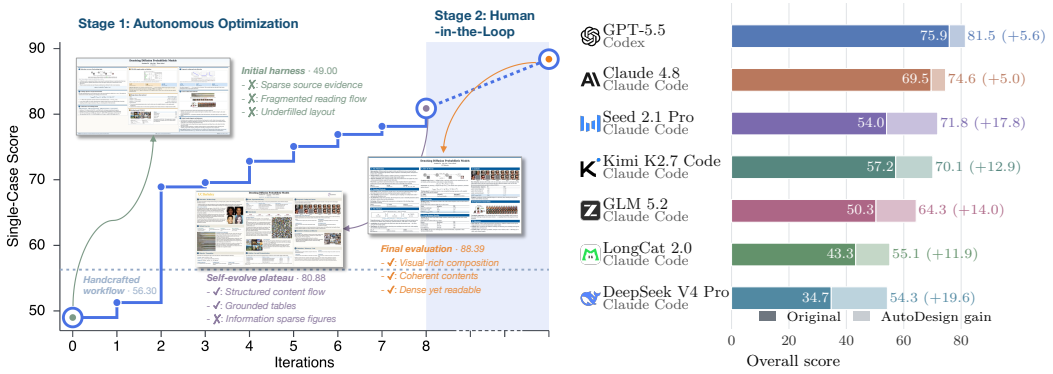
AUTODESIGN: META-HARNESS OPTIMIZATION FOR LONG-HORIZON AGENTIC DESIGN

Anonymous authors
Paper under double-blind review

ABSTRACT

Transforming multimodal sources into condensed and structured media outputs can be fundamentally conceptualized as a long-horizon agentic process centered on a model-harness system. While an ideal harness system should align with human design priors and accumulate reusable experience through empirical exploration to drive recursive self-improvement, existing paradigms remain static and fall short of this capability. In this paper, we present **AutoDesign**, a framework that aligns with human design priors, where a meta-harness optimizer guides a code agent to recursively improve harness based on rollout feedback. To instantiate and evaluate this framework, we focus on the academic paper-to-poster generation task and introduce **PosterBench**, comprising a 100-paper Main Track spanning five disciplines and **PosterBench-mini**, a shared 10-paper subset for controlled evaluation. On the PosterBench Main Track, AutoDesign achieves the highest score of 78.32, surpassing the closed-source commercial system Claude Design by 7.45 points. Across seven controlled code agent–model configurations, integrating the learned **DesignHarness** consistently improves performance, increasing the average PosterBench Score from 54.99 to 67.39 (+12.4). In a fully autonomous long-horizon loop, it executes 253 tool calls and 11 editing turns within 40 minutes for under $3, reaching average conference-poster quality in human evaluation. A system-blind human study further demonstrates that AutoDesign achieves the highest human preference among evaluated systems.



(a) Meta-harness optimization trace (b) Gains from **DesignHarness**

Figure 1: **AutoDesign improves harnesses and artifacts.** (a) One representative paper's poster score across meta-harness iterations: autonomous optimization plateaus before human guidance yields further gains. (b) **DesignHarness** improves all seven code agent–model configurations by 5.0–19.6 **PosterBench** points, reaching 81.5.

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.

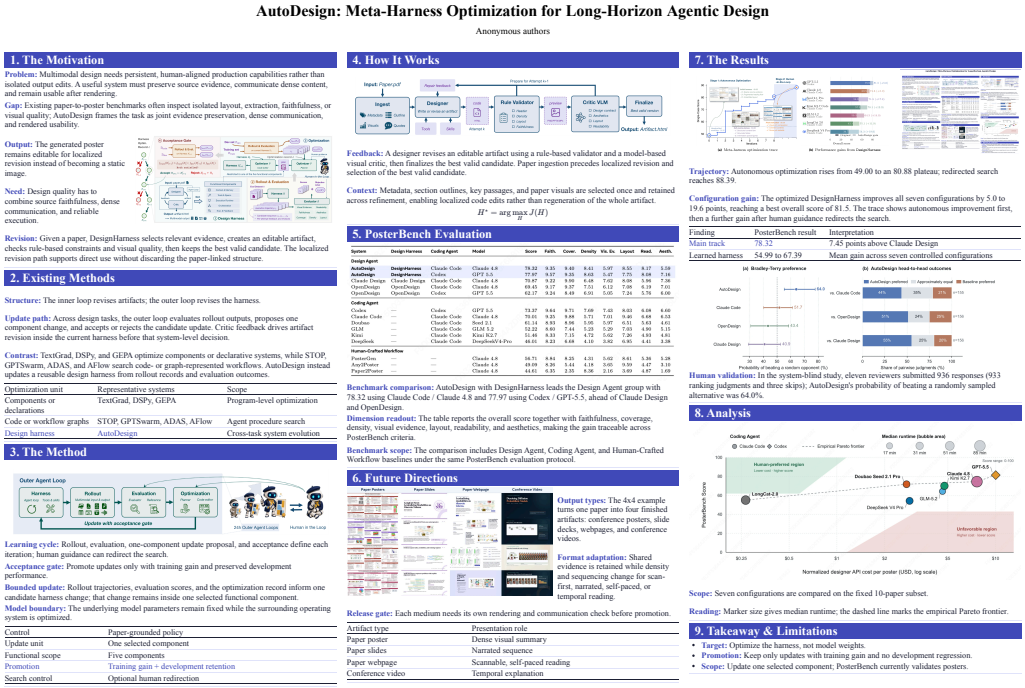


Figure 2: **AutoDesign for AutoDesign.** Its self-paper poster, generated through source ingestion, iterative refinement, critic feedback, and finalization in approximately 40 minutes with negligible human intervention.

1 Introduction

Human communication often involves understanding, organizing, and presenting information from diverse multimodal sources into human-facing artifacts, *e.g.,* webpages, slides, posters, and videos (Fu et al., 2022; Pang et al., 2025; Zheng et al., 2025; Chen et al., 2025; Zhu et al., 2025). Achieving such multimodal input-to-output transformation (Wu et al., 2023; Chameleon Team, 2024; Kim et al., 2026; Meituan LongCat Team et al., 2026) requires the ability to extract relevant evidence, reason over heterogeneous information, plan intermediate steps, and iteratively improve outputs based on feedback (Choi et al., 2026; Liu et al., 2026), which naturally positions multimodal design as a suitable yet challenging long-horizon task for agentic coding. However, developing such systems remains difficult due to complexity of real-world workflows and reliance on extensive human feedback.

Current multimodal design systems seek human-aligned design priors through cycles of generation, critique, and revision, often using visual references or design-specific feedback (Sun et al., 2026; Zheng et al., 2025; Liu et al., 2026). At the response level, feedback can revise the current output (Madaan et al., 2023), while agentic systems can retain reflections, skills, or task experience across attempts (Shinn et al., 2023; Wang et al., 2023; Zhao et al., 2024). However, unlike human creators who continuously accumulate knowledge from successful revisions and failures, such systems treat individual human-aligned feedback as transient signals rather than reusable design knowledge. The unresolved question for multimodal design is how to convert multimodal evidence, structural constraints, feedback, and human preferences into persistent design-aligned capabilities of the production system (Ren et al., 2026).

To close this gap, we propose **AutoDesign**, which frames human-aligned design generation as a meta-harness optimization problem (Robeyns et al., 2025; Lee et al., 2026b;a; Zhang et al., 2026a; Lin et al., 2026; Ren et al., 2026): an agentic system recursively optimizes the design harness itself, rather than an individual artifact, based on an evaluation grounded in human preferences, *e.g.*, annotated reference artifacts or natural language guidance. As illustrated in Figure 3, `AutoDesign` operates through two nested loops. The inner loop is the design harness, which transforms the source context into an editable output and iteratively revises it under critic feedback. The outer loop is the

meta-harness, which optimizes the design harness across tasks. It first grounds human preferences by initializing an evaluator from annotated reference artifacts. Given this human-aligned evaluator, the meta-harness aggregates rollouts and evaluation scores across tasks to identify recurrent failures, and directs a coding agent as the optimizer to propose a bounded update to the current design harness at each iteration. To prevent overfitting, the update is admitted through an acceptance gate, which accepts it only when it improves performance on the training set without degrading performance on the development set (Nguyen et al., 2026).

Accepted updates accumulate to **DesignHarness**, an executable system that autonomously ingests the source, generates and iteratively refines the artifact under critic feedback, and finalizes it into a readable, visually coherent artifact. We instantiate this system for academic paper-to-poster generation, a challenging design task that must condense long, multimodal scientific sources into a single legible, visually coherent poster while preserving traceable evidence (Jaisankar et al., 2025; Pang et al., 2025; Sun et al., 2026; Vinaykumar et al., 2026). In this task, DesignHarness can produce ready-to-use conference posters that align closely with human preferences while reducing the time and manual effort required for high-quality poster production.

Evaluating a design harness requires a benchmark and evaluation protocol that jointly measure source fidelity, dense scientific communication, and rendered usability. Existing paper-to-poster benchmarks each cover only part of these aspects, such as layout, extraction, faithfulness, or visual quality, and fall short of a comprehensive, task-level protocol (Wang et al., 2024; Jaisankar et al., 2025; Pang et al., 2025; Sun et al., 2026; Vinaykumar et al., 2026). **PosterBench** addresses this gap with a 100-paper spanning five disciplines, together with PosterBench-mini, a 10-paper subset for rapid testing. Its seven-dimensional rubric combines rule-based algorithm checks, rubric VLM judgments, and hybrid methods where both signals apply. One main track evaluates the complete system, while three controlled tracks isolate component-level effects. A system-blind human study independently validates the automatic protocol.

Across PosterBench-mini and PosterBench evaluations, the learned DesignHarness improves system quality while remaining practical to deploy. On PosterBench-mini, attaching DesignHarness to each of seven Code Agents raises the average PosterBench Score from 54.99 to 67.39 (+12.40 points). On the PosterBench Main Track, AutoDesign scores 78.32; under the same Claude Code and Claude 4.8 configuration, it outperforms the closed-source commercial system Claude Design by 7.45 points. **AutoDesign** also makes strong paper-to-poster generation accessible at low cost: with **DesignHarness**, LongCat-2.0 reaches 55.13 at approximately \$0.27 per poster[1]. Finally, across 933 valid system-blind pairwise judgments, AutoDesign receives the highest Bradley–Terry preference estimate, 64.0% (95% interval: 55.2–77.8%); for pairs separated by at least 20 PosterBench points, participants prefer the PosterBench preferred poster in 74.4% of cases.

Our contributions are:

1. **AutoDesign**, a meta-harness optimization framework that converts human-designed reference artifacts, rollout traces, rendering diagnostics, and evaluator feedback into persistent design priors. Over seven days, it invokes 224 subagents, records at least 123 recursive iterations, and accumulates 54 harness updates for human-aligned multimodal design.
2. **DesignHarness**, the academic paper-to-poster system evolved by AutoDesign, combines a paper-grounded Designer, editable generation, rendering, rule-based validation, and visual critic feedback for localized revision, producing usable, source-grounded posters.
3. **PosterBench**, a protocol for scientific communication and artifact reliability covering seven dimensions: faithfulness, coverage, density, visual evidence, layout, readability, and aesthetics. Attaching DesignHarness raises Claude Code (Claude 4.8) from 70.01 to 78.32 (+8.31), exceeding Claude Design by 7.45 points.
4. **DesignHarness** produces human-level academic posters through a fully autonomous loop: one run uses 253 tool calls and 11 editing turns in 40 minutes for under \$3, with negligible human intervention. A research-preview platform supports interactive use and localized revision of editable posters.

[1]The reported cost reflects LongCat-2.0's pricing policy at evaluation time: cached context incurs no charge on a cache hit.

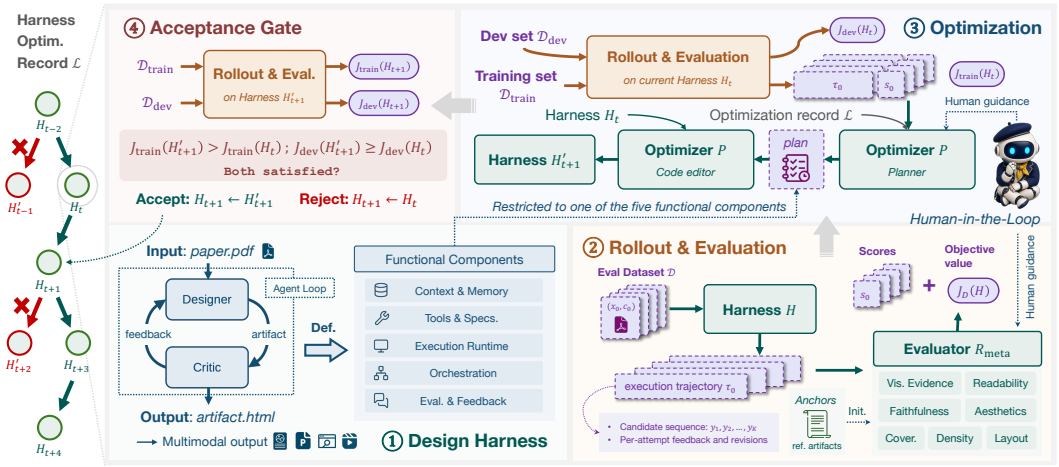


Figure 3: **Overview of AutoDesign**. The inner loop generates and revises artifacts using critic feedback. The outer loop evaluates rollouts and accepts or rejects proposed updates to one harness component, running autonomously with optional human guidance.

2 META-HARNESS OPTIMIZATION

`AutoDesign` optimizes the executable system surrounding a fixed model. Figure 3 distinguishes artifact refinement in the inner loop from persistent harness updates in the outer loop. We specify the update object, procedure, and acceptance gate below; complete definitions and pseudocode are in Section A.3.

2.1 OPTIMIZATION OBJECT

A design harness H maps a multimodal source x and design requirements c to an artifact $y \sim H(\pi_\theta, x, c)$ through an execution trajectory τ. Here c specifies the medium and user constraints; the LLM or MLLM π_θ remains fixed. The trajectory records actions, states, revisions, and feedback. The search object includes prompts, tools, and control logic (Lee et al., 2026b; Ren et al., 2026).

We decompose H into five functional components to define the scope of an update:

- **Context and memory:** source management, prompts, skills, reusable assets, and persistent state.
- **Tools and specifications:** tools and editable artifact specifications for layout, typography, and provenance.
- **Execution runtime:** the workspace and runtime for authoring, rendering, validating, and exporting artifacts.
- **Orchestration:** task routing, attempt budgets, loop control, candidate selection, fallback, and finalization.
- **Evaluation and feedback:** rule-based validation, model-based critique, and localized feedback for revision.

These components bound update responsibilities, not implementations: each outer-loop iteration changes one component, potentially across several files.

Given a user specification q and optional initial harness H_0, the meta-harness produces H_T to maximize expected artifact quality over task distribution p_{task}:

$$J(H) = \mathbb{E}_{(x,c)\sim p_{\text{task}},\, y\sim H(\pi_\theta,x,c)}\left[R_{\text{meta}}(y,x,c)\right], \qquad H^\star = \arg\max_H J(H). \tag{1}$$

The evaluator assesses artifacts against their sources and requirements. Accepted harness changes persist across tasks; repairing only the current poster leaves the policy unchanged.

2.2 Artifact Generation

We initialize a minimal inner-loop scaffold with a designer M_{design} and a critic M_{critic} (Figure 4). At step k, the designer conditions on the current artifact, previous feedback, source, and requirements; the critic examines the candidate:

$$y_k = M_{\text{design}}(y_{k-1}, f_{k-1}; x, c),$$
$$f_k = M_{\text{critic}}(y_k; x, c). \tag{2}$$

Starting from empty y_0 and f_0, the first step generates a draft from (x, c). Evaluation and revision under fixed H produce trajectory τ, retaining intermediate artifacts and execution history for subsequent harness optimization.

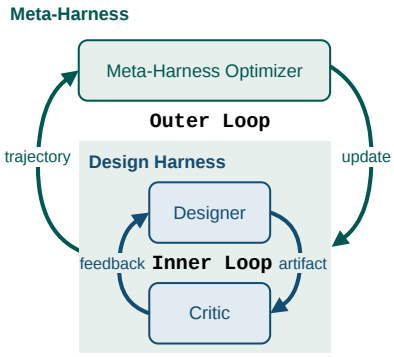


Figure 4: Artifact refinement in the inner loop; harness updates in the outer loop.

The modules' implementation remains open. Across outer-loop iterations, the meta-harness may refine prompts, tools, feedback mechanisms, and stopping policies; within each artifact run, these policies stay fixed. **DesignHarness** instantiates this abstraction with editable HTML, deterministic checks, and visual critique (Section 2.5).

The inner-loop critic supplies repair feedback. It is distinct from R_{meta}, which scores rollouts for harness optimization, and the frozen PosterBench evaluator used for final system comparison.

2.3 Harness Updates and Acceptance

The outer loop maintains one active harness through rollout, evaluation, code-update proposal, and acceptance (Figure 5; Algorithm 1 in Appendix A.3). The prime in H'_{t+1} denotes a candidate awaiting acceptance.

Rollout. At iteration t, H_t runs on the training task set $\mathcal{D}_{\text{train}} = \{(x_i, c_i)\}_{i=1}^{N_{\text{train}}}$. Each execution produces an artifact y_t^i and trajectory τ_t^i; the trajectories form $\boldsymbol{\tau}_t$. Their actions and revisions let the optimizer identify recurrent procedural failures across runs.

Evaluation. Before optimization, an evaluator coding agent constructs R_{meta} from human-annotated reference artifacts. The annotations cover seven dimensions: faithfulness, coverage, density, visual evidence, layout, readability, and aesthetics. The agent combines rule-based checks for directly measurable properties with VLM judgments for perceptual properties. Once constructed, R_{meta} remains fixed during autonomous optimization. It scores each training artifact as $s_t^i = R_{\text{meta}}(y_t^i, x_i, c_i)$, yielding the batch $\boldsymbol{s}_t$.

The optimizer receives both scores and trajectories. The optimization evaluator is separate from the frozen **PosterBench** protocol in Section A.7; a change to the former through the explicit human-guidance channel does not change the latter.

Code-update proposal. A coding agent P consumes the current harness, training evidence, and a persistent optimization record $\mathcal{L}$:

$$H'_{t+1} = P(H_t, \boldsymbol{\tau}_t, \boldsymbol{s}_t, \mathcal{L}). \tag{3}$$

The optimizer in Figure 5 acts first as a planner, dispatching trace-analysis subagents to inspect trajectories and scores. It synthesizes their findings into a plan identifying recurrent failures, component to modify, and intended changes. It then acts as a code editor, applying plan to obtain H'_{t+1}.

The single-component boundary ties each proposal to one system responsibility: orchestration covers attempt budgets, selection, and fallback, while evaluation and feedback covers validator and critic signals. A proposal can span files within that component but cannot modify another component. This boundary alone does not establish a causal gain; the gate evaluates the candidate.

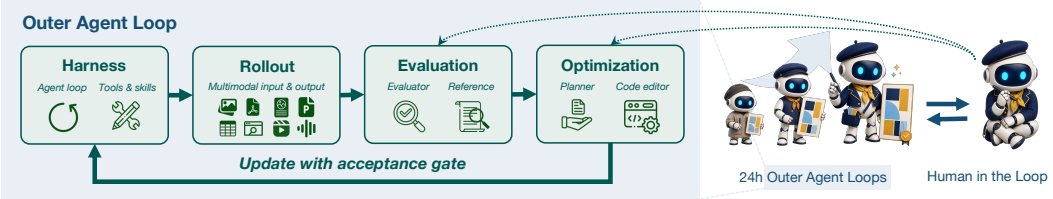


Figure 5: **AutoDesign outer loop.** Each iteration proceeds through rollout, evaluation, update proposal, and acceptance. The loop runs autonomously, with optional human guidance to redirect the evaluation and optimization.

Acceptance. The candidate is evaluated on both training tasks and an independent development set $\mathcal{D}_{\text{dev}}$. Let J_{train} and J_{dev} denote empirical versions of Equation (1) on these sets. A candidate is promoted only if

$$\text{Accept}(H'_{t+1}) \iff J_{\text{train}}(H'_{t+1}) > J_{\text{train}}(H_t) \ \wedge \ J_{\text{dev}}(H'_{t+1}) \geq J_{\text{dev}}(H_t). \tag{4}$$

The gate requires training improvement without development regression. Acceptance retains the candidate harness and its training evidence; rejection retains the current harness and existing rollout evidence. The selected harness's development score is retained for subsequent comparisons.

Raw development scores and trajectories are withheld from P and its persistent record. Acceptance decisions remain observable, so the gate limits but does not eliminate development feedback. Passing this empirical test does not guarantee improvement over the full task distribution; generalization across papers is assessed in the final system comparisons.

Persistent record. After each decision, $\mathcal{L}$ stores the repository checkpoint, training trajectories and scores, selected component, plan, code changes, and acceptance decision. It provides context for subsequent planning, comparison, reproducibility, and rollback. Rejected attempts remain recorded, allowing the planner to propose different updates. Search follows one active harness sequence rather than a tree of retained variants.

2.4 Human Guidance and Optimization Record

Optional human guidance can redirect a stalled planner or request a coding-agent revision of R_{meta} to address evaluator bias; the frozen `PosterBench` protocol remains unchanged. The seven-day optimization record includes 224 subagents, at least 123 iterations, and 54 harness updates. Figure 1a illustrates one representative paper and its intervention phases, not average cross-paper performance. Section A.3.3 details the guidance channels.

2.5 Resulting DesignHarness

The final **DesignHarness** implements the four stages in Figure 6. Full implementation details and the shared generation interface are in Sections A.4 and A.5.

Source ingestion and editable generation. Ingestion extracts paper's metadata, section outline, key passages, figures, and tables, retaining source locations for provenance (Jaisankar et al., 2025; Pang et al., 2025; Sun et al., 2026). These materials form a content brief and medium-specific artifact plan specifying the target format, claims to convey, and their supporting visual evidence. Source references let statements and visual materials be traced back to the paper and checked during revision. This context is constructed once and retained across all inner-loop steps. The coding-agent designer creates or revises editable HTML conditioned on this context, the current artifact, and repair feedback (Qu et al., 2025; Liu et al., 2026). Localized code edits allow revision without regenerating the entire output. Each candidate can be rendered or exported as a medium-specific preview for critique.

Validation and critique. Deterministic blocking checks cover asset safety and availability, provenance, severe overflow or overlap, and typographic and layout constraints. A passing candidate proceeds directly to finalization. Otherwise, validator diagnostics, including non-blocking coverage, density, and numerical-consistency checks, combine with a VLM's visual assessment of context

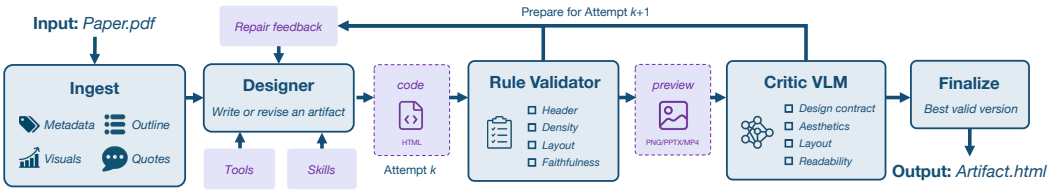


Figure 6: **Overview of DesignHarness**, optimized for human-facing artifact generation. Given source materials, the designer iteratively generates and revises an editable artifact using feedback from dual critics: a rule-based validator and a model-based visual critic. `DesignHarness` then finalizes the best valid candidate.

compliance, layout, readability, and aesthetics to form the repair signal f_k in Equation (2) (Madaan et al., 2023; Zhuge et al., 2025). The validator localizes constraint violations, while the VLM assesses perceptual properties of the rendered artifact. Together they instantiate M_{critic} and supply feedback for the designer's next attempt.

Stopping and finalization. Each run allows at most $K = 12$ attempts, stopping at the first candidate that passes all blocking checks. If none passes, fallback mechanisms select a candidate from the retained history while preserving essential safety and integrity constraints. Final rendering, mathematical typesetting, and asset inlining produce a self-contained output. The execution trajectory records these revisions while H remains fixed.

3 Evaluation

3.1 Main Results

PosterBench evaluates paper-to-poster generation on 100 papers across five disciplines: AI/ML, biomedicine and health, climate and earth environment, economics and policy, and physics and astronomy. All systems receive the same source paper and assets, and their rendered posters are scored by a frozen evaluator combining rule-based checks and VLM judgments across seven quality dimensions. The overall score averages poster scores after applying record-level ceilings. This evaluator is distinct from the optimization-time R_{meta} and is not modified by the outer loop. Ablations use PosterBench-mini, a shared 10-paper subset. The full evaluation protocol is in Section A.7; the mini leaderboard and controlled-track results are in Section A.2. Outputs are rendered to a common poster format before scoring. The shared generation interface and fixed-versus-varied factors are documented in Section A.5 and table 3.

Table 1 reports the 100-paper PosterBench Main Track, comparing design agents, standalone coding agents, and task-specific handcrafted workflow under the fixed PosterBench evaluator. **AutoDesign** attains the highest Score at 78.32. Under the matched Claude Code and Claude 4.8 configuration, AutoDesign scores 78.32, exceeding Claude Design by 7.45 points and OpenDesign by 8.87 points. Curated qualitative comparisons, which are excluded from the aggregate PosterBench results, appear in Figure 14.

Interpreting the scores. The evaluator forms a weighted rubric score from seven 0–10 dimensions, then caps severe layout damage, insufficient presentation viability, confirmed visible failures, and render-integrity violations before averaging over papers. The table reports dimension means; reweighting those means generally does not recover the mean of capped poster scores.

Human preference and benchmark alignment. In a system-blind study of four systems on the Main Track, 11 reviewers provided 933 non-skip judgments. AutoDesign has the highest Bradley–Terry estimated probability of beating a uniformly sampled alternative (64.0%; 95% interval: 55.2–77.8%).

Table 1: **PosterBench** Score and dimension scores for the `PosterBench` Main Track on the 100-paper evaluation set. Systems are grouped by their primary design mechanism. "—" denotes a layer that a system does not use.

System	Design Harness	Coding Agent	Model	Score	Faith.	Cover.	Density	Vis. Ev.	Layout	Read.	Aesth.
Design Agent											
AutoDesign	**DesignHarness**	Claude Code	Claude 4.8	78.32	9.35	9.40	8.41	5.97	8.55	8.17	5.59
AutoDesign	**DesignHarness**	Codex	GPT 5.5	77.97	9.57	9.35	8.63	5.47	7.75	8.08	7.16
Claude Design	Claude Design	Claude Code	Claude 4.8	70.87	9.22	9.90	6.48	7.62	8.08	5.96	7.36
OpenDesign	OpenDesign	Claude Code	Claude 4.8	69.45	9.17	9.37	7.51	6.12	7.08	6.19	7.01
OpenDesign	OpenDesign	Codex	GPT 5.5	62.17	9.24	8.49	6.91	5.05	7.24	5.76	6.00
Coding Agent											
Codex	—	Codex	GPT 5.5	73.37	9.64	9.71	7.69	7.43	8.03	6.08	6.60
Claude Code	—	Claude Code	Claude 4.8	70.01	9.25	9.88	5.71	7.01	9.46	6.68	6.53
Doubao	—	Claude Code	Seed 2.1	61.14	8.93	8.96	5.95	5.97	6.51	5.63	4.61
GLM	—	Claude Code	GLM 5.2	52.22	8.60	7.44	5.23	5.29	7.03	4.90	5.15
Kimi	—	Claude Code	Kimi K2.7	51.46	8.33	7.15	4.72	5.62	7.26	4.93	4.81
DeepSeek	—	Claude Code	DeepSeekV4-Pro	46.01	8.23	6.68	4.10	3.82	6.95	4.41	3.38
Human-Crafted Workflow											
PosterGen	—	—	Claude 4.8	56.71	8.84	8.25	4.31	5.62	8.61	5.36	5.28
Any2Poster	—	—	Claude 4.8	49.09	8.26	5.44	4.18	3.65	9.59	4.47	3.10
Paper2Poster	—	—	Claude 4.8	44.61	6.35	2.35	8.36	2.16	3.69	4.87	1.69

Poster-level PosterBench scores have a positive, imperfect association with tie-adjusted human preference ($r = 0.34$; 95% interval: $[0.22, 0.44]$). Agreement also depends on the score margin: the benchmark-preferred poster agrees with human choice in 51.9% of comparisons with 0–3-point gaps and 74.4% with gaps of at least 20 points. The full study design, pairwise outcomes, score-margin analysis, and uncertainty estimates are in Section A.8.1.

3.2 Ablation Studies

3.2.1 Effect of the Design Harness

Table 2: DesignHarness attachment with fixed model and coding agent (Posterbench-mini).

Configuration	Original	DesignHarness	Gain
GPT-5.5 Codex	75.87	**81.46**	+5.59
Claude 4.8 Claude Code	69.55	**74.56**	+5.01
Seed 2.1 Pro Claude Code	54.01	**71.83**	+17.82
Kimi K2.7 Code Claude Code	57.20	**70.12**	+12.92
GLM 5.2 Claude Code	50.32	**64.33**	+14.01
LongCat 2.0 Claude Code	43.26	**55.13**	+11.87
DeepSeek V4 Pro Claude Code	34.73	**54.29**	+19.56

To isolate the design harness contribution, we hold the model and coding agent fixed and compare each configuration before and after harness attachment in Table 2. Across the seven completed configurations, the harness improves the PosterBench Score by 5.01–19.56 points. Under the native Codex–GPT-5.5 configuration, it raises the score from 75.87 to 81.46 (+5.59); with Claude Code and Kimi K2.7, it rises from 57.20 to 70.12 (+12.92). The largest gain is 19.56 points for DeepSeek V4 Pro with Claude Code. These improvements span multiple agent pairs. MLLMs have an additional repair signal unavailable to text-only LLMs: at each attempt, the rendered preview from the preceding attempt is supplied as visual context for the next repair. This lets the model inspect the artifact it is editing and localize layout, clipping, or visual-evidence failures that are not fully captured by textual diagnostics alone.

3.2.2 Cost–Performance Trade-off

Figure 7 compares seven AutoDesign configurations by PosterBench Score and a normalized designer-only API cost proxy on the fixed 10-paper PosterBench-mini subset. The observed Pareto frontier spans LongCat-2.0 (55.13 at $0.27/poster), Doubao Seed 2.1 Pro (71.83 at $2.75), Claude 4.8 (74.56 at $7.63), and GPT-5.5 (81.46 at $10.02). Doubao reaches 88% of GPT-5.5's leading score at 27% of its cost.

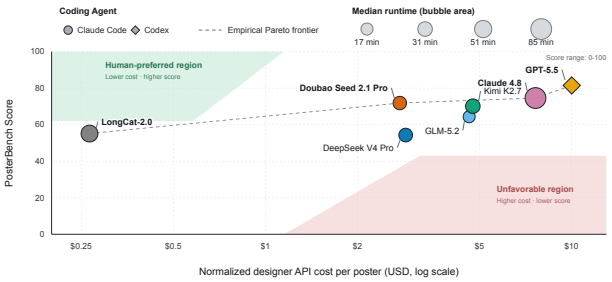


Figure 7: Cost versus score. Area shows median runtime; the dashed line marks the empirical Pareto frontier.

4 Related Work

Academic poster generation combines source understanding, content selection, and visual organization. Paper2Poster uses a parser–planner–painter pipeline with visual feedback, while P2P couples specialized agents with dedicated checkers (Pang et al., 2025; Sun et al., 2026). PosterForest jointly refines content and layout through a hierarchical Poster Tree (Choi et al., 2026); Any2Poster extends source inputs across modalities and domains (Vinaykumar et al., 2026). For editable artifacts, IGD generates multimodal layers, while DesignAsCode synthesizes HTML/CSS and revises code through visual reflection (Qu et al., 2025; Liu et al., 2026).

Feedback supports output refinement and reuse of task experience. Self-Refine iteratively critiques and revises responses without weight updates (Madaan et al., 2023). Reflexion stores verbal reflections, Voyager accumulates executable skills, and ExpeL extracts reusable insights from task experience (Shinn et al., 2023; Wang et al., 2023; Zhao et al., 2024). `AutoDesign` uses artifact repair as the inner loop and aggregates its traces to guide executable harness changes across tasks.

At the program level, TextGrad, DSPy, and GEPA optimize components and prompts in compound systems (Yuksekgonul et al., 2024; Khattab et al., 2024; Agrawal et al., 2026). STOP, GPTSwarm, ADAS, and AFlow search code- or graph-represented agentic systems (Zelikman et al., 2024; Zhuge et al., 2024; Hu et al., 2025; Zhang et al., 2025). Meta-Harness searches harness programs, while HarnessX evolves composable primitives from execution traces (Lee et al., 2026b; Chen et al., 2026). Source-rewriting agents likewise improve their code using execution evidence (Robeyns et al., 2025; Cai et al., 2026; Zhang et al., 2026b).

Update mechanisms also differ in their evaluation signals. Recursive Harness Self-Improvement uses pairwise revision feedback to specialize a prompt-level multi-agent harness (Lee et al., 2026a); Recursive Self-Evolving Agents gate persistent updates on a disjoint development split (Nguyen et al., 2026). AutoDesign applies harness optimization to faithful, editable academic artifacts. Its inner loop repairs outputs, its outer loop updates the harness with fixed model weights, and a frozen PosterBench protocol plus a system-blind human study assess the final system.

5 Conclusion

AutoDesign turns recurring design failures into improvements to the system that generates future multimodal outputs. Its **MetaHarnessOptimizer** aggregates trajectories, source and rendering diagnostics, evaluator feedback, and reference posters to update one **DesignHarness** component at a time while keeping model weights fixed. This makes paper-to-poster generation a persistent learning process that accumulates design priors and produces editable outputs for direct use or local revision on the Demo Page. We also introduce **PosterBench**, a unified evaluation protocol for academic posters. DesignHarness achieves the top score (78.32) and, with Claude Code and Claude 4.8 fixed, surpasses Claude Design by 7.45 points. It also receives the highest Bradley–Terry estimate in a system-blind human study.

AI USE STATEMENT

Generative AI is integral to the research method. The `MetaHarnessOptimizer` uses a coding agent to analyze execution trajectories and evaluation feedback, plan harness updates, and modify the `DesignHarness` codebase while keeping model weights fixed (Section 2). A coding agent also constructs the optimization-time evaluator from human-annotated reference artifacts (Section 2.3). Generative models produce the design artifacts, and VLMs contribute to their evaluation (Section 3 and Appendix A.7).

Generative AI also supported discussions of research ideas and experimental design. It was not used for theoretical proofs, data cleaning, translation, or interpretation of experimental results.

Verification was based on generated artifacts: each harness-evolution iteration produced actual posters, and candidate code updates were assessed by the training/development acceptance gate. We evaluated generated HTML/poster outputs using our manually constructed benchmark and system-blind human judgments, as detailed in Appendix A.8.

Generative AI additionally assisted with literature search and citation checking, drafting and revising manuscript text, summarizing method details, and LaTeX template migration and layout review. During manuscript revision, the authors provided iterative feedback on text and figure/table layout. The authors are responsible for the final text, claims, code, and artifacts, including material produced with generative AI assistance.

ETHICS STATEMENT

This work studies automated scientific communication. Generated posters may omit evidence, distort source claims, or present inaccurate content convincingly. Source provenance, rule-based checks, and visual critique help identify such failures, but outputs still require checking against the source paper before academic use. Our human evaluation involved 11 volunteer reviewers comparing posters with system and model identities withheld; the protocol and response options, including ties and skips, are documented in Appendix A.8. Preference estimates describe these reviewers and evaluated systems and should not be treated as a universal standard of design quality.

REPRODUCIBILITY STATEMENT

Section 2 specifies the optimization object, update procedure, and acceptance gate. Appendices A.3 and A.4 provide detailed method and implementation descriptions. Appendix A.1 documents the source packages, controlled factors, runtime versions, and reasoning settings; Appendix A.2 gives complete configurations and scores. The shared generation instruction is in Appendix A.5. The frozen evaluation rubric, aggregation rules, and per-case record schema are in Appendix A.7; the human-comparison protocol, ranking model, uncertainty procedure, and decision schema are in Appendix A.8. These specifications make the reported comparisons and scoring procedure explicit.

REFERENCES

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RQm2KQTM5r.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952. doi: 10.1093/biomet/39.3-4.324.

Qianshu Cai, Yonggang Zhang, Xianzhang Jia, Wei Xue, Jun Song, Xinmei Tian, and Yike Guo. MOSS: Self-evolution through source-level rewriting in autonomous agent systems. *arXiv preprint arXiv:2605.22794*, 2026. URL https://arxiv.org/abs/2605.22794.

Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*, 2024. URL https://arxiv.org/abs/2405.09818.

Tingyang Chen, Shuo Lu, Kang Zhao, Weicheng Meng, Hanlin Teng, Tianhao Li, Chao Li, Xule Liu, Jian Liang, Zhizhong Zhang, Yuan Xie, Heng Qu, Kun Shao, and Jian Luan. HarnessX: A composable, adaptive, and evolvable agent harness foundry. *arXiv preprint arXiv:2606.14249*, 2026. URL https://arxiv.org/abs/2606.14249.

Yuhang Chen, Tianpeng Lv, Siyi Zhang, Yixiang Yin, Yao Wan, Philip S. Yu, and Dongping Chen. Paper2Web: Let's make your paper alive! *arXiv preprint arXiv:2510.15842*, 2025. URL https://arxiv.org/abs/2510.15842.

Jiho Choi, Seojeong Park, Seongjong Song, and Hyunjung Shim. PosterForest: Hierarchical multi-agent collaboration for scientific poster generation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 379–401. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.15. URL https://aclanthology.org/2026.acl-long.15/.

Tsu-Jui Fu, William Yang Wang, Daniel McDuff, and Yale Song. DOC2PPT: Automatic presentation slides generation from scientific documents. In *AAAI*, 2022. URL https://arxiv.org/abs/2101.11796.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=t9U3LW7JVX.

Vijay Jaisankar, Sambaran Bandyopadhyay, Kalp Vyas, Varre Suman Chaitanya, and Shwetha Somasundaram. Deep submodular optimization and LLM for multimodal content extraction and automatic poster generation from long document. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 24221–24229, 2025. doi: 10.1609/aaai.v39i23.34598. URL https://ojs.aaai.org/index.php/AAAI/article/view/34598.

Seth Karten, Joel Zhang, Tersoo Upaa, Ruirong Feng, Wenzhe Li, Chengshuai Shi, Chi Jin, and Kiran Vodrahalli. Continual harness: Online adaptation for self-improving foundation agents. *arXiv preprint arXiv:2605.09998*, 2026. URL https://arxiv.org/abs/2605.09998.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=sY5N0zY5Od.

Jaeik Kim, Woojin Kim, Jihwan Hong, Yejoon Lee, Sieun Hyeon, Mintaek Lim, Yunseok Han, Dogeun Kim, Hoeun Lee, Hyunggeun Kim, and Jaeyoung Do. Dynin-Omni: Omnimodal unified large diffusion language model. *arXiv preprint arXiv:2604.00007*, 2026. URL https://arxiv.org/abs/2604.00007.

Hyunin Lee, Jinglue Xu, Jeffrey Seely, Donghyun Lee, Matei Zaharia, and Yujin Tang. Recursive harness self-improvement. *arXiv preprint arXiv:2607.15524*, 2026a. URL https://arxiv.org/abs/2607.15524.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026b.

Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, and Yu-Gang Jiang. Agentic harness engineering: Observability-driven automatic evolution of coding-agent harnesses. *arXiv preprint arXiv:2604.25850*, 2026. URL https://arxiv.org/abs/2604.25850.

Ziyuan Liu, Shizhao Sun, Danqing Huang, Yingdong Shi, Meisheng Zhang, Ji Li, Jingsong Yu, and Jiang Bian. DesignAsCode: Bridging structural editability and visual fidelity in graphic design generation. *arXiv preprint arXiv:2602.17690*, 2026. URL https://arxiv.org/abs/2602.17690.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Meituan LongCat Team, Bin Xiao, Chao Wang, Chengjiang Li, Chi Zhang, Chong Peng, Hang Yu, Hao Yang, Haonan Yan, Haoze Sun, et al. LongCat-Next: Lexicalizing modalities as discrete tokens. *arXiv preprint arXiv:2603.27538*, 2026. URL https://arxiv.org/abs/2603.27538.

Michael Nguyen, Quoc Nguyen, and Paul Vuong. Recursive self-evolving agents via held-out selection. *arXiv preprint arXiv:2606.28374*, 2026. URL https://arxiv.org/abs/2606.28374.

Wei Pang, Kevin Qinghong Lin, Xiangru Jian, Xi He, and Philip H. S. Torr. Paper2Poster: Towards multimodal poster automation from scientific papers. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/17337b1d5eeac8b59c80e025a552fa7a-Abstract-Datasets_and_Benchmarks_Track.html.

Yadong Qu, Shancheng Fang, Yuxin Wang, Xiaorui Wang, Zhineng Chen, Hongtao Xie, and Yongdong Zhang. IGD: Instructional graphic design with multimodal layer generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 18218–18228, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Qu_IGD_Instructional_Graphic_Design_with_Multimodal_Layer_Generation_ICCV_2025_paper.html.

Zhe Ren, Yimeng Chen, Dandan Guo, Guowei Rong, Tonghui Li, R. B. Xiong, Qingfeng Lan, Wenyi Wang, Nanbo Li, Yibo Yang, Mingchen Zhuge, and Jürgen Schmidhuber. Self-improvements in modern agentic systems: A survey. *arXiv preprint arXiv:2607.13104*, 2026. URL https://arxiv.org/abs/2607.13104.

Maxime Robeyns, Martin Szummer, and Laurence Aitchison. A self-improving coding agent. *arXiv preprint arXiv:2504.15228*, 2025. URL https://arxiv.org/abs/2504.15228.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Tao Sun, Enhao Pan, Zhengkai Yang, Kaixin Sui, Jiajun Shi, Xianfu Cheng, Tongliang Li, Wenhao Huang, Ge Zhang, Jian Yang, and Zhoujun Li. P2P: Automated paper-to-poster generation and fine-grained benchmark. In *International Conference on Learning Representations*, 2026. URL https://iclr.cc/virtual/2026/poster/10010167.

Amogh Vinaykumar, Aiden Li, Suozhi Huang, and Shilong Liu. Any2Poster: Any-source poster generation across modalities and domains. *arXiv preprint arXiv:2606.02915*, 2026. URL https://arxiv.org/abs/2606.02915.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Hao Wang, Shohei Tanaka, and Yoshitaka Ushiku. SciPostLayout: A dataset for layout analysis and layout generation of scientific posters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 8136–8141, 2024. URL https://openaccess.thecvf.com/content/CVPR2024W/GDUG/html/Wang_SciPostLayout_A_Dataset_for_Layout_Analysis_and_Layout_Generation_of_CVPRW_2024_paper.html.

Shengqiong Wu, Hao Fei, Leigang Qu, Wei Ji, and Tat-Seng Chua. NExT-GPT: Any-to-any multimodal LLM. *arXiv preprint arXiv:2309.05519*, 2023. URL https://arxiv.org/abs/2309.05519.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. TextGrad: Automatic "differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024. URL https://arxiv.org/abs/2406.07496.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-taught optimizer (STOP): Recursively self-improving code generation. In *First Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2310.02304.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-Harness: Harnesses that improve themselves. *arXiv preprint arXiv:2606.09498*, 2026a.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel machine: Open-ended evolution of self-improving agents. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://arxiv.org/abs/2505.22954.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=z5uVAKwmjf.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024. URL https://arxiv.org/abs/2308.10144.

Hao Zheng, Xinyan Guan, Hao Kong, Wenkai Zhang, Jia Zheng, Weixiang Zhou, Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. PPTAgent: Generating and evaluating presentations beyond text-to-slides. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 14402–14418. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.728. URL https://aclanthology.org/2025.emnlp-main.728/.

Zeyu Zhu, Kevin Qinghong Lin, and Mike Zheng Shou. Paper2Video: Automatic video generation from scientific papers. *arXiv preprint arXiv:2510.05096*, 2025. URL https://arxiv.org/abs/2510.05096.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTswarm: Language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62743–62767, 2024. URL https://proceedings.mlr.press/v235/zhuge24a.html.

Mingchen Zhuge, Changsheng Zhao, Dylan R. Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, Yangyang Shi, Vikas Chandra, and Jürgen Schmidhuber. Agent-as-a-judge: Evaluate agents with agents. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 80569–80611, 2025. URL https://proceedings.mlr.press/v267/zhuge25a.html.

A SUPPLEMENTARY EXPERIMENTAL MATERIALS

This appendix documents the complete method and implementation, controlled comparisons, shared generation instruction, frozen scoring protocol, released records, and matched visual evidence that complement the main paper.

Appendix guide

Jump to	Contents
A.1 Benchmark Inputs and Comparison Matrix	Source packages, the factors held fixed in each comparison, and the factor varied by every reported track.
A.2 Full Benchmark Results	Complete configurations, dimension scores, and analyses of the mini leaderboard and controlled tracks.
A.3 Complete Method Details	Full harness definitions, learning-loop details, pseudocode, and human-guidance channels.
A.4 The Optimized DesignHarness	Complete source-ingestion, artifact-revision, validation, and finalization implementation.
A.5 AutoDesign Generation Interface	The system-instruction excerpt and the shared task prompt used across compared systems.
A.6 DesignHarness Evolution	The staged capabilities accumulated in the optimized design harness and their mapping to the five-component abstraction.
A.7 PosterBench Evaluation Interface	Benchmark definition, scoring equations, evaluation diagram, operational rubric, gates, and per-case record.
A.8 Released Records and Human Evaluation	The benchmark archive schema, blind-review interface, human-preference results, and benchmark–human alignment.
A.9 Qualitative Trajectory and Comparisons	The local diagnostic and repair sequence of one poster generation run.
Matched Qualitative Comparisons	Curated three-system poster comparisons on identical source papers.
Additional Poster Demonstrations	The self-paper run record and four additional source-grounded, editable paper posters.
B Future Directions	Cross-medium pilot artifacts and directions for multimodal design and harness adaptation.

A.1 Benchmark Inputs and Comparison Matrix

Each case provides the source PDF and available paper assets. All systems in a matched row receive the same package and produce one editable poster artifact; the source paper remains the authority for claims, numbers, and visual evidence. Table 3 identifies the controlled factor for every reported track.

Table 3: Controlled comparison matrix. Main tracks compare complete systems; controlled tracks vary only the factor named in the final column.

Track	Papers	Fixed across a row comparison	Varied factor
`PosterBench` Main Track	100	Source paper, source assets, output contract, and frozen `PosterBench` protocol	System configuration
`PosterBench-mini` Main Track	10	Shared `PosterBench-mini` papers, output contract, and frozen `PosterBench` protocol	System configuration
Design Harness Track	10	Claude Code and Claude 4.8; shared `PosterBench-mini` papers and frozen `PosterBench` protocol	Design harness
Coding Harness Track	10	`AutoDesign`, GLM 5.2, shared `PosterBench-mini` papers, and frozen `PosterBench` protocol	Coding harness
Model Track	10	`AutoDesign`, Claude Code, shared `PosterBench-mini` papers, and frozen `PosterBench` protocol	Model
Harness-Attachment Ablation	10	Model, code agent, shared `PosterBench-mini` papers, and frozen `PosterBench` protocol	Presence of the `AutoDesign` design harness

Runtime Versions and Reasoning Configuration. Under our controlled configurations, we used `codex-cli` v0.142.3 for Codex and Claude Code v2.1.119 for Claude Code. Each model used its highest available thinking-effort setting.

A.2 Full Benchmark Results

The complete configurations and dimension scores for the 100-paper Main Track are in Table 1 in the main text. This appendix reports the shared mini-subset leaderboard and controlled comparisons in Tables 4 and 5.

A.2.1 PosterBench-mini Main Track

Table 4 reports results on `PosterBench-mini`, the original 10-paper subset used for the controlled ablations.

On `PosterBench-mini`, `AutoDesign` reaches 81.46 with Codex, compared with 75.87 for the native Codex baseline, and 74.56 with Claude Code, compared with 69.55 for the corresponding standalone baseline.

Table 4: `PosterBench` Score and dimension scores for the `PosterBench-mini` Main Track on the original 10-paper subset.

System	Design Harness	Coding Agent	Model	Score	Faith.	Cover.	Density	Vis. Ev.	Layout	Read.	Aesth.
Design Agent											
AutoDesign	**DesignHarness**	Codex	GPT 5.5	**81.46**	9.88	9.90	8.64	6.20	8.01	7.96	7.45
AutoDesign	**DesignHarness**	Claude Code	Claude 4.8	**74.56**	9.28	8.80	8.15	5.00	9.00	7.86	6.75
OpenDesign	OpenDesign	Claude Code	Claude 4.8	70.36	9.10	9.00	7.39	6.15	7.13	6.38	7.31
Claude Design	Claude Design	Claude Code	Claude 4.8	66.83	8.95	10.00	5.24	7.90	8.54	6.09	6.77
OpenDesign	OpenDesign	Codex	GPT 5.5	60.58	9.03	9.20	6.13	5.45	7.68	5.75	5.75
Coding Agent											
Codex	—	Codex	GPT 5.5	75.87	9.47	9.40	7.88	7.55	8.44	6.00	7.00
Claude Code	—	Claude Code	Claude 4.8	69.55	9.19	10.00	5.21	7.50	9.76	6.90	6.40
Kimi	—	Claude Code	Kimi K2.7	57.20	8.20	8.20	4.91	6.40	7.91	5.60	5.95
Doubao	—	Claude Code	Seed 2.1	54.01	7.99	8.20	6.30	5.55	6.01	5.52	6.15
GLM	—	Claude Code	GLM5.2	50.32	8.83	7.80	3.79	3.85	8.21	5.11	5.45
DeepSeek	—	Claude Code	DeepSeekV4-Pro	34.73	8.08	7.90	2.94	3.33	5.98	3.71	5.05
Human-Crafted Workflow											
PosterGen	—	—	Claude 4.8	51.82	8.95	9.33	3.39	6.67	7.67	4.96	5.50
Any2Poster	—	—	Claude 4.8	46.88	8.09	4.90	3.69	3.65	9.55	4.40	3.20
Paper2Poster	—	—	Claude 4.8	42.06	6.16	2.10	8.25	2.30	2.76	4.82	1.80

A.2.2 CONTROLLED TRACKS

Design Harness Track. Table 5(a) isolates the design-harness contribution by holding Claude Code and Claude 4.8 fixed. `AutoDesign` reaches 74.56, while Claude Design and OpenDesign score 66.83 and 70.36, respectively, under the same configuration.

Coding Harness Track. Table 5(b) fixes `AutoDesign` and GLM 5.2, varying only the coding harness. Kimi Code achieves the highest score at 82.31, followed by ZCode at 69.53.

Model Track. Table 5(c) fixes `AutoDesign` and Claude Code to separate model choice from harness variation. Claude 4.8 achieves the highest score at 74.56, followed by Seed 2.1 Pro at 71.83 and Kimi K2.7 at 70.12.

Table 5: Controlled track analysis on `PosterBench-mini`, the fixed 10-paper subset. (a) varies the design harness while holding the coding harness and model fixed. (b) varies the coding harness while holding `AutoDesign` and GLM 5.2 fixed. (c) varies the model while holding `AutoDesign` and Claude Code fixed.

Variant	Overall ↑	Faith.	Cover.	Density	Vis. Ev.	Layout	Read.	Aesth.
(a) Design Harness Track								
Coding Harness: Claude Code | *Model: Claude 4.8*								
AutoDesign	**74.56**	9.28	8.80	8.15	5.00	9.00	7.86	6.75
OpenDesign	70.36	9.10	9.00	7.39	6.15	7.13	6.38	7.31
Claude Design	66.83	8.95	10.00	5.24	7.90	8.54	6.09	6.77
(b) Coding Harness Track								
Design Harness: `AutoDesign` | *Model: GLM 5.2*								
Kimi Code	**82.31**	9.29	9.60	7.95	6.75	8.75	8.46	7.68
ZCode	69.53	9.42	8.90	7.04	5.10	7.23	6.93	6.77
OpenCode	67.87	9.32	8.20	7.28	6.90	7.84	7.21	6.47
Claude Code	64.33	8.81	7.30	7.38	4.50	7.74	6.30	5.68
(c) Model Track								
Design Harness: `AutoDesign` | *Coding Harness: Claude Code*								
Claude 4.8	**74.56**	9.28	8.80	8.15	5.00	9.00	7.86	6.75
Seed 2.1 Pro	71.83	9.06	9.00	6.53	6.55	9.16	7.11	6.85
Kimi K2.7	70.12	8.54	7.00	8.30	6.27	7.81	6.82	6.00
GLM 5.2	64.33	8.81	7.30	7.38	4.50	7.74	6.30	5.68
LongCat 2.0	55.13	9.11	8.10	5.24	5.25	7.56	5.81	5.65
DeepSeek V4 Pro	54.29	9.47	9.60	4.32	5.50	8.38	5.62	6.00

A.3 COMPLETE META-HARNESS DEFINITIONS AND LEARNING LOOP

A.3.1 DESIGN HARNESS DEFINITION

Following recent work that treats harnesses as an optimization target distinct from model weights (Lee et al., 2026b; Ren et al., 2026), we define a *design harness* H as the system surrounding a fixed model that turns a multimodal source (*e.g.*, an academic paper or report) into a human-facing artifact (*e.g.*, a presentation slide, poster, video, or web page):

$$y \sim H(\pi_\theta, x, c), \tag{5}$$

where π_θ is an LLM or MLLM, x is the multimodal input, and c specifies the context including the target medium and user constraints. The harness produces an artifact y through an execution trajectory τ, which records the sequence of intermediate actions, states, and revisions leading to the final output.

To enable systematic meta-harness optimization and facilitate credit assignment, we decompose the design harness H into five functional components:

- **Context and Memory**: source management, prompts, skills, reusable assets and persistent state.
- **Tools and Specifications**: tools and editable artifact specifications for layout, typography, and provenance.
- **Execution Runtime**: the workspace and runtime for authoring, rendering, validating, and exporting artifacts.
- **Orchestration**: task routing, attempt budgets, loop control, candidate selection, fallback, and finalization.
- **Evaluation and Feedback**: rule-based validation, model-based critique, and localized feedback for revision.

This decomposition specifies only the high-level abstraction of the design harness. The concrete implementation of each component is instantiated and iteratively improved by the meta-harness.

A.3.2 Meta-Harness Definition

We define a *meta-harness* as a system that operates on the design harness (Lee et al., 2026b; Zhang et al., 2026a; Lin et al., 2026). Given a user specification q and an optional initial design harness H_0, the meta-harness instantiates or iteratively improves the concrete implementation of the design harness, yielding an optimized harness H_T. Unlike a design harness, which transforms multimodal sources into human-facing artifacts, the meta-harness transforms harness requirements and implementations into an improved design harness.

The optimization target is the expected quality of the artifacts produced by the design harness. Let p_{task} denote the task distribution. We define the performance of a design harness H as

$$J(H) = \mathbb{E}_{(x,c)\sim p_{\text{task}},\, y\sim H(\pi_\theta,x,c)} \left[R_{\text{meta}}(y, x, c) \right], \tag{6}$$

where $R_{\text{meta}}(y, x, c)$ denotes the evaluator used by the meta-harness to assess the quality of artifact y with respect to its source x and design context c. Accordingly, the meta-harness optimization objective is

$$H^\star = \arg\max_{H} J(H). \tag{7}$$

Throughout this process, the parameters θ of the underlying model π_θ in the harness remain fixed. The optimization therefore acts on the system surrounding the model rather than on the model itself, consistent with the model-versus-scaffold distinction in recent self-improving-agent taxonomies (Ren et al., 2026).

A.3.3 Learning Loop

`AutoDesign` organizes artifact generation (*i.e.*, design harness) and harness optimization (*i.e.*, meta-harness) as two nested feedback loops, as illustrated in Figure 4. For each design task (x, c), the design harness executes an *inner loop* that repeatedly generates and revises the current artifact. This loop operates on the artifact y under a fixed design harness H and records the resulting execution trajectory τ. Across tasks, the meta-harness executes an *outer loop* that analyzes these trajectories and their evaluation results to identify recurrent failures and update the design harness H. Thus, the inner loop improves a single artifact without changing H, whereas the outer loop improves H based on evidence collected from multiple generation runs.

Inner loop. We initialize the design harness with a minimal inner-loop scaffold consisting of two abstract modules: a *designer* M_{design} and a *critic* M_{critic}. As shown in Figure 4, the designer generates and revises the artifact, while the critic evaluates its current state and provides feedback for the next revision. At refinement step k, their interaction is defined as

$$\begin{aligned} y_k &= M_{\text{design}}(y_{k-1}, f_{k-1}; x, c), \\ f_k &= M_{\text{critic}}(y_k; x, c), \end{aligned} \tag{8}$$

where y_k is the artifact at step k and f_k is the corresponding feedback, with y_0 and f_0 left empty so that the first step produces an initial draft from (x, c) alone. Repeated evaluation and revision produce an execution trajectory τ under the current design harness.

This formulation specifies only the roles of the two modules and the information flow between them. It defines the basic structure of the initial design harness while leaving its concrete realization open. During outer-loop optimization, the meta-harness may refine the two modules and their interaction, including their prompts, tools, feedback mechanisms, and loop-control policies, based on observed task performance.

Outer loop. The outer loop improves the design harness across tasks, as illustrated in Figure 3. Each iteration proceeds in four stages: rollout, evaluation, update proposal, and acceptance.

Rollout. At outer-loop iteration t, the current design harness H_t is executed on a training task set $\mathcal{D}_{\text{train}} = \{(x_i, c_i)\}_{i=1}^{N_{\text{train}}}$, where x_i is a multimodal source and c_i specifies the target medium and the corresponding design requirements. Each execution produces an artifact y_t^i and a corresponding trajectory τ_t^i. We denote the collection of trajectories by $\boldsymbol{\tau}_t = \{\tau_t^i\}_{i=1}^{N_{\text{train}}}$.

Evaluation. Before outer-loop optimization, we provide an evaluator coding agent with reference artifacts annotated by humans along seven quality dimensions: (i) Faithfulness, (ii) Coverage, (iii) Density, (iv) Visual Evidence, (v) Layout, (vi) Readability, and (vii) Aesthetics. The agent uses these examples to implement the evaluator R_{meta}, combining rule-based checks for directly measurable properties with VLM-based judgments for perceptual properties such as aesthetics. Once constructed, R_{meta} remains fixed during autonomous optimization. The resulting evaluator is then used to assess every generated artifact, $s_t^i = R_{\text{meta}}(y_t^i, x_i, c_i)$, and the resulting scores are collected into a batch denoted by $\boldsymbol{s}_t$. This optimization-time evaluator is distinct from the frozen `PosterBench` protocol used for final system comparison; the latter is specified in Section A.7.

Update proposal. We denote the *meta-harness optimizer* by P. In addition to the current design harness and the evidence collected at the current iteration, the meta-harness maintains an optimization record $\mathcal{L}$, serving as persistent context across outer-loop iterations. At iteration t, the optimizer takes the current harness H_t, the collected trajectories $\boldsymbol{\tau}_t$, their evaluation scores $\boldsymbol{s}_t$, and the optimization record $\mathcal{L}$ as input, and produces a candidate updated harness H'_{t+1}:

$$H'_{t+1} = P\left(H_t, \boldsymbol{\tau}_t, \boldsymbol{s}_t, \mathcal{L}\right). \tag{9}$$

The prime indicates that H'_{t+1} is an update proposal, whose acceptance is determined by the subsequent gating stage.

We instantiate P as a coding agent that sequentially assumes the roles of a planner and a code editor, as illustrated in Figure 3. In the planner role, the agent analyzes the current trajectories and scores together with the optimization history in $\mathcal{L}$. It dispatches parallel subagents to inspect the trajectories and their scores, synthesizes their findings into structured evidence of recurrent failures, and formulates a harness update plan. The plan specifies the observed failure modes, the harness component to be modified, and the intended changes. In the code-editor role, the agent implements these changes in the current design harness H_t, yielding the candidate harness H'_{t+1}.

Each outer-loop iteration is restricted to exactly one of the five harness components defined in Section A.3.1. An update may span multiple files within the selected component, but it cannot modify another component in the same iteration. This restriction keeps credit assignment interpretable, as each gain or regression is attributable to a single coherent intervention rather than to several simultaneous changes.

Acceptance gate. Once an update has been proposed, the meta-harness determines whether it should replace the current harness through a separate acceptance gate. Let J_{train} and J_{dev} denote the objective in Equation (6) evaluated on the training set $\mathcal{D}_{\text{train}}$ and on an independent development set $\mathcal{D}_{\text{dev}}$, respectively. A candidate is accepted only when its performance on $\mathcal{D}_{\text{train}}$ improves and its performance on $\mathcal{D}_{\text{dev}}$ does not decline:

$$\text{Accept}(H'_{t+1}) \iff J_{\text{train}}(H'_{t+1}) > J_{\text{train}}(H_t) \ \wedge \ J_{\text{dev}}(H'_{t+1}) \geq J_{\text{dev}}(H_t). \tag{10}$$

If the condition holds, the meta-harness promotes H'_{t+1} to H_{t+1}; otherwise H_t is retained. Results on the development set are used exclusively by the acceptance gate and are never exposed to P when

constructing an update proposal. Therefore, $\mathcal{D}_{\text{dev}}$ serves as a guard against overfitting the harness to the training tasks.

After the acceptance decision, the meta-harness appends the completed iteration to the optimization record $\mathcal{L}$. For each iteration t, $\mathcal{L}$ stores the harness H_t, the trajectories and scores, the selected harness component, the update plan and the corresponding code changes, and the acceptance decision, with a repository checkpoint preserving the harness implementation at that iteration. Trajectories and scores from the development set are not included in the record.

The updated $\mathcal{L}$ is supplied to P as persistent context in the next iteration. When a candidate is rejected, the record allows the next iteration to propose a different update while retaining the evidence of what has already been tried. $\mathcal{L}$ thus supports comparison, reproducibility, and rollback across iterations. Note that the outer loop maintains a single active harness at each iteration and does not perform tree search over harness variants. The complete meta-harness optimization procedure is summarized in Algorithm 1.

Algorithm 1 `AutoDesign` meta-harness optimization

Require: Fixed model π_θ; initial harness H_0; evaluator R_{meta}
Require: Training and development tasks $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{dev}}$; iterations T
Evaluate H_0 on train/dev; retain training evidence and gate scores; initialize $\mathcal{L}$.
for $t = 0$ to $T - 1$ **do**
  P analyzes training evidence and $\mathcal{L}$; plan an update to one component.
  Implement the plan with the code editor to obtain candidate H'_{t+1}.
  Evaluate H'_{t+1} on train/dev with R_{meta}.
  if Equation (4) holds **then**
    Promote H'_{t+1} and retain its training evidence and development gate score.
  else
    Retain H_t, its training evidence, and its development gate score.
  end if
  Record checkpoint, training evidence, component, plan, patch, and decision in $\mathcal{L}$.
end for
return $H_T, \mathcal{L}$.

Human-in-the-Loop. `AutoDesign` also supports an optional human-in-the-loop mode, as illustrated in Figure 3. Human intervention can operate through two channels. First, at iteration t, a user may provide directional guidance g_t in natural language, which is supplied to the planner alongside the trajectories and evaluation scores, so that the update proposal becomes $H'_{t+1} = P(H_t, \boldsymbol{\tau}_t, \boldsymbol{s}_t, \mathcal{L}, g_t)$. We introduce this mechanism because the coding agent acting as P may converge prematurely to a locally satisfactory harness configuration, at which point outer-loop optimization stagnates. Guidance can inject task-specific heuristics or redirect the search toward alternative improvements.

Second, human guidance may be provided to the coding agent responsible for implementing the evaluator when visual inspection reveals a systematic artifact bias not captured by R_{meta}. Such evaluator revision requires explicit human input. Otherwise, R_{meta} remains fixed because the meta-harness receives no external signal with which to identify or correct evaluator bias.

In both cases, the human provides observations or high-level directions rather than directly editing the harness or evaluator implementation. When no guidance is provided, the outer loop operates autonomously as defined in Equation (9).

A.4 The Optimized DesignHarness

The meta-harness optimization described in Section 2 yields `DesignHarness`. The resulting implementation supports multiple output media, including academic posters, presentation slides, videos, and web pages. We characterize its resulting architecture by examining the final implementation obtained through meta-harness optimization. As summarized in Figure 6, our analysis identifies four main stages: source ingestion, iterative artifact generation and revision by a designer module, feedback from a rule-based validator and a VLM-based critic, and finalization that prepares the selected candidate for delivery.

A.4.1 PAPER INGESTION

The ingestion stage transforms the input source x and the design context c into a structured, provenance-aware context for subsequent generation and revision. It extracts the document metadata and section outline, identifies key passages supporting the main claims, and records figures and tables together with their source locations, reflecting the joint textual and visual evidence selection required in paper-to-poster generation (Jaisankar et al., 2025; Pang et al., 2025; Sun et al., 2026). These materials are then organized into a content brief and a medium-specific artifact plan, which specify the target output format, the claims to be conveyed, and the visual evidence supporting each of them. Every extracted element retains a reference to its location in x, so that source-derived statements and visual materials used in the artifact can be traced back to the source and checked during revision. The resulting context is constructed once, retained across all inner-loop refinement steps, and provided to the designer as source-grounded input.

A.4.2 ARTIFACT GENERATION AND REVISION

The designer module is implemented as a coding agent that generates or revises the artifact from the ingested source context using the tools and skills available in the harness. At refinement step k, it conditions on the current artifact y_{k-1}, the feedback f_{k-1}, and the ingested context to produce the next candidate y_k, thereby instantiating M_{design} in Equation (8). Consistent with layer- and code-based design generation, the artifact remains as editable HTML files throughout refinement (Qu et al., 2025; Liu et al., 2026), allowing revisions to be implemented as localized code edits without requiring regeneration of the entire output. For visual critique, it can be rendered or exported as a medium-specific preview, such as PNG, PPTX, or MP4.

A.4.3 VALIDATION AND FINALIZATION

At each refinement step k, the candidate artifact y_k produced by the designer is examined by the rule-based validator, which applies a set of deterministic blocking checks. At a high level, these checks determine whether the candidate satisfies the requirements for terminating refinement and cover issues such as unsafe or missing assets, broken provenance links between incorporated materials and their sources, severe overflow or overlap, and violations of the required typographic and layout constraints. If the candidate passes all of them, the inner loop terminates and the candidate proceeds directly to finalization. Otherwise, the validator returns localized diagnostics for the detected violations, together with the results of non-blocking checks on properties such as content coverage, information density, and numerical consistency with the source.

When a candidate fails the blocking checks, it is also rendered into a medium-specific preview and inspected by a critic VLM. The critic assesses rendered properties of the candidate, including compliance with the design context, layout, readability, and aesthetics. The two sources of feedback are consolidated into the repair signal f_k and passed to the designer for the next attempt. This feedback-to-revision pattern is related to recursive self-refinement and agent-as-a-judge approaches (Madaan et al., 2023; Zhuge et al., 2025). In the notation of Equation (8), rule-based validation and visual critique jointly instantiate M_{critic}.

The final implementation permits at most $K = 12$ refinement attempts. As soon as a candidate y_k passes all blocking checks, the attempt loop terminates and the candidate is passed to the finalization stage. This stage applies the remaining post-processing, such as final rendering adjustments, mathematical typesetting, and inlining of referenced assets, to produce a self-contained output. If the attempt budget is exhausted without any candidate passing all blocking checks, the harness uses the retained attempt history and applies a sequence of fallback mechanisms to identify a deliverable candidate while retaining essential safety and integrity constraints. The selected candidate is then passed to the same finalization stage. The sequence of candidate generation, validation, critique, and revision constitutes the execution trajectory τ in Section 2.2, while the design harness H remains fixed throughout this process.

A.5 AUTODESIGN GENERATION INTERFACE

The `AutoDesign` system excerpt is shown first. The shared user prompt below fixes the target artifact, source-grounding requirements, and visible quality constraints for every compared system. System-specific orchestration, tools, and repair policies remain part of each system configuration.

AutoDesign System Prompt (excerpt)

```
You are the AutoDesign designer.  Turn the supplied brief into
editable visual artifacts.  Keep final text native and editable.
Ground paper-derived claims in the supplied source; do not
invent numbers, authors, venues, URLs, benchmark deltas,
compute, or citations.  For an academic paper poster, use the
source figures and tables as primary visual evidence, author a
complete editable artifact, render it, and repair objective
layout or grounding defects before delivery.
```

Shared User Prompt (verbatim)

```
Generate a polished academic paper poster from the attached
   paper. The visible artifact must be the poster itself: a
   3072 x 1536 px / 2:1 landscape academic conference poster,
   not a landing page, dashboard, marketing graphic, or slide
   collage. Use browser-native authored HTML/CSS, CSS grid/flex
   /normal document flow, and native editable text.

Header requirements:
- Use a full-width top header band.
- The header is limited to exactly these three visible paper-
   identity lines: paper title, author list, and school/
   institution/company names.
- Place those fields as three compact centered text rows only:
   title line, authors line, school/institution/company line.
- The school/institution/company line should contain only
   organization names grounded in the paper, rendered as plain
   text only. Do not invent missing organizations.
- Do not add a fourth header/meta/subtitle row or side identity
    rail. Do not put any other visible content in the header:
   no logos, image badges, icons, QR codes, venue/year text,
   conference names, arXiv/archive labels, citation/contact
   text, project/code/resource links, topic badges, method
   slogans, contribution bullets, benchmark claims, source
   figures/tables, or explanatory captions. If any of those
   fields are available, leave them out of the header; users
   can add them after export.

Poster structure and density:
- Use three balanced vertical columns below the header.
- Use a compact editorial hierarchy: usually 7-10 numbered
   sections across the three columns, with 2-4 sections per
   column.
- Prefer section titles such as Motivation, Method, Data/
   Benchmark, Results, Ablation/Analysis, Limitations, and
   Takeaways. Keep titles literal and academic; do not use
   marketing headlines.
- Make the poster dense like a real conference poster, but
   avoid the AI-dashboard look: no repeated large cards, no
```

Shared User Prompt (continued)

```
    grid of generic metric tiles, no decorative badge rows, and
    no nested card-in-card panels.
- Do not add a horizontal metric/KPI/result band: a row of
    standalone big-number-plus-label tiles that highlights
    headline numbers (e.g. Inception Score, FID). State those
    numbers inside the running results prose or a distilled
    native table, never as a separate summary strip that
    duplicates figures already given elsewhere.
- Build visual evidence units in the DOM shape expected by
    validation. Each placed source figure/table/image must live
    in its own direct child of the section panel root: `<section
     class="source-flow-unit figure-flow-unit" data-source-id="
    ingest_fig_01" data-layer-id="ingest_fig_01"> ... </section
    >`.
- Inside that source-flow unit, keep the source crop and its
    explanation as direct siblings. Use a bound `<figure class="
    flow-asset source-asset float-right" data-source-id="..."
    data-layer-id="...">` with an `<img src="{{layer:...}}">`,
    plus direct sibling `p`, `ul`, `table`, or `div` readout
    content in the same source-flow unit.
- The `.source-flow-unit` / `.figure-flow-unit` element itself
    must be normal flow: `display: flow-root` or `display: block
    `, not grid or flex. Do not wrap source crops and readouts
    in `.media-row`, `.visual-shell`, `.source-grid`, `.evidence
    -grid`, or other grid/flex split wrappers. If side wrapping
    is useful, float the `.flow-asset` left or right with `shape
    -outside`; otherwise make the asset stacked/full-width with
    the readout directly below inside the same source-flow unit.
- Build visual evidence units deliberately. A paper figure/
    table crop may be stacked/full-width, or floated with a
    wrapped readout only when the local readout meaningfully
    fills the side flow. If the side flow would be empty, stack
    the crop and readout instead.
- Floated source-flow lists must reserve a marker gutter. If a
    direct sibling `ul`/`ol` wraps beside a floated source asset
    , style it with `display: flow-root`, `padding-inline-start:
     1.25em` or more, `list-style-position: outside`, and small
    `li` padding; do not use `padding: 0`, negative text indents
    , or absolutely positioned pseudo bullets. If the side lane
    is too narrow for the marker and readable wrapped text, use
    a stacked/full-width source-flow instead.
- Every section should earn its space with paper-specific
    content. If a section has obvious blank space, first add
    concise paper facts, benchmark context, mechanism notes,
    ablation caveats, compact comparison table rows, source-
    grounded bullets, or concise visual interpretation. If
    content still does not fit, shorten prose, rebalance section
     heights, split a dense section, or remove low-value text.
- Do not enlarge figures, tables, cards, or body text just to
    fill space. Do not shrink text below readable poster scale
    to force in extra content.

Evidence and content requirements:
- Use real figures, tables, charts, and diagrams from the paper
     whenever they support the story.
```

Shared User Prompt (continued)

```
- Paper source figures and paper source tables must appear as
    original PDF crops when used as source evidence. Do not
    replace a source paper table with a native reconstruction,
    selected-column subset, or re-rendered table image.
- Every source figure/table crop should appear near a concise
    local explanation of what it proves.
- Native authored tables are only for distilled poster
    summaries: compact benchmark readouts, method taxonomies,
    training-stage summaries, ablation summaries, or limitation
    evidence. They must summarize rather than duplicate a full
    paper source table, and must not degenerate into a
    horizontal strip of big-number metric tiles.
- Do not show the same paper table as both an original source
    crop and a full native reconstruction. If a source crop is
    placed, any nearby native table must be clearly smaller and
    more distilled.
- All claims, numbers, benchmarks, dataset details, method
    descriptions, limitations, and takeaways must be grounded in
     the attached paper.
- If a number or claim cannot be verified from the paper, omit
    it or rewrite it qualitatively.
- Results sections should include more than headline numbers.
    Add benchmark context: competing baselines, task families,
    secondary metrics, and the claim those results support.
- Training, method, and analysis sections should use paper-
    specific details: stages, datasets, losses, sequence lengths
    , modules, ablation results, assumptions, or future-work
    claims where available.
- Prefer concrete paper terms over generic filler. Avoid empty
    phrases such as "core thesis", "grounded poster
    interpretation", "forward path", "evidence target", "
    benchmark signal", "what it supports", or "key insight" as
    visible labels unless those exact terms are natural to the
    paper.

Visual and typography requirements:
- Use a formal academic poster visual system with a strong
    title header, clear section title bands, compact body text,
    thin dividers, native tables, charts, and disciplined
    spacing.
- Use an academic serif typography system such as Times New
    Roman / Times / Georgia for the poster text. Body, local
    readout, table-prose, caption, ordinary bullet, and table-
    cell parent text should stay at font-weight 400.
- Use short inline `<strong class="lead-key">...</strong>`
    emphasis as sentence-start lead phrases in motivation,
    method, results, analysis, limitations, takeaways, and
    source readouts, for example `<strong class="lead-key">
    Training signal:</strong> ...` or `<strong class="lead-key">
    Evidence:</strong> ...`. Keep each lead phrase 2-5 words
    when possible; one-word academic labels such as `Problem:`
    or `Risk:` are fine when natural.
- Avoid large blocks of bold body copy, bold whole bullets/
    readouts/table rows, scattered bold keywords, all-caps
    labels, negative letter spacing, and oversized display type
    inside dense sections. Use italics sparingly or not at all.
```

Shared User Prompt (continued)

```
- Prefer concise bullets for explanatory text when they improve
    scanability.
- Prefer real paper visuals over decorative imagery. Do not use
    generated or stock imagery as scientific evidence.
- Preserve readable source figures and source table crops.
   Avoid oversized white wrappers around images. Crop only
   obvious external white margins; never crop into axes,
   legends, labels, captions, diagrams, table rules, or
   meaningful visual content.
- Use restrained academic color: neutral paper background, high
   -contrast dark ink, one disciplined muted accent, and at
   most one subtle secondary accent. Use light section bands
   and thin rules rather than saturated fills.
- Avoid one-note blue/purple dashboard palettes, decorative
   gradients, orbs, stock-like backgrounds, heavy drop shadows,
    and repeated pill/badge decoration.

Native academic table requirements:
- Poster-native summary/readout tables should use booktabs-
   style academic treatment: no full boxed grid, no heavy outer
    frame, no vertical rules by default, no double rules, and
   no saturated dark header bars.
- Use a top rule, header/mid rule, optional light row
   separators, and a bottom rule. Headers should usually be
   white or lightly tinted with dark text.
- Left-align every `th` and `td`, including numeric values. Do
   not right-align or decimal-align numeric columns in ordinary
    poster-native tables.
- Use all-center alignment only for short pure symbol/numeric
   matrices with no prose cells, no method/dataset row labels,
   no sentence fragments, and uniformly short values.
- Keep native tables poster-readable: usually 3-7 body rows and
    3-6 columns. Move paragraph explanations into a nearby
   readout instead of stuffing long prose into cells.
- Use only one restrained emphasis mechanism per table: one
   focus row, one key column, or a small number of best cells.
   Avoid zebra striping unless the table is unusually long or
   wide.

Do not:
- Do not create a web landing page, dashboard, marketing
   graphic, promotional poster, or grid of large repeated cards
   .
- Do not show process labels such as "source-backed", "authored
    HTML", "paper poster", "no generated evidence imagery", "
   ingested", "retrieved", "grounded", "evidence pack", or "
   pipeline".
- Do not display source ids, layout notes, planning notes, or
   internal instructions.
- Do not bake final text into images; keep text native and
   editable.
- Do not invent logos, institutions, venues, links, benchmark
   numbers, dataset details, or paper limitations.

After rendering, visually inspect the full poster and iterate
   locally until it looks intentionally composed: no clipped
```

Shared User Prompt (continued)

```
figures, overlapped text, unreadable tables, padded-looking
sections, oversized wrappers, decorative filler, repeated
generic cards, or obvious blank panel bottoms.
```

A.6 DesignHarness Evolution

Figure 8 provides an implementation-level architecture summary for the paper-to-poster instantiation of `DesignHarness`. It is neither a second taxonomy nor a record of individual outer-loop iterations: its stages collectively instantiate the five functional components of H in Section 2.1. The illustration moves from a planning-and-critique loop to a coding-agent designer that authors editable HTML, while source grounding, specialist prompt and vision support, repair and validation gates, image-native evaluation, candidate promotion, and recovery operations accumulate around that core.

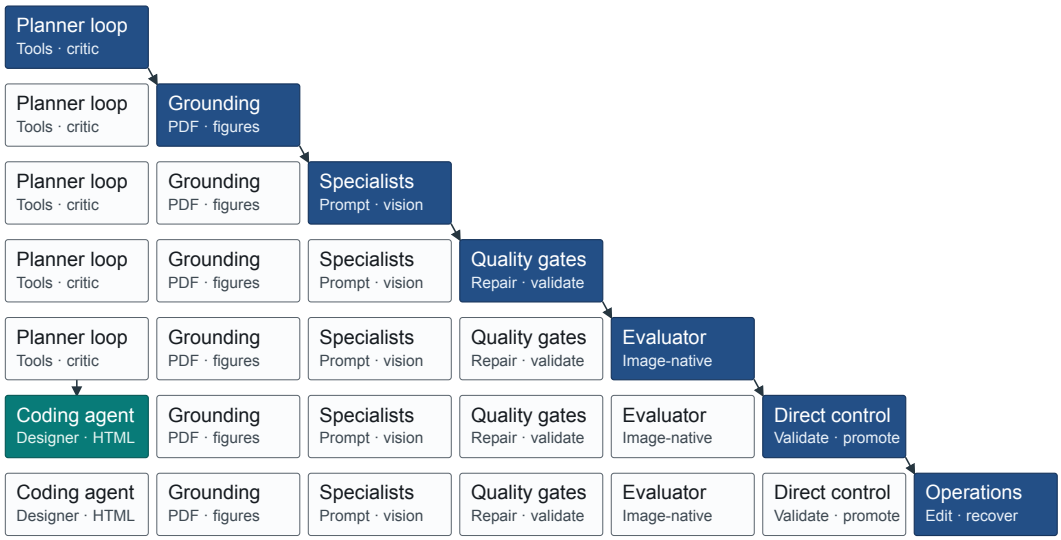


Figure 8: **DesignHarness architecture evolution for paper-to-poster.** An implementation-level summary of accumulated capabilities, not a second harness taxonomy or a chronological outer-loop trace. The diagonal path highlights additions in this architectural view; it does not denote individual meta-harness iterations. The final harness combines source grounding, specialist support, quality gates, image-native evaluation, controlled promotion, and operations for editable HTML artifacts.

The figure shows why the optimized result is more than a static prompt or a single repair loop: it accumulates a coherent system around the fixed model. In the five-component abstraction, grounding supplies context and memory; specialist support and editable HTML define tools and specifications; the workspace, browser, renderer, and export environment support the coding-agent authoring path as the execution runtime; direct control and operations implement orchestration; and quality gates together with image-native evaluation provide feedback for revision. The image-native evaluator shown here belongs to evaluation and feedback inside the design harness. It is distinct from the outer-loop evaluator R_{meta} and from the frozen `PosterBench` protocol used for final comparison (Sections A.7 and 2.3).

A.7 PosterBench Evaluation Interface

A.7.1 Benchmark and Scoring Protocol

`PosterBench` contains a 100-paper Main Track and `PosterBench-mini`, a shared 10-paper subset reported in Table 4. The papers span five disciplines: *AI/ML*, *biomedicine and health*, *climate and earth environment*, *economics and policy*, and *physics and astronomy*. Every system receives the

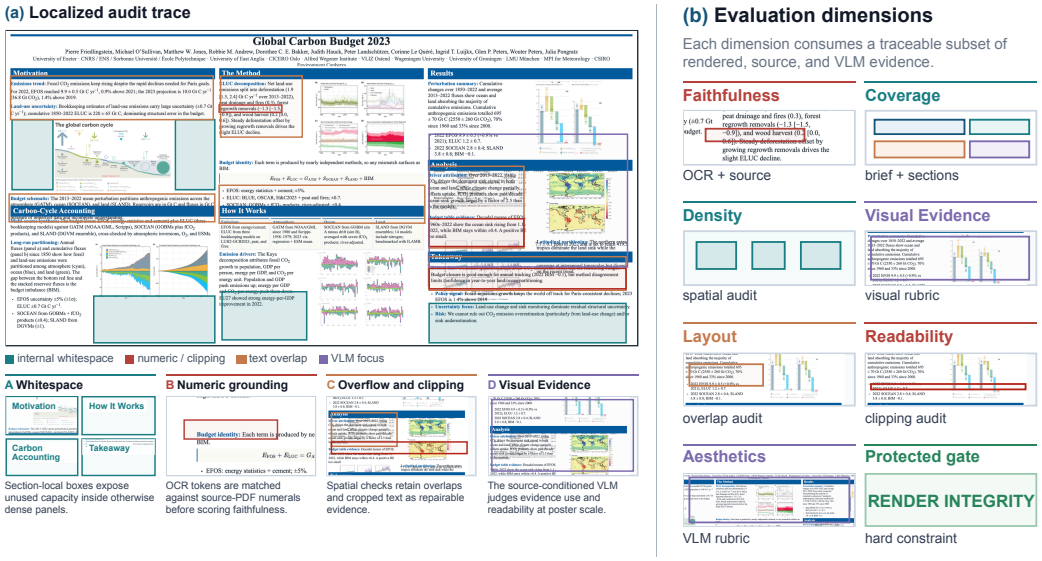


Figure 9: **PosterBench evaluation protocol.** `PosterBench` combines rule-based algorithms for traceable spatial, OCR, numeric-grounding, and render-integrity checks with rubric-guided VLM judges for source-grounded assessment of visual evidence, layout, readability, and aesthetics.

same source paper and associated source assets, and its output is rendered to a common poster format before scoring. The shared paper-to-poster generation interface and the exact fixed-versus-varied factors for every reported track are documented in Section A.5 and Table 3.

For paper p_i and candidate artifact A_i, the fixed `PosterBench` evaluator returns $\mathbf{q}_i \in [0, 10]^7$, ordered as Faithfulness, Coverage, Density, Visual Evidence, Layout, Readability, and Aesthetics. It forms the weighted rubric score

$$R_{\text{rubric}}(p_i, A_i) = \sum_{j=1}^{7} \alpha_j q_{i,j}/10, \qquad \boldsymbol{\alpha} = (10, 10, 15, 10, 20, 25, 10). \tag{11}$$

The final score applies the strictest active record-level ceiling before benchmark averaging:

$$\begin{aligned} R_{\text{poster}}(p_i, A_i) = \min\Big(&R_{\text{rubric}}(p_i, A_i), C_i^{\text{layout}}, \\ &C_i^{\text{viability}}, C_i^{\text{failure}}, C_i^{\text{gate}}\Big), \\ \text{Overall} = \frac{1}{N}\sum_{i=1}^{N} &R_{\text{poster}}(p_i, A_i). \end{aligned} \tag{12}$$

Here, C_i^{layout}, $C_i^{\text{viability}}$, C_i^{failure}, and C_i^{gate} bound severe layout damage, insufficient presentation viability, confirmed visible failures, and protected render-integrity violations, respectively; inactive ceilings are 100. A standard P0 gate caps a score at 40, and more severe gate types may set a lower cap. Thus, the reported metric columns are dimension means, while **Overall** is the mean of capped poster scores and cannot generally be recovered by reweighting those displayed means.

`PosterBench` is a frozen external evaluator, separate from the optimization-time evaluator R_{meta} used in the meta-harness outer loop (Section 2.3). R_{meta} supplies feedback for updating `DesignHarness`; `PosterBench` evaluates completed systems and is neither optimized nor modified by the outer loop. The operational rubric, protected gates, and per-case record schema are detailed below.

Figure 9 shows the evaluator's evidence path on a rendered poster. Programmatic spatial and OCR audits localize unused regions, text overflow or clipping, and source-inconsistent numeric claims. The source-conditioned VLM then assesses paper-grounded visual quality from the rendered artifact and paper brief. These dimension-level signals are aggregated with fixed weights and record-level ceilings.

Table 6: `PosterBench` scoring protocol. "Programmatic" denotes image-native and source-grounded checks; "VLM" denotes a dimension-specific judgment conditioned on the rendered poster and compact source context. The weights sum to 100.

Dimension	Wt.	Score mode	Operational definition
Faithfulness	10	Programmatic + VLM	Checks numeric and source grounding, then judges whether claims, entities, and visual evidence remain consistent with the paper.
Coverage	10	VLM	Assesses whether the poster preserves the paper's problem, method, evidence, and takeaway against a compact source brief.
Density	15	Programmatic	Measures information occupancy, OCR text coverage, blank interiors, and pasted paper-body screenshots.
Visual Evidence	10	Programmatic + VLM	Judges whether figures and tables are relevant, readable, and explained locally; guards reject raw paper-body crops.
Layout	20	Programmatic	Audits render size and aspect, OCR fallback, clipping, overlap, export-edge damage, and visible placeholders.
Readability	25	Programmatic + VLM	Combines poster-scale text and spatial checks with hierarchy, scan-path, balance, and crowding judgments.
Aesthetics	10	VLM	Rates academic visual craft, including typography, palette discipline, and compositional coherence.

A.7.2 Operational Rubric and Records

`PosterBench` evaluates a rendered artifact against its source paper with seven scores $q_j \in [0, 10]$. Its rubric, weights, and protected gates are manually specified and frozen before comparative evaluation. The seven dimensions use the same quality vocabulary as the outer-loop evaluator R_{meta} in Section 2.3, but serve a different role: `PosterBench` reports completed-system quality, whereas an evaluator coding agent constructs R_{meta} from annotated reference artifacts to provide update feedback during harness optimization. R_{meta} is then fixed within each optimization run.

For record i, `PosterBench` first forms the fixed weighted score from the seven dimension scores, applies the record-level ceiling C_i, and finally averages capped poster scores across the benchmark. Its ceiling families address severe layout damage, insufficient presentation viability, confirmed visible failures, and protected render-integrity gates; inactive ceiling families take value 100. A standard P0 gate has a ceiling of 40, while more severe gate types may impose a lower ceiling. Thus, if $\bar{\mathbf{q}} = N^{-1} \sum_i \mathbf{q}_i$ is the vector displayed by a table row, $\frac{1}{10}\boldsymbol{\alpha}^\top \bar{\mathbf{q}}$ does not generally equal **Overall**: the ceiling is applied before the benchmark average. For batches of at least 20 readable posters, a blinded style-homogeneity check may only reduce the professional-aesthetics score; it is not applied to `PosterBench-mini`, whose 10-poster scale is below that threshold.

The released benchmark rows score rendered poster images. Programmatic signals cover render integrity, occupancy, OCR readability, and source grounding. Each VLM judgment receives the rendered image, a compact paper brief, and selected grounding signals, but no system identity or generation prompt.

Per-case evaluation record (schema excerpt)

```
{
  "case_id": "2017-attention-is-all-you-need",
  "system": "AutoDesign + Claude Code + Claude 4.8",
  "overall_score": "<aggregate-score>",
  "dimension_scores": { "faithfulness": "<0-10>", ...},
  "evaluation_status": "scored"
}
```

Released records identify the system configuration, source case, evaluation status, aggregate score, and dimension scores, enabling each reported row to be audited and reaggregated.

A.8 Released Records and Human Evaluation

The released score archive records system identity, source case and discipline, evaluation status, aggregate score, and per-dimension scores. It distinguishes fresh evaluations from reaggregated records and supports independent auditing of the reported tables.

Released benchmark record fields	
Configuration	System, design harness, coding harness, and model.
Source case	Paper identifier and discipline.
Evaluation	Status, aggregate score, and seven dimension scores.

Human-evaluation record	
Included reviewer accounts	11
Submitted responses	936 (933 rankings; 3 skips)
Judgment design	System-blind pairwise comparison on a shared paper
Complete task roster	600 tasks per reviewer, covering all paper–system pairs
Ranking analysis	Bradley–Terry; ties contribute one half-win
Uncertainty	2,000 crossed paper–reviewer bootstrap resamples
Agreement diagnostic	Nominal Krippendorff coefficient: 0.101

A.8.1 Human Evaluation

We recruited 11 volunteer reviewers for a fully system-blind pairwise evaluation of `AutoDesign`, Claude Code, OpenDesign, and Claude Design on the 100 source papers of the `PosterBench` Main Track. For each comparison, reviewers saw only two posters generated from the same paper; no method, system, or model identity was disclosed. They selected the left poster, the right poster, approximate equality, or skip. We fit a Bradley–Terry model (Bradley & Terry, 1952),

$$\Pr(i \succ j) = \frac{\exp(\beta_i)}{\exp(\beta_i) + \exp(\beta_j)},$$

where β_i is the latent preference strength of system i. Ties contribute one half-win to each system and skips are excluded. We report the probability of beating a uniformly sampled alternative, with 95% intervals from 2,000 crossed bootstrap resamples of papers and reviewers. The complete task roster and decision schema are detailed below, together with the blind-review interface in Figure 12.

As shown in Figure 10, the volunteer reviewers submitted 936 responses: 933 ranking judgments and three skips. `AutoDesign` has the highest Bradley–Terry point estimate at 64.0% (95% interval: 55.2–77.8%). Its tie-adjusted empirical preference scores are 61.3% against Claude Code, 63.1% against OpenDesign, and 67.6% against Claude Design.

Benchmark–human alignment. The blind study also assesses whether `PosterBench` provides an informative comparison signal at the poster level. Figure 11(a) compares each poster's `PosterBench` Score with its tie-adjusted human preference. Each human score aggregates the blind pairwise judgments involving that poster. The two measurements have a positive, albeit imperfect, association ($r = 0.34$; a paper-cluster bootstrap gives a 95% interval of $[0.22, 0.44]$). This is a useful property of the protocol rather than a requirement that it duplicate human preference: `PosterBench` also evaluates Faithfulness, Coverage, Density, Visual Evidence, Layout, Readability, and Aesthetics, whereas each blind judgment asks for an immediate pairwise choice.

More importantly, the `PosterBench` Score margin calibrates how informative an automatic comparison is. Figure 11(b) uses 919 of the 933 non-skip judgments: the remaining 14 assign equal `PosterBench` Scores to both posters and therefore have no benchmark-preferred direction. The probability that the benchmark-preferred poster agrees with the human decision rises from 51.9% for 0–3-point gaps to 74.4% for gaps of at least 20 points. The error bars are 95% intervals from 2,000 crossed paper–reviewer bootstrap replicates, which resample both source papers and reviewers. Thus,

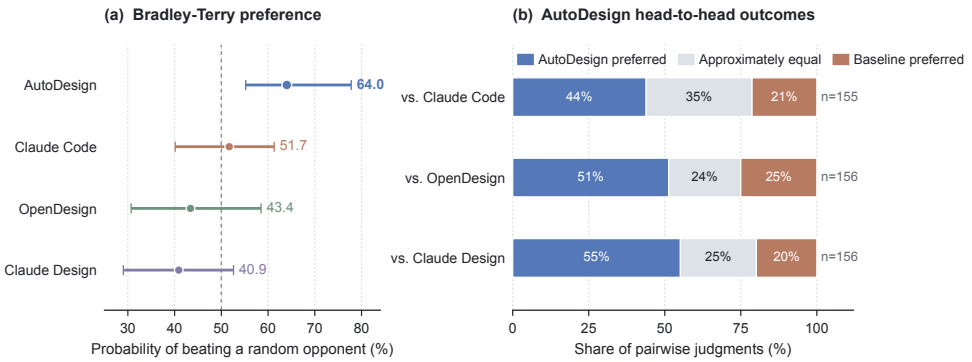


Figure 10: System-blind human evaluation. (a) Bradley–Terry probability of beating a uniformly sampled alternative, with 95% intervals crossed by paper–reviewer bootstrap replicates. (b) `AutoDesign` head-to-head outcomes against each baseline. Eleven volunteer reviewers submitted 936 responses, including 933 ranking judgments and three skips.

the benchmark offers more than a global system ranking: a large score gap identifies comparisons in which human preference is substantially more consistent.

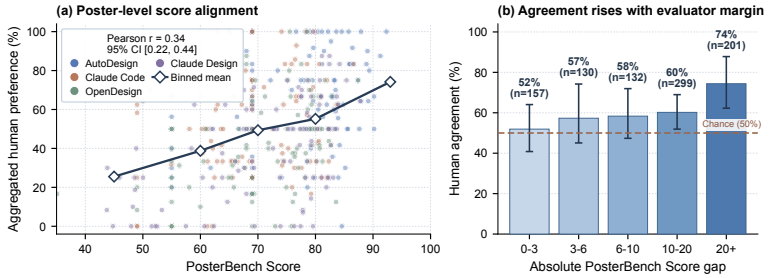


Figure 11: `PosterBench` Score alignment with system-blind human preference. (a) Poster-level score association. (b) Human agreement increases with the `PosterBench` Score margin.

Blind-review interface and complete task roster. Figure 12 shows the website used for the system-blind study. It presents two anonymous posters for the same source paper with a shared title, abstract, and PDF; system, model, and harness identities are withheld. The complete roster balances left–right presentation and records Poster A, about equal, Poster B, skip, and an optional critical-failure flag.

The roster visible to each reviewer contains every paper–system-pair task:

$$|\mathcal{T}| = N_{\text{papers}} \binom{N_{\text{systems}}}{2} = 100 \binom{4}{2} = 600.$$

It evaluates each of the six unordered system pairs on every paper, providing equal paper coverage and a connected graph for Bradley–Terry estimation. The "0/600" counter denotes this complete roster; submitted non-skip decisions are retained and uncompleted assignments are not imputed.

Human review decision record (schema excerpt)

```
{
  "paper_id": "2017-attention-is-all-you-need",
  "poster_a_id": "anonymous-artifact-a",
  "poster_b_id": "anonymous-artifact-b",
  "choice": "A | B | approximately-equal | skip",
  "severe_a": false, "severe_b": false, "skip_reason": null
}
```

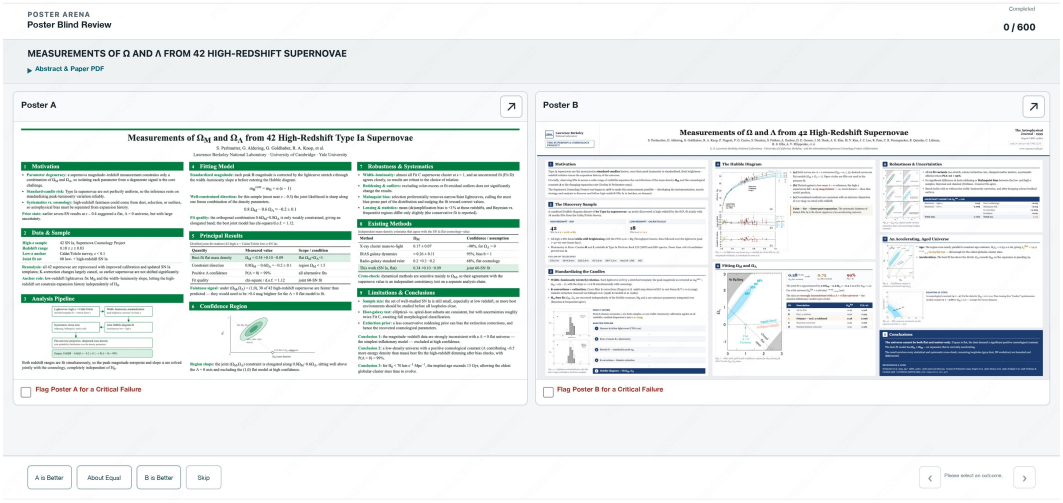


Figure 12: System-blind Poster Arena interface. Each task presents two anonymous posters for the same paper, with the abstract and PDF as common context. Reviewers select a preference, approximate equality, or skip, and can flag a critical failure. The "0/600" counter denotes the complete roster, rather than a requirement to submit 600 judgments.

The released decision record preserves each anonymous comparison and critical-failure flag. The coefficient is a nominal agreement diagnostic over paper–system-pair items, not the ranking estimator; Figure 11 reports a separate paper-cluster bootstrap for benchmark–human alignment.

A.9 Qualitative Trajectory and Comparisons

Figure 13 traces one poster run across five attempts. The critic first identifies a clipped analysis lane at A1 (0.36); the reallocation of the row removes the constraint at A3 (0.42), and later the refit of the header and the scaling of evidence produce a more balanced hierarchy at A5/A6 (0.62). A9 preserves the repaired composition and is accepted at 0.78. The trace illustrates the intended use of diagnostic feedback: edits stay localized to the failing region while valid layout and source-derived content are retained across revisions.

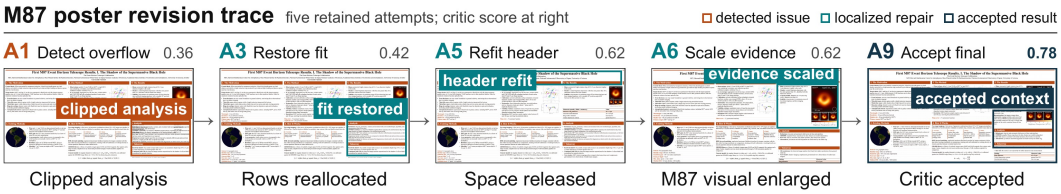


Figure 13: Qualitative trajectory of a poster generation run. Five selected attempts show the local change associated with each diagnostic or repair event. Colored boxes mark the implicated region: an analysis overflow at A1, its fit restoration at A3, header refitting at A5, evidence scaling at A6, and the critic-accepted final at A9.

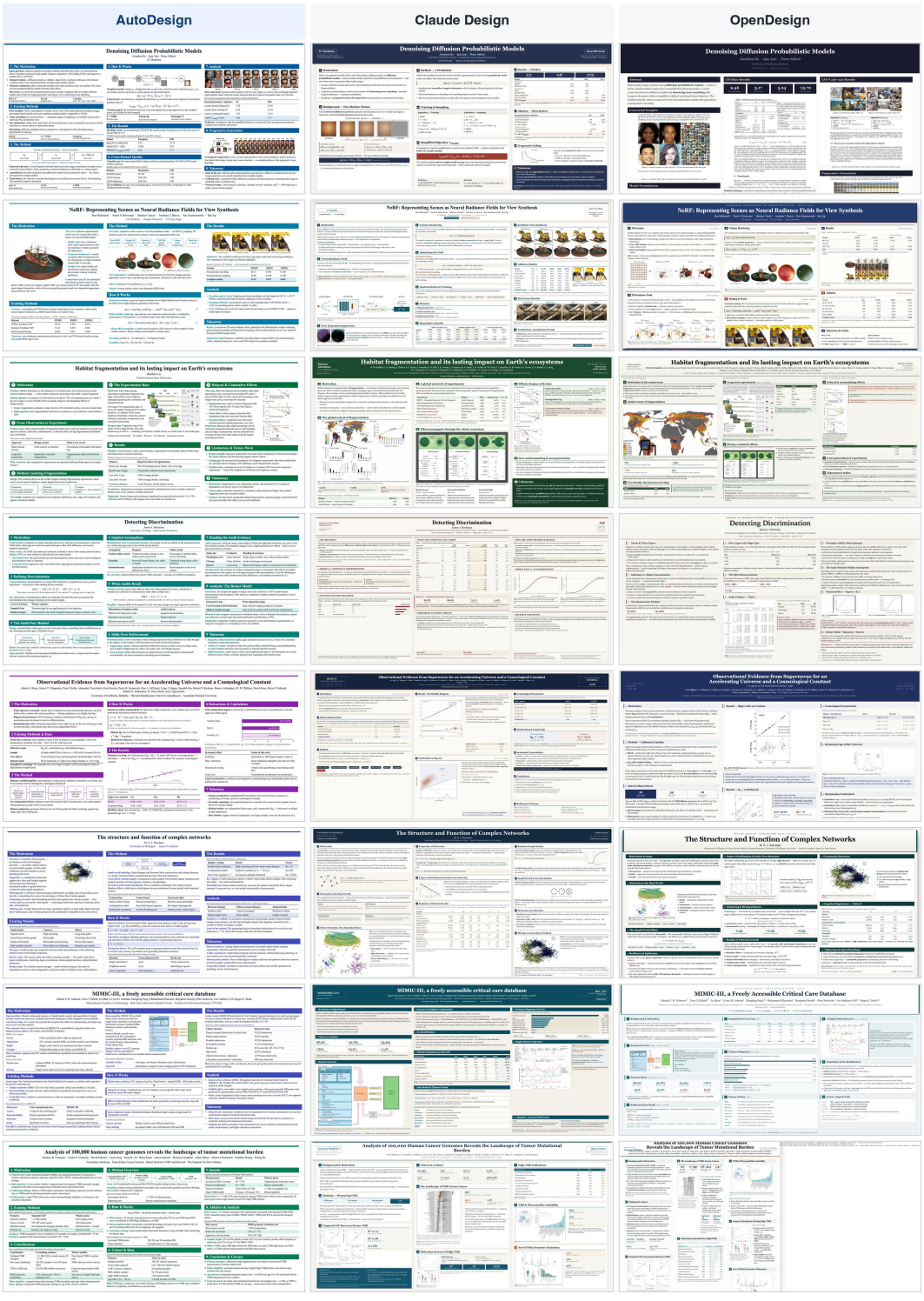


Figure 14: Additional matched qualitative comparisons across Design Agents with the same generation prompt; each row shows three independently generated posters for one source paper, at a common scale.

A.10 Additional AutoDesign Poster Demonstrations

The self-paper demonstration in Figure 2 used 253 tool calls and 11 editing turns in approximately 40 minutes for under $3. This is a single artifact-generation run; it is distinct from the multi-day harness-optimization record.

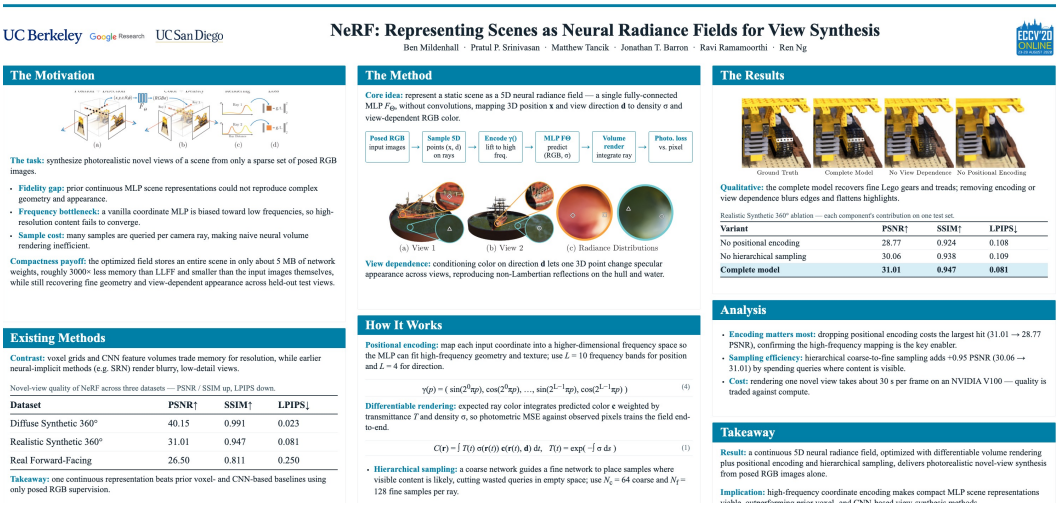


Figure 15: **Additional AutoDesign poster demonstrations.** NeRF is rendered from its source paper.

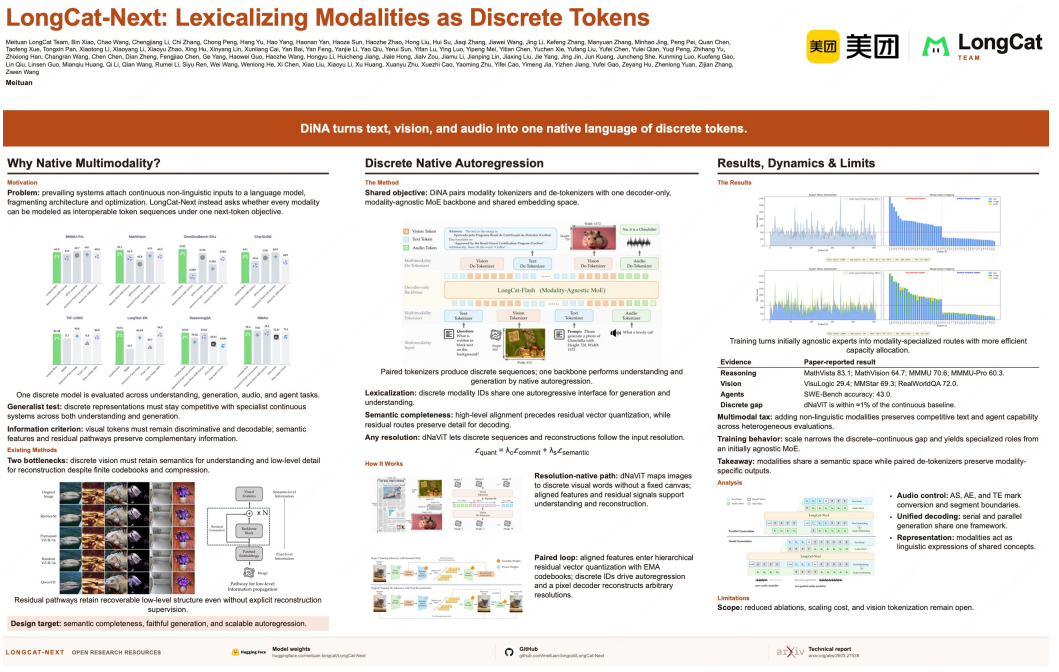


Figure 16: **Additional AutoDesign poster demonstrations (continued).** LongCat-Next is rendered from its source paper.

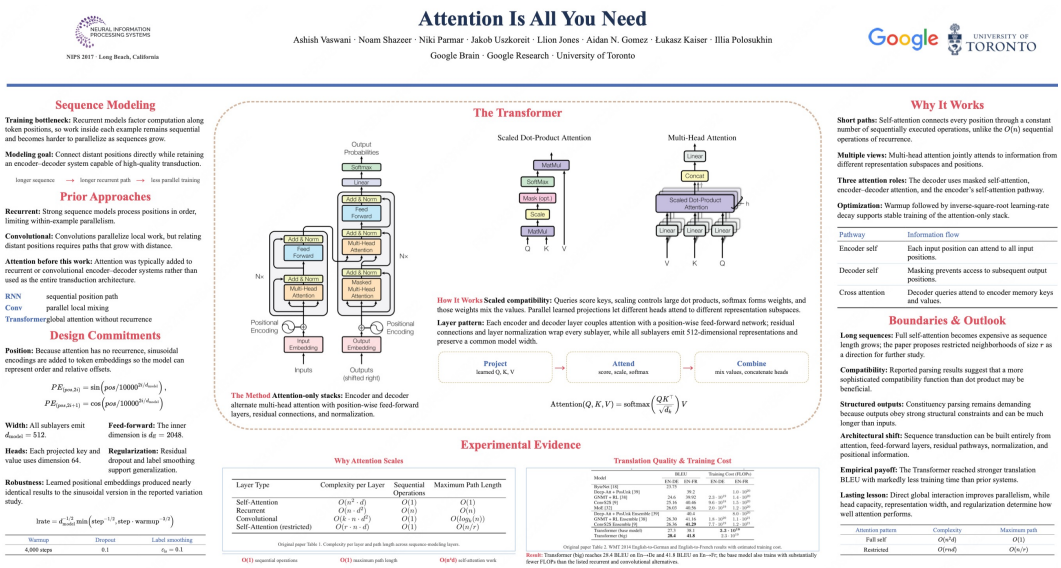


Figure 17: **Additional AutoDesign poster demonstrations (continued).** Attention Is All You Need is rendered from its source paper.

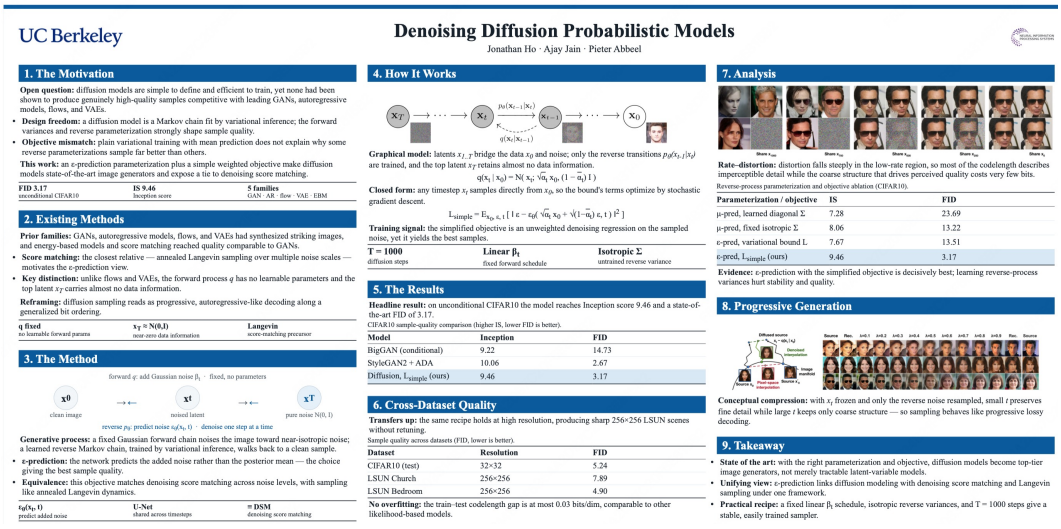


Figure 18: **Additional AutoDesign poster demonstrations (continued).** DDPM is rendered from its source paper.

B Future Directions

`AutoDesign` is presently validated for academic paper-to-poster generation, but the underlying agentic design pattern is not tied to a single input or output medium. The pilot artifacts in Figure 20 show that the current `DesignHarness` can also produce paper-to-slide, paper-to-webpage, and paper-to-conference-video outputs. Figure 19 illustrates the long-term direction: a multimodal-in, multimodal-out agentic design system that integrates papers, visual evidence, code, data, and human guidance, then iteratively creates the medium-appropriate output for a target communication setting. Making this expansion reliable requires more than exposing new output formats. Each medium needs source–output data, an evaluator, a rendering and validation gate, and an objective tailored to its communication setting. `PosterBench` formally evaluates academic posters only; the slide, webpage, and video artifacts therefore remain pilots. Shared context construction, preference memo-

ries, and repair histories could nevertheless provide a substrate for reusing experience across media, provided that the transfer is evaluated against medium-specific objectives. At meta-harness level, better component selection and evaluator evolution remain open problems. A selector should choose next bounded update from failure attribution, uncertainty, expected improvement, and component interactions. Any adaptive evaluator must remain versioned and anchored by frozen reference tasks, adversarial probes, and periodic human audits so optimization doesn't reward-hack a moving target.

Finally, recent work has begun to connect continual harness adaptation, self-improving harnesses, and model–harness co-evolution (Karten et al., 2026; Zhang et al., 2026a; Lee et al., 2026a). In this direction, harness optimization can complement model post-training: long-horizon trajectories and repair outcomes provide execution-time supervision, whereas the model supplies the reasoning and coding capabilities. Joint training should preserve this division while evaluating both layers against shared held-out objectives.

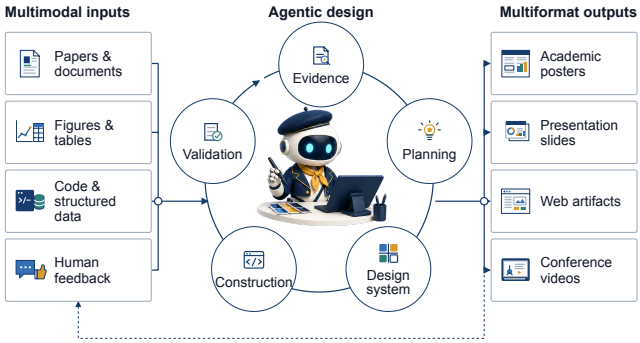


Figure 19: Future direction: a multimodal-in, multimodal-out agentic design system that integrates diverse sources and human guidance to iteratively create medium-specific outputs.



Figure 20: **Future directions for AutoDesign.** The initially optimized `DesignHarness` already extends paper-to-poster generation to slide decks, webpages, and conference videos. Applying the same meta-harness optimization methodology to these and other multimodal outputs offers a path toward a general multimodal-in & out agentic design system.