Declare, Compile, Look: Coordinate-Free Layout Generation with Vision-in-the-Loop Repair

Guian Fang[1], Mengsha Liu[2], and Mike Zheng Shou[1]

[1] Show Lab, National University of Singapore
[2] Nanyang Technological University

Abstract. Multimodal digital agents are usually studied as consumers of interfaces. We examine the complementary role, the agent as a producer of on-screen content, and the failure mode specific to it: a model that writes a layout by predicting absolute coordinates produces semantically right but structurally broken output, with overlapping boxes, dangling connectors, and drifting baselines. We describe a recipe in which the model never emits a coordinate: it declares elements and relations in a typed specification, a deterministic compiler solves the geometry, and the agent inspects the rendered result through a structural parse of the emitted SVG and a vision–language review of the pixels, repairing by patching the specification and recompiling. On a 292-sample public benchmark of scientific method figures, the recipe surpasses the strongest public baseline (63.9 vs. 60.2) and more than halves the mean structural error rate. We discuss what the recipe offers agents that generate or repair screen layouts, and draw one sharper point: an agent holding a complete, ground-truth DOM of its own output still gains from looking at the render, and the widest repair gain is faithfulness to the source document, a property the output alone cannot decide.

Keywords: Digital agents · Layout generation · Vision–language review

1 Introduction

Vision-centric digital agents are almost always studied as consumers of interfaces. They ground instructions in screenshots [3, 4], operate applications through visual perception [5], and are evaluated in environments built from software that already exists [10, 12]. Yet many agent tasks end the other way around: the agent must put content on a screen. Generating a front end from a design [8], assembling a slide, or drawing a diagram [2, 6] all terminate in markup that a browser or renderer turns into pixels.

Production has a characteristic failure of its own. Generative models write layouts by predicting absolute geometry: pixel coordinates in SVG, offsets in CSS, positions in slide XML. The content is usually right; the geometry is where quality is lost. Boxes overlap, arrows end in empty space, rows drift off a shared baseline. These are the properties a reader uses to parse a page, and they are hard to repair in the rendered artifact, which no longer records which relations were intended.

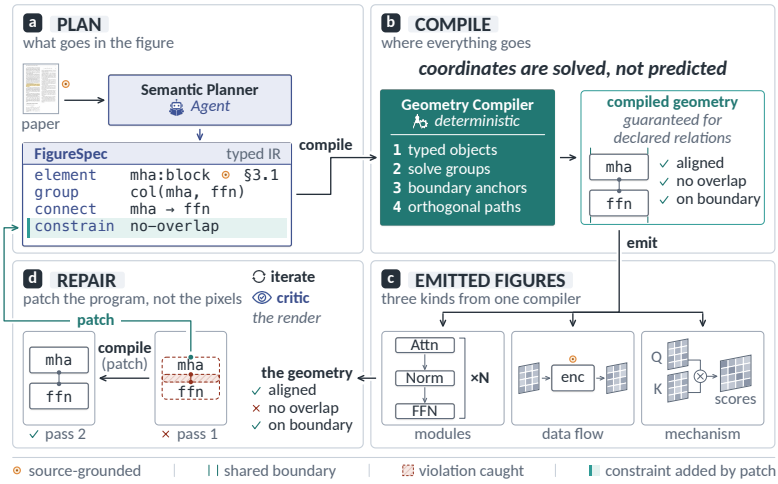

Fig. 1: The model decides content; a compiler decides geometry; the agent looks at the render. A planner emits a typed, coordinate-free specification (a); a compiler solves positions, anchors, and orthogonal paths (b), emitting vector figures (c); a checker and a vision–language reviewer diagnose the render, and repairs patch the specification, not the pixels (d). This figure is unedited output of the system it depicts.

We argue that semantic and geometric decisions should not share one generative mechanism, and describe a system built on that split (Fig. 1). The model plans what the layout contains in a typed specification with no coordinates; a deterministic compiler solves all geometry; the agent then verifies its output the way this workshop advocates perceiving interfaces, by looking at the render, and repairs by patching the specification.

We instantiate the recipe on scientific method figures, a benchmarked domain whose artifacts live in the browser stack: the compiler emits SVG, a browser renders it, and every structural property is measurable in the DOM; nothing in the recipe is figure-specific. From the results we draw three points for this community: (i) a coordinate-free action space for producing layouts; (ii) a reference-free structural audit that scores produced layouts without ground truth; and (iii) on the production side of this workshop's structure-versus-vision question, a measurement of what visual review adds when the agent already owns the structure of its own output.

2 The Recipe

Declare. A language model turns the source document into a typed, declarative program (FigureSpec): panels, typed visual elements, semantic relations, grouping and nesting, style roles, and, for every non-decorative element, a pointer to the source text that licenses it. It fixes what the layout contains and how its parts relate, and contains no coordinate, size, or path.

Table 1: Each stage of the recipe removes a different error class. Reference-free structural audit: per-figure error rates (N=292, lower is better).

Error class	M1	M2	M3	M4	**M5**
Per-shape overlap	0.267	0.140	0.161	0.123	**0.075**
Dangling endpoints	0.288	0.182	0.158	0.140	**0.123**
Alignment errors	0.182	0.151	0.147	0.082	**0.034**
Non-orthogonal connectors	0.250	0.161	0.188	0.120	**0.072**
Connector crossings	0.219	0.089	0.106	0.089	**0.017**
Mean	0.241	0.145	0.152	0.111	**0.064**

Compile. A deterministic compiler sizes each element by measuring its text, converts grouping declarations into hard constraints (canvas containment, minimum padding, legal nesting, no unintended overlap) and soft objectives, and solves them [1]. Every declared relation is then realized as a connector: a port is chosen on each endpoint's boundary and an orthogonal path is routed around the inflated boxes of all other objects [9]. Alignment, non-overlap, and connection termination hold by construction for every declared relation. The output is an SVG DOM with stable identifiers.

Look, then patch. The agent inspects the rendered result through two channels [7]. A geometry checker parses the emitted SVG and flags overflow, overlap, and alignment deviation. A vision–language reviewer examines a browser screenshot of the compiled page, together with the specification and the source document, for content coverage, relation correctness, visual hierarchy, and readability. A patch generator turns the diagnostics into edits of the specification: adding or removing elements, correcting labels, regrouping, switching composition templates. The loop ends when no hard geometric violation remains and the visual review passes.

3 Results on Scientific Figures

Setup. PaperBananaBench [13] provides 292 method-figure samples. A reference-based protocol has a judge model compare each generated figure with the author's own, mapping win/tie/loss to 100/50/0, so the author figure scores 50 by construction; the judge (Gemini 2.5 Pro) is from a different family than the generation backbone (Claude Opus 4.8), and candidate order is swapped [11]. We compare five systems: **M1**, direct text-to-image; **M2**, PaperBanana, the strongest public system [13]; **M3**, the same planned content with the model predicting SVG coordinates directly; **M4**, our compiler without the look-and-patch loop; and **M5**, the full system. M5 reaches an overall score of 63.9 against 60.2 for M2 and 11.5 for M1; faithfulness (47.6) stays below the author's 50.

Where each stage acts. Table 1 reports a structural audit that parses the final SVG of every system, with raster baselines audited after structure recovery; each rate is a within-figure fraction of the objects the error applies to, averaged over the split, and no reference figure is needed. Adding the compiler (M3 to M4) removes the errors that are decided once positions are solved: alignment falls from 0.147 to 0.082 and non-orthogonality from 0.188 to 0.120. Adding the look-and-patch loop (M4 to M5) removes the errors the compiler cannot fix because the specification itself asks for something infeasible: overlap falls from 0.123 to 0.075 and crossings from 0.089 to 0.017. In direct pairwise judgment M5 is preferred over M2 on publishability by 192 wins to 49 losses and over M4 by 132 to 86 (exact sign tests, Holm-corrected across four comparisons; $p<10^{-4}$ and $p=0.0022$). These numbers price the loop as a whole: the geometric gains could come from the structural checker alone, and a channel ablation remains to be run. Content is another matter: the widest pairwise margin over M4 is faithfulness to the source document, 150 wins to 74, and the structural checker never reads the source.

4 What This Offers Vision-Based Digital Agents

A production-side action space. For consuming interfaces the field debates pixels against accessibility trees; for producing them the choice is between predicting geometry and declaring structure. Every action in our loop is a typed edit (add an element, regroup, rewire a relation), and the compiler re-derives all geometry after each patch, so no action can introduce a misalignment. Agents that emit HTML and CSS, slide markup, or dashboard configurations can act at this level today; the solvers already exist.

A reference-free audit for produced layouts. Evaluating produced content today requires a reference, as when generated front ends are compared with the source design [8], or a judge model. The audit of Table 1 needs neither: its five error classes are computed deterministically from the rendered DOM alone, a cheap gate before an expensive visual review, and it runs unchanged on any structured render at benchmark scale.

Vision complements structure even when the agent owns the structure. The case for vision in this workshop's call is that structured representations of others' interfaces are incomplete or unavailable. Production inverts that premise: our agent holds a complete, ground-truth DOM of its own output, and the structural channel is the cheaper, more precise judge of geometric legality. The visual channel earns its place on what the output alone cannot decide: whether the render is faithful to a source the DOM never contains, and whether it reads well. An agent that only trusts its tree ships layouts that parse but do not read.

Our evidence comes from scientific figures; whether the same division of labor holds for full web pages and application interfaces is a question this community is best placed to test, and the audit and the loop are the instruments we bring.

References

1. Badros, G.J., Borning, A., Stuckey, P.J.: The Cassowary linear arithmetic constraint solving algorithm. ACM Transactions on Computer-Human Interaction **8**(4), 267–306 (2001)
2. Belouadi, J., Lauscher, A., Eger, S.: AutomaTikZ: Text-guided synthesis of scientific vector graphics with TikZ. In: International Conference on Learning Representations (2024)
3. Cheng, K., Sun, Q., Chu, Y., Xu, F., Li, Y., Zhang, J., Wu, Z.: SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In: Annual Meeting of the Association for Computational Linguistics (2024)
4. Gou, B., Wang, R., Zheng, B., Xie, Y., Chang, C., Shu, Y., Sun, H., Su, Y.: Navigating the digital world as humans do: Universal visual grounding for GUI agents. In: International Conference on Learning Representations (2025)
5. Hong, W., Wang, W., Lv, Q., Xu, J., Yu, W., Ji, J., Wang, Y., Wang, Z., Dong, Y., Ding, M., Tang, J.: CogAgent: A visual language model for GUI agents. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (2024)
6. Huang, S., Zhou, Y., Gao, Y., Yin, Z., Bai, J., Liu, X., Chellappa, R., Lau, C.P., Peng, C., Nag, S., Pramanick, S.: SciFig: Towards automating editable figure generation for scientific papers. arXiv preprint arXiv:2601.04390 (2026)
7. Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al.: Self-refine: Iterative refinement with self-feedback. In: Advances in Neural Information Processing Systems. vol. 36 (2023)
8. Si, C., Zhang, Y., Yang, Z., Liu, R., Yang, D.: Design2Code: How far are we from automating front-end engineering? arXiv preprint arXiv:2403.03163 (2024)
9. Wybrow, M., Marriott, K., Stuckey, P.J.: Orthogonal connector routing. In: Graph Drawing. pp. 219–231. Springer (2010)
10. Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T.J., Cheng, Z., Shin, D., Lei, F., et al.: OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In: Advances in Neural Information Processing Systems. vol. 37 (2024)
11. Zheng, L., Chiang, W.L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E.P., et al.: Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In: Advances in Neural Information Processing Systems. vol. 36 (2023)
12. Zhou, S., Xu, F.F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., Alon, U., Neubig, G.: WebArena: A realistic web environment for building autonomous agents. In: International Conference on Learning Representations (2024)
13. Zhu, D., Meng, R., Song, Y., Wei, X., Li, S., Pfister, T., Yoon, J.: PaperBanana: Automating academic illustration for AI scientists. arXiv preprint arXiv:2601.23265 (2026)