ReViz: Structured Visual Synthesis of Research Methodology Diagrams

Anonymous ACL submission

Abstract

Generating publication-quality diagrams using free-form methodology text is a difficult problem. Researchers need diagrams that faithfully represent how a method works: its components, their relationships, and the flow of information through the system. This requirement is what we call **scientific fidelity**. Existing approaches either treat diagram generation as unconstrained image synthesis, which sacrifices semantic precision, or depend on labeled samples to guide generation, which places a heavy curation burden on the author. We present **ReViz**, a multi-agent system that converts a free-form methodology description into a publication-quality diagram. ReViz organizes this task as a **Research**, **Strategy**, **Execution**, and **Rasterization** lifecycle. Each phase is carried out by a dedicated set of agents matched to those demands. Intermediate representations are made explicit and inspectable at every stage, so errors are caught early rather than propagated silently. As a result, researchers can examine and adjust the output at each step rather than accepting the diagram as a black box. Evaluation on established benchmark shows that ReViz generates diagrams with high scientific fidelity and visual quality.

1 Introduction

A methodology diagram is a structured abstraction of a research paper's technical argument, compressing several kinds of information into a single visual artifact. A reader should be able to reconstruct the high-level logic of the method from the figure alone. Producing such a figure automatically requires understanding the scientific process, deciding which entities become visual components, organizing those components into a readable layout, and rendering the result under publication constraints. Errors at any step propagate to the final output: a wrong semantic structure produces a misleading diagram even when the layout is clean; a poor layout makes a faithful graph unreadable; an unconstrained rendering may lose text labels, reverse arrow directions, or drop technical symbols that carry scientific meaning (Zhang et al., 2025; Hu et al., 2026).

Recent advances across three areas have made automated scientific figure generation practically feasible: (i) Large language models now show strong reasoning ability, allowing them to interpret complex methodologies and infer structured relationships between components; (ii) Text-to-image models produce high-fidelity visuals from detailed prompts; (iii) Vision-language models can evaluate generated images against structured criteria, enabling automated quality checks without human review. Beyond individual models, the agentic paradigm has shifted the generic outcome oriented approaches into task specifc workflows (Wang et al., 2025b; Chen et al., 2025). These developments together create the conditions for a system that can handle the full chain from methodology text to publication-quality figure.

The most direct approach feeds methodology text to a general-purpose image model and asks it to produce a diagram. But this appraoch requires the model to peform several important tasks at once implicitly (Zhang et al., 2025; Parthasarathy et al., 2026). Text-based unstructured agentic systems like PaperBanana (Zhu et al., 2026a) orchestrate LLMs and image models in iterative refinement loops but still operate without an explicit structured intermediate. While such unstructured agentic approaches perform distinct tasks, both they have a core limitation; the lack of clearly deifned structured intermediataries makes the outcomes less robust. This may result in missed components, incorrect flows, and inconsistent information extraction across independent generation attempts of the same diagram.

More organized approaches introduce structured intermediate representations (a component-and-

connection graph or a vector diagram) before the final image is produced (Zhu et al., 2026b; Zala et al., 2023; Huang et al., 2026). These systems improve label control and editability over direct generation. However, three problems remain. First, spatial arrangement is committed to after a single candidate is produced; without exploring alternative floor plans, a layout that is topologically valid but tighly matched to the initial structure reaches the rendering stage unchallenged. Second, symbolic assets such as icons and scene panels are either omitted or described inline within the same prompt that governs global layout. This forces the model to resolve per-component visual detail and diagram-wide structure simultaneously; With the competing instructions, adding well detailed problem specific visual elements becomes challenging. Third, no deterministic check verifies that every connection present in the extracted source survived the rendering process. Structural omissions are therefore only detectable at the final raster image stage, where the model's own judgment is used to evaluate the final output. This creates a risk of undetected structural errors, that propaate through the intermediate data structures and into the final output.

To address these limitations, we present ReViz, a multi-agent pipeline for generating publication-quality methodology diagrams from research paper text. ReViz addresses the gaps above through four contributions: ❶ a four-stage structured pipeline where every intermediate artifact is inspectable and correctable before the next stage begins; ❷ explicit layout exploration before commitment; ❸ independent detailed artifact generation; and, ❹ a deterministic connection verification step that ensures that intermediate steps faithfully transfer structural information. The result is a system that can be inspected and corrected at every stage, not only at the final output, and where the model assigned to each stage is can be matched to what that stage demands (Zhu et al., 2026b; Wei et al., 2025; Huang et al., 2026).

2 Background and Related Work

This section reviews the challenges of automatic methodology diagram generation and situates our approach relative to existing work. We identify two key limitations of single-step text-to-image generation: the difficulty of balancing multiple conflicting objectives in a single prompt, and the heterogeneity of subtask requirements that no single model can optimally handle. We then discuss how these limitations motivate a modular pipeline architecture with explicit intermediate representations, and how this design compares to prior approaches in the literature.

2.1 From Free-Form Generation to Structured Visual Synthesis

A more robust approach introduces **intermediate representations** that separate the problem into stages
blue (Lin et al., 2026; Ghosh et al., 2025). The methodology can first be converted into an explicit component-and-connection structure, capturing entities, phase memberships, and typed relationships. A layout can then be planned over that structure before any rendering is attempted. A vector representation can serve as an inspectable intermediate for labels, boxes, arrows, and phase regions. Raster generation can then be constrained by the vector structure rather than asked to invent the diagram from free-form text. This staged formulation changes the task from unconstrained image synthesis to structured visual synthesis, where each step receives a well-defined input and produces a well-defined output.

The need for explicit intermediate structure is especially strong for scientific methodology diagrams. General image models struggle with small text, precise arrow directions, and repeated technical labels at the density typical of research figures blue (Rodriguez et al., 2023; Garg et al., 2026). Beyond rendering quality, scientific figures carry semantic constraints that purely aesthetic criteria cannot capture. The same visual component may represent different scientific roles depending on its phase context. A single arrow may imply a data flow, a control signal, or a feedback path. The representation supplied to layout and rendering stages must therefore preserve not only the entities but also the typed relationships between them —otherwise a visually clean figure may silently misrepresent the described method.

2.2 Heterogeneous Task Requirements and Model Selection

A second limitation of single-step generation is that different subtasks require fundamentally different model capabilities. No single model is uniformly best at all of them, which motivates a modular architecture that can assign each subtask to a model

	PaperBanana (Zhu et al., 2026a)	**AutoFigure** (Zhu et al., 2026b)	**DiagrammerGPT** (Zala et al., 2023)	**ReViz (ours)**
Agentic pipeline	✓	✓	✓	✓
Structured intermediates	✗	✓	✓	✓
Dedicated layout exploration	✗	✗	✗	✓
Standalone asset generation	✗	✗	✗	✓
Agent-specific model support	✓	✓	✗	✓
Human-in-the-loop support	✗	✓	✗	✓

Table 1: Comparison of ReViz with related agentic scientific figure generation systems. Structured intermediates refers to machine-parsable representations (e.g. JSON plans, SVG blueprints, coordinate graphs) that decouple semantic structure from rendering. All four systems are agentic, but only ReViz supports dedicated divergent layout exploration, standalone asset generation, and human-in-the-loop review.

suited for it
blue (Zhu et al., 2026b; Wei et al., 2025).

Semantic structuring. Extracting a structured visual inventory from methodology prose requires high reasoning ability: inferring the scientific purpose of each component, discriminating real computational steps from abstract operations, and preserving only direct connections rather than implied or transitively derived ones. It also requires controlled creativity, because the system must produce concise and concrete visual hints for components that will later need icon representations. Reasoning and creativity must work together here; neither alone is sufficient.

Layout planning. Organizing extracted components on a canvas requires understanding graph topology and phase structure — which calls for reasoning — but also the ability to generate and compare alternative spatial organizations, which calls for divergent thinking. A linear layout communicates a sequential process; a multi-lane arrangement communicates parallel phases; a feedback-centered layout communicates iteration. Choosing among these requires exploring options before committing, not simply computing the optimal answer from the graph structure
blue (Zala et al., 2023; Huang et al., 2026).

Symbol construction. Assigning or authoring icons for components requires both semantic grounding and visual abstraction. A model must understand what a component means in context, decide whether a symbol would aid or clutter comprehension, and then select or describe a symbol that communicates the concept to a general technical reader. This involves reasoning about the component's role and creativity in mapping technical concepts to concrete visual analogues. Set-level consistency adds another constraint: different components in the same figure must not receive visually similar symbols with different meanings.

Vector implementation and repair. Producing a valid SVG is a program synthesis task
blue (Belouadi et al., 2025; Wang et al., 2025a). A model must write well-formed XML, maintain coordinate consistency, preserve text labels, route connectors without overlaps, and satisfy rendering constraints such as flat design and grayscale distinguishability. Repair is harder still. It requires interpreting a structured critic report, identifying the underlying structural cause of each issue, and modifying only the necessary parts of the SVG without introducing new inconsistencies. This combines strong coding ability with high reasoning ability in a way that differs from either unconstrained code generation or unconstrained analytical reasoning.

Prompt construction and styling. Writing an effective image-generation prompt requires technical writing skill and visual reasoning simultaneously. The prompt must communicate the exact structure of the figure — component identities, layout, connectors, labels — while also guiding typography, color, and provider-specific rendering behavior. Overly terse prompts omit essential detail. Overly long prompts introduce noise that degrades generation quality
blue (Zhu et al., 2026a; Liu et al., 2024). The best prompt balances scientific specificity with clear visual direction.

Raster image generation. Generating the final figure requires multimodal reasoning and the ability to follow external structural guidance. The model must respect an SVG layout, preserve labels, place

icon assets in the intended regions, and produce a polished figure without inventing components or omitting labeled nodes. These demands differ substantially from the skills needed for SVG code generation or semantic extraction.

This heterogeneity motivates a modular pipeline architecture in which the model assigned to each stage can be selected based on the capabilities that stage demands
blue (Guo et al., 2025; Meng et al., 2025). Models with stronger reasoning can handle semantic structuring and refinement. Models with stronger coding ability can handle vector implementation and repair. Models with stronger writing or multimodal rendering behavior can handle prompt construction and final image synthesis. This assignment is practical because the pipeline exposes well-defined intermediate artifacts, so each model operates on a constrained input rather than the full open-ended problem.

2.3 Positioning Relative to Existing Work

General image models are known to struggle with small text, precise arrow directions, and repeated technical labels (Tang et al., 2026; Fu et al., 2025) — all of which are central to research methodology diagrams, not peripheral details.

Vector-first workflows
blue (Wei et al., 2025; Rodriguez et al., 2025) improve editability and label control by producing structured SVG output. However, these approaches still need an accurate semantic representation of the methodology as input, and a mechanism for quality control after the first generation attempt. Without semantic extraction, a vector renderer may faithfully draw an incorrect graph. Without critique and repair, a first draft SVG can retain missing components, overlapping labels, or ambiguous connectors that carry incorrect scientific meaning.

Multi-agent LLM systems have been applied to code generation
blue (Wei et al., 2025; Dibia, 2023), scientific reasoning
blue (Huang et al., 2026; Chen et al., 2025), and complex task decomposition
blue (Zhu et al., 2026a,b). However, structured visual synthesis of scientific methodology figures presents a distinct combination of requirements — semantic fidelity, layout exploration, symbolic grounding, vector program synthesis, iterative quality control, and constrained raster generation — that existing multi-agent frameworks do not directly address.

ReViz combines these directions into a unified pipeline. It uses explicit methodology structure to anchor all downstream stages. It applies divergent layout exploration before committing to a spatial arrangement. It handles icon construction as a separate grounded reasoning and generation step. It implements and iteratively refines a vector diagram before any raster generation is attempted. Finally, it uses the vector artifact as a structural constraint on the raster model rather than as a secondary reference. The contribution is not only that the system produces a diagram, but that it decomposes the problem into stages whose outputs can be individually inspected, evaluated, and improved — and where the model assigned to each stage is matched to the specific capability that stage requires. Table 1 summarises how ReViz compares with the closest related systems across these dimensions.

3 Methodology

Generating publication quality figures from methodology text requires solving several distinct problems; understanding the addressed problem, important components/processes to represent, the interactions between those components, and the overall structure that correctly represents the methodology with easy to understand visual asthetics. ReViz decomposes these problems into a series of structured steps with specialized agents for each appropriate step. The pipeline is organized into four stages; ❶ Research: extracts structured information and produces a visual treatment plan, ❷ Strategy: explores suitable layouts and generates visual assets, ❸ Execution: implements the selected design as a vector diagram, ❹ Rasterization: generates a polished raster image through a refinement loop. Fig. 1 illustrates the end-to-end architecture.

The key design principle here is that each stage operates on the structured output of the previous stage, not on the original free-form text. This prevents ambiguity from propagating silently across stages and makes every intermediate artifact inspectable and correctable. In addition to the fully automated pipeline, ReViz also supports human-in-the-loop checkpoints after each major stage, allowing researchers to inspect intermediate artifacts and ensure scientific fidelity of the generated diagram. To make sure that the agents perform well structured reasoning, each agent is designed with

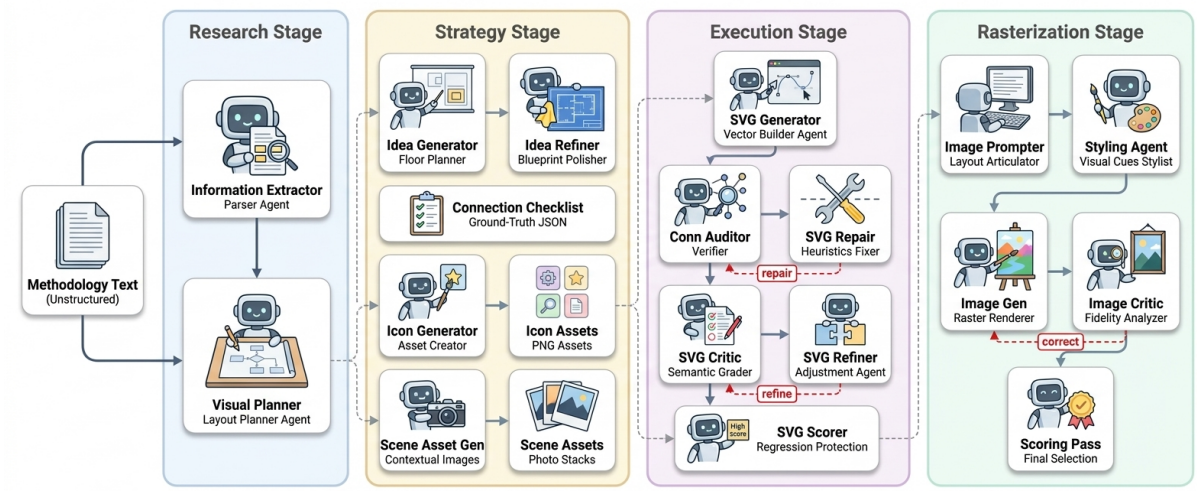


Figure 1: The ReViz pipeline consists of four stages: Research, Strategy, Execution, and Rasterization. Each stage produces inspectable artifacts that feed into the next. The final output is a publication-quality raster image and an editable SVG.

scratchpads that explores agent specific questions before generating their outputs.

3.1 Research Stage

The first step converts unstructured methodology text into a canonical structured representation that all downstream stages consume. The **information extractor** does not merely summarize the input. It identifies the exact visual inventory of the diagram: which components appear, in which phase, with what local role, and connected to what other components through what type of relationship. This strictness is deliberate. Later stages do not re-read the original paper text. They operate entirely over the extracted artifact. If a component name is incorrect, a connection is missing, or a phase boundary is wrong in the extraction, those errors propagate into the pipeline that follows. Getting the representation right at this stage is therefore a prerequisite for scientific fidelity downstream.

Knowing what components exist is not the same as knowing how they should be represented. Different components call for different visual treatments: a hardware sensor communicates more clearly as a photographic image than as a schematic icon; a learned feature extractor may be better represented by an abstract symbol than by a literal photograph of a GPU. If these decisions are left to individual downstream agents, they will be made inconsistently, because each agent has only a local view of the diagram and no single agent can enforce coherence across the full component set. To this end, ReViz introduces a **visual planner** that runs immediately after extraction, before any layout or asset work begins. The visual planner reads the complete extracted methodology and produces a structured JSON covering every component. For each component, the plan records whether the component needs an icon, whether it needs a scene asset such as a contextual photographic image, what visual treatment to apply, and the stylings of the artifact to generate. Every downstream agent that touches visual decisions gets this plan as a shared contract, allowing the pipeline to enforce consistent styles, avoid mixed visual registers within the same diagram region, and select palettes that remain distinguishable in grayscale print.

3.2 Strategy Stage

A methodology graph does not uniquely determine a good figure. The same set of components and connections can be arranged different visual layouts. The best choice depends on the dominant topology, the number of components, the presence of feedback, and how phases are emphasized in the paper's narrative. Committing to a layout too early, before comparing alternatives, risks producing a figure that is visually plausible but poorly matched to the method's structure.

To this end, ReViz separates layout exploration from layout commitment. The **idea generator** produces several candidate floor plans as abstract SVGs, using the extracted methodology and the visual plan as context. These floor plans encode component blocks, phase regions, and approximate connector routes. They do not yet apply final

styling; their purpose is to explore divergent spatial organizations that all represent the same extracted methodology.

The **idea refiner** then evaluates these candidates using the extracted methodology and rendered visual evidence from each. It selects the candidate that best balances methodology fidelity, flow clarity, phase separation, alignment, canvas utilization, and visual balance. Having selected a floor plan, the refiner improves it: adjusting spacing, normalizing block sizes, tightening phase groupings, and routing connectors so that the intended reading order is clear. The result is a refined spatial blueprint that the Execution stage will implement.

After the idea refiner completes, ReViz runs a semi-deterministic **connection checklist** step. This step parses the extracted methodology and builds a ground-truth JSON record of every required connection: source component, target component, edge kind, and label. This checklist is used in the Execution stage to verify the SVG's structural correctness before entering the critic loop. Separating ground-truth construction from SVG evaluation is important because it makes the verification step independent of the SVG generator's own assumptions. An LLM-based critic may rationalize an incorrect edge as intentional; a checklist comparison cannot.

Before SVG synthesis, ReViz generates two types of visual assets guided by the visual plan. The **icon generator** runs when the plan designates at least one component as needing an icon; it batches icon briefs into a grid sheet (eg: 2×2, 3×2, 4×2, or 3×3), generates all icons as one image for visual consistency, and splits the sheet into individual PNGs. The **scene asset generator** runs when the plan designates at least one component as needing a scene asset, producing contextual photographs for components where a photographic image communicates the concept more clearly than a schematic symbol. Both asset types are generated separately from the final raster prompt so that per-component visual detail does not compete with global layout and connection instructions at generation time.

3.3 Execution Stage

The **SVG generator** takes the refined floor plan, the extracted methodology, and the visual plan, and produces the first complete SVG implementation with remaing details (ie. labels), enforcing structural integrity. Before entering the critic loop, the **connection auditor** checks every connector in the SVG against the ground-truth checklist built in the Strategy stage. This step is semi-deterministic within the generation cycle and independent of the SVG generator's own assumptions; an LLM critic may rationalize an absent edge as intentional, but a checklist comparison will be able to detect that. If the audit fails, the **SVG connection repair** tool applies targeted fixes to missing or misrouted arrows and the audit runs once more. Repair is attempted at most once to avoid destabilizing the layout.

Critic–Refiner Loop. After structural verification, the **SVG critic** inspects the diagram for semantic fidelity and visual quality, producing a severity-tagged report where structural errors (missing nodes, incorrect edge directions, hallucinated components) take precedence over stylistic suggestions. The **SVG refiner** then applies targeted updates: correcting labels, repositioning blocks, rerouting connectors, or adjusting contrast. For structural failures where the floor plan itself is the root cause, the refiner can escalate to regenerate the layout rather than patch over a fundamentally incorrect arrangement; this escalation is permitted at most once to prevent runaway regeneration cycles.

SVG Scorer. After the critic–refiner loop completes, **SVG scorer** scores all intermediate SVGs by a weighted count of critical, major, and minor issues. It selects the highest-scoring intermediate SVG as the final `svg_refined.svg`. This scoring step protects against regression: a refiner that partially fixes a critical error while introducing a new one should not be preferred over an earlier intermediate that had fewer total issues.

3.4 Rasterization Stage

After the SVG is finalized, ReViz constructs the raster-generation prompt. The **image prompter** reads the final SVG, the extracted methodology, the visual plan, and the icon and scene assets, and generates a prompt that articulate the visual structure; specifying layout structure, component labels, connector semantics, and typography. The **styling agent** then enriches this prompt with domain-aware visual cues and color schemes. The styling agent also reinforces print-oriented constraints: high contrast, sufficient label padding, and text that remains readable at paper scale.

The **image generator** receives the final SVG as a structural guide (as rasterized image or code), the styled prompt as a visual directive, and the icon

and scene assets as reference inputs. This framing gives the raster model a constrained generation problem; render a polished version of an already specified diagram, rather than an open-ended one. The SVG preserves topology and labels. The styled prompt governs visual quality and domain character. The icon and scene assets anchor the symbolic representations to the decisions made by the asset generation stage.

Image Critic–Regeneration Loop. After the initial raster image is produced, ReViz applies an evaluation–correction loop analogous to the SVG critic–refiner loop. The **image critic** compares the generated image against source references: the final SVG, the styled prompt, the icon assets, the scene assets, and the connection checklist. It produces a report with a `passed/failed` flag and a list of `well specified corrections` describing specific failures. Grounding the evaluation in the connection checklist is important because a raster model can visually blur or merge connector paths in ways that satisfy an aesthetic criterion but violate the methodology's graph structure.

Image Finalization. After all regeneration attempts complete, a scoring pass selects the best generated image across all rounds using a penalty scheme that prioritises structural failures (missing or hallucinated blocks and incorrect connections) over aesthetic issues, and copies the selected image to the final output.

3.5 Checkpointing and Human Review

ReViz is implemented as a state-managed graph with explicit checkpointing. Each step reads named artifacts written by earlier steps and writes its own inspectable outputs: extracted methodology markdown, visual plan, candidate floor plans, refined floor plans, connection checklists, icon briefs, icon assets, scene assets, intermediate SVGs, audit reports, critic reports, final SVGs, image prompts, and generated raster images. This artifact chain supports two important operational properties. First, an interrupted run can resume from the last completed step without re-executing earlier work. Second, an intermediate failure can be diagnosed by inspecting the artifact at the point of failure rather than re-running the entire pipeline.

ReViz also supports **human-in-the-loop checkpoints**. A researcher can pause the pipeline after any major stage, inspect the current artifact, and proceed only when the output is scientifically faithful to the source methodology. This is important because small visual errors in research diagrams carry real consequences. A reversed edge implies a different data dependency. A missing component omits a step from the scientific description. An incorrect phase boundary changes how the reader interprets the method's modularity. These errors are easy to miss in a final rendered image but visible in the structured intermediate artifacts that ReViz preserves at each step.

4 Experiments

4.1 Experimental Setup

We evaluate ReViz against a direct-generation baseline on 25 papers randomly sampled from PaperBananaBench (Zhu et al., 2026a). The baseline feeds the methodology text directly to `gemini-2.0-flash-preview-image-generation` using a single prompt, with no intermediate structured representation. ReViz is evaluated in two configurations that differ only in the agent model: **ReViz-Flash** uses `gemini-2.0-flash-preview` for all pipeline stages, and **ReViz-Pro** uses `gemini-2.5-pro-preview`; both configurations use `gemini-2.0-flash-preview-image-generation` for the final raster generation stage.

4.2 Evaluation Methodology

We adopt the three-tier automated evaluation framework from Paper2SysArch (Guo et al., 2025) rather than PaperBanana's pure text based evaluation, which does not evaluate structural fidelity in a robust manner. The framework assesses each diagram across three dimensions: the **Semantic judge** extracts a structured component-and-connection graph from both the source text and the generated figure, and scores node, edge, and hierarchy consistency against the ground truth; the **Layout judge** detects spatial defects such as edge crossings, element overlaps, and text overflows, scoring readability through a penalty mechanism; and the **Visual judge** assesses icon semantic relevance and text legibility. The overall score is a weighted combination ($0.3 \times$ **Semantic** $+ 0.3 \times$ **Layout** $+ 0.4 \times$ **Visual**), and both frameworks have demonstrated correlation with human judgment (Zhu et al., 2026a; Guo et al., 2025).

Table 2: Evaluation results on 25 papers from PaperBananaBench. All systems use `gemini-3.1-flash-image-preview` for raster generation. Evaluation judges run on `gemini-3.1-pro-preview`.

Method	Semantic	Layout	Visual	Overall
ReViz (`gemini-3-flash-preview`)	30.34	82.91	82.65	67.03
ReViz (`gemini-3.1-pro-preview`)	36.94	80.4	80.41	67.36

4.3 Results

Table 2 reports results for both ReViz configurations. ReViz-Flash (`gemini-3-flash-preview`) achieves an overall score of 67.03, compared to 67.36 for ReViz-Pro (`gemini-3.1-pro-preview`) — a gap of only 0.33 points. The pro model yields a higher semantic score (36.94 vs. 30.34), reflecting its stronger reasoning ability during structured extraction and layout planning, while the flash model matches or slightly exceeds the pro model on layout (82.91 vs. 80.4) and visual quality (82.65 vs. 80.41). The near-identical overall scores demonstrate that the structured pipeline architecture absorbs much of the capability difference between models: by decomposing the task into well-defined stages with explicit intermediate artifacts, ReViz constrains each agent to a bounded sub-problem rather than the full open-ended generation task, allowing a lighter model to perform comparably to a significantly stronger one on the dimensions that matter most for the final output. The appendix provides illustrative examples of the generation process, showing the intermediate artifacts produced at each pipeline stage alongside the final output.

5 Conclusion

We presented ReViz, a multi-agent pipeline for generating publication-quality methodology diagrams from research paper text. Rather than treating figure generation as a single end-to-end task, ReViz decomposes the problem into four structured stages, each producing an inspectable intermediate artifact that the next stage operates on directly. Within this architecture, ReViz introduces explicit layout exploration before spatial commitment, independent asset generation decoupled from global layout synthesis, and deterministic connection verification that tracks structural fidelity across stages without relying on model judgment. Experimental results demonstrate that these contributions produce diagrams that more accurately reflect the described methodology and achieve higher visual quality, suggesting that finegrained structural decomposition is a productive direction for reliable automated scientific figure generation.

Limitations

Raster connection fidelity. Although the SVG stage enforces correct edges through the connection auditor and ground-truth checklist, the raster generation model does not operate over explicit graph structure. It synthesises the figure from visual and textual prompts, so connector paths can be blurred, merged, or omitted even when the SVG reference is structurally correct, which is an observable limitation. The image critic–regeneration loop mitigates this, but cannot guarantee elimination: each regeneration attempt introduces new visual variation, and the critic must detect structural failures in pixel space rather than graph space. Further prompt engineering reduces the frequency of such errors, but the root cause is fundamental to how current image generation models handle precise relational geometry (Zhu et al., 2026b).

Text rendering and pipeline cost. Image generation models continue to struggle with small, precise labels at the density typical of research figures (Zhu et al., 2026a; Zhang et al., 2025); the SVG artifact therefore remains the primary source of label accuracy, with the raster output serving presentation quality. Additionally, the multi-stage architecture requires many sequential model calls, making figure generation more expensive than single-step approaches. Both constraints are expected to relax as model capabilities and inference efficiency improve, but they represent practical considerations for current deployment.

Acknowledgments

Large language models were used for language correction and code assistance during the preparation of this work.

References

Jonas Belouadi, Eddy Ilg, Margret Keuper, Hideki Tanaka, Masao Utiyama, Raj Dabre, Steffen Eger, and Simone Ponzetto. 2025. Tikzero: Zero-shot text-guided graphics program synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17793–17806.

Qi Chen, Jingxuan Wei, Zhuoya Yao, Haiguang Wang, Gaowei Wu, Bihui Yu, Siyuan Li, and Cheng Tan. 2025. Researchpulse: Building method-experiment chains through multi-document scientific inference. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 9110–9119.

Victor Dibia. 2023. Lida: A tool for automatic generation of grammar-agnostic visualizations and infographics using large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 113–126.

Pei Fu, Tongkun Guan, Zining Wang, Zhentao Guo, Chen Duan, Hao Sun, Boming Chen, Qianyi Jiang, Jiayao Ma, Kai Zhou, and 1 others. 2025. Multimodal large language models for text-rich image understanding: A comprehensive review. *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19941–19958.

Shivank Garg, Sankalp Mittal, and Manish Gupta. 2026. Text2arch: A dataset for generating scientific architecture diagrams from natural language descriptions. *arXiv preprint arXiv:2604.14941*.

Akash Ghosh, Aparna Garimella, Pritika Ramu, Sambaran Bandyopadhyay, and Sriparna Saha. 2025. Infogen: Generating complex statistical infographics from documents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 20552–20570.

Ziyi Guo, Zhou Liu, and Wentao Zhang. 2025. Paper2sysarch: Structure-constrained system architecture generation from scientific papers. *arXiv preprint arXiv:2511.18036*.

You Hu, Chenzhuo Zhao, Changfa Mo, Haotian Liu, and Xiaobai Li. 2026. Scifigdetect: A benchmark for ai-generated scientific figure detection. *arXiv preprint arXiv:2604.08211*.

Siyuan Huang, Yutong Gao, Juyang Bai, Yifan Zhou, Zi Yin, Xinxin Liu, Rama Chellappa, Chun Pong Lau, Sayan Nag, Cheng Peng, and 1 others. 2026. Scifig: Towards automating scientific figure generation. *arXiv preprint arXiv:2601.04390*.

Honglin Lin, Chonghan Qin, Zheng Liu, Qizhi Pei, Yu Li, Zhanping Zhong, Xin Gao, Yanfeng Wang, Conghui He, and Lijun Wu. 2026. Scientific image synthesis: Benchmarking, methodologies, and downstream utility. *arXiv preprint arXiv:2601.17027*.

Luping Liu, Chao Du, Tianyu Pang, Zehan Wang, Chongxuan Li, and Dong Xu. 2024. Improving long-text alignment for text-to-image diffusion models. *arXiv preprint arXiv:2410.11817*.

Fanqing Meng, Jin Wang, Chuanhao Li, Quanfeng Lu, Hao Tian, Tianshuo Yang, Jiaqi Liao, Xizhou Zhu, Jifeng Dai, Yu Qiao, and 1 others. 2025. Mmiu: Multimodal multi-image understanding for evaluating large vision-language models. In *International Conference on Learning Representations*, volume 2025, pages 38405–38453.

Rishab Parthasarathy, Jasmine Collins, and Cory Stephenson. 2026. What makes a good generated image? investigating human and multimodal llm image preference alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 24801–24810.

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

Juan A Rodriguez, David Vazquez, Issam Laradji, Marco Pedersoli, and Pau Rodriguez. 2023. Figgen: Text to scientific figure generation. *arXiv preprint arXiv:2306.00800*.

Yinghao Tang, Xueding Liu, Boyuan Zhang, Tingfeng Lan, Yupeng Xie, Jiale Lao, Yiyao Wang, Haoxuan Li, Tingting Gao, Bo Pan, and 1 others. 2026. Igenbench: Benchmarking the reliability of text-to-infographic generation. *arXiv preprint arXiv:2601.04498*.

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

Zihang Wang, Yilun Zhao, Kaiyan Zhang, Chen Zhao, Manasi Patwardhan, and Arman Cohan. 2025b. Scisketch: An open-source framework for automated schematic diagram generation in scientific papers. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 403–417.

Jingxuan Wei, Cheng Tan, Qi Chen, Gaowei Wu, Siyuan Li, Zhangyang Gao, Linzhuang Sun, Bihui Yu, and Ruifeng Guo. 2025. From words to structured visuals: A benchmark and framework for text-to-diagram generation and editing. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 13315–13325.

Abhay Zala, Han Lin, Jaemin Cho, and Mohit Bansal. 2023. Diagrammergpt: Generating open-domain, open-platform diagrams via llm planning. *arXiv preprint arXiv:2310.12128*.

Leixin Zhang, Steffen Eger, Yinjie Cheng, Weihe Zhai, Jonas Belouadi, Fahimeh Moafian, and Zhixue Zhao. 2025. Scimage: How good are multimodal large language models at scientific text-to-image generation? In *International Conference on Learning Representations*, volume 2025, pages 6923–6948.

Dawei Zhu, Rui Meng, Yale Song, Xiyu Wei, Sujian Li, Tomas Pfister, and Jinsung Yoon. 2026a. Paperbanana: Automating academic illustration for ai scientists. *arXiv preprint arXiv:2601.23265*.

Minjun Zhu, Zhen Lin, Yixuan Weng, Panzhong Lu, Qiujie Xie, Yifan Wei, Sifan Liu, Qiyao Sun, and Yue Zhang. 2026b. Autofigure: Generating and refining publication-ready scientific illustrations. *arXiv preprint arXiv:2602.03828*.

A ReViz Pipeline Artifacts

The following sections show example artifacts produced at each stage of the ReViz pipeline. They illustrate the intermediate representations that flow through the four stages.

A.1 Layout Exploration and Refinement

The idea generator produces several candidate floor plans representing divergent spatial organizations of the extracted components. The idea refiner evaluates these candidates and produces a refined blueprint by selecting the best-matching layout and adjusting spacing, block sizes, and connector routes. An example set of four candidates and the resulting refined layout are shown below.

A.2 Artifact Generation

Before SVG synthesis, the pipeline generates symbolic and contextual visual assets guided by the visual plan. Example outputs for both asset types are shown below. Icons are generated as a single batched grid to enforce set-level visual consistency; scene assets are generated separately for components where a photographic image communicates the concept more clearly than a schematic symbol.

A.3 Styled Prompt

After the SVG is finalized, the image prompter and styling agent jointly construct the generation prompt. The prompter encodes diagram structure and connector semantics; the styling agent adds domain-aware visual cues and print-oriented constraints. An example prompt is shown below.

Styled Image-Generation Prompt

Figure type: Multi-stage agentic pipeline diagram for a computer science research paper. Flat design. Publication quality. Suitable for print at two-column textwidth.

Layout: Four horizontal phase swimlanes labeled Research, Strategy, Execution, and Rasterization, arranged left to right. Each swimlane contains a vertical column of labeled rectangular component blocks. Phase labels appear above their respective swimlane regions in a larger, bold sans-serif font. A thin horizontal rule separates each swimlane from its neighbors.

Components and placement:
Research lane — Information Extractor (top), Visual Planner (below).
Strategy lane — Idea Generator, Idea Refiner, Connection Checklist, Icon Generator, Scene Asset Generator (top to bottom).
Execution lane — SVG Generator, Connection Auditor, SVG Critic, SVG Refiner (top to bottom).
Rasterization lane — Image Prompter, Styling Agent, Image Generator, Image Critic (top to bottom).

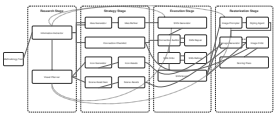
Connectors: Data-flow arrows are solid with a filled arrowhead. Control-flow arrows are dashed. Feedback arrows are curved and colored in a muted red accent. All arrow weights are uniform. No connector crosses a phase boundary unless the connection is explicitly labeled as inter-phase.

Icons: Place the provided icon asset for each component inside the upper-left corner of its block at 20% of the block height. Do not distort or recolor icons. Maintain the same icon size across all blocks.

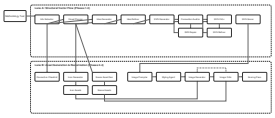
Typography: Component labels in 9pt medium sans-serif. Phase labels in 11pt bold sans-serif. Connection labels in 7pt italic sans-serif, centered above their connector line. No decorative text. No title text inside the figure.

Color: Use the palette from the visual plan: blue accents for Research, teal for Strategy, orange for Execution, purple for Rasterization. Block fill is a very light tint (5%) of the phase accent color. Swimlane background is white. All colors must remain distinguishable at 200 dpi grayscale print.

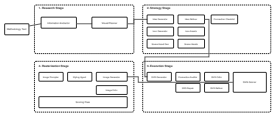
Negative guidance: No photorealistic textures. No drop shadows or beveled edges. No decorative background gradients. No extraneous icons or clip art not specified above. No hallucinated components. No omitted labeled nodes. No hand-drawn or sketch aesthetic.



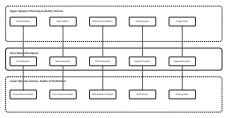
(a) Candidate 1 **(selected)**



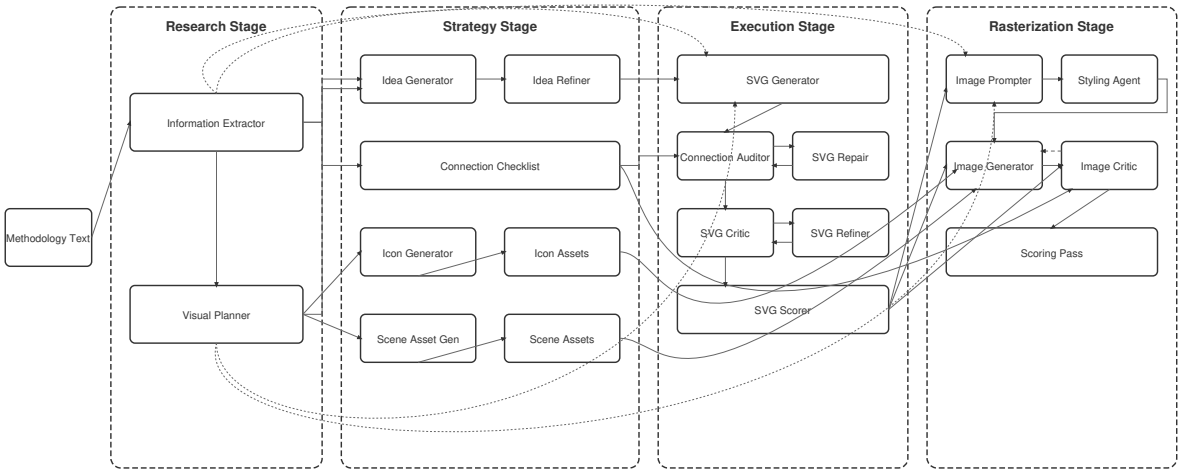
(b) Candidate 2



(c) Candidate 3

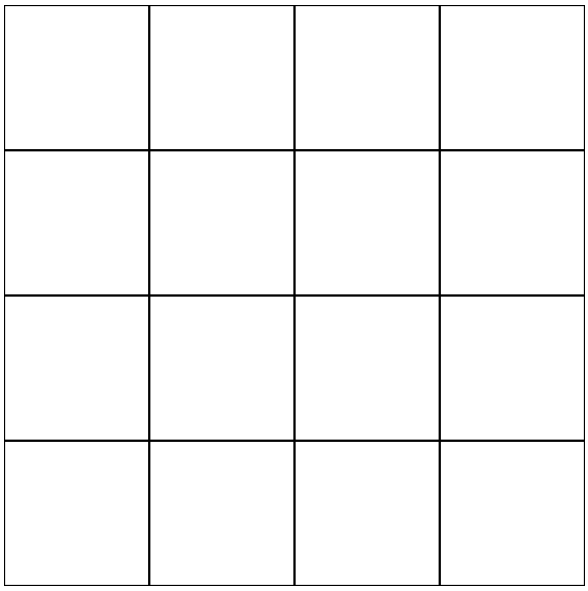


(d) Candidate 4

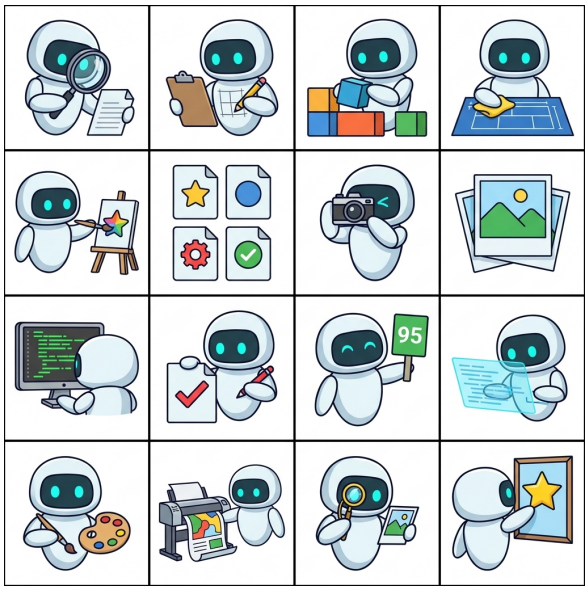


(e) Refined layout: selected and improved from Candidate 1, with adjusted phase groupings, normalized block sizes, and routed connectors

Figure 2: Four candidate floor plans explored by the idea generator (a–d) and the selected refined blueprint (e). Each candidate encodes the same extracted components and connections under a different spatial organization; Candidate 1 was selected as the best match for the input methodology's sequential, phase-structured topology.



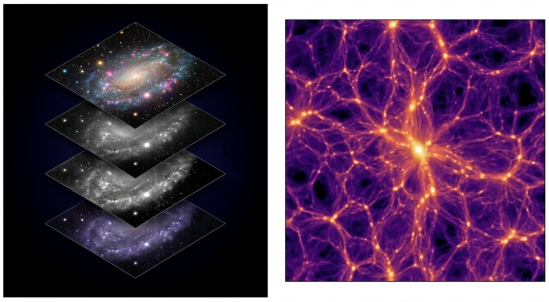
(a) Icon brief template grid: each cell specifies component name, role, and icon hint for a single icon in the batch



(b) Generated icon sheet: all icons produced in a single generation call and subsequently split into individual PNGs

Figure 3: Example icon generation. The brief template (a) batches all component icon descriptions into a structured grid layout. Generating all icons together in one image (b) enforces stroke weight, style, and visual register consistency across the full component set.

(a) Scene asset brief template: each cell specifies the component, the scene context, and photographic framing guidance



(b) Generated scene assets: contextual photographs produced for components where a photographic image communicates the concept more clearly than a schematic symbol

Figure 4: Example scene asset generation. The brief template (a) specifies photographic context and framing for each component designated as needing a scene asset by the visual plan. Generated assets (b) are passed as reference inputs to the image generator alongside the final SVG.