ITERAFIGURE: EDITABLE STRUCTURED SCIENTIFIC DIAGRAM GENERATION WITH ATOMIC LOCAL EDITING

Anonymous authors
Paper under double-blind review

ABSTRACT

A generated scientific figure is rarely final. Authors rename a module, enlarge a label, or shift a group, and each of these requests is small and local. Most systems nevertheless deliver an image or an editable file with no persistent state behind it, so one small revision triggers full regeneration and disturbs already-verified content. We present ITERAFIGURE, whose product of generation is not a rendered image but a persistent object graph, so generation and revision operate on one structured state. A revision is an atomic local edit on this state, not a regeneration. In the editor, the user's selection states which region should change, an interaction design that lets authors express a revision more precisely and conveniently than language alone. The selected scope is then enforced mechanically. A deterministic guard validates the whole operation batch against the scope before mutation, and an accepted batch commits atomically, leaving every object outside the scope with identical content, style, geometry, hierarchy, and endpoints. Locality is thereby turned from a request in natural language into a property decidable at commit time. On FigureBench, ITERAFIGURE attains the highest overall quality among ranked systems while every element stays addressable, and under a matched-input protocol that gives every system the same editing interface it strictly passes 99.3% of single-turn local edits while preserving every unrelated visible element. The advantage compounds over ten-step mixed editing chains, where it still completes 96.7% at the final step, and a blinded study rates its outputs highest for visual quality, topology accuracy, and publication willingness.

1 INTRODUCTION

Text-to-figure generation has advanced rapidly, from multimodal pipelines to specialized scientific-illustration systems (Zhu et al., 2026b;a; Zhang et al., 2025; Zala et al., 2024), yet generating a scientific figure is rarely a single act. After the first draft, researchers rename modules, enlarge labels, and move groups, and each of these requests is small, local, and frequent. A figure system therefore must produce a faithful, publication-grade figure and then let an author change exactly what was intended while everything already verified survives. Answering a local request by regenerating the whole figure forces the author to re-check settled content, and usually to re-fix what the regeneration disturbed.

Current systems meet parts of this problem. Image generators and planning pipelines improve first-pass appearance (OpenAI, 2026; Zhu et al., 2026a;b), but a raster holds no independently addressable element. Systems that emit SVG, PPTX, drawing code, or a diagram representation (Sveidqvist & Contributors to Mermaid, 2014; Jiang, 2025; Shao et al., 2026; Liu et al., 2025; Guo et al., 2026; Wei et al., 2025) deliver an editable format without an editable state. Section 4.4 measures the gap directly, where the most capable editing baseline completes 93.4% of the requested edits yet preserves only 88.6% of unrelated visible content. What is missing is a mechanism that decides locality at commit time on a persistent representation.

We frame the task as editable structured diagram generation over a persistent state, which raises three challenges. A method description must become a graph with explicit objects, hierarchy, geometry, and connections rather than merely plausible pixels (Cui et al., 2025; Guo et al., 2026).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

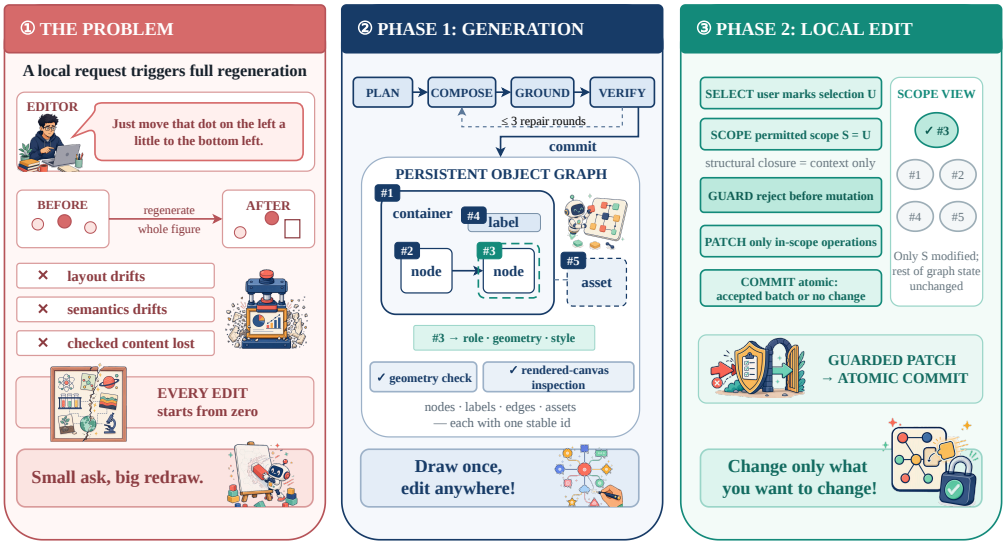


Figure 1: ITERAFIGURE. (i) A local request answered by full regeneration disturbs already-verified content. (ii) Generation. Plan, Compose, Ground, and Verify commit a persistent object graph in which every node, label, edge, and asset carries one stable ID. (iii) Atomic local editing. The user selection U fixes the permitted scope S, elements outside S are non-editable context, and accepted patches apply atomically. This figure was authored in ITERAFIGURE itself.

Structure alone does not guarantee rendering quality, because text overflow, overlap, and erroneous routing still compromise accuracy (Liang & You, 2025; Zhang et al., 2026). And a natural-language modification must be confined to an explicit permitted scope. Instruction-guided editing leaves the changed region implicit (Brooks et al., 2023; Hertz et al., 2023), and asking a model to edit locally, or exposing object IDs to it, is not enforcement. The first two challenges concern generation. The third separates an editable product from an editable format.

ITERAFIGURE[1] makes the object graph the product of generation rather than an intermediate serialization, and most of the pipeline exists to build that graph. It constructs independently addressable nodes, labels, edges, and assets, verifies the rendered canvas against both source content and deterministic geometry, and keeps the verified graph as the authoritative state for every later revision. When a revision is requested, the user selection U is an interaction affordance that states, more precisely and conveniently than language alone, which region the author means. In the editor it is also the permitted scope S enforced at commit time, and a deterministic structural closure around it is shown to the model as non-editable context that grounds the request without widening S. The commit path rejects any batch containing an out-of-scope operation before mutation, and an accepted batch applies atomically. People decide *which* region changes, and the system executes and verifies *how*, in line with guidance on meaningful human control (Amershi et al., 2019; Shneiderman, 2022).

Scope of the contribution. The contribution is architectural and methodological rather than a new backbone or learning algorithm. A model proposes an operation batch, and a deterministic checker admits or rejects it before the state changes, in the spirit of verifier-gated generation (Chen et al., 2026). The locality condition is therefore enforceable in the mxGraph implementation (JGraph, 2021), and the content-locator design transfers to other addressable representations (Sections 3.1 and 5).

We make three contributions.

[1]The complete system implementation is available in an anonymous repository at `https://anonymous.4open.science/r/IteraFigure`. Model-facing prompts and task-level evaluation records are released in the anonymized supplement, submitted as a separate attachment.

- **ITERAFIGURE.** A generation framework whose product is the object graph itself. Direct-to-mxGraph construction, deterministic design-contract normalization, and rendered-canvas plus geometric verification yield independently addressable nodes, labels, edges, and assets in one persistent state.
- **A checkable locality property.** Because the generated graph persists, revision can be made safe by construction. A user selection U yields a permitted scope S, and an operation batch Π is validated against S before mutation and then committed atomically. The property is decidable at commit time, and any violation aborts the batch.
- **A bounded lifecycle evaluation and provenance-aware asset handling.** Five empirical studies cover first-pass generation, single-turn and ten-step revision, a blinded human study, and an ablation under one matched-input protocol with a shared deterministic content locator. For identity-sensitive assets, the system separates planning from retrieval and gates candidates before preserving provenance for user review. ITERAFIGURE attains the highest automated Overall among ranked systems, preserves all unrelated visible text on the evaluated editing tasks, and has the numerically highest aggregate human-evaluation scores, although not every pairwise preference gap is significant.

2 RELATED WORK

2.1 SCIENTIFIC FIGURE GENERATION

Three axes separate the systems in this line of work. The first is the product. Raster pipelines end in pixels (Zhu et al., 2026a;b), while structured pipelines end in a vector, code, or XML artifact (Guo et al., 2026; Liu et al., 2025; Jiang, 2025; Sveidqvist & Contributors to Mermaid, 2014; Zala et al., 2024). Draw-with-Thought also emits mxGraph XML but reconstructs an existing figure from its published raster image: the structure it recovers is already on the canvas, whereas a generator must decide that structure from a text specification (Cui et al., 2025). The second axis is what happens when the request changes. Most systems regenerate. A continuation component appears in the multi-route Paper2Any system (Guo et al., 2026; Liu et al., 2025), but it holds no verified graph as authoritative state across revisions. The third axis is how quality is checked. Visual priors (Shao et al., 2026), tool-using agents (Wei et al., 2025), and graph-based checks (Liang & You, 2025) assess quality, inverse-parsing benchmarks score structure preservation (Zhang et al., 2026), and multimodal benchmarks probe text-to-image generation directly (Zhang et al., 2025). ITERAFIGURE is compared quantitatively on first-pass generation with the five configurations from these lines of work that run end to end under one common setup (Sections 4.2 and 4.3), and after generation keeps one structured state across modification, restricting every post-initialization change to guarded local operations (Sections 4.4 and 4.5).

2.2 EDITABLE GRAPHICS, EDITING, AND EVALUATION

Vector and programmatic representations expose editable primitives through visual-program generation (Reddy et al., 2021; Carlier et al., 2020; Jain et al., 2023; Xing et al., 2023; Rodriguez et al., 2025), text-guided graphics-program synthesis (Belouadi et al., 2024; 2025), layered-SVG recovery (Song et al., 2025), and code- or XML-based authoring tools (Sveidqvist & Contributors to Mermaid, 2014; Jiang, 2025). Graph-based metrics assess generated diagrams structurally (Liang & You, 2025), iterative self-correction and verbal feedback improve agent outputs (Gou et al., 2024; Shinn et al., 2023), and verifier-in-the-loop pipelines increasingly gate model output behind deterministic checks (Chen et al., 2026). Image editing research studies instruction-guided (Brooks et al., 2023) and prompt-controlled (Hertz et al., 2023) modification, but on pixels rather than addressable elements. Graph drawing research treats the stability of the viewer's spatial understanding across edits as a first-class constraint, through mental-map preservation in layout adjustment (Misue et al., 1995; Purchase et al., 2007), online incremental re-layout (Frishman & Tal, 2008), and dynamic graph visualization (Beck et al., 2017). Those methods keep a drawing geometrically close to its predecessor, whereas ITERAFIGURE keeps the out-of-scope graph state unchanged at the artifact level. Tool-using agents (Yao et al., 2023; Schick et al., 2023; Hong et al., 2024) and vision-language models (Liu et al., 2023) support agentic tool use and rendered-output diagnosis. Because automated rubrics are model-scored and inherit known judge biases (Zheng et al., 2023; Wang et al., 2024), we

add a blinded human study. For identity-sensitive assets, ITERAFIGURE records reviewable application provenance rather than claiming license or trademark clearance (Coalition for Content Provenance and Authenticity, 2026).

Where the increment lies. Per-element identifiers already exist in several structured systems, but identifiers alone do not enforce that unrelated content survives a local edit. ITERAFIGURE differs at three levels. At the product level it treats the object graph itself as the output of generation, so the delivered artifact is the last verified state rather than a serialization of something else. At the mechanism level it restricts every post-generation change to scope-guarded atomic local edits, which makes preservation a property checked at commit time on an editable mxGraph substrate rather than a hope expressed in a prompt. At the evaluation level it measures the full lifecycle, from first-pass generation through single-turn and ten-step revision, under one matched-input protocol with a blinded human study.

3 ITERAFIGURE

3.1 PROBLEM FORMULATION

We model a scientific figure as an attributed object graph $G = (V_G, E_G)$, where V_G is the set of node cells (modules, labels, icon zones, asset regions), E_G the set of connector cells, and each cell carries a semantic role $\tau(c) \in \mathcal{T}$ and a persistent identifier. Every cell holds a state tuple

$$\kappa(c, G) \;=\; \big(\ell(c),\, \sigma(c),\, \gamma(c),\, \pi(c),\, \epsilon(c)\big), \tag{1}$$

comprising its label ℓ, style attributes σ, geometry γ, hierarchical parent π, and, for connectors, the endpoints ϵ. Given a specification x and optional reference material R, generation computes a verified initial state

$$G_0 \;=\; \mathcal{V}\big(\mathcal{G}(x, R)\big), \tag{2}$$

where $\mathcal{G}$ constructs the object graph directly from the specification and $\mathcal{V}$ denotes the verification-and-repair loop of Section 3.5; only a state that passes $\mathcal{V}$ is committed. The delivered artifact is the graph G_0 itself, together with a rendering $I(G_0)$ and asset metadata P (Section 3.3).

Editing is a controlled state transition. An instruction u pairs with a **user selection** $U \subseteq V_{G_t} \cup E_{G_t}$ recorded in the editor, and U is exactly the **permitted modification scope** S that the system enforces. The model proposes an **operation batch** $\Pi = (\pi_1, \ldots, \pi_k)$ against the current state G_t, where each operation updates, deletes, or inserts one cell addressed by persistent identifier. A deterministic guard admits the batch only under the condition

$$\Gamma(G_t, S, \Pi) = 1 \iff \mathrm{tgt}(\Pi) \subseteq S \;\wedge\; \mathrm{wf}\big(\mathrm{apply}(\Pi, G_t)\big), \tag{3}$$

where $\mathrm{tgt}(\Pi)$ is the set of identifiers that any operation in Π reads or writes, and $\mathrm{wf}(\cdot)$ asserts structural well-formedness of the resulting graph: valid identifiers, an acyclic hierarchy, and resolvable connector endpoints. The session transition is then

$$G_{t+1} = \begin{cases} \mathrm{apply}(\Pi, G_t) & \text{if } \Gamma(G_t, S, \Pi) = 1, \\ G_t & \text{otherwise,} \end{cases} \tag{4}$$

so an out-of-scope target or a structurally invalid result leaves the state unchanged, and an accepted batch applies atomically.

Definition 1 (Locality by construction) *A transition $G_t \to G_{t+1}$ under scope S is* local *iff every cell outside the scope keeps its full state:*

$$\forall c \in \big(V_{G_t} \cup E_{G_t}\big) \setminus S : \quad \kappa(c, G_{t+1}) = \kappa(c, G_t). \tag{5}$$

By the guard in Eq. (3) and the atomic commit in Eq. (4), every admitted transition satisfies Definition 1. Content, style, geometry, hierarchy, and endpoints of out-of-scope objects are therefore preserved by invariant, which makes locality a decidable property of the commit path rather than a request in natural language.

3.2 Pipeline Overview

Figure 1 groups the implementation into four stages realized as seven steps, each a typed operator on the emerging state. *Plan* covers `parse` and `layout`, *Compose* covers `sketch`, *Ground* covers `asset`, and *Verify* covers `validate`, `relayout`, and `final`. The generation operator of Eq. (2) is their composition applied to (x, R),

$$\mathcal{G} \;=\; \texttt{final} \circ \texttt{relayout} \circ \texttt{validate} \circ \texttt{asset} \circ \texttt{sketch} \circ \texttt{layout} \circ \texttt{parse}, \qquad (6)$$

Four steps delegate judgment to the backbone model, and their tools run deterministic bookkeeping. `parse` and `layout` produce the structured strategy and the adaptive canvas plan. `sketch` emits the first mxCell XML, which the normalizer of Section 3.3 completes into graph state. `asset` is plan-only: the client retrieves and screens candidates before a user binds one (Appendix E). `validate` and `relayout` realize the loop $\mathcal{V}$ (Section 3.5), and `final` cannot emit XML, so the delivered artifact is exactly the last verified graph. Appendix H tabulates what the model and the deterministic code each supply at every step.

3.3 Structured State and Design Contract

We adopt draw.io's mxGraph XML object-graph format of individually addressable `mxCell` objects (JGraph, 2021), chosen for its mature visual editor, which lets users keep editing the delivered artifact by hand, with SVG, PDF, and PNG export downstream. The delivered state is the triple $(G_t, I(G_t), P)$ of Eq. (2). A retrieved asset in P records its source page, downloaded-byte fingerprint, selection status, and graph binding.

The *design contract* fixes palette roles, typography hierarchy, stroke rhythm, spacing, shape language, routing policy, and asset policy. A deterministic normalizer N_c applies it to every emitted cell,

$$\sigma\big(N_c(e)\big) \;=\; \sigma(e) \;\cup\; \big\{\big(a,\, v_c(a)\big) \;:\; a \text{ unspecified in } \sigma(e)\big\}, \qquad (7)$$

so any attribute the model left unspecified takes a contract value $v_c(a)$ rather than a renderer default. The contract values are tabulated in Appendix A. Cells bound to an image or a stencil are exempt, and XML-emitting model calls receive the contract as their style specification.

3.4 Semantic Elementization and Local Editing

The graph is decomposed into the roles of Section 3.1 (title band, module, label, connector, icon, asset region), and each role τ carries a constraint set Φ_τ over style, geometry, and connectivity. Connectors require explicit endpoints ϵ, long labels use controlled wrapping, and icon zones stay separate from text zones, so a rendered defect maps back through its ID to the smallest editable cause. An editing request pairs a user selection U with a natural-language instruction u, where the selection is an interaction affordance that expresses the intent language leaves underspecified and, on the interactive pathway, defines S as in Eq. (3). Before the model sees the request, a deterministic structural closure $C(U) \supseteq U$ is attached as non-editable context, which grounds the request without widening S. The execution trace attributes each visual difference to a concrete operation.

Eq. (3) fixes only the problem-level contract. The remaining guard semantics, including the exemption set that S cannot include, the closure $C(U)$, structurally anchored additions, and the whole-graph re-layout lock, are fixed in code and specified in Appendix A, and scope expansion follows one fixed structural rule rather than learned inference of correlated changes, which remains future work.

3.5 Separated Validation and Repair

Validation runs in two channels that cannot substitute for one another. Content validation compares labels, entities, and relations against x and R, and its tool contract restricts the model call to text-value corrections of roughly the original length, so a factual correction cannot turn into a relayout in disguise. Rendered-canvas verification evaluates overlap, edge interference, readability, crowding, and composition consistency on a screenshot. The vision-language verifier never emits cell IDs and reports at most five label-referenced findings with executable suggestions, while the twenty deterministic inspectors embed owning cell IDs in findings marked critical or warning. Deterministic

Table 1: Mechanism comparison with every ranked baseline that emits a revisable artifact. Full circle present, half-filled circle partial, empty circle absent.

System	Source	Locality by construction			Checks	Provenance	Editor
		Persistent IDs	Scope guard	Atomic commit			
Mermaid	Software, 2014	○	○	○	○	○	●
Paper2Any	CVPR Findings, 2026	○	○	○	○	○	●
AutoFigure	ICLR, 2026	○	○	○	◐	○	●
LiveFigure	ICML, 2026	●	○	○	◐	○	●
ITERAFIGURE	Ours	●	●	●	●	●	●

findings therefore carry their target IDs, and label-referenced VLM findings are localized by the acting model before issuing ID-addressed edits.

Merged findings enter the repair loop that realizes $\mathcal{V}$ in Eq. (2),

$$G_{i+1} \;=\; \mathrm{patch}\big(G_i,\, \mathrm{top}(Q_i)\big), \qquad Q_i \;=\; \mathrm{merge}\big(F_{\mathrm{cont}}(G_i),\, F_{\mathrm{vis}}(I(G_i)),\, F_{\mathrm{geom}}(G_i)\big), \tag{8}$$

where F_{cont}, F_{vis}, and F_{geom} collect content, rendered-canvas, and deterministic-geometry findings, merged in a fixed priority order and capped at eight per round. The loop stays alive only while Q_i contains a critical finding, for at most three rounds followed by one final check. Dense routing constraints can conflict, so repair targets locally attributable violations and patches the implicated edge IDs, favoring structural stability over routing optimality. When the repair budget is exhausted the loop neither errors nor restarts. Verification returns a no-repair verdict, the pipeline commits, and unresolved findings are declared as residual defects. Appendix J gives the inspector set, the queue order, the budgets, and the failure taxonomy.

3.6 Mechanism Comparison

Table 1 contrasts ITERAFIGURE with every ranked revisable baseline, on the artifact each system is evaluated on in Section 4.2. The distinction is not editability in principle, since each system ends in a file a human can reopen and revise, but whether locality is enforced during automated revision. We call this property *locality by construction*. The commit path admits only an in-scope patch or an unchanged graph, so the graph state of out-of-scope objects is unchanged by invariant, not because the model declined to touch them, and the guarantee transfers to any addressable representation with a similarly constrained commit path. The two half-filled-circle entries behave the same way. AutoFigure and LiveFigure send a rendered screenshot to a model that reports overlaps and overflow, and compute no geometry in code. The route we benchmark renders only to emit its deliverable, and its unbenchmarked draw.io and reconstruction routes carry a model critique or source-fidelity checks, not both.

4 Experiments

4.1 Research Questions and Protocol

Five questions organize the evaluation. **First-pass generation quality** asks whether a persistent object graph can match specialized generation systems. **Single-turn revision quality** asks whether each system applies the same intent to its own native artifact while preserving unrelated content, under a matched-input interface covering three atomic edit families. **Sustained revision** asks how completely a system survives ten mixed editing operations applied in sequence. **Human utility** asks which outputs researchers judge clear, correct, and publication-ready, and it evaluates outputs only. **Mechanism evidence** tests whether the agent workflow accounts for the observed gains under fixed conditions.

Every system receives matched prompts and optional references. Native artifacts are retained for the editability tests and standardized images only for visual assessment. A generation counts as

Table 2: Main comparison on 300 FigureBench prompts (all scores 0–10). †non-ranked, non-editable raster reference, excluded from editing. ‡reference value exceeding the best ranked result. Succ. = success rate.

			Content Fidelity				Visual Quality			Structural Validity			
Method	**Source**	**Succ.**	**Sem.**	**Fact.**	**Comp.**	**Avg.**	**Read.**	**Pub.**	**Avg.**	**Topo.**	**Flow**	**Avg.**	**Overall**
GPT Image 2†	Model, 2026	1.00	9.11	8.11	8.81	8.68	**7.94**‡	**7.04**‡	**7.49**‡	9.35	8.53	8.94	8.37
Mermaid	Software, 2014	0.98	7.37	6.45	6.00	6.61	3.08	3.19	3.14	6.96	5.52	6.24	5.33
Paper2Any	CVPR Findings, 2026	**1.00**	8.66	7.96	7.98	8.20	6.68	5.73	6.21	8.54	7.91	8.23	7.54
AutoFigure	ICLR, 2026	**1.00**	8.82	7.59	7.96	8.12	5.93	4.76	5.35	9.47	7.92	8.69	7.39
LiveFigure	ICML, 2026	0.99	8.67	7.35	8.30	8.11	6.69	5.46	6.08	9.10	7.87	8.48	7.56
ITERAFIGURE	Ours	**1.00**	**9.33**	**8.66**	**8.93**	**8.97**	**7.53**	**6.95**	**7.24**	**9.56**	**8.65**	**9.11**	**8.44**

successful when its artifact parses and renders, so crashes, timeouts, and unparsable artifacts are failures. Quality scores use successful generations only, success rate is reported separately, and human evaluators are blinded. Prompts, rule sets, thresholds, and repair limits were fixed before the 300-prompt test set was scored and never tuned on test outcomes.

4.2 DATASET, BASELINES, AND METRICS

We use FigureBench (Zhu et al., 2026b). Its designated 300-prompt test set drives generation, a common fixed 100-prompt subset drives editing and ablation, and human evaluation uses an independently sampled 100-prompt set. FigureBench was introduced by the authors of AutoFigure, so we use its released split without modification and disclose that provenance.

We compare with GPT Image 2 (OpenAI, 2026), Mermaid (Sveidqvist & Contributors to Mermaid, 2014), the SVG-input continuation component of the multi-route Paper2Any system (Guo et al., 2026; Liu et al., 2025), AutoFigure (Zhu et al., 2026b), and LiveFigure (Shao et al., 2026). These five span the two probed axes, namely raster versus structured products and regeneration versus in-place modification, and each has a runnable implementation under the common configuration. GPT Image 2, the most recent OpenAI image model as of August 2026, is the shared backbone of all image-producing baselines. Two further systems discussed in Section 2.1 (Wei et al., 2025; Cui et al., 2025) are cited but not compared, because the protocol requires running each system's generation and editing paths over the full test set under one fixed configuration, which we could set up only for the five above. GPT Image 2 is a non-ranked raster reference only. The common configuration uses `gpt-5.6-sol` for reasoning and `qwen3.7-plus` for automated judging, with adapters and the editing-stage configuration of AutoFigure documented in Appendices C and M.

Seven source-grounded sub-metrics form three tiers. **Content Fidelity** covers semantic coverage, factual accuracy, and completeness. **Visual Quality** covers readability and publication readiness. **Structural Validity** covers topology fidelity and flow coherence. Overall is the unweighted mean of the three tier means, each tier the unweighted mean of its displayed sub-metrics, aggregated from unrounded scores. We use 20,000 bootstrap resamples for confidence intervals and 100,000 sign-permutation samples for paired tests, with Holm correction across reported contrasts. In all main-text tables, boldface marks the best value per column (ties boldfaced) and underlining the second-best, omitted when the best is tied. In Table 2 the marking applies among ranked methods only.

4.3 FIRST-PASS QUALITY

Table 2 reports successful-output means and success rates separately. On paired Overall comparisons over jointly successful prompts, ITERAFIGURE scores higher than every ranked configuration, with all Holm $p < .001$ and mean differences from $+0.88$ to $+3.12$.

Structure is not paid for with quality. ITERAFIGURE reaches 8.44 Overall (95% CI $[8.31, 8.56]$) against 8.37 (95% CI $[8.22, 8.51]$) for the non-editable GPT Image 2 raster reference, with every node, edge, and label addressable. It leads on the content tier, 8.97 against 8.68, and on the structural tier, 9.11 against 8.94, while GPT Image 2 stays ahead on the visual tier at 7.49 against 7.24. Generation wall time also favors ITERAFIGURE at 637 s mean against 964 s for AutoFigure and 878 s for LiveFigure (Appendix M.4). The complete unedited case comparison is in Appendix K.

4.4 MATCHED-INPUT LOCAL EDITING OVER ONE TURN

The separating capability appears after generation, where a change on three atomic edit families should succeed without reopening settled parts of the figure. Because the systems expose different editing interfaces, we evaluate all of them under one matched-input interface. Each system receives its own complete native artifact plus one natural-language edit request, with one edit attempt that returns a complete artifact when successful. The target is identified only by a deterministic locator over unique visible text in that system's own pre-edit artifact (Appendix C). No native identifier, coordinate, selection, or scope list is given to any system. GPT Image 2 lacks an auditable editable artifact and Mermaid cannot deterministically represent the geometric operations, so both are excluded here. The figures therefore measure post-hoc properties of returned artifacts, not the interactive selected-scope guard of Section 3.1, which is an editor-pathway mechanism.

The suite applies three code-verifiable operations to the shared 100-prompt subset, giving 300 planned tasks per system. The operations replace a fixed label, increase a font size by 4 px, and translate a group right by 10% of the primary element's width. **Target completion** records whether the requested edit happened, and **unrelated-element preservation** records what share of non-target baseline visible text survives one-to-one afterwards. We report only a strict pass rate over successful generated-and-parsed artifacts. All four predeclared components (**edit requirement satisfied**, **non-target content/structure preservation**, **structural validity**, and **expected structure delta satisfied**) must pass together, and failed artifacts are excluded from the denominator (Appendix C).

Table 3 reports a strict pass rate of 99.3% (298/300) for ITERAFIGURE over successful artifacts, against 67.3% (148/220) for AutoFigure, 26.5% (75/283) for LiveFigure, and 14.3% (43/300) for Paper2Any. Every non-target visible-text item survives every ITERAFIGURE task. The strict pass rate requires the target request and the structural conditions to pass together. Appendix C reports per-family pass rates, preservation, timing, failure accounting, and the underlying task-level fields.

Table 3: Matched-input single-turn editing on three atomic edit families (300 planned tasks per system). Strict pass rates are over successful artifacts under the shared native-artifact plus content-locator interface (Appendix C).

System	Succ. (%)	Pass (%)	Pres. (%)
AutoFigure	73.3	67.27	99.97
LiveFigure	94.3	26.50	96.69
Paper2Any	**100.0**	14.33	88.59
ITERAFIGURE	**100.0**	**99.33**	**100.00**

4.5 SUSTAINED REVISION OVER TEN-STEP MIXED CHAINS

Single-turn results bound one operation, while real revision sessions compose many. Each system runs a fixed ten-step *mixed* chain on 30 stratified seeds of the common subset. The chain mixes text, style, geometry, and structure changes, steps are strictly sequential, and each call receives the previous artifact through content locators only. A checkpoint counts as complete when either the visual judgment or the native target check confirms the requested edit, and the cumulative pass through checkpoint K requires this combined condition at every assessed checkpoint through K. All four systems use the same fixed 30-chain denominator, and native baseline-relative preservation is reported independently at each checkpoint. Appendix D gives the step sequence, the exact decision rule, and the full accounting.

Table 4: Long-horizon mixed editing over 30 preregistered chains. Cells report cumulative completion pass rate / native baseline-relative preservation at each checkpoint (Appendix D).

System	K=1	K=3	K=5	K=10
AutoFigure	80.0 / 99.5	30.0 / 99.5	30.0 / 96.3	16.7 / 96.3
LiveFigure	80.0 / 86.4	80.0 / 86.1	80.0 / 86.1	80.0 / 86.1
Paper2Any	**100.0 / 100.0**	56.7 / 98.5	53.3 / 98.5	30.0 / 89.2
ITERAFIGURE	**100.0 / 100.0**	**100.0 / 100.0**	**100.0 / 100.0**	**96.7 / 96.7**

At $K = 10$, the combined completion pass rate is 96.7% for ITERAFIGURE, 30.0% for Paper2Any, 80.0% for LiveFigure, and 16.7% for AutoFigure (Table 4). The separation between completion and preservation makes the failure modes visible. A returned artifact can retain much of the non-target content while failing the requested edit, or satisfy the request while disturbing unrelated content. Figures 3 and 4 in Appendix D give the full diagnostics.

Table 5: Controlled ablations on the fixed 100-prompt subset (0–10). All rows share prompts, backbone model, and scoring protocol; the last row is an alternative raster-drafting construction, not a component removal (Appendix G).

Configuration	Succ.	Content	Visual	Struct.	Overall
Full **IteraFigure**	**1.00**	**9.02**	**7.38**	**9.23**	**8.54**
w/o content validator	**1.00**	8.95	6.94	9.08	8.32
w/o visual verifier	**1.00**	8.50	6.17	8.86	7.84
w/o incremental by-ID repair	**1.00**	8.90	6.97	9.16	8.34
Visual-draft variant	**1.00**	8.80	6.86	9.08	8.24

4.6 Mechanism Evidence from Component Ablations

We ablate three components on the fixed 100-prompt subset (Table 5); the full-system Overall of 8.54 there differs from the 300-prompt 8.44 of Table 2.

The visual verifier is the most consequential removal, dropping Overall from 8.54 to 7.84. The reduction spreads from Visual Quality into the other two tiers, which is consistent with its repair loop touching shared structure. Removing incremental by-ID repair (generation-time refinement) costs 0.20, removing content validation 0.22. The visual-draft variant reaches 8.24, and its shortfall is concentrated in the visual tier, from 7.38 to 6.86, exactly where a draft image should have helped. Raster drafting is therefore no substitute for structured construction and verification (configurations in Appendix G).

4.7 Human Evaluation of Outputs

Thirty-two researchers completed 324 blinded pairwise trials on a separate 100-prompt sample. Three cannot-judge responses are excluded, and left/right placement shows no detectable bias (per-pair accounting in Appendix F). IteraFigure has the highest aggregates on the 0–10 scale, namely Visual Quality 7.40, Topology Accuracy 7.12, and tie-adjusted publication willingness 79.8%. Holm-corrected pairwise preferences favor it over Mermaid and Paper2Any (both $p < .001$); Table 12 and Figure 6 in Appendix F give the full accounting.

5 Limitations and Future Work

IteraFigure targets method and architecture diagrams, and data-driven charts would need constraints on axes, scales, and encodings. The editing studies cover three atomic families and report strict pass rates over successful artifacts. Because no baseline exposes a selection interface, the shared input carries a content locator rather than a manual selection, so the figures credit the persistent graph state under post-hoc checks rather than the interactive selected-scope pathway (Appendices H and C). The long-horizon study uses one fixed ten-step chain on 30 seeds, and because the locator depends on unique visible text, purely geometric nodes, unlabelled connectors, and duplicated labels fall outside it, while the mean per-edit wall time of 156.7 s bounds interactive fluency. Dense edge routing remains open (Wybrow et al., 2010; Dwyer et al., 2009), and learning which changes travel together, rather than one fixed closure rule, is the natural next step.

6 Conclusion

IteraFigure makes a persistent, verifiable object graph the product of generation and the state for revision, attaining 99.3% single-turn strict pass, full unrelated-text preservation, 96.7% ten-step completion, and top blinded human ratings. The editor's selection delimits intent, the guard checks every batch before it touches the state, and locality holds by construction rather than by model discretion. Because the guard needs only addressable identifiers and a checkable scope, the mechanism transfers to any representation whose commit path validates a batch before applying it. Learning which correlated changes should travel together, and extending coverage to data-driven charts, are the natural next steps.

AI USE STATEMENT

We used generative AI tools to assist implementation and testing of system and evaluation software; review the experimental methodology; clean and reformat released task records and audit workbooks; check calculations and interpret results; and draft and edit manuscript text. AI-assisted literature search suggested potential related-work sources, whose relevance and bibliographic metadata were checked before citation. Experimental outputs and human judgments were collected through the procedures reported in this paper. AI-assisted release preparation did not generate new experimental observations. The reported aggregates were independently recomputed from released records. The authors take responsibility for the final text, claims, and artifacts.

REPRODUCIBILITY STATEMENT

The appendix specifies the task protocols, metrics, prompts, deterministic constants, model configurations, and statistical procedures. The anonymized supplement, submitted as a separate attachment, contains the task-level records, audit workbooks, hashes, the scripts for checking headline results and regenerating figures, and the complete verbatim prompt set. The anonymous repository at `https://anonymous.4open.science/r/IteraFigure` holds the system implementation. The asset appendix documents the retrieval, safety, selection, and user-review mechanisms. Model-driven components remain stochastic, and the single-run ablations are accordingly interpreted descriptively.

REFERENCES

Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi T. Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. Guidelines for human-AI interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019. doi: 10.1145/3290605.3300233. URL `https://doi.org/10.1145/3290605.3300233`.

Fabian Beck, Michael Burch, Stephan Diehl, and Daniel Weiskopf. A taxonomy and survey of dynamic graph visualization. *Computer Graphics Forum*, 36(1):133–159, 2017. doi: 10.1111/cgf.12791. URL `https://doi.org/10.1111/cgf.12791`.

Jonas Belouadi, Anne Lauscher, and Steffen Eger. AutomaTikZ: Text-guided synthesis of scientific vector graphics with TikZ. In *International Conference on Learning Representations (ICLR)*, 2024. URL `https://openreview.net/forum?id=v3K5TVP8kZ`.

Jonas Belouadi, Eddy Ilg, Margret Keuper, Hideki Tanaka, Masao Utiyama, Raj Dabre, Steffen Eger, and Simone Ponzetto. TikZero: Zero-shot text-guided graphics program synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 17793–17806, 2025. doi: 10.1109/ICCV51701.2025.01653. URL `https://doi.org/10.1109/ICCV51701.2025.01653`.

Tim Brooks, Aleksander Holynski, and Alexei A. Efros. InstructPix2Pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18392–18402, 2023. doi: 10.1109/CVPR52729.2023.01764. URL `https://doi.org/10.1109/CVPR52729.2023.01764`.

Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. DeepSVG: A hierarchical generative network for vector graphics animation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/hash/bcf9d6bd14a2095866ce8c950b702341-Abstract.html`.

Junhao Chen, Xiang Li, Mingjin Chen, Boran Zhang, Henghaofan Zhang, Yibin Xu, Yuehan Cui, Fangsheng Weng, Fei Ma, Qi Tian, Ruqi Huang, and Hao Zhao. Paircoder++: Pair programming as a universal paradigm for verified code-driven multimodal and structured-artifact generation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026. URL `https://arxiv.org/abs/2607.01883`. Preprint arXiv:2607.01883.

Coalition for Content Provenance and Authenticity. Content credentials: C2PA technical specification. Technical specification, 2026. URL https://spec.c2pa.org/specifications/specifications/2.4/specs/C2PA_Specification.html. Version 2.4, April 2026; accessed 2026-07-24.

Zhiqing Cui, Jiahao Yuan, Hanqing Wang, Yanshu Li, Chenxu Du, and Zhenglong Ding. Draw with thought: Unleashing multimodal reasoning for scientific diagram generation. In *Proceedings of the 33rd ACM International Conference on Multimedia (MM)*, pp. 5050–5059, 2025. doi: 10.1145/3746027.3755731. URL https://doi.org/10.1145/3746027.3755731. Preprint arXiv:2504.09479.

Tim Dwyer, Kim Marriott, and Michael Wybrow. Topology preserving constrained graph layout. In *Graph Drawing (GD 2008)*, volume 5417 of *Lecture Notes in Computer Science*, pp. 230–241. Springer, 2009. doi: 10.1007/978-3-642-00219-9_22. URL https://doi.org/10.1007/978-3-642-00219-9_22.

Yaniv Frishman and Ayellet Tal. Online dynamic graph drawing. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):727–740, 2008. doi: 10.1109/TVCG.2008.11. URL https://doi.org/10.1109/TVCG.2008.11.

Emden R. Gansner and Stephen C. North. An open graph visualization system and its applications to software engineering. *Software: Practice and Experience*, 30(11):1203–1233, 2000. URL https://doi.org/10.1002/1097-024X(200009)30:11%3C1203::AID-SPE338%3E3.0.CO;2-N.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=Sx038qxjek.

Ziyi Guo, Zhou Liu, and Wentao Zhang. Paper2SysArch: Structure-constrained system architecture generation from scientific papers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 1996–2006, 2026. URL https://openaccess.thecvf.com/content/CVPR2026F/html/Guo_Paper2SysArch_Structure-Constrained_System_Architecture_Generation_from_Scientific_Papers_CVPRF_2026_paper.html.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross-attention control. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=_CDixzkzeyb.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=VtmBAGCN7o.

Ajay Jain, Amber Xie, and Pieter Abbeel. VectorFusion: Text-to-SVG by abstracting pixel-based diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1911–1920, 2023. doi: 10.1109/CVPR52729.2023.00190. URL https://doi.org/10.1109/CVPR52729.2023.00190.

JGraph. draw.io. Computer software, 2021. URL https://www.drawio.com/. Version 15.5.2, tagged and published 2021-10-12; Apache-2.0; the repository's official CITATION.cff (added 2024-11-19, default branch) credits JGraph; accessed 2026-07-28.

Dayuan Jiang. Next AI Draw.io repository. Software repository, 2025. URL https://github.com/DayuanJiang/next-ai-draw-io. Repository handle DayuanJiang; created 2025-03-23; latest release v0.4.16 (2026-05-21); Apache-2.0; accessed 2026-07-28.

Chumeng Liang and Jiaxuan You. Evaluating LLM-generated diagrams as graphs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12678–12690, 2025. doi: 10.18653/v1/2025.emnlp-main.640. URL https://aclanthology.org/2025.emnlp-main.640.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 34892–34916, 2023. doi: 10.52202/075280-1516. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/6dcf277ea32ce3288914faf369fe6de0-Abstract-Conference.html.

Zhou Liu, Ziyi Guo, Zeli Su, Ligang Huang, Yihe Yang, Zhaoyang Han, Zewei Pan, and Wentao Zhang. Paper2Any: Turn paper/text/topic into editable research figures, technical route diagrams, and presentation slides. Computer software, 2025. URL https://github.com/OpenDCAI/Paper2Any. Repository created 2025-10-17; Apache-2.0; no tagged release as of 2026-09-09; accessed 2026-07-24.

Kazuo Misue, Peter Eades, Wei Lai, and Kozo Sugiyama. Layout adjustment and the mental map. *Journal of Visual Languages & Computing*, 6(2):183–210, 1995. doi: 10.1006/jvlc.1995.1010. URL https://doi.org/10.1006/jvlc.1995.1010.

OpenAI. GPT Image 2 model. OpenAI API documentation, 2026. URL https://developers.openai.com/api/docs/models/gpt-image-2. Model ID: gpt-image-2; snapshot: gpt-image-2-2026-04-21; released 2026-04-21; most recent OpenAI image model as of August 2026; accessed 2026-07-28.

Helen C. Purchase, Eve Hoggan, and Carsten Görg. How important is the "mental map"? – an empirical investigation of a dynamic graph layout algorithm. In *Graph Drawing (GD 2006)*, volume 4372 of *Lecture Notes in Computer Science*, pp. 184–195. Springer, 2007. doi: 10.1007/978-3-540-70904-6_19. URL https://doi.org/10.1007/978-3-540-70904-6_19.

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 (CVPR)*, pp. 7338–7347, 2021. doi: 10.1109/CVPR46437.2021.00726. URL https://doi.org/10.1109/CVPR46437.2021.00726.

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)*, pp. 16175–16186, 2025. doi: 10.1109/CVPR52734.2025.01508. URL https://doi.org/10.1109/CVPR52734.2025.01508.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 68539–68551, 2023. doi: 10.52202/075280-2997. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

Christoph Daniel Schulze, Miro Spönemann, and Reinhard von Hanxleden. Drawing layered graphs with port constraints. *Journal of Visual Languages and Computing*, 25(2):89–106, 2014. doi: 10.1016/j.jvlc.2013.11.005. URL https://doi.org/10.1016/j.jvlc.2013.11.005.

Chenyang Shao, Jiahe Liu, Fengli Xu, and Yong Li. LiveFigure: Generating editable scientific illustration with VLM agents. In *International Conference on Machine Learning (ICML)*, 2026. URL https://openreview.net/forum?id=bJJLW1gjqh.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Ben Shneiderman. *Human-Centered AI*. Oxford University Press, 2022. ISBN 9780192845290. URL https://global.oup.com/academic/product/human-centered-ai-9780192845290.

Yiren Song, Danze Chen, and Mike Zheng Shou. LayerTracer: Cognitive-aligned layered SVG synthesis via diffusion transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19731–19741, 2025. doi: 10.1109/ICCV51701.2025.01835. URL https://doi.org/10.1109/ICCV51701.2025.01835.

Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981. doi: 10.1109/TSMC.1981.4308636. URL https://doi.org/10.1109/TSMC.1981.4308636.

Knut Sveidqvist and Contributors to Mermaid. Mermaid: Generate diagrams from markdown-like text. Computer software, 2014. URL https://mermaid.js.org/. First tagged release 0.1.0, tagged commit dated 2014-11-16; MIT License; source code at https://github.com/mermaid-js/mermaid; accessed 2026-07-01.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9440–9450, 2024. doi: 10.18653/v1/2024.acl-long.511. URL https://aclanthology.org/2024.acl-long.511.

Jingxuan Wei, Cheng Tan, Qi Chen, Gaowei Wu, Siyuan Li, Zhangyang Gao, Linzhuang Sun, Bihui Yu, and Ruifeng Guo. From words to structured visuals: A benchmark and framework for text-to-diagram generation and editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13315–13325, 2025. doi: 10.1109/CVPR52734.2025.01243. URL https://doi.org/10.1109/CVPR52734.2025.01243.

Michael Wybrow, Kim Marriott, and Peter J. Stuckey. Orthogonal connector routing. In *Graph Drawing (GD 2009)*, volume 5849 of *Lecture Notes in Computer Science*, pp. 219–231. Springer, 2010. doi: 10.1007/978-3-642-11805-0_22. URL https://doi.org/10.1007/978-3-642-11805-0_22.

Ximing Xing, Chuang Wang, Haitao Zhou, Jing Zhang, Qian Yu, and Dong Xu. DiffSketcher: Text guided vector sketch synthesis through latent diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 15869–15889, 2023. doi: 10.52202/075280-0698. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/333e67fc4728f147d31608db3ca78e09-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Abhay Zala, Han Lin, Jaemin Cho, and Mohit Bansal. DiagrammerGPT: Generating open-domain, open-platform diagrams via LLM planning. In *Proceedings of the First Conference on Language Modeling (COLM)*, 2024. URL https://arxiv.org/abs/2310.12128.

Leixin Zhang, Steffen Eger, Yinjie Cheng, Weihe Zhai, Jonas Belouadi, Fahimeh Moafian, and Zhixue Zhao. ScImage: How good are multimodal large language models at scientific text-to-image generation? In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=ugyqNEOjoU.

Tong Zhang, Honglin Lin, Zhou Liu, Chong Chen, and Wentao Zhang. SciFlow-Bench: Evaluating structure-aware scientific diagram generation via inverse parsing. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17747–17765, 2026. doi: 10.18653/v1/2026.acl-long.807. URL https://doi.org/10.18653/v1/2026.acl-long.807.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 46595–46623, 2023. doi: 10.52202/075280-2020. URL https://doi.org/10.52202/075280-2020.

Dawei Zhu, Rui Meng, Yale Song, Xiyu Wei, Sujian Li, Tomas Pfister, and Jinsung Yoon. PaperBanana: Automating academic illustration for AI scientists, 2026a. URL https://arxiv.org/abs/2601.23265. arXiv preprint.

Minjun Zhu, Zhen Lin, Yixuan Weng, Panzhong Lu, Qiujie Xie, Yifan Wei, Sifan Liu, Qiyao Sun, and Yue Zhang. AutoFigure: Generating and refining publication-ready scientific illustrations. In *International Conference on Learning Representations (ICLR)*, 2026b. URL https://openreview.net/forum?id=5N3z9JQJKq.

Supplementary Material (Appendix)

This appendix documents the evaluation protocols, the implementation semantics behind each claim, the full ledgers behind every number in the main text, and the mechanism-critical model-facing prompts verbatim. The anonymized supplement releases the task-level records, audit workbooks, hashes, scripts, and the complete verbatim prompt set that these sections refer to.

A Complete Generation and Repair Procedure

Generation and repair procedure. Given specification x, references R, and defaults D, the system forms contract c and graph G, then runs content, rendered-canvas, and geometry checks. The contract values include one orthogonal connector family, palette-bound fills, strokes, and text colors, a role-based font size, and top alignment with reserved padding for dense labels. Blocking defects enter a fixed-priority queue of ID-addressed patches, for at most three repair rounds followed by one final check. For editing, instruction u and selection U produce the operation batch Π. The selection U is the permitted scope S that the guard enforces, and the deterministic closure computed around it is shown to the model as non-editable context. The guard rejects the whole batch before mutation on any scope violation. Otherwise application is atomic, and an identifier, collision, cell, or structural error returns G unchanged. Implementation semantics and the seven-step mapping appear in Appendix H.

B Evaluation Rubric and Score Aggregation

B.1 Metric hierarchy

Generation quality uses seven sub-metrics in three pre-specified tiers. Content Fidelity covers Semantic Coverage, Factual Accuracy, and Content Completeness. Visual Quality covers Layout Readability and Publication Readiness. Structural Validity covers Topology Fidelity and Flow Coherence. Table 6 gives each field's operational meaning. The rubric and field mapping were fixed before scoring.

B.2 Scale conversion and aggregation

The main paper gives the tier and Overall aggregation and the resampling protocol. Here we retain the audit mapping. Coverage and label fidelity form Semantic Coverage. Accuracy forms Factual Accuracy. Completeness and appropriateness form Content Completeness. Readability, hierarchy, alignment and spacing, and routing form Layout Readability. Consistency, frame discipline, and publication readiness form Publication Readiness. Grouping and relation fidelity form Topology Fidelity. Logical flow and intent conveyance form Flow Coherence. Compatibility-only visual fields are excluded from the composite. All aggregation uses unrounded values. Fields on a $[0, 1]$ scale are multiplied by 10, and 1–5 fields use $(q - 1) \times 2.5$.

Element Addressability is a deterministic binary property, reported separately rather than averaged into quality. It holds for all 300/300 successful IteraFigure artifacts, every element of which is an

Table 6: Operational definitions of the seven generation-quality sub-metrics. All reported generation scores are on a 0–10 scale.

Tier	Sub-metric	Operational interpretation
Content Fidelity	Semantic Coverage	Coverage and correct naming of expected objects and labels.
	Factual Accuracy	Correct technical claims, values, formulas, and statements.
	Content Completeness	Necessary coverage without unsupported or inappropriate content.
Visual Quality	Layout Readability	Legibility, clarity, hierarchy, alignment, spacing, occlusion, and routing.
	Publication Readiness	Aesthetic quality, professional polish, stylistic consistency, frame discipline, and readiness for paper use.
Structural Validity	Topology Fidelity	Correct grouping, hierarchy, directed relations, and connectivity.
	Flow Coherence	Natural logical progression and faithful communication of the overall method intent.

addressable `mxCell` (Table 2, Succ. = 1.00). Quality scores condition on successful generations. Success rate is a separate reliability outcome, and paired main-comparison tests condition on joint success.

C Full Ledger for Matched-Input Local Editing

C.1 Protocol

Each method receives the same semantic editing intent on its own native artifact through the matched-input interface. The input is the complete pre-edit artifact, one natural-language request, and a deterministic content locator over unique visible text derived from that artifact. When no stable description exists, the task is recorded as *unavailable* rather than guessed. No native identifiers, coordinates, selections, scope lists, operation scripts, or permitted-scope fields are provided to any system. Each plan permits one model edit call returning a complete artifact, and network retries resend byte-identical requests only. The IteraFigure record consequently has `editor_mode=full_native_artifact_one_shot` and `explicit_selection_sent_to_model=false`, and input-hash mismatches abort execution. For IteraFigure the permitted scope is derived by rule from this request and locator rather than from a manual selection. The suite contains three atomic operation families. Family (1), `replace_label`, changes the target content's label to the fixed string `Edited element SxxxxTxx`. Family (2), `increase_font`, raises the target font size by exactly 4 px. Family (3), `move_right`, translates the target group right by 10% of the primary element's width, keeping internal geometry and attached straight connectors consistent. Every family is applied to the common 100-prompt subset, giving 300 planned tasks per system.

Metrics. **Target edit completion** is the mean of predeclared acceptance conditions with partial credit. The label family requires the new text without unrelated label changes. The font family requires the exact +4 px on the matched unique-text element with other style attributes preserved. The move family requires a right translation of the target amount at preserved size and vertical position. Where a system's artifact cannot expose the needed attribute, the condition is recorded as `null` rather than zero-filled, and its measurable sample size is reported. Artifact execution and target completion are separate fields: *Executed* means that a post-edit artifact was produced and parsed, while the reported pass rate counts only executed rows that satisfy all four named conditions. **Unrelated-element preservation** performs one-to-one multiset matching of every baseline visible text except the target content and reports the preserved share. **Structural validity** requires the post-edit artifact to parse and pass basic structural validation. IteraFigure additionally reports object-count preservation, edge-endpoint preservation under stable IDs, and expected-delta satisfaction for every executed task. We report the strict pass rate over the executed-artifact denominator. It combines edit requirement satisfied, non-target content/structure preservation, structural

validity, and expected structure delta satisfied. Native object-count and endpoint fields contribute when exposed; blank fields are not zero-filled. An offline audit of the existing ITERAFIGURE XML pairs additionally compares stable IDs and canonical value, style, geometry, hierarchy, endpoint, and child-XML fingerprints. All 468 audited non-move pairs preserve every protected cell exactly under the frozen direct-target-plus-declared-addition policy. The 130 move pairs are reported separately. Their historical plans name target text but do not serialize the semantic group closure, so their direct-target comparison is a closure-sensitive diagnostic, not evidence that every observed non-target geometry change is invalid or permitted. Cost is reported as wall time, API calls, and provider-returned tokens without collapse into a composite. Cross-system preservation comparisons use jointly executed and measurable tasks; pass rates use each system's executed-artifact denominator.

C.2 EXECUTION ACCOUNTING

Provider-returned token counts are available only for LiveFigure in the matched-input export, because the IteraFigure and Paper2Any adapters did not persist token fields in these runs. Those cells are left null rather than reconstructed or imputed.

Failed AutoFigure rows with produced SVG remnants retain diagnostic text-matching scores in the task-level export, but remain failures and do not enter completed-run means. Their recorded zero wall time is not treated as a measured edit duration.

Table 7: Artifact accounting over 300 planned matched-input tasks per system. *Executed* denotes a completed artifact-producing run. Every other planned task is a *failure*. Unavailable records locator abstentions. [d]LiveFigure's target field is measurable on 213 of 283 completed artifact-producing runs; the strict pass denominator is all 283 executed artifacts.

System	Executed	Failed	Unavailable	Pass denominator n
ITERAFIGURE	300	0	0	300
Paper2Any	300	0	0	300
LiveFigure	283	17	0	283
AutoFigure	220	80	0	220

Structural validity follows the same accounting. All 283 completed LiveFigure artifact-producing runs pass the basic structural check, so the completed-artifact pass rates are 300/300 for ITERAFIGURE, 300/300 for Paper2Any, 283/283 for LiveFigure, and 220/220 for AutoFigure. The remaining records stay outside those denominators, namely the 17 LiveFigure tasks and the 80 AutoFigure failed attempts. The strict pass rate uses the executed-artifact denominator, while preservation keeps its own measurable field denominator.

C.3 PER-FAMILY RESULTS

Table 8: Matched-input strict pass rate and preservation by operation family. Pass rates use successful generated-and-parsed artifacts as the denominator; preservation retains its own measurable denominator. Boldface marks the best value per column (all ties boldfaced) and underlining the second-best, omitted when the best is tied.

	Replace label		Increase font		Move right	
System	Pass	Pres.	Pass	Pres.	Pass	Pres.
AutoFigure	<u>69.9 (73)</u>	<u>99.9</u>	<u>68.5 (73)</u>	**100.0**	<u>63.5 (74)</u>	**100.0**
LiveFigure	42.4 (92)	98.5	12.5 (96)	97.7	25.3 (95)	93.9
Paper2Any	17.0 (100)	89.5	24.0 (100)	89.2	2.0 (100)	87.1
ITERAFIGURE	**100.0 (100)**	**100.0**	**99.0 (100)**	**100.0**	**99.0 (100)**	**100.0**

Table 8 decomposes Table 3 using the same strict conjunction. ITERAFIGURE passes 100% of label edits and 99% of font and move edits. Paper2Any and LiveFigure lose strict passes when the

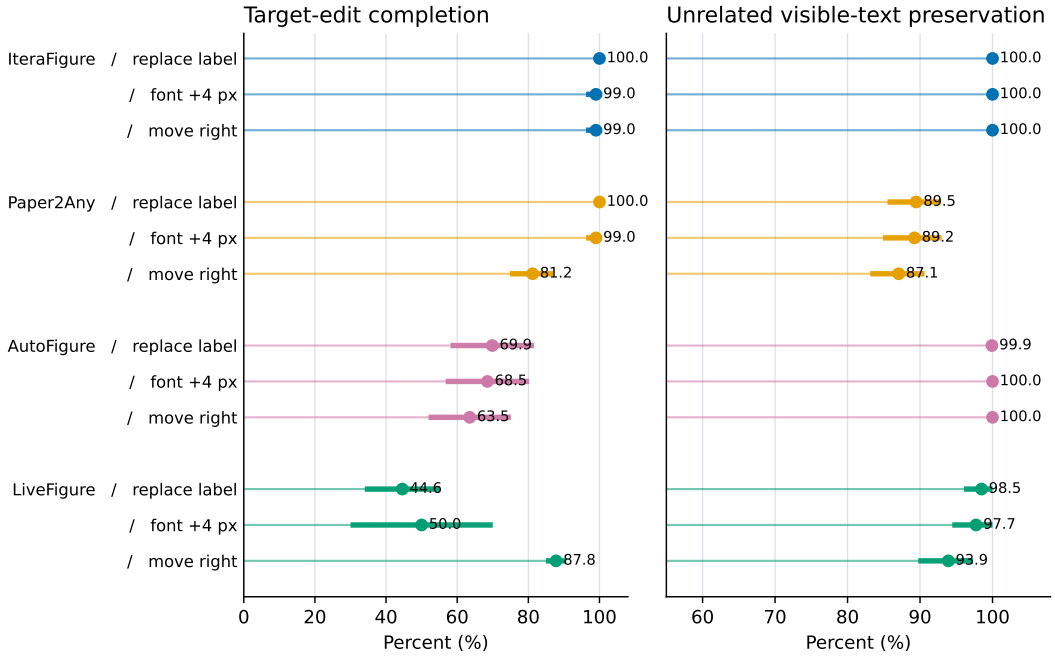


Figure 2: Matched-input strict pass rate by operation family. Values are proportions of successful generated-and-parsed artifacts; AutoFigure uses 73, 73, and 74 such artifacts, LiveFigure 92, 96, and 95, and the other systems 100 per family. IteraFigure attains near-ceiling strict pass rates with exact preservation.

requested edit or preservation condition is incomplete, while AutoFigure passes 63.5–69.9% of its executed family records despite near-perfect visible-text retention.

C.4 Paired contrasts

Table 9: Paired contrasts between IteraFigure and each baseline over jointly executed, jointly measurable tasks. CIs bootstrap the paired mean difference, and p values use two-sided sign-permutation tests with Holm correction across the three reported preservation contrasts.

Comparator	Metric	Δ (pts)	95% CI	n	p_{Holm}
Paper2Any	Preservation	+11.4	[9.5, 13.5]	300	$< .001$
LiveFigure	Preservation	+3.3	[1.9, 5.0]	283	$< .001$
AutoFigure	Preservation	+0.03	[0.01, 0.05]	220	.125

C.5 Editing behavior and cost

Mean per-edit wall time under matched inputs was 156.7 s for IteraFigure, 137.7 s for LiveFigure, 180.5 s for AutoFigure, and 232.4 s for Paper2Any. Timing is descriptive and was not optimized for comparison. All IteraFigure post-edit artifacts passed object-count, edge-endpoint, and expected-delta checks, and no unrelated visible text was lost or altered on any task. The accompanying object-level audit adds exact protected-cell fingerprints for the non-move operations. The separately reported move diagnostics retain the historical group-closure limitation.

D Chain Protocol and Full Results for Sustained Revision

D.1 Design

Each chain executes one fixed *mixed* sequence of ten operations. The 30 seeds are a frozen stratified subsample of the common 100-prompt subset, with 20 paper, 2 blog, 4 survey, and 4 textbook prompts, and exact indices in Appendix L. Step (1) applies the chain's label-edit instruction. Step (2) raises font by 4 px on the focus content, (3) recolors fill to `#DDEBFF`, (4) moves right by 10% of width, (5) inserts an annotation node with a unique label near the focus, (6) connects focus to annotation, (7) inserts a companion node, (8) connects focus to companion, (9) recolors the annotation node to `#FFF2CC`, and (10) raises font by 3 px on the companion node. Steps 5–8 add expected structural deltas of one node and one edge each. Scoring checks that the expected increments are satisfied and that seed text, object counts, and original edge endpoints are preserved.

D.2 Acceptance and states

The native acceptance checker remains an audit channel: a native step is accepted when its artifact parses, passes structural validation, and satisfies the native target, preservation, and planned-delta checks. It is not the reported completion pass criterion. For the reported long-horizon pass rate, each assessed checkpoint is marked complete when either the visual field `edit_requirement_satisfied` equals one or the native field `target_edit_success` equals one. The cumulative pass through K requires this logical-OR condition at every assessed checkpoint through K. The visual and native channels are never averaged and a missing value satisfies neither side. The checker is a measurement rule rather than an artifact-repair stage: it reads the native post-edit artifact, records target completion, baseline-relative preservation, and the planned object/edge delta, but never changes an artifact or retries it. Duplicate identifiers abort a draw.io step, and such genuine model errors remain failed. *Failed* steps record the error, *unavailable* steps could not be seeded or located, and *blocked* steps inherit from a missing predecessor input. Checkpoints $K \in \{1, 3, 5, 10\}$ reuse the same longest chain, and every method uses the fixed 30-chain denominator. Native baseline-relative preservation is computed independently at each checkpoint from that checkpoint artifact and is not pooled across checkpoints. The native acceptance ledger, the combined-rule first-failure distribution, and the continuation credit remain available as diagnostics. For IteraFigure, per-checkpoint native code metrics are complete over 29–30 records; for LiveFigure, a native PresentationML evaluator reads each PPTX step's shape text, font size, fill, geometry, inserted labels, connector endpoints, and cumulative label preservation. The AutoFigure reader uses the native SVG acceptance record whenever a required semantic field is not directly recoverable from SVG geometry alone.

D.3 LiveFigure seed audit and conditional coverage

The current LiveFigure long-horizon submission is evaluated from its PPTX artifacts. Twenty-six chains contain complete step-1–10 artifact sequences, while four are unavailable before producing a usable seed. Under the combined completion rule, 24 of 30 chains pass at each reported checkpoint. The stricter native acceptance ledger has 13 accepted step-1 chains and 11 chains accepted through step 10. These two counts describe distinct criteria.

As a separate visual-quality diagnostic on the common 0–100 display scale, the fixed-denominator, credit-weighted $K = 10$ values plotted in Figure 4 are 90.0 edit satisfaction, 73.3 non-target preservation, 57.3 change quality, 54.0 integration, and 44.0 publication readiness for IteraFigure. Paper2Any follows at 1.5, 2.4, 6.0, 4.2, and 3.9. LiveFigure records 51.7, 33.3, 36.9, 35.2, and 34.5, and AutoFigure 0.7, 1.2, 0.7, 0.7, and 0.6. Every value divides by the same 30 preregistered chains, so a system that has lost most of its chains cannot appear visually competitive on end-state quality alone. Figure 4 plots the same credit weighting across all checkpoints. These weighted visual scores are not the combined completion or native preservation rates of Table 11.

At $K = 1$, Paper2Any's credit-weighted edit and visual-preservation scores in Figure 4 are 100.0 and 90.0 on the 0–100 display scale.

D.4 Combined-rule first-failure distribution

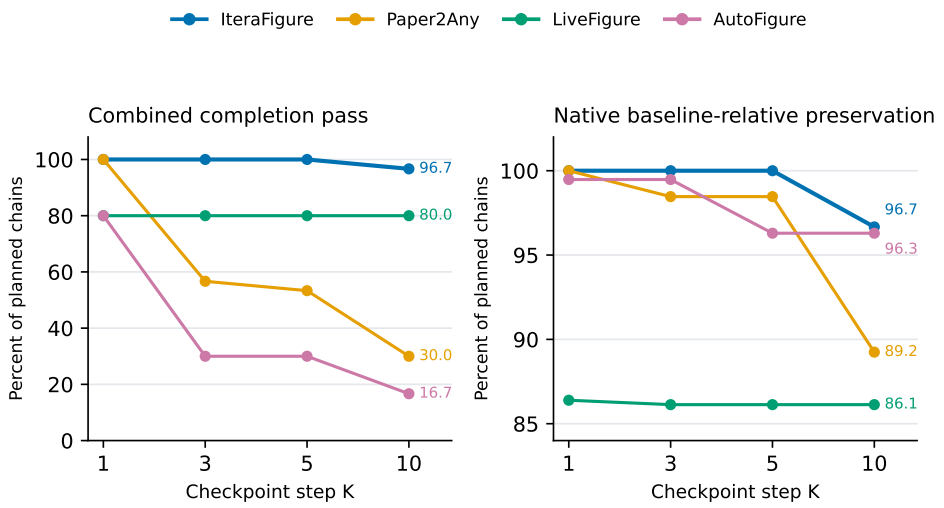


Figure 3: Long-horizon editing under the matched-input protocol. The left panel reports cumulative completion pass through each checkpoint using the visual-or-native rule; the right panel reports native baseline-relative preservation independently at each checkpoint. Every point uses the same 30 planned chains, and end labels give the exact percentages. The native acceptance ledger and graded continuation-credit/VLM diagnostics are retained separately in the accompanying tables and Figure 4.

Table 10: First non-passing assessed checkpoint under the reported combined visual/native rule (30 chains per system). A chain is assigned to the first checkpoint at which its cumulative pass condition fails; chains passing all four checkpoints are counted in the final column.

First failed checkpoint	$K = 1$	$K = 3$	$K = 5$	$K = 10$	**pass through 10**
IteraFigure	0	0	0	1	29
Paper2Any	0	13	1	7	9
LiveFigure	6	0	0	0	24
AutoFigure	6	15	0	4	5

D.5 Combined completion and independent preservation

Table 11 is the primary checkpoint accounting for the long-horizon study. It uses all 30 planned chains at every checkpoint. Completion is the logical OR of the visual edit judgment and the native target check at every assessed checkpoint, and the cumulative pass through K requires that condition at all assessed checkpoints through K. Native preservation is computed independently for the artifact at that checkpoint relative to the chain's initial artifact; it is not pooled across checkpoints.

For Paper2Any, every checkpoint uses the same non-target SVG text multiset from the chain's initial artifact, excluding the original target label, and measures its survival in that checkpoint's artifact. The 30 planned chains remain the denominator even when a checkpoint artifact is unavailable.

The first-failure table above is derived from the same combined rule as the reported completion rates. The native acceptance ledger remains a separate deterministic audit: for example, AutoFigure has visual or native completion evidence on many checkpoints even though none of its chains is native-accepted under the stricter all-condition ledger. The continuation-credit and credit-weighted VLM plots are complementary diagnostics of partial progress and rendered quality.

D.6 Why some visual trajectories rise

Fixing the denominator removes changing-denominator bias, and it does not force a visual trajectory to be monotone. Let $c_{i,K} \in [0, 1]$ be a chain's continuation credit and $v_{i,K}$ its VLM measure on

Table 11: Long-horizon checkpoint accounting with the fixed 30-chain denominator. Combined completion pass is cumulative through the named checkpoint and accepts a step when either the visual `edit_requirement_satisfied` field or native `target_edit_success` field equals one. Native preservation is the independent baseline-relative proportion at that checkpoint; missing records remain in the denominator.

	$K = 1$		$K = 3$		$K = 5$		$K = 10$	
System	Pass	Pres.	Pass	Pres.	Pass	Pres.	Pass	Pres.
AutoFigure	24/30 (80.0)	99.5	9/30 (30.0)	99.5	9/30 (30.0)	96.3	5/30 (16.7)	96.3
LiveFigure	24/30 (80.0)	86.4	24/30 (80.0)	86.1	24/30 (80.0)	86.1	24/30 (80.0)	86.1
Paper2Any	**30/30 (100.0)**	**100.0**	17/30 (56.7)	98.5	16/30 (53.3)	98.5	9/30 (30.0)	89.2
ITERAFIGURE	**30/30 (100.0)**	**100.0**	**30/30 (100.0)**	**100.0**	**30/30 (100.0)**	**100.0**	**29/30 (96.7)**	**96.7**

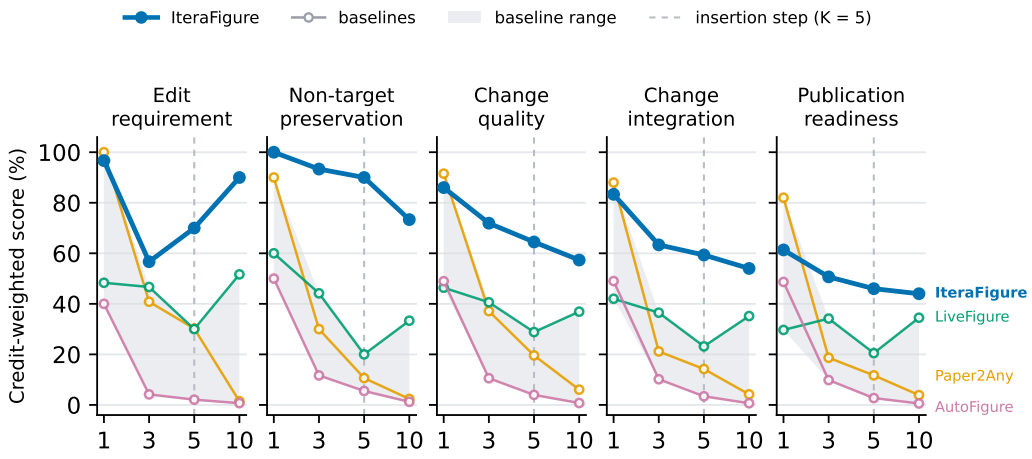
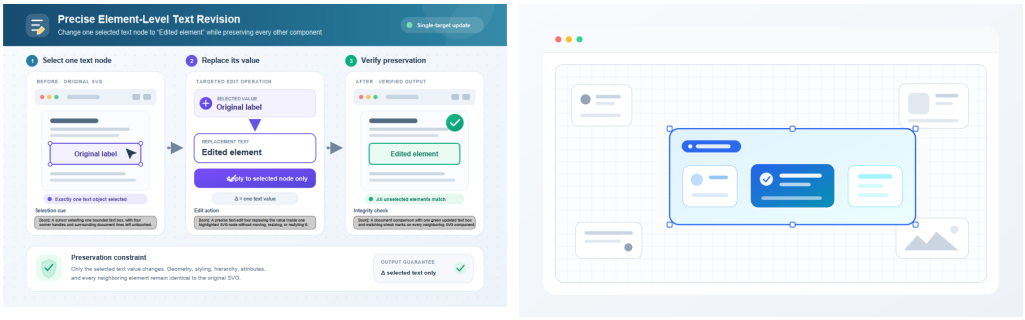


Figure 4: Fixed-denominator, continuation-credit-weighted VLM trajectories for the five reported long-horizon visual measures. Every point has the same denominator of 30 preregistered chains. A completed artifact's VLM score is multiplied by its chain's current continuation credit, which starts at 100 and is halved for each available uncompleted or unconfirmed current edit. Failed, unavailable, unparsable, and unscored artifacts contribute zero. Thus partial attempts remain visible without being treated as strict success. The shaded band spans the minimum and maximum across the three baselines, and the dashed vertical line marks the insertion step at $K = 5$. Edit requirement and non-target preservation are binary/proportion measures. Change quality, integration, and publication readiness use the 1–5 VLM rubric and are linearly displayed as score$/5 \times 100$.

the plotted 0–100 scale. The plotted value is $\frac{1}{30}\sum_i c_{i,K} v_{i,K}$, so a later successful edit can raise the visual term even while credit has declined after earlier misses. The combined completion pass remains absorbing across the assessed checkpoints, while the graded visual view can rise. AutoFigure illustrates the distinction: its combined pass at $K = 1$ is 24/30 even though its stricter native acceptance ledger has no accepted chain; the credit-weighted step-1 values are 40.0 edit satisfaction, 50.0 non-target preservation, and 49.0 change quality. The full native-acceptance transition decomposition remains in the supplement audit workbook.

The operation sequence explains the pattern. $K = 3$ follows the long replacement label, its $+4$ px font change, and a fill edit. These style operations can expose text wrapping or contrast defects. $K = 5$ additionally contains a move and a newly inserted, uniquely labelled annotation. The annotation is a fresh, salient requested object, so a VLM can recognize the cumulative request more readily even when earlier styling remains imperfect. This explains the edit-requirement rebounds without implying that all prior visual defects were repaired. By $K = 10$, the final operation is a local $+3$ px change on a newly added companion node. It can be visually clean and therefore



(a) (b)

Figure 5: Representative AutoFigure step-1 failure mode. (a) is sample 0003 with the `replace_label` request; (b) is sample 0090 with the `move_right` request. Each returned SVG is a newly composed explanatory graphic about editing, with headings, panels, arrows, and synthetic before/after examples, rather than an in-place revision of the supplied figure. The native evaluator records target-edit success 0 for both examples; they illustrate the failure mode and are not additional aggregate observations.

improve target-focused or local integration judgments, while the accumulated moved labels, added boxes, and connectors still lower global preservation, quality, or publication readiness. For example, IteraFigure's $K = 5 \rightarrow 10$ edit score rises by 20.0 points while its non-target preservation falls by 16.7 points and its quality falls by 7.1 points. For Paper2Any, no plotted checkpoint transition is a positive rebound, since its $K = 3 \rightarrow 5$ and $K = 5 \rightarrow 10$ changes are non-positive and its large drops are dominated by absorbing attrition. The distinction exposes a failure mode that any long-horizon editor should expect, namely that satisfying the currently salient request is easier than preserving global visual economy as new objects accumulate.

The deterministic checker is therefore necessary for interpretation and is not a substitute for VLM or human visual assessment. It certifies a deliberately narrow representation contract, namely requested text, style, and geometry changes, parseability, expected graph deltas, and preserved tracked content or endpoints. It certifies nothing about overlap, legibility, whitespace balance, or publication quality. The divergence in Figure 4 is consequently informative rather than contradictory. It separates structural continuity from the remaining visual-layout problem and prevents a later plausible artifact from erasing an earlier failed edit. The full per-transition decomposition, including zero and negative transitions and chain-level examples, is retained in the supplement audit workbook.

Why AutoFigure fails on local edits. The low AutoFigure editing score has a concrete qualitative failure mode. On some requests, the system treats the instruction as a topic for a new explanatory graphic rather than as an operation on the supplied figure. The two representative step-1 artifacts in Figure 5 are valid, well-rendered SVGs, but they are newly composed tutorials about single-element text editing: they introduce their own headings, panels, arrows, and before/after examples instead of preserving the submitted figure and changing one addressed element. The native checker therefore records target-edit success 0 and object-count preservation 0 for these records, even though the returned XML parses. Across the 300 single-turn tasks, 80 attempts failed to produce a completed run, and among the 220 completed runs the requested edit was confirmed in only 67.3%, showing that parseability and text retention do not guarantee instruction fulfillment. Geometry and style requests are especially difficult to confirm from an SVG when the requested change is not represented by a uniquely identifiable native object. The examples explain why a visually coherent output can still fail a native editing benchmark: the benchmark requires an in-place change with the original content retained, whereas a topical redraw changes the artifact identity and cannot establish locality.

E Asset Retrieval, Selection, and Provenance

Candidate retrieval and eligibility. For a retrieved asset, the client combines configured public search providers with a direct user URL or pages discovered from the query. A `site:` term is a

hard source-page constraint. An image hosted by a CDN is permitted only when referenced by a qualifying source page. The system scores and retains at most 12 candidates, but source-host agreement alone never proves an image's identity. For identity requests, automatic eligibility also requires entity evidence in the asset-specific label/path or qualified page context, a logo/wordmark/seal-like mark signal, no conflicting entity signal, and a qualifying score. Site icons are never automatically selected.

Fetch and content screening. Each eligible image URL is retried up to three times before the next candidate is tried. The client accepts only screened public HTTP(S) inputs and blocks `localhost`, `.local`, IPv4 loopback, and the 10/8, 172.16/12, and 192.168/16 ranges before fetch. It caps an image at 4 MB and a page at 1.5 MB, with a 12 s fetch timeout. It rejects HTML masquerading as an image and verifies type from an allowlist plus file signatures. Raster inputs are decoded before use. Tiny images, wholly transparent images, and QR-like images are rejected. The provenance record retains candidate scores and reasons, search-provider result/error summaries, retry diagnostics, final URL, MIME declaration and detected type, byte size, dimensions, and content flags, but never API keys.

Optional vision-based selection. An optional vision selector receives at most three already downloaded PNG/JPEG/WebP/GIF candidates that have passed all deterministic source, safety, content, and eligibility checks. Its prompt supplies only the closed candidate-ID set, trustworthy text metadata, byte fingerprints, and the attached images. It must return one structured assessment per supplied ID (description, asset kind, entity match, confidence, visible evidence, and review flag) plus an optional selected ID. A response with an outside ID, incomplete assessments, malformed fields, low confidence, a non-mark type, an entity mismatch, or a review flag cannot select an asset. Automatic adoption requires *all* of logo/wordmark/lockup/seal/emblem type, entity match, confidence at least 0.85, and no review flag. The model cannot provide a URL or bypass preceding guards. On a missing or failed vision configuration, the deployment records `not_configured`/`disabled` and may use its deterministic ordering for interactive compatibility. This is explicitly not visual verification. Candidates that fail automatic selection remain previewable for a user to choose or replace with an official upload.

Provenance record and user review. When a retrieved candidate is saved, its source URL and page URL, source type, scorer reasons, download/content diagnostics, search attempts, visual description, candidate assessments, model/revision, final selection status, and graph cell bindings are stored in the project asset provenance record. This supports inspection and project export, but it neither establishes rights clearance nor replaces human verification of an identity mark. In particular, a matching source page, filename, or vision-model choice is not proof of identity. Ambiguous candidates remain subject to user confirmation or replacement by an uploaded official file.

F Blind Human Evaluation

F.1 Composition and procedure

The study uses an independently sampled set of 100 FigureBench test prompts through 324 blinded pairwise trials completed by 32 participants recruited from AI and related technical research communities. Each trial presents two anonymous outputs. Participants score both for Visual Quality and Topology Accuracy on a 1–5 scale and then report Publication Willingness as a left preference, right preference, tie, or cannot judge. Each scalar rating is converted as $(q - 1) \times 2.5$, and standard deviations are multiplied by 2.5. The export contains 1,296 scalar ratings and 324 pairwise responses, of which three are cannot-judge and are excluded from preference summaries.

Participation was voluntary. Participants were told the purpose of the study before starting, could stop at any time, and saw anonymized figure renderings only. The only participant information recorded is coarse background, namely year of birth, education stage, research field, and publication count, stored without names or contact details. The study involves no sensitive data and no more than minimal risk.

Left/right placement was randomized. Among the 260 decisive trials, 133 chose the left image and 127 the right, a left rate of .512 with 95% Clopper–Pearson interval $[.449, .574]$ and exact binomial $p = .757$. There is no evidence of position bias.

Table 12: Blind human ratings on the common 0–10 scale. Pref. is the tie-adjusted publication-willingness score, $(W + 0.5T)/(W + L + T)$.

System	**Visual** (mean ± SD)	**Topology** (mean ± SD)	**Pref.** (%)
Mermaid	2.08 ± 2.09	2.56 ± 2.38	16.8
Paper2Any	4.90 ± 2.07	5.25 ± 1.81	42.2
AutoFigure	5.21 ± 2.05	5.29 ± 1.97	39.1
LiveFigure	6.72 ± 2.50	6.45 ± 2.49	71.9
ITERAFIGURE	7.40 ± 2.69	7.12 ± 2.46	79.8

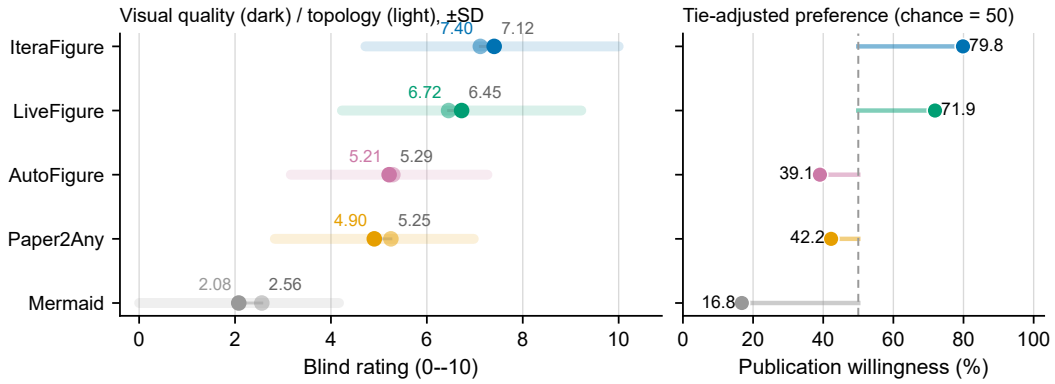


Figure 6: Blind human evaluation. The left panel shows mean Visual Quality and Topology Accuracy on the 0–10 scale (dots, with light bars for ±1 SD). The right panel shows tie-adjusted publication willingness per system, with the 50% chance reference line.

F.2 PAIR ALLOCATION

Table 13 reports the number of trials allocated to each of the ten unordered system pairs. The counts sum to the 324 completed trials. Allocation was fixed by design rather than by output quality, and the ITERAFIGURE row supplies the per-pair inference in Appendix F.3.

Table 13: Number of pairwise trials per system pair.

	Mer.	**P2A**	**Auto**	**Live**	**Itera**
Mermaid	—	50	30	20	30
Paper2Any	50	—	15	22	43
AutoFigure	30	15	—	57	27
LiveFigure	20	22	57	—	30
ITERAFIGURE	30	43	27	30	—

F.3 PER-PAIR INFERENCE FOR PUBLICATION PREFERENCE

Table 14 provides the pairwise preference analysis supporting the aggregate human-evaluation summary in the main paper.

Table 14: ITERAFIGURE's publication choices against each comparator. Win rate and CI are computed over decisive trials, and p values are exact two-sided binomial tests with Holm correction.

Comparator	W–L–T	Dec. n	Win rate	95% CI	p_{Holm}
Mermaid	28–1–0	29	.966	$[.822, .999]$	$< .001^*$
Paper2Any	36–4–3	40	.900	$[.763, .972]$	$< .001^*$
AutoFigure	16–6–5	22	.727	$[.498, .893]$	.105
LiveFigure	19–11–0	30	.633	$[.439, .801]$	.201

F.4 COVERAGE AND REPEATED RATINGS

The 100 prompts and five systems define 500 possible outputs, of which 435 were rated at least once, namely Mermaid 88, Paper2Any 89, AutoFigure 85, LiveFigure 86, and ITERAFIGURE 87. The 65 unrated outputs follow from pairwise allocation, since each trial exposes only two of the five systems for a given prompt, and not from any filtering on quality or system identity. The near-equal per-system coverage confirms this. Of the rated outputs, 247 were rated once, 163 twice, and 25 three times. Most outputs therefore carry a single human rating.

F.5 HUMAN–VLM ALIGNMENT

For a post-hoc alignment check, repeated human ratings are first averaged within each sample, system, and criterion cell and paired with the corresponding automated score. All 435 unique judged outputs per criterion are matched. Visual Quality yields Pearson $r = .42$, Spearman $\rho = .38$, MAE $= 2.19$, RMSE $= 2.72$, and a mean automated-minus-human offset of $+0.33$ points. Topology Accuracy yields $r = .37$, $\rho = .32$, MAE $= 3.17$, RMSE $= 3.77$, and an offset of $+2.83$. The large topology offset is a scale artefact rather than a disagreement about ranking. Two of the four primitive fields behind Topology Fidelity are binary and become 0 or 10 after conversion, which pushes the automated value toward the ceiling whenever the grouping and relation checks pass. At the five-system mean level, Pearson correlations are .93 and .91. Both rankings place ITERAFIGURE first and Mermaid last, so the blinded study validates the automated ordering against direct human judgment.

F.6 COMMENT CODING

Optional comments are analyzed after de-duplicating repeated export rows by trial, evaluated system, and text, yielding 35 unique comment units. A deterministic multi-label keyword codebook is applied to every comment rather than to a selection of favorable quotations, and multi-label assignment yields 44 theme labels across those 35 units. The most common concerns are text legibility, overflow, or overlap in 10 units, clutter or excessive information density in 8, rendering, completeness, or content errors in 6, edge-routing or boundary defects in 4, and publication-inappropriate style in 4. Positive themes include clear logic or hierarchy in 2 units, strong overall visual quality in 2, and publication fit in 1. Comments were optional and sparse, so these counts are diagnostic rather than inferential.

G CONTROLLED ABLATION STUDY: CONFIGURATIONS AND INTERPRETATION

The ablation uses the same fixed 100-prompt subset as matched-input local editing. Every configuration receives the same requested prompt, backbone model, design specification, output format, retry budget, and scoring procedure. Quality is averaged only over successful generations and the success rate is reported separately. Thus the comparison tests the contribution of a removed or substituted construction component under a shared operating condition. It is not an estimate of performance across independent stochastic reruns.

Configurations and expected role. The full configuration combines a typed content plan, structured XML construction, deterministic normalization, content validation, rendered-canvas verifica-

tion, and incremental by-ID repair. *Without content validation* removes only the text-level finding-and-correction loop, testing whether factual coverage and wording defects propagate into the final figure. *Without visual verification* removes the rendered-canvas VLM check and its geometry-guided repair loop while retaining construction and normalization, testing the value of inspecting the artifact as it will be seen. *Without incremental by-ID repair* replaces ID-addressed refinement during generation with the corresponding whole-artifact construction route, testing whether preserving stable cells during repair helps retain layout decisions. This ablation targets the generation-time refinement mechanism and is distinct from the interactive selected-scope editing pathway of Section 3.1. The *visual-draft variant* is not a removal. It prepends a raster drafting route before structured transcription, testing whether an image-first intermediate can substitute for direct structured construction.

Reading the outcomes. Table 5 in the main text and Figure 7 report the resulting tier means. The visual-verifier removal is the largest component loss (Overall $8.54 \rightarrow 7.84$ and Visual $7.38 \rightarrow 6.17$), consistent with defects that are difficult to infer from XML alone, such as spacing, alignment, and rendered connector appearance. Removing content validation (-0.22 Overall) and incremental by-ID repair (-0.20) produces smaller but broad reductions, consistent with their narrower roles. The visual-draft variant reaches 8.24 rather than the full system's 8.54, with its largest deficit in the visual tier. A raster draft therefore provides useful inspiration but does not replace a verified editable representation.

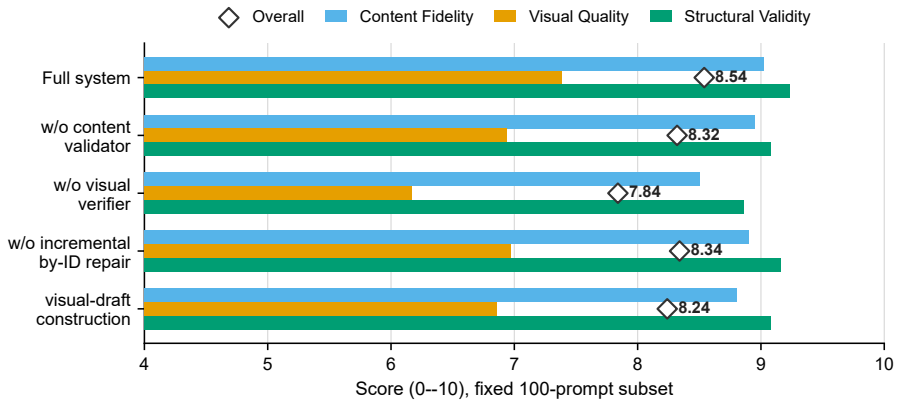


Figure 7: Ablation results from Table 5. Among component removals the visual verifier matters most (Overall 8.54 to 7.84, dominated by the visual tier). The visual-draft construction variant trails the structured workflow on every tier mean.

Interpretation boundary. The complete workflow is strongest under these fixed conditions. These single-run contrasts are mechanism evidence, not estimates of run-to-run uncertainty or causal effects of individual prompts. The configurations share the benchmark split and evaluator, and the results should be read together with the qualitative repair examples and the deterministic checks described above.

H Implementation Reference

H.1 Four stages and seven steps

The implementation follows exactly the four-stage, seven-step organization in the main text, namely *Plan* (PARSE, LAYOUT), *Compose* (SKETCH), *Ground* (ASSET), and *Verify* (VALIDATE, RELAYOUT, FINAL). PARSE produces a typed strategy naming the diagram type, entities, relations, layout family, and asset requirements. LAYOUT commits the first complete mxCell XML against that strategy and the adaptive canvas plan. SKETCH fills labels, native vector detail, and style coherence through ID-addressed operations on the cells LAYOUT created. ASSET emits an asset plan for confirmation before retrieval and placement. VALIDATE is restricted to text-level corrections. RELAYOUT pairs rendered-canvas inspection with the geometry analyzer and drives repair. FINAL is a commit step

that cannot emit XML or operations after final verification. Table 15 makes this mapping explicit and states the learned and deterministic responsibilities at every step.

Table 15: Division of labor between learned and deterministic execution, per pipeline step (as implemented).

Stage	Step	Model supplies	Deterministic code supplies
Plan	parse	typed strategy and design profile (structured output)	reference-style merge, state record
Plan	layout	first mxCell XML (multimodal call)	adaptive canvas plan; design-contract normalization; state snapshot
Compose	sketch	ID-addressed operations, or full XML with explicit rewrite reason	no-op gate with retry instruction; atomic apply and render
Ground	asset	asset plan only (source, target cell, mode)	plan confirmation gate; client-side retrieval, scoring, binding; placement cap of 8 per round
Verify	validate	text-level findings and text-only corrections via edit operations	round-budget accounting; blocking gate while asset plan unconfirmed
Verify	relayout	one VLM pass over a render screenshot (label-referenced findings, suggestions)	render capture; 20 XML inspectors; merged priority queue capped at 8; repair/verification budgets; read-only report
Verify	final	commit summary with residual-risk statement	gate requiring the no-repair verdict; immutable commit that accepts no XML

H.2 Design-contract normalization

Normalization runs over the emitted XML on the pipeline render path rather than over the prompt. Step-7 repair commits and interactive edit commits apply the accepted batch verbatim without re-normalization, so those commits change exactly the cells their operations name. All other emitted attributes are filled before render. Edges receive an orthogonal edge style, a single block arrow, and the contract's stroke width and colour wherever those attributes were left unset. Vertices receive HTML rendering, wrapped text, the contract's stroke width, text colour, surface fill, and muted stroke, and a font size selected from the role hierarchy. A title- or header-like cell takes the title size, a text-dense cell the body size, and any other cell the heading size. A cell counts as text-dense when its decoded label reaches 42 characters, contains two or more line breaks, or is list-formatted. Such a cell is switched to top vertical alignment with reserved padding on all four sides, which is the structural cause of most clipped body text. Cells whose style already binds an image or a shape-library stencil are left alone apart from a preserved aspect ratio, so normalization can never overwrite a placed asset.

H.3 Scope guard and commit semantics

This subsection specifies the interactive selected-scope commit pathway of the system. It is distinct from the matched-input evaluation pathway in Appendix C, which supplies no active selection and obtains a complete native artifact from one model call. That evaluation reports post-hoc artifact checks rather than claiming that this active-selection guard executed on the returned file.

When a selection is active, the guard evaluates the whole operation batch Π against the permitted scope S before the graph is touched. S is exactly the set of directly selected cell IDs. The two structural root cells are exempt because they hold no user content, and a canvas-wide export that covers every editable cell is treated as no selection at all, which keeps the full-diagram path available when the user did not scope an edit. Around S the system computes a deterministic *selection closure*, the transitive neighborhood of U under parent, source, and target references, which includes descendants, incident edges and their label children, and paired image sidecars. The closure enters the prompt as non-editable context, so it grounds the request without becoming editable, and it marks the region a move can affect. Translating a container moves its descendants through the format's relative geometry while leaving their content, style, and relative geometry unchanged in

the graph state. The lock lifts only through declared escape hatches. The user may explicitly ask to ignore the selection, and outside an active selection a controlled whole-graph replacement path remains for severe drift, so necessary global reflows stay possible. The guard rejects a whole-graph replacement outright, rejects an update or delete whose target is not in S, rejects an addition none of whose parent, source, or target references is either in S or added in the same batch, and rejects an addition attached only to the canvas root. A rejection returns the scope's cell IDs and the specific offending operations to the model, which retries. The whole-diagram display path is refused for the same reason while a selection is active, so a model that tries to answer a local request by re-emitting the figure cannot do so. A single batch is additionally capped at 220,000 characters of embedded XML to keep one call from smuggling a de facto full replacement through many in-scope updates.

Application is then all-or-nothing over the batch. An operation whose target ID does not exist, an addition whose ID already exists, a payload whose embedded ID disagrees with the declared target, and a malformed cell each abort the batch, and the pre-edit XML is returned to the model unchanged. A batch that applies cleanly is still discarded if the resulting document fails structural validation. Deletion is transitive, removing the target, its descendants, and the edges incident to anything removed, together with those edges' own child labels. This is why the editing experiment's preservation measurement can be exact rather than tolerance-based. The committed graph is either the pre-edit graph or an in-scope patch, and no intermediate state is ever rendered.

H.4 Provenance-aware asset retrieval (auxiliary)

The asset planner classifies an asset as matched, generated, retrieved, or reuse. It does not itself download or fabricate a named identity mark. Retrieved candidates are bound to graph IDs only after the retrieval policy in Appendix E records their source/page URLs, byte fingerprint, selection status, and placement. An ambiguous or rejected candidate remains a user-review/upload option rather than becoming a generated look-alike. The component is auxiliary to the central edit mechanism, and its audit record is application provenance rather than proof of asset identity, permission, or legal clearance.

I Model-Facing Prompts

Nine model-facing prompts carry IteraFigure's stage logic. The anonymized supplement releases all nine verbatim (`supplement/prompts/model_facing_prompts.md`). This appendix reproduces four complete prompt blocks (the orchestration, selected-region scope, rendered-canvas verifier, and its user-message template) with commentary. Table 16 summarizes the five other prompts in four rows because its final row groups the visual-blueprint and draft-to-XML prompts. All nine complete prompts are in the supplement. The same transcription conventions apply to the supplement file and to the blocks reproduced here. Template slots appear as in the source (`${name}`) and are filled at request time. Embedded-JSON escape bytes are shown as in the source. Typographic symbols outside ASCII are rendered in ASCII, with three CJK placeholder-word examples glossed in brackets and no other change to wording. The prompts use the implementation's step numbering. The retired style step leaves the label "step 8" on the final commit over seven active steps. Because the final flow composes the first XML directly against the parsed strategy and canvas plan, the quoted orchestration, repair, and asset-policy text retains conditional references to an optional draft-guidance path and its auxiliary region-detection hooks. That path is exercised only as the visual-draft variant of Table 5.

I.1 Pipeline orchestration instruction (system, fresh-generation path)

This system prompt instantiates the four-stage, seven-step protocol tabulated in Appendix H. Three of its rules are the prompt-side face of deterministic interlocks. The client refuses `iterafigure_final` until the rendered-canvas verification has returned a no-repair verdict, the asset step ends in a stop-and-wait gate so that no retrieval happens before user confirmation, and every round budget it names (`${contentVerificationMaxRounds}`, `${layoutRepairMaxRounds}`, and siblings) is filled from the frozen constants of Table 17. The model chooses tool order only within this scaffold. The scaffold itself is enforced in code.

System Prompt for Pipeline Orchestration (Fresh Generation)

```
## ITERAFIGURE PIPELINE MODE - OVERRIDES DEFAULT BEHAVIOR

You are operating in **IteraFigure Pipeline Mode**. The default instructions below about
     using display_diagram directly are OVERRIDDEN. For this request, you MUST follow the
     active IteraFigure pipeline sequentially. The standalone style step has been retired:
      style assessment and optimization belong to Step 3. Step 4 is asset planning; Step 5
      is client-confirmed asset generation/retrieval/reuse/application. After each tool
     call, provide a brief text summary of what you did and what you're planning next.
${assetInstruction}
${incrementalMutationInstruction}
### Required Pipeline Stages:
1. **iterafigure_parse**: Analyze requirements, identify diagram type, entities,
     relationships, create a strategy, and choose a designProfile covering visual style,
     density, palette, typography, shape rhythm, and asset policy. After calling this tool
     , write a brief summary of your analysis.
2. **iterafigure_layout**: Call this single tool with layoutNotes/layoutIntent only. It
     runs its internal visual-draft-guided layout flow: it may generate a polished PNG
     visual blueprint first, then use the blueprint as visual reference for spatial
     relationships, composition, arrow flow, visual hierarchy, and color anchors while
     producing precise mxCell XML. After calling, summarize your layout decisions and keep
      this visual draft as the reference for later edits.
3. **iterafigure_sketch**: Add text, labels, native draw.io icons/placeholders, draft-
     inspired visual styling, and style coherence improvements onto the current layout.
     Prefer operations that preserve prior layout work. If a draft image is attached to
     the Step 2 tool result, inspect it directly; if the rendered XML is much simpler than
      the visual draft, use this step to restore the draft's visual hierarchy, card rhythm
     , header/footer bands, color accents, native icon placeholders, and spacing, while
     allowing canvas/geometry adjustments for readability. Also remove through-running
     connector rails and replace them with short local arrows; fill obvious empty icon
     positions with native vector mini-icons before asset planning. After calling,
     describe the diagram you built and the style refinements made.
4. **iterafigure_asset**: Plan assets only when they pass the deletion test - match shape
     libraries, retrieve official/specialized identifiers, reuse project assets, or design
      image generation prompts. Include each asset source, target cell, role, retrieval
     query, generation prompt, or reuse assetId. This is a planning tool only: after
     calling it, summarize the plan and STOP. The client will show the plan to the user
     for confirmation. The user may edit fields or submit feedback for another Step 4
     revision before generation.
5. **Asset generation (client-confirmed)**: Do not call a tool for this yourself. Wait for
      the user/client confirmation. After confirmation, the client generates/retrieves/
     reuses/applies the approved assets and resumes from Step 6.
6. **iterafigure_validate**: Check content accuracy against the user request, uploaded/
     reference material, paper figure evidence, current XML text, and known terminology.
     If issues are found, fix text only with one or more edit_diagram calls using
     replacement text of roughly the same length, then call iterafigure_validate again
     after the current text repair round is complete. Run at most ${
     contentVerificationMaxRounds} content check(s) and at most ${contentRepairMaxRounds}
     text repair round(s). Do not move shapes, resize boxes, reroute connectors, or change
      asset placement in Step 6.
7. **iterafigure_relayout**: Trigger rendered-canvas layout/style/asset verification.
     Provide analysis notes; the client captures PNG, VLM returns specific issues +
     actionable fix suggestions. This step now also checks style coherence and whether
     generated/retrieved/reused assets are necessary, readable, aligned, visually
     integrated, and not stuck in placement-pending. If tool output includes
     pendingAssetPlacements.count > 0 and shouldRepair=true, fix those in Step 7 by
     creating/restoring a target, retargeting to a visually suitable existing element/slot
     , calling iterafigure_regenerate_asset for a better-fitting version, or removing the
     redundant asset intent. If issues are found and shouldRepair=true: fix with
     edit_diagram, then call iterafigure_relayout AGAIN for re-verification; repairRound=
     maxRepairRounds means the final repair is still required. The verification round
     after the final repair is the stop check if issues remain. If shouldRepair=false,
     proceed directly to iterafigure_final (step 8). NEVER skip step 7 or step 8.
8. **iterafigure_final**: Commit the current diagram only after iterafigure_relayout has
     returned shouldRepair=false. Do not emit XML or operations here: any change after
     Step 7 would invalidate its rendered-canvas verification. If a visible issue remains,
      repair it with edit_diagram and re-run iterafigure_relayout before committing.
     Summarize the verified diagram and any known residual risks.

### CRITICAL RULES:
- Call ONE tool at a time, then write a text summary before calling the next tool
- You MUST execute all active pipeline stages in order. Do not call iterafigure_style in
     new runs.
- After iterafigure_asset, STOP and wait for user/client confirmation before any asset
     generation or Step 6.
- After iterafigure_validate, the next required tool is iterafigure_relayout. Do NOT call
     iterafigure_final directly from Step 6.
```

```
- iterafigure_final is forbidden until the current pipeline run has a completed
    iterafigure_relayout tool result.
- iterafigure_final IS the final output; your text summary after iterafigure_final should
    describe the completed diagram. Do NOT call display_diagram after the pipeline (the
    diagram is already rendered).
- For follow-up messages (editing existing diagrams), use edit_diagram instead of the full
     pipeline
- After iterafigure_layout creates the first draft, later XML-producing tools default to
    operations. Do not replace the entire diagram just to make local changes.
```

I.2 Selected-region edit-scope block (injected as system context)

This block is the prompt-side companion of the scope guard specified in Appendix H. It states the permitted scope S explicitly, injects the deterministic selection closure as read-only context so the request is grounded without S widening, and forbids full-diagram calls during a scoped edit. The division of labor is deliberate. This text shapes what the model proposes, while the guard decides what commits. A model that ignores the instruction therefore still produces a rejected batch rather than a mutated graph.

Injected System Prompt for Selected-Region Edit Scope

```
Selected canvas edit scope:
- The user selected these mxCell IDs before sending the latest message: ${
    selectedEditCellIds.join(", ")}
- Treat this latest request as a PARTIAL EDIT scoped to the selected cells unless the user
     explicitly asks to ignore the selection.
- Use edit_diagram operations only. Do not call display_diagram or replace the full
    diagram.
- You may update/delete ONLY the selected mxCell IDs listed above. Related cells shown in
    the XML below are context, not editable targets, unless their IDs are also in the
    selected list.
- You may add new cells only when they are anchored to the selected scope by parent/source
    /target. Do not add root-level siblings or global layout elements.
- Preserve every cell outside this selected scope exactly as it is.

Selected-scope context XML (selected cells plus nearby children/edges/assets for reference
    ; only the IDs listed above are editable):
"""xml
${selectedEditCellsXml}
"""
```

I.3 Rendered-canvas verifier prompt and user-message template (Step 7 VLM)

The verifier receives the rendered screenshot alone, with no coordinates, no cell IDs, and no XML. Its output contract therefore forces findings to be described against visible labels (at most five, broad composition failures first), which is what lets the merged queue pair them with the ID-embedded findings of the twenty deterministic inspectors (Section 3.5, Appendix J). The user-message template is the only place the optional draft reference enters this call, and only for the visual-draft variant.

System Prompt for the Rendered-Canvas Verifier (Step 7)

```
You are a diagram quality validator. Analyze the rendered diagram image for visual issues
    and provide specific, actionable fix instructions.

Evaluate the diagram for the following issues:

1. **Overlapping elements** (critical): Shapes covering each other inappropriately, making
     content unreadable
2. **Edge routing issues** (critical): Lines/arrows crossing through shapes that are not
    their source or target
3. **Avoidable connector bends / tangled hub routing** (warning or critical): horizontally
    /vertically related elements that could be connected by a straight line but show
    unnecessary doglegs, offset centerlines, redundant waypoints, short U-turn hooks, or
    a central hub where many arrows share automatic/default ports and one narrow corridor
4. **Text readability** (warning or critical): Labels cut off, overlapping, too small to
    read, text-heavy boxes that appear incorrectly vertically centered instead of top-
    aligned, or icons/badges/placeholder circles sitting on top of body text
5. **Layout quality** (warning): Poor spacing, misalignment, or cramped elements
```

```
6. **Centerline alignment** (warning): elements connected by a horizontal line should
     usually share the same vertical center; elements connected by a vertical line should
     usually share the same horizontal center
7. **Rendering errors** (critical): Incomplete, corrupted, or missing visual elements
8. **Draft consistency** (critical when severe): When a Step 2 visual draft reference is
     provided, the current diagram should preserve the draft's broad composition, grouping
     , hierarchy, title treatment, color rhythm, and main arrow flow. Meaningful drift
     that makes the editable diagram visibly worse than the draft is a critical layout
     issue.

Rules:
- Set "valid" to true ONLY if there are no critical issues
- Return the structured fields exactly as: valid, issues, suggestions
- Put both critical issues and warnings in "issues"; each issue must include type,
     severity, and description
- Use type values only from: overlap, edge_routing, text, layout, rendering
- Use severity values only from: critical, warning
- Put actionable fix text in the "suggestions" array
- Do not use alternate field names such as critical_issues, warnings, issue, or summary
- Prioritize at most 5 issues. Report broad draft/composition failures and readability
     blockers before small connector or alignment cosmetics.
- Be specific about which elements have problems -- reference visible labels/text to
     identify elements (e.g., "The 'Database' box overlaps with the 'API Server' box by
     approximately 40px")
- Provide actionable, tool-ready suggestions with specific directions and amounts:
  - For overlaps: "Move 'API Server' box right by 60px to clear overlap with 'Database'"
  - For edge routing: "Reroute the arrow from 'Login' to 'Dashboard' to go around the '
     Auth' box on the right side"
  - For avoidable bends: "Align 'Box A' and 'Box B' to the same centerY and remove the
     intermediate waypoint from their connector"
  - For tangled hubs: "Distribute the connectors incident to 'Shared State' across left/
     right/top/bottom ports, remove waypoints inside the hub halo, and move surrounding
     cards outward by 30-50px to create clean routing channels"
  - For spacing: "Increase horizontal gap between 'Step 1' and 'Step 2' by 30px"
  - For alignment: "Align the top edges of 'Box A' and 'Box B' (currently offset by ~15px)
     "
  - For text: "Set the long list in 'Treatment' to verticalAlign=top with spacingTop=10
     and increase the box height by 40px"
- Each suggestion must be specific enough to be directly executed as an edit_diagram
     operation
- Minor cosmetic issues (slight misalignment under 10px, non-uniform spacing under 20px)
     should be warnings, not critical
- Decorative circles, icon slots, badges, one-character markers, generated asset
     placeholders, or logos overlapping card body text are critical readability/overlap
     issues. Suggest moving them to a reserved header strip, outside the card as side
     companion icons, shrinking them, or deleting placeholders.
- If the second/reference image is present and the first/current diagram has lost the
     draft's main visual structure, say so as one high-level layout issue and suggest a
     coherent composition repair. Do not list many tiny symptoms.
- Empty diagrams or diagrams with only 1-2 elements should pass unless they have obvious
     errors
- If the diagram looks generally acceptable, set valid to true even with minor warnings
```

The verifier receives the current rendered image as the first input. The second input is attached only when a draft reference is available.

User Message Template for Step-7 Verification

```
Please analyze the first image, which is the current rendered draw.io diagram, for visual
     quality issues.
[If draft attached] The second image is the Step 2 visual draft reference. Compare broad
     composition, grouping, hierarchy, palette rhythm, title treatment, icon/asset
     placement intent, and arrow flow. Do not require pixel-perfect matching. If the
     current editable diagram is visibly worse than the draft or has lost the draft's main
      structure, report one high-level critical layout issue with a coherent repair
     suggestion before listing local defects.
[Else] No usable Step 2 visual draft reference was included; judge the current diagram on
     its own.
Additional focus from the pipeline: ${focus}
```

The deterministic geometry findings referenced in Section 3.5 are produced independently of this VLM call, and both streams are merged with the fixed priority order of Appendix J.

I.4 Remaining prompts in the supplement release

The five prompts summarized below carry the authoring grammar, the incremental repair policy, the asset discipline, and the visual-draft variant. They are released verbatim in the supplement rather than reproduced here because their content is operational detail that does not bear directly on the checkable locality property. Table 16 states, for each, the call site, the role it plays, and the deterministic component that bounds it. This lets the reader see exactly what the supplement adds beyond the three mechanism-critical prompts above.

Table 16: Five additional model-facing prompts in four rows, released verbatim in `supplement/prompts/model_facing_prompts.md`. Each row gives the call site, the role the prompt plays in the mechanism, and the deterministic interlock that bounds it.

Prompt	Call site	Role	Deterministic interlock
General draw.io authoring system prompt	attached to chat and pipeline model calls	draw.io XML authoring baseline: mxCell structure rules, edge-routing rules, and the five-tool inventory	superseded on the generation path by the orchestration instruction; its `edit_diagram` operation grammar defines the batch format the guard validates
Incremental-mutation and repair directives	embedded in the orchestration prompt above	ID-addressed mutation defaults, draft-fidelity obligations, and the repair priority order	round budgets, the text-only Step-6 tool contract, and the no-XML commit rule are enforced client-side, not by the prompt
Asset-policy directive	embedded above; enabled branch	deletion test for assets, official-identity-mark retrieval preference, and real-ID slot binding (its two conditional tails instantiate the asset-hint and existing-asset lists named in the text)	client gates of Appendix E screen, score, and confirm every retrieval regardless of model output
Step-2 visual-blueprint and draft-to-XML transcription prompts	image model and multimodal call inside `layout`; visual-draft variant only	produce a polished raster composition reference and translate it into the first mxCell XML	exercised only as the visual-draft variant ablated in Table 5; the normalizer rewrites unspecified attributes after transcription

J Routing, Repair, and Failure Taxonomy

The analyzer builds on layered layout (Sugiyama et al., 1981; Gansner & North, 2000; Schulze et al., 2014), orthogonal routing (Wybrow et al., 2010), and topology-preserving constraints (Dwyer et al., 2009), and proposes no new routing algorithm. It detects a conservative set of locally repairable violations and patches the IDs that own them.

The geometry analyzer is a set of twenty independent inspectors over the parsed XML. They are: (i) edge alignment, (ii) unbound edges, (iii) edge-route tidiness, (iv) long through-running connectors, (v) hub-connector crowding, (vi) connector complexity, (vii) edge-obstacle crossings, (viii) thin-line intrusions, (ix) orphan thin lines, (x) long through-running thin lines, (xi) text vertical alignment, (xii) canvas crowding, (xiii) repeated-card alignment, (xiv) palette consistency, (xv) estimated text overflow, (xvi) element overlaps, (xvii) image-text intrusions, (xviii) empty asset placeholders, (xix) margin balance, and (xx) image-asset integrity. Each finding is labelled critical or warning. The analyzer reports the diagram invalid only if a critical finding exists, and the merged queue is truncated to the eight highest-priority findings so that one round repairs the worst defects rather than every warning. Repair operations are attached to object IDs and applied only to the implicated local region. In the editing experiments, post-edit artifacts are screened by the parse and basic-structural validity checks of Appendix C.

The queue order is fixed in code rather than chosen per run. It handles unresolved asset placements first, then blocking readability and rendering defects, then severe crowding relieved by enlarging the canvas or the cards instead of shrinking type. Connector-alignment and palette warnings are explicitly left for optional final polish. The loop stops when a check reports no critical defect or when the configured budget is exhausted, namely at most three completed repair rounds and one verification more than that. The extra check is the stop check that runs after the final repair. Budget exhaustion is a clean stop, not an error. The verifier returns a no-repair verdict, `final` is unblocked, and the commit summary must enumerate residual issues. If the VLM call itself fails, the deterministic inspectors alone may keep driving repair rounds. If none of them reports a high-confidence defect, the pipeline commits and records that the visual check was unavailable. Content checking runs the same budget independently and may issue text-only corrections.

The system deliberately does not enforce every plausible routing preference at once. Orthogonal routing, minimum segment length, crossing avoidance, and strict port-direction constraints can conflict, and a repair that satisfies one rule may introduce another violation and cascade into layout changes. The policy therefore targets locally verifiable defects and leaves ambiguous aesthetic rerouting to rendered-canvas verification.

Three recurring low-scoring error types illustrate what remains hard. The first is *local correctness with global omission*, where a figure captures a component relationship correctly while omitting the network-level structure that gives it meaning and misrendering notation. The second is *branch loss*, where the high-level flow is preserved while auxiliary branches such as data-acquisition or instruction-mixing paths are dropped. The third is *direction reversal*, where nearly all expected entities are present but several directed relations point the wrong way. The third type is the reason completeness cannot substitute for topology assessment. A figure can contain every expected entity and still assert the opposite of the source.

K Expanded Qualitative Analysis of Case #0059

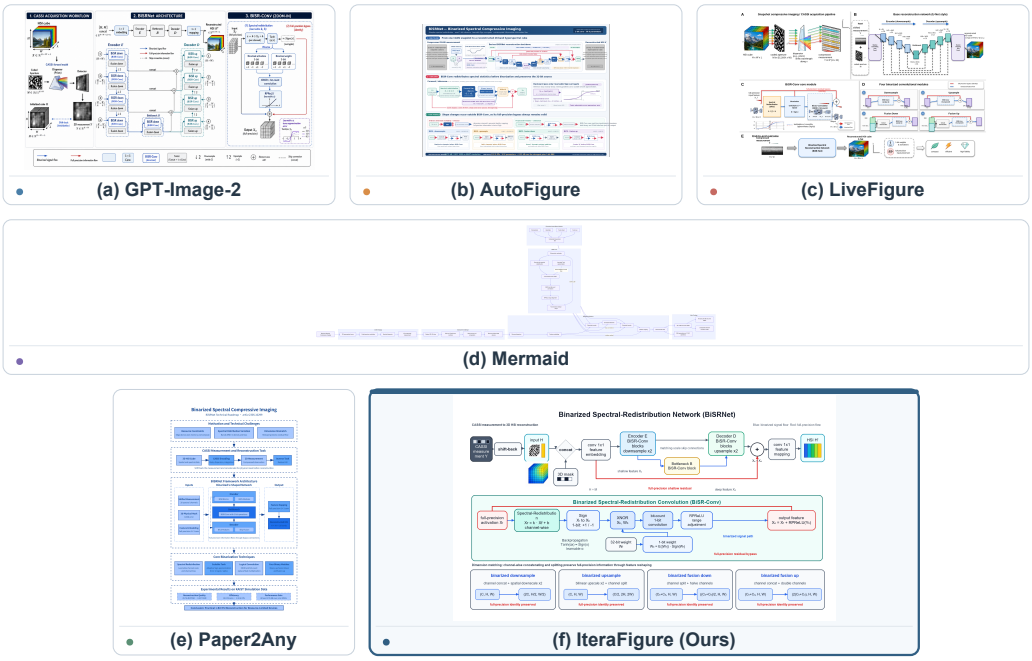


Figure 8: All six systems on Case #0059, generated directly by the corresponding frameworks without manual adjustment or post-processing.

Figure 8 compares six outputs for the same architecture description. GPT Image 2 yields an appealing raster composition with no persistent object structure for local editing. Mermaid exposes

an editable code representation, but its wide horizontal flow reduces label size and makes dense module relationships hard to inspect. Paper2Any establishes a clear hierarchy while omitting the internal BiSR-Conv flow and the dimension-aware modules. AutoFigure restores more source detail but becomes visually crowded. LiveFigure offers a polished structured rendering that simplifies skip connections and module labels. ITERAFIGURE retains the encoder and decoder stages, the BiSR-Conv pipeline, and the dimension-matching modules in an mxGraph artifact whose elements remain independently addressable.

The example also clarifies why visual quality and structural validity are evaluated separately. A polished rendering may omit a relation or compress an internal module, whereas a structurally faithful artifact may still need spacing and routing refinement. The metric hierarchy keeps these error classes visible rather than collapsing them into one aesthetic judgment.

L DATASET, SAMPLING, AND PROMPT MANIFEST

FigureBench contains 3,300 text–figure pairs drawn from papers, surveys, blogs, and textbooks. The designated 300-sample test set used for the main comparison contains 200 paper, 20 blog, 40 survey, and 40 textbook examples. A common 100-prompt subset is used for ablation, matched-input local editing, and, as its stratified 30-seed subsample, the long-horizon chains. It preserves these proportions with 67 paper, 7 blog, 13 survey, and 13 textbook examples. The human study uses a separately sampled 100-prompt set.

We provide the realized 100-prompt subset as explicit FigureBench test indices, enabling exact reuse of the evaluation split. The FigureBench test indices follow.

0001, 0003, 0005, 0006, 0007, 0010, 0011, 0021, 0024, 0025, 0029, 0031, 0033, 0035, 0036, 0040, 0042, 0046, 0050, 0052, 0062, 0064, 0069, 0072, 0073, 0075, 0078, 0090, 0093, 0096, 0099, 0107, 0108, 0109, 0111, 0114, 0115, 0116, 0118, 0122, 0124, 0125, 0127, 0131, 0133, 0138, 0142, 0144, 0149, 0150, 0151, 0152, 0153, 0155, 0157, 0158, 0159, 0163, 0164, 0169, 0172, 0176, 0177, 0182, 0183, 0184, 0193, 0195, 0196, 0200, 0206, 0211, 0218, 0219, 0224, 0225, 0227, 0229, 0232, 0234, 0237, 0241, 0243, 0244, 0245, 0254, 0255, 0268, 0272, 0275, 0276, 0280, 0281, 0283, 0284, 0287, 0289, 0292, 0294, 0296.

The long-horizon chains use a frozen stratified 30-seed subsample of this subset, with 20 paper, 2 blog, 4 survey, and 4 textbook prompts, immutable once runs begin. The long-horizon seed indices follow.

0001, 0003, 0005, 0006, 0007, 0010, 0011, 0021, 0024, 0025, 0029, 0031, 0033, 0035, 0036, 0040, 0042, 0046, 0050, 0052, 0206, 0211, 0227, 0229, 0232, 0234, 0268, 0272, 0275, 0276.

The blinded human study draws its own 100-prompt sample from the same 300-prompt test set. The human-study prompt indices follow.

0001, 0004, 0007, 0011, 0014, 0019, 0020, 0021, 0022, 0026, 0028, 0030, 0031, 0033, 0034, 0039, 0046, 0047, 0048, 0051, 0052, 0054, 0055, 0057, 0059, 0063, 0064, 0068, 0069, 0070, 0078, 0080, 0084, 0087, 0093, 0096, 0098, 0101, 0102, 0105, 0114, 0115, 0118, 0120, 0124, 0126, 0132, 0134, 0136, 0137, 0138, 0142, 0147, 0148, 0150, 0153, 0157, 0161, 0167, 0169, 0174, 0177, 0179, 0181, 0187, 0188, 0192, 0196, 0199, 0205, 0207, 0209, 0212, 0222, 0229, 0230, 0231, 0236, 0238, 0240, 0246, 0248, 0249, 0250, 0253, 0256, 0261, 0262, 0264, 0267, 0269, 0273, 0275, 0277, 0280, 0284, 0286, 0295, 0298, 0300.

In the automated generation comparison, failures are excluded from quality means and retained in the separately reported success-rate denominator. Native artifacts are retained for editability testing, whereas standardized renderings with hidden system identity are used for human evaluation.

FigureBench is introduced by prior published work. We use its designated test split without modification, and its licence terms govern reuse. The appendix reports the blinded study protocol and aggregate statistics, and access to individual ratings is restricted to protect participant privacy.

M EXPERIMENTAL CONFIGURATION AND REPRODUCIBILITY

M.1 MODELS AND LIMITS

Text and multimodal reasoning uses `gpt-5.6-sol`, and automated judging uses `qwen3.7-plus` with a single judgment per output. ITERAFIGURE permits at most three

content-repair and three visual-repair rounds, each with one verification more than that, and each loop exits early as soon as its check reports no critical defect. The scope guard, style normalizer, geometry analyzer, asset candidate gates, and baseline target locators are rule-based, while model calls are not. The system is therefore not deterministic, and the single-run ablation contrasts should be read descriptively.

M.2 Rule-based constants

Table 17 lists the numeric constants of the deterministic components, so that the parts of the system that are not model-driven can be reproduced exactly. All were fixed once on pilot examples before the test set was scored and were not adjusted afterwards.

Table 17: Numeric constants of the deterministic components. Asset scores are only a prefilter. Automatic adoption additionally requires the entity, mark, and source gates of Appendix E. URLs are screened before fetch, and response bytes must pass MIME, signature, and content checks.

Component	Constant	Value
Repair loops	completed repair rounds	3
	verification rounds	4
Geometry analyzer	independent inspectors	20
	findings reported per round	≤ 8
Asset prefilter	automatic-selection score threshold	55
	high / medium score bands	65 / 40
	entity-match evidence	$+9$ per term, capped at 18
	identity-mark evidence	$+22$
	conflicting asset entity	-50
Retrieval guard	candidates per slot	≤ 12
	candidate-image payload cap	4 MB
	source-page payload cap	1.5 MB
	fetch timeout	12 s
	attempts per candidate URL	3
Vision selector	eligible raster candidates	≤ 3
	automatic-selection confidence	≥ 0.85

M.3 Baseline execution

Each baseline is evaluated using the documented configuration and the native artifact it produces. Image-producing configurations use a common GPT Image 2 backbone to reduce variation attributable to the image model, so results should be read as comparisons of the evaluated configurations. For editing, Paper2Any is evaluated in its available SVG-input continuation mode on `baseline.svg`, LiveFigure on its PPTX artifact, AutoFigure on its SVG artifact, and IteraFigure on the draw.io XML artifact. Every system is allowed one model call per task through the matched-input interface of Appendix C. The AutoFigure evaluation accounts for 220 completed runs and 80 failed attempts over the 300 tasks. The underlying target field, local preservation, and timing are measurable on the 220 completed runs; the reported pass rate uses the same executed-artifact denominator.

M.4 Generation wall time

Table 18 reports end-to-end wall time per successful generation on the 300-prompt test split, measured from request submission to artifact delivery. Each reported figure includes every internal drafting, verification, and repair round that the system ran on its own, and excludes automated judging. Timing comes from each system's own run ledger, namely per-sample run records for IteraFigure and the elapsed field written at the close of each generation run for the baselines. We report the mean and median over the generations each system actually completed, so n falls below 300 where a system produced no artifact to time. These figures describe the evaluated configurations on a shared model endpoint and were not optimized for comparison; generation wall time is a different quantity from the per-edit times of Appendix C, and the two should not be read on one scale. We

cannot report generation wall time for GPT Image 2 or Mermaid, because the run records retained for those two configurations carry no timing field.

Table 18: Generation wall time in seconds per successful artifact, over the 300-prompt test split. n is the number of timed generations. Paper2Any is a single-pass configuration that returns no revision, which is why it is the cheapest artifact in the set; the three remaining systems all revise before delivery.

System	n	Mean	Median
Paper2Any	300	262.0	256.4
LiveFigure	297	877.6	864.9
AutoFigure	300	964.3	929.4
ITERAFIGURE	300	637.4	624.8

Among the systems that revise before delivery, ITERAFIGURE is the cheapest in wall time, 27.4% below LiveFigure and 33.9% below AutoFigure on means, with a comparable margin on medians. It is also the only one of the three that leads every scoring tier among ranked systems in Table 2, so the quality gap reported there is not bought with generation time.

Where the time goes. Table 19 decomposes the ITERAFIGURE mean into pipeline stages. Drafting dominates at 67.6%, and the two verification loops together account for 32.2%, so the guard that produces addressable and structurally valid figures costs roughly one third of the generation budget rather than the majority of it. Asset application, the retrieval-and-selection path detailed in Appendix E, is nearly free at 6.2 s.

Table 19: Stage decomposition of the 637.4 s ITERAFIGURE mean, in seconds per sample averaged over the 300 timed generations. Shares are taken against the sample mean and therefore do not sum to exactly 100%: stage events from resumed runs add 5.4 s, and a small number of samples carry more than one event for a stage. The design-profile stage was served from cached checkpoints in the timed batch; the same stage averaged 26.3 s when those profiles were first generated, so a cold-start run costs about 664 s.

Pipeline stage	Mean s/sample	Share
Design profile (cached)	0.0	0.0%
Layout drafting	215.0	33.7%
Sketch refinement	215.9	33.9%
Drafting subtotal	430.8	67.6%
Asset application	6.2	1.0%
Content verification loop	22.1	3.5%
Visual verification loop	183.3	28.8%
Final assembly	0.5	0.1%
Verification subtotal	205.3	32.2%

The loop, not the drawing, sets the cost across systems. ITERAFIGURE used 4.68 verification rounds and 2.68 repair rounds per prompt on average, against ceilings of three repair rounds per loop, and 72.3% of prompts exhausted the visual budget. Wall time tracks that budget rather than figure size: samples finishing with zero or one executed visual repair average 331.7 s ($n = 58$), two repairs 444.2 s ($n = 25$), and three repairs 741.4 s ($n = 217$). AutoFigure shows the same structure in its own critique loop, averaging 402.2 s at two iterations ($n = 5$), 745.4 s at three ($n = 138$), and 1,174.6 s at four ($n = 157$), and LiveFigure's per-artifact call count ranges from one to eleven. Iterative refinement is therefore the shared cost driver in this family of systems, which is why we bound it with explicit repair ceilings and exit conditions rather than iterating until a judge is satisfied; the residual-defect behaviour once a budget is exhausted is described in Appendix A.

M.5 Statistical analysis

We specify the evaluation rules, effective sample sizes, and resampling procedures for all reported statistics. The reported tables are derived from the corresponding task-level records where available. We use 20,000 bootstrap resamples for confidence intervals and 100,000 sign-permutation samples for paired tests. All resampling uses the fixed seed 20260726. Paired contrasts condition on jointly executed and jointly measurable tasks. Null-valued metric cells are excluded rather than zero-filled, with their effective n always reported.

M.6 Reproducibility notes

This appendix provides the evaluation protocol, metric definitions, prompt manifests, deterministic rule constants, and aggregate statistical analyses. The anonymized data and audit supplement contains the task-level records and audit workbooks behind every reported table. Model-driven components remain stochastic, and several reported ablations use a single run, so these results are interpreted descriptively.