SVG code, the method uses pretrained image generation models to guide vector optimization. Similarly, SVG-Dreamer [7] uses diffusion-based optimization to generate vector graphics by iteratively refining SVG representations according to learned visual priors. While these approaches benefit from strong image generation models, they rely on optimization procedures and are not directly compatible with general-purpose VLM reasoning.

Other work has investigated fine-tuning on training multimodal models specifically for SVG generation. **StarVector** [7] introduces a vision-language model trained to directly generate SVG code from images by combining visual encoding with code generation capabilities. **OmniSVG** [7] proposes a large-scale SVG understanding and generation framework trained on diverse vector graphics data, enabling both text-to-SVG and image-to-SVG tasks. **Chat2SVG** [7] explores conversational SVG generation, allowing users to iteratively interact with a model to modify vector graphics. **InstructSVG** [7] studies instruction-following SVG generation through specialized training on vector editing tasks.

These methods demonstrate strong SVG generation capabilities, but they generally require dedicated training on SVG-specific datasets. Our work explores a complementary direction: whether existing VLMs can be adapted at inference time through structured interaction and output constraints, allowing improvements from general multimodal model scaling to transfer naturally to SVG generation.

2.2 Structured Generation and Code Models

SVG generation is closely related to structured code generation, where models must produce outputs satisfying both semantic and syntactic requirements. Recent advances in code generation have shown that inference-time strategies can significantly improve model reliability without additional training.

Grammar-constrained decoding methods restrict generation to valid formal languages, preventing syntax errors during program generation [7, 7]. Similar approaches have improved reliability in tasks such as SQL generation, program synthesis, and structured JSON generation. Self-refinement approaches [7] and reflective coding agents extend this idea by allowing models to iteratively inspect and improve their own outputs using feedback.

These approaches motivate our use of inference-time scaffolding for SVG generation. Unlike much prior code generation, however, SVG generation requires models to reason about both program structure and visual appearance. Our framework combines structured output constraints from code generation with visual feedback mechanisms from multimodal systems.

3 Method

3.1 System Overview

We introduce Constrained Iterative Refinement, a framework that equips off-the-shelf VLMs with two inference-time mechanisms designed to address common SVG generation failures: unreliable structured output and limited visual self-correction.

Figure 1 provides an overview of our framework. Given either a text prompt or reference image, the model generates an SVG representation through one of four pipelines: single-turn generation, unstructured refinement, semi-structured refinement, or structured refinement. For iterative approaches, the model repeatedly receives the original instruction, current SVG code, and rendered SVG image, and produces an updated representation.

All generation pipelines require the model to output structured JSON, containing both reasoning and generation information. For example, single-turn generation produces:

```
{
  "reasoning": "...",
  "code": "<svg>...</svg>"
}
```

where the code field must contain valid SVG XML. Iterative pipelines extend this format by requiring either complete replacement code or explicit edit actions.

3.2 Iterative Refinement

Rather than requiring a model to generate a complete SVG in a single pass, iterative refinement treats SVG generation as a sequence of edits. At each iteration, the model observes the original instruction, current SVG source code, and the rasterized output of the SVG. The model does not receive previous reasoning traces or earlier iterations.

We evaluate three refinement strategies.

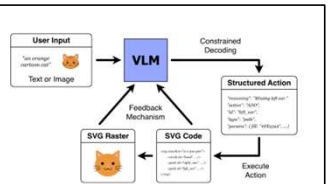


Figure 1: Overview of the Constrained Iterative Refinement framework. A VLM generates and refines SVG programs using rendered visual feedback and structured decoding constraints.

Unstructured Refinement. At every iteration, the model generates an entirely new SVG program. This provides maximum flexibility but requires the model to repeatedly reconstruct the entire image representation.

Semi-Structured Refinement. The model produces an explicit action type (*Add, Update, Remove, or Exit*), a target element identifier, and an SVG code block when necessary. This reduces the burden of regenerating unchanged elements while still allowing the model to directly write SVG code.

Structured Refinement. The model produces the same action space but instead specifies parameterized SVG elements through JSON. For example, an update operation may specify a target circle and its attributes:

```
{
  "action":"Update",
  "id":"circle1",
  "type":"circle",
  "cx":125,
  "cy":250,
  "r":150,
  "fill":"#FF0000"
}
```

Internally, structured refinement maintains a lightweight mapping from element identifiers to their parameters, which is serialized into SVG XML after each update.

Unstructured refinement is limited to ten iterations, while semi-structured and structured refinement allow up to twenty iterations. These limits were selected based on empirical observations and computational constraints, as additional iterations rarely produced useful edits.

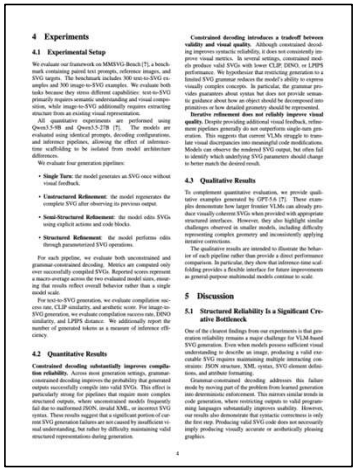
3.3 Grammar-Constrained Decoding

Many VLM failures during SVG generation occur before visual quality is considered: malformed JSON, invalid XML, or incorrect SVG syntax can prevent outputs from being rendered entirely. We therefore investigate inference-time constrained decoding.

For closed-source models, we apply JSON schema constraints that enforce the expected output format, including required fields and property ordering. However, these constraints do not directly restrict the generated SVG content itself.

For open-source models, we additionally apply context-free grammar (CFG) constraints through vGrammar integrated with vLLM. The grammar defines a subset of valid SVG syntax, restricting generation to supported primitives such as rectangles, circles, ellipses, paths, and text. The grammar enforces valid element names, attribute names, and attribute types, while allowing model-generated values such as coordinates and colors.

This constraint intentionally does not encode higher-level semantic constraints. For example, it does not restrict the number of elements, guarantee reasonable coordinate ranges, or fully validate complex path structures. These limitations remain challenging failure modes for current models and motivate future extensions of the framework.



4 Experiments

4.1 Experimental Setup

We evaluate our framework on MMSVG-Bench [7], a benchmark containing paired text prompts, reference images, and a SVG targets. The benchmark includes 300 text-to-SVG examples and 300 image-to-SVG examples. We evaluate both tasks because they stress different capabilities: text-to-SVG primarily requires semantic understanding and visual composition, while image-to-SVG additionally requires extracting structure from an existing visual representation.

All quantitative experiments are performed using Qwen3-5-9B and Qwen3-5-27B [7]. The models are evaluated using identical prompts, decoding configurations, and inference pipelines, allowing the effect of inference-time scaffolding to be isolated from model architecture differences.

We evaluate four generation pipelines:

- **Single Turn:** the model generates an SVG once without visual feedback.
- **Unstructured Refinement**: the model regenerates the complete SVG after observing its previous output.
- **Semi-Structured Refinement**: the model edits SVGs using explicit actions and code blocks.
- **Structured Refinement**: the model performs edits through parameterized SVG operations.

For each pipeline, we evaluate both unconstrained and grammar-constrained decoding. Metrics are computed only over successfully compiled SVGs. Reported scores represent a macro-average across the two evaluated model sizes, ensuring that results reflect overall behavior rather than a single model scale.

For text-to-SVG generation, we evaluate compilation success rate, CLIP similarity, and aesthetic score. For image-to-SVG generation, we evaluate compilation success rate, DINO similarity, and LPIPS distance. We additionally report the number of generated tokens as a measure of inference efficiency.

4.2 Quantitative Results

Constrained decoding substantially improves compilation reliability. Across most generation settings, grammar-constrained decoding improves the probability that generated outputs successfully compile into valid SVGs. This effect is particularly strong for pipelines that require more complex structured outputs, where unconstrained models frequently fail due to malformed JSON, invalid XML, or incorrect SVG syntax. These results suggest that a significant portion of current SVG generation failures are not caused by insufficient visual understanding, but rather by difficulty maintaining valid structured representations during generation.

Constrained decoding introduces a tradeoff between validity and visual quality. Although constrained decoding improves syntactic reliability, it does not consistently improve visual metrics. In several settings, constrained models produce valid SVGs with lower CLIP, DINO, or LPIPS performance. We hypothesize that restricting generation to a limited SVG grammar reduces the model's ability to express visually complex concepts. In particular, the grammar provides guarantees about syntax but does not provide semantic guidance about how an object should be decomposed into primitives or how detailed geometry should be represented.

Iterative refinement does not reliably improve visual quality. Despite providing additional visual feedback, refinement pipelines generally do not outperform single-turn generation. This suggests that current VLMs struggle to translate visual discrepancies into meaningful code modifications. Models can observe the rendered SVG output, but often fail to identify which underlying SVG parameters should change to better match the desired result.

4.3 Qualitative Results

To complement quantitative evaluation, we provide qualitative examples generated by GPT-5.6 [7]. These examples demonstrate how larger frontier VLMs can already produce visually coherent SVGs when provided with appropriate structured interfaces. However, they also highlight similar challenges observed in smaller models, including difficulty representing complex geometry and inconsistently applying iterative corrections.

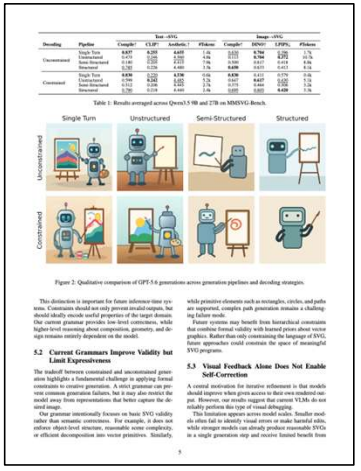
The qualitative results are intended to illustrate the behavior of each pipeline rather than provide a direct performance comparison. In particular, they show that inference-time scaffolding provides a flexible interface for future improvements as general-purpose multimodal models continue to scale.

5 Discussion

5.1 Structured Reliability Is a Significant Creative Bottleneck

One of the clearest findings from our experiments is that generation reliability remains a major challenge for VLM-based SVG generation. Even when models possess sufficient visual understanding to describe an image, producing a valid executable SVG requires maintaining multiple interacting constraints: JSON structure, XML syntax, SVG element definitions, and attribute formatting.

Grammar-constrained decoding addresses this failure mode by moving part of the problem from learned generation into deterministic enforcement. This mirrors similar trends in code generation, where restricting outputs to valid programming languages substantially improves usability. However, our results also demonstrate that syntactic correctness is only the first step. Producing valid SVG code does not necessarily imply producing visually accurate or aesthetically pleasing graphics.



Decoding	Pipeline	Text→SVG Compile↑	Text→SVG CLIP↑	Text→SVG Aesthetic↑	Text→SVG #Tokens	Image→SVG Compile↑	Image→SVG DINO↑	Image→SVG LPIPS↓	Image→SVG #Tokens
Unconstrained	Single Turn	**0.837**	0.255	4.655	1.4k	0.830	**0.704**	0.266	1.7k
	Unstructured	0.753	0.253	4.618	1.4k	0.753	0.697	0.271	1.7k
	Semi-Structured	0.140	0.228	4.275	7.9k	0.595	0.615	0.414	8.0k
	Structured	0.760	0.226	4.480	3.5k	**0.850**	0.653	0.413	8.1k
Constrained	Single Turn	**0.936**	0.220	**4.730**	0.6k	**0.870**	0.411	0.579	0.6k
	Unstructured	0.799	**0.241**	4.655	5.2k	0.817	**0.637**	0.430	5.1k
	Semi-Structured	0.747	0.234	4.645	5.2k	0.660	0.602	0.447	5.1k
	Structured	0.780	0.218	4.660	1.4k	0.693	0.602	**0.428**	1.3k

Table 1: Results averaged across Qwen3-5 9B and 27B on MMSVG-Bench.

Figure 2: Qualitative comparison of GPT-5.6 generations across generation pipelines and decoding strategies.

This distinction is important for future inference-time systems. Constraints should not only prevent invalid outputs, but should ideally encode useful properties of the target domain. Our current grammar provides low-level correctness, while higher-level reasoning about composition, geometry, and design remains entirely dependent on the model.

5.2 Current Grammars Improve Validity but Limit Expressiveness

The tradeoff between constrained and unconstrained generation highlights a fundamental challenge in applying formal constraints to creative generation. A strict grammar can prevent common generation failures, but it may also restrict the model away from representations that better capture the desired image.

Our grammar intentionally focuses on basic SVG validity rather than semantic correctness. For example, it does not enforce object-level structure, reasonable scene complexity, or efficient decomposition into vector primitives. Similarly, while primitive elements such as rectangles, circles, and paths are supported, complex path generation remains a challenging failure mode.

Future systems may benefit from hierarchical constraints that combine formal validity with learned priors about vector graphics. Rather than only constraining the language of SVG, future approaches could constrain the space of meaningful SVG programs.

5.3 Visual Feedback Alone Does Not Enable Self-Correction

A central motivation for iterative refinement is that models should improve when given access to their own rendered output. However, our results suggest that current VLMs do not reliably perform this type of visual debugging.

This limitation appears across model scales. Smaller models often fail to identify visual errors or make harmful edits, while stronger models can already produce reasonable SVGs in a single generation step and receive limited benefit from additional refinement. This suggests that scaling alone may not automatically produce better iterative behavior.

The challenge is not simply recognizing that an output differs from a target image. Rather, the model must understand how changes to an underlying symbolic representation affect the rendered result. Developing stronger visual reasoning over editable representations remains an important direction for future multimodal systems.

5.4 Action Spaces Must Match the Model's Training Distribution

The difference between semi-structured and structured refinement provides an additional insight into designing inference-time interfaces. Explicit action spaces can reduce generation complexity, but only when the interface matches representations the model can naturally reason about.

Semi-structured refinement combines two different forms of generation: deciding an editing action while simultaneously writing arbitrary SVG code. This hybrid representation may introduce unnecessary difficulty. In contrast, both full SVG generation and fully structured JSON operations provide clearer targets.

This suggests that future scaffolding methods should not only constrain outputs, but should carefully design intermediate representations that align with the capabilities learned during pretraining.

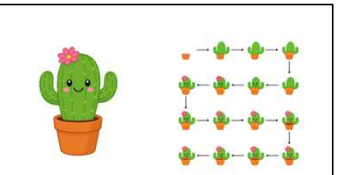


Figure 3: Intermediate SVG renders produced during image-to-SVG iterative refinement.

6 Future Work

Our results highlight several opportunities for improving inference-time SVG generation. One promising direction is developing stronger constraint mechanisms that move beyond syntactic validity toward semantic guidance. For example, future grammars could incorporate learned priors about scene structure, primitive selection, or efficient vector decomposition.

Another direction is improving visual feedback mechanisms. The current framework provides the model only with rendered images and SVG source code. More informative feedback signals, such as object-level comparisons, spatial alignment information, or automated SVG analysis, may help models better connect visual errors with specific code changes.

Improved refinement strategies may also require better intermediate representations. Rather than directly editing SVG code, future systems could operate over higher-level scene graphs, object relationships, or learned vector representations.

Finally, evaluation remains an open challenge. Existing similarity metrics capture only limited aspects of SVG quality. Future work could investigate metrics that better measure editability, compactness, semantic correctness, and human preference.

More broadly, this work explores a hypothesis that increasingly capable general-purpose VLMs may eventually outperform specialized generation systems when paired with effective inference-time scaffolding. While current models still struggle with structured visual reasoning, improving the interface between models and structured environments may provide a scalable path toward more capable multimodal systems.

Next Steps

- Finalize REU Poster and Report
- Assist James with meta-optimization experiments
- Learn more about graduate school opportunities
  - NSF GRFP
  - Other Fellowships
  - Undergrad Research Programs
  - Grad School Candidates