ReDesign: Recovering Editable Design Structures from Images via Agentic Decomposition

Jooyeol Yun*[1], Jintae Park*[1], Hyesu Lim[2], Junha Hyung[1], Hyungjin Chung[34], and Jaegul Choo[1]

[1] KAIST AI, Daejeon, Korea
{blizzard072, sonjt00 , sharpeeee, jchoo}@kaist.ac.kr
[2] Helmholtz Munich, Neuherberg, Germany
hye.lim@helmholtz-munich.de
[3] Korea University, EverEx
hj.chung@korea.ac.kr

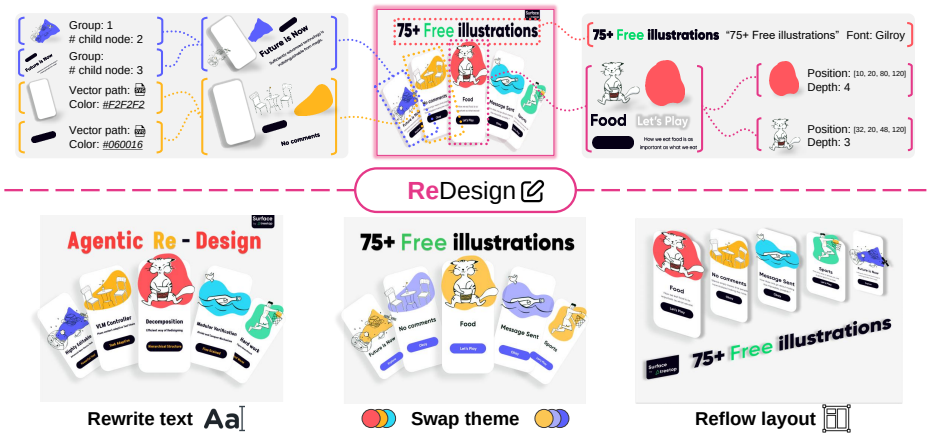


Fig. 1: ReDesign turns raster images into editable representations that enable real-time adjustments, such as rewriting text, color theme swaps, and repositioning layouts, directly to designs. Visit the project page for interactive demos.

Abstract. Recovering an editable design file from a raster image is a common and costly bottleneck in modern design workflows, yet remains challenging since editability depends on recovering multi-modal attributes, such as typography, vector geometry, colors, grouping, and layer ordering. We present ReDesign, an agentic framework that grows an editable layer hierarchy by selecting and composing specialized tools across modalities. To keep this long decision process reliable despite imperfect tool outputs, we introduce graceful verification at each expansion, which provides local accept, prune, or retry feedback that prevents error accumulation and avoids large scale reruns. To evaluate editability at scale, we introduce the Figma Edit Replay Benchmark, consisting of 909 raw Figma files and 14,796 controlled edit instructions that replay edits on reconstructed outputs. Across this benchmark and standard reconstruction metrics, ReDesign achieves strong visual fidelity while delivering the highest editability across layout, color, and text edits, outperforming layered decomposition baselines and serial tool use pipelines.

* indicates equal contribution.

1 Introduction

Designers routinely need to adapt a single design across media platforms, repurpose layouts for new campaigns, and make adjustments for accessibility. In practice, many real world design assets and handoff deliverables exist only as raster exports or screenshots, so the preceding task is to reconstruct an editable version (*e.g.*, Figma or Adobe Illustrator file) from a raster image. Today, that reconstruction is largely manual and error prone, forcing designers into tedious recreation before any meaningful edits can begin. Thus, a system that automatically recovers editable components such as text layers, vector shapes, and layouts can remove this tedious step and dramatically reduce the turnaround time for designers.

Making a raster image re-editable means recovering the same structured representation that designers work with. Design tools, such as Adobe Illustrator, Figma, and Canva, keeps typography, color, and layout as separate objects with their own parameters because that aligns with how humans perceive and edit a design. This structure is exactly what makes editing easy, as these representations expose the correct "knobs", such as font attributes for text, vector paths for primitive shapes, and position and scale for layouts. In essence, rasterization is a many-to-one mapping from editable layer structures to pixels, so recovering the original structure from an image is an ill-posed inverse problem. While the final appearance may look identical, a raster export no longer states which pixels belong to which object, which attributes produced them, or how elements were layered. Therefore, a system has to infer a plausible set of objects and parameters across multiple modalities including text, geometry, color, and layout, so the result both matches the image and behaves like an editable design.

Many previous studies solve key subproblems for editable reconstruction, such as recognizing text [7], decomposing layers [18,30,41], and extracting shapes [23, 38]. A natural next step is to combine these components into an end-to-end system, but a straightforward composition is often unreliable, since errors from one tool can cascade and a single fixed sequence rarely fits the diversity of real designs. In this paper, we present ReDesign, which addresses this gap by focusing on the structure of the editable output. Editable designs naturally come as a layer hierarchy, with groups and layers organized in a tree, and we therefore cast reconstruction as recovering this hierarchy from a raster image. Starting from the full image as the root, ReDesign operates as a vision-language model (VLM) agent that repeatedly selects a node to expand and composes a tool sequence from a fixed action set to produce child nodes, progressively refining the hierarchy into atomic editable elements.

In practice, tool outputs are often imperfect, and the controller can make incorrect expansion choices, producing branches that drift away from the actual structure of the original design. An accept or reject decision on the final output [4,26] may be easy to compute but offers little guidance about which expansion failed or how to repair it. We therefore introduce a graceful verification at every node expansion, where a verifier evaluates the proposed children, and either accepts the expansion, prunes invalid branches, or retries with a different

tool or configuration. The verifier targets failure modes specific to hierarchical growth, such as duplication across sibling nodes and incomplete coverage of the parent region. This keeps tree growth efficient by pruning and retrying only the offending branch at the moment it diverges, rather than letting errors accumulate until they force a hard failure.

To evaluate our method, we introduce the Figma Edit Replay Benchmark, a collection of diverse real world designs from raw Figma files [8] with ground-truth layer hierarchies and attributes. Using this benchmark, we evaluate both visual fidelity and editability, with editability measured by replaying 14,796 controlled edit instructions spanning layout, color, and text. Our evaluations show that ReDesign achieves superior performance compared to strong baselines.

Our contributions are threefold: Our contributions are threefold:

- We introduce ReDesign, an agentic raster-to-editable reconstruction system that casts the problem as structured tree expansion and grows an editable layer hierarchy into atomic elements through tool composition.
- The graceful verification enforces local correctness at each node expansion with accept, prune, and retry outcomes, enabling targeted repair and preventing hard failures from cascaded tool errors.
- We construct the Figma Edit Replay Benchmark and conduct extensive experiments showing that ReDesign achieves strong visual fidelity and state-of-the-art editability across layout, color, and text edits, with supporting ablations and comparisons against strong baselines.

2 Related Work

2.1 Layered Image Generation and Decomposition

A growing line of work studies layer-aware image generation [3, 5, 9, 13, 20, 21, 45] and decomposition [30, 33], motivated by the need for controllable composition and post-hoc editing [4, 12, 42, 44]. Closest to our setting are methods that decompose raster graphics into layers or generate PSD-like layered outputs, aiming for inherent editability [18, 30, 41]. These approaches provide useful primitives for isolating elements and for constraining edits to localized regions, and they become even more practical when combined with modern image editing models [1, 11]. However, producing layers is only part of reconstructing an editable design file, which also requires correct semantics, hierarchy, and parameters so that edits behave as expected. We therefore treat layered decomposition models as tool for a larger decomposition workflow, rather than as a complete solution.

2.2 Image Vectorization and SVG Generation

Image vectorization has a long history, from classical tracing pipelines [6] to learned models that generate vector primitives or scalable vector graphics (SVG) programs [2, 23, 28, 38]. Vectorization is an important ingredient for reconstructing editable shapes and icons, since vector paths expose direct controls for geometry, scale, and appearance. At the same time, vectorization is not universally

desirable, since forcing complex textures or photographic content into paths can reduce editability and distort the representation relative to how the design was authored. Practical reconstruction therefore requires deciding when vector paths are appropriate, and when regions should instead be represented as text or raster layers. Thus, similarly, we utilize vectorization methods as one set of tools within our hierarchical decomposition process, selectively applying them where they best support downstream editing.

2.3 Tool-Using Agents

Tool-using agents extend language and vision-language models [27, 31] with external modules, enabling multi-step problem solving through action selection and execution [25, 39]. In parallel, modern image editing models provide practical primitives for localized manipulation, content removal, and background completion that can support iterative pipelines [1, 11]. Most existing agentic pipelines for vision emphasize serial tool invocation, often paired with an end-stage validation of the final output [22, 26]. Editable reconstruction differs in that it requires a long sequence of structural decisions, where early commitments change the state of the reconstruction and constrain what later tools can recover. This makes reliability depend on how intermediate decisions are guided and revised during the process, rather than only on the strength of individual tools or a final verification step. Our work builds on these agentic and editing primitives, and focuses on structured sequential decision making that supports local revision while constructing an editable layer hierarchy.

3 Method

3.1 Problem Setup and Overview

Given an input raster image, our goal is to recover an editable design structure, as we depict in Figure 2. The desired output is a JSON hierarchy that has editable meta-data, such as vector shapes, colors, text contents, fonts, groups, and z-orders, and this tree hierarchy must be inferred from the root node which is the raster image.

We maintain a partial reconstruction tree, which is a tree that contains nodes whose metadata fields may be incomplete and are filled in over time. The partial reconstruction tree is expanded as the controller policy selects an action from a fixed set of actions. A graceful verifier evaluates the proposed expansion at every step and either accepts the proposed children nodes the or provides signals to revise the actions. The process terminates when the tree has sufficient metadata to be exported as an editable JSON hierarchy.

3.2 Action Space and Tools

Converting a raster design into an editable format requires handling heterogeneous element types, such as text, vector shapes, photographs, and layered compositions, each demanding different specialized processing. We therefore define

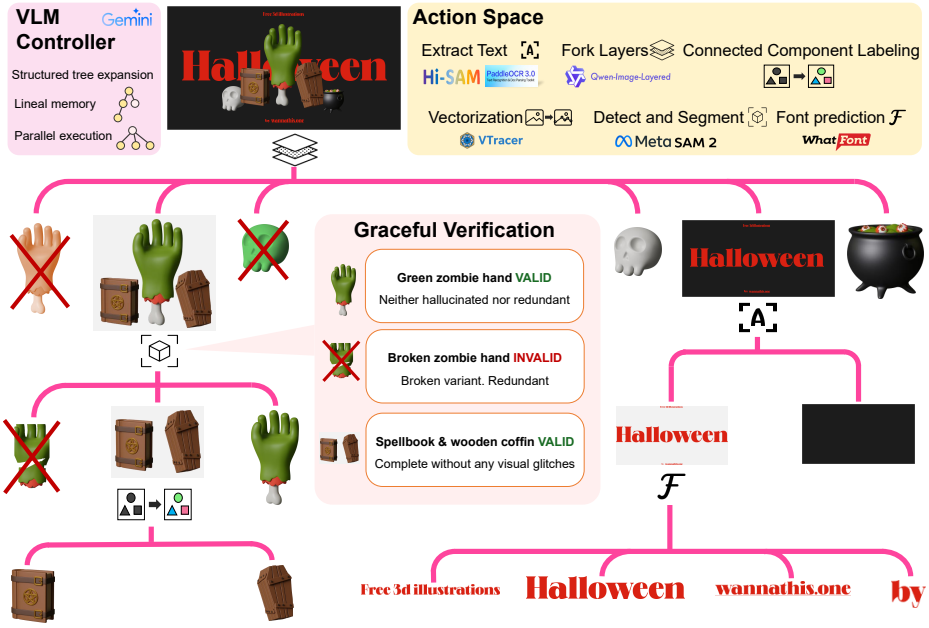


Fig. 2: Overview of ReDesign. A VLM controller grows an editable layer hierarchy by selecting from a fixed set of tool actions for text, objects, and layered decomposition, and runs graceful verification at each parent-to-children expansion to accept valid splits, prune redundant or hallucinated children, or trigger a retry, until atomic editable elements are recovered.

a discrete action space in which each action encapsulates a tool chain tailored to a specific decomposition scenario. This space is modular by design and can be extended as new tools become available.

Text extraction. An optical character recognition tool [7] localizes and reads visible text, after which the text pixels are segmented [40] and the vacated region is inpainted [29], producing a text layer and a remainder layer with the remainder background.

Multi-layer decomposition. A layered image generation model [41] offers powerful decomposition capability separating the raster image into distinct layers, with an adaptively chosen layer count based on visual complexity.

Connected Component Labeling. When a layer contains multiple elements that are spatially disjoint, it can be split into non-overlapping children using a simple connected component labeling algorithm [24].

Detection and segmentation. For entangled objects, a VLM [10] identifies the foremost element, an open-set detector [19] localizes it, a segmentation

model [15] extracts it, and an inpainting model [29, 46] fills the cropped out region, yielding a foreground node and a background node.

Vectorization. Leaf nodes are finalized as either vectorized SVG paths [6] for shape-like content, or raster layers with placement metadata for photographic content. Text leaves undergo font recognition [34] and typographic property fitting to produce fully editable text elements carrying necessary font attributes.

3.3 Controller Policy for Tree Expansion

The core idea of the controller policy is to follow the same structure as an editable design file, instead of a serial or unstructured sequence of actions (tool calls). We therefore represent the reconstruction as a tree that is grown progressively from the full canvas into smaller, more specific regions, so early decisions establish high level grouping and ordering, and later decisions refine each group into editable leaves. Each node represents a localized region together with its current metadata, and any attributes already inferred by previous tools. At each step, the controller selects an action given the node's partial rendering, and the expansion history stored for that node, which records prior tool choices and verification outcomes.

Furthermore, since an expansion only depends on the parent node and its lineal state, and not on sibling states, our structured tree growth naturally enables parallel expansion across all current leaf nodes. Thus, we can expand multiple leaf nodes simultaneously, which improves throughput without changing the decision logic or sacrificing the coarse-to-fine consistency of the recovered hierarchy.

3.4 Graceful Verification

Structured expansion alone does not guarantee a correct reconstruction due to imperfect tools and occasional controller mistakes that propose duplicate content across siblings, leave parts of the parent region unexplained, or introduce content that was never present in the input. These errors are especially harmful since they immediately change the state of the tree and therefore constrain all later actions, resulting in a hard failure, where a system would have to re-run from the begining.

However, our structured expansion effectively limits the scope of each error to a single parent-to-children proposal, and therefore makes verification both local and well-defined, as the goal of an expansion is simply to decompose one parent region into a set of children that jointly explain it. Thus, we attach a graceful verification step to every proposed expansion and evaluate whether the candidate children form a valid decomposition under two simple criteria, (1) does the union of child nodes cover the parent content and (2) does the child node expand beyond the parent content. The verifier either accepts the expansion, prunes invalid or redundant children, or triggers a retry by requesting a different tool choice or adjusted hyperparameters for the same parent. In this way, errors are corrected locally before deeper branches are built on top of them, and the

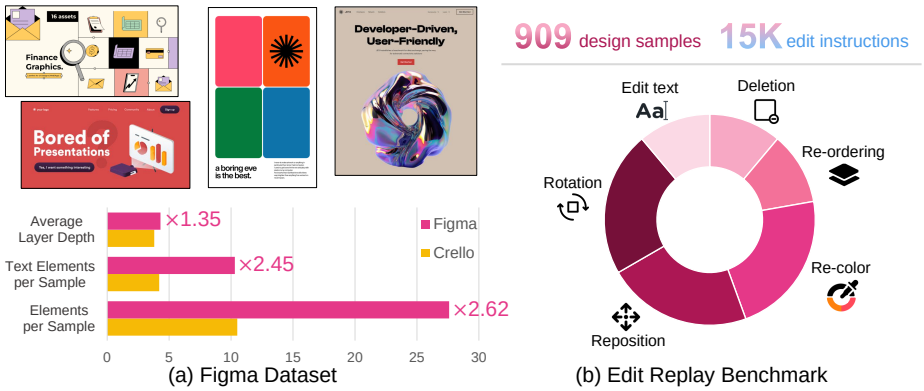


Fig. 3: (a) Figma examples and statistics compared to the Crello dataset. (b) Edit Replay Benchmark and detailed edit types.

controller receives structured feedback tied to a single parent to children decision, which in turn reduces hard failures that would require large-scale reruns.

3.5 Memory Management and Repair Signals

We maintain a *lineal* memory per node, meaning that each expansion stores only its own history along the path from the root rather than tracking sibling states, and this design avoids exponential growth while also preventing concurrency issues due to parallel execution. Specifically, for every node we record the tool inputs and outputs, the tool choice and hyperparameters, and any verification outcomes or failure reasons, and this compact record makes the controller's next decision better grounded because it can reuse what worked and avoid repeating configurations that already failed. Therefore, when an expansion is rejected or pruned by the verifier, the controller can perform local repair by selecting a different action or adjusted settings based on the stored failure signal, and continuing growth without disturbing the rest of the tree, which reduces redundant computation and keeps parallel progress stable.

4 Experiments

4.1 Figma Dataset and the Edit Replay Benchmark

To evaluate reconstruction on real-world design artifacts with ground-truth structure, we collect a dataset of 909 raw Figma design files. Each file provides the layer hierarchy along with element types and attributes, including text content and typography, vector shapes, colors, and ordering. As seen in Figure 3 (a), we find that the Figma dataset consists of complex real-world designs compared to the original Crello dataset [37], with an average 2.62 times more elements per sample.

We use these files to benchmark editability by generating paired supervision. Specifically, for each design, we apply a controlled edit directly to the original Figma document (*e.g.*, reposition, text edit, and recolor) and render the post-edit image as ground truth. In total, we curate 14,796 edit instructions, averaging 15 edits per design.

We also evaluate visual reconstruction accuracy on the Crello dataset [37], which provides raster design images with annotations. Following standard practice, we report reconstruction metrics on Crello to enable comparison with prior methods that emphasize visual fidelity and element recovery.

4.2 Baselines

Layered image decomposition. We compare against Qwen-Image-Layered [41] and LayerD [30], which decompose an input image into a set of RGBA layers. Since RGBA images are not directly editable, we post process the layers using OCR and vectorization tools, as discussed in the original papers.

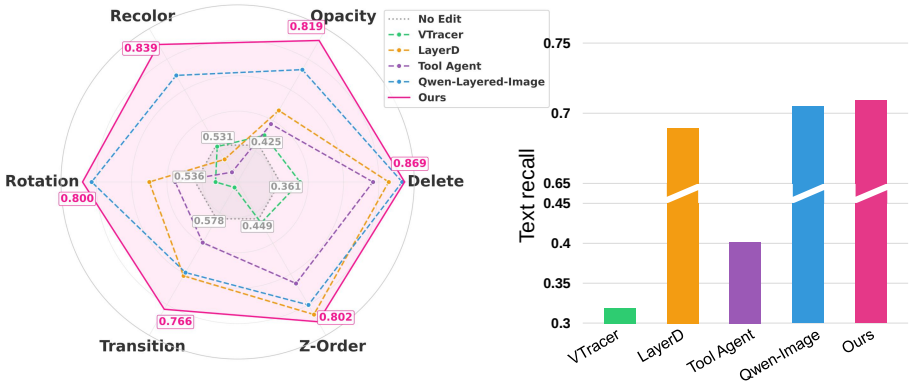
Tool using agent. We compare against a standard tool using agent, similar to a ReAct framework [39], that has access to the same tools and memory and uses the same VLM backbone. The agent produces an editable JSON output through a single linear sequence of tool calls, with validation at the end.

Image vectorization. We compare against VTracer [6], a widely used image vectorization tool, and treat their outputs as an editable design. We also experimented with recent vectorization models (*e.g.*, OmniSVG [38] and StarVector [23]), but found that they perform poorly on design images due to a domain gap, since many are trained primarily on icons or simplified graphics.

4.3 Editability

Edit replay protocol. We evaluate editability on the Figma design dataset using an edit replay protocol, meaning that we apply the same edit operation to the ground-truth design and to the reconstructed design, and then compare the rendered results. Each example provides a ground-truth editable design file and its corresponding raster image, after which each method takes the raster image as input and produces an editable output in the same target format. We then replay the edit instruction on the ground truth file to obtain a rendered reference, replay the same instruction on the predicted editable output, and finally measure agreement between the two edited renderings.

Specifically, to apply the same instruction to a predicted editable output, we first match the target ground-truth element to an element in the prediction. Since predictions made by different models have different representations of a reconstructed design, we match the target and ground-truth using the highest intersection over union (IoU) among all decomposed elements. If no suitable match exists, the edit is not executed. Otherwise, we apply the same property change to the matched predicted element and render the edited prediction. We



(a) SSIM after edit replay (b) Text recall after text edits

Fig. 4: (a) Edit replay SSIM for different edit types, where our method performs best across all edits. (b) Text editability measured by text recall after replaying text edits, where our method achieves the highest recall among baselines.

measure the structural similarity (SSIM) [32] between the ground-truth rendered image and the edited image. For text edits, we additionally evaluate whether the edited text is correctly realized by running OCR on the rendered result and reporting text recall against the ground truth rendered text.

Editability results. As seen in Figure 4, our method achieves the best editability across all edit types, showing that the recovered hierarchy supports both appearance edits and structural edits without breaking the design. In particular, we perform strongly on attribute edits such as recolor and opacity, where baselines often fail because the target element is not cleanly separated and the edit bleeds into background pixels or nearby layers. Qwen-Image-Layered [41] and LayerD [30] remain competitive on simpler geometric edits such as delete and rotation, but their performance drops on edits that depend on correct element attribution and ordering, which requires the model to recover which pixels belong to which layer and how layers are stacked. We observe the same trend for text, where Figure 4 (b) shows that our method achieves the highest text recall after replaying text edits, while the Tool Agent baseline degrades sharply because error cascades in long serial tool chains often corrupt the reconstruction early, and text is typically the first content to become distorted and undetectable in later steps.

Qualitative examples in Figure 5 illustrate this failure mode, where edits intended for one element unintentionally affect several other elements. Overall, these results indicate that structured reconstruction yields representations that can be precisely targeted, and that graceful verification prevents early decomposition mistakes from propagating into failures for edits that require correct grouping and z-order.

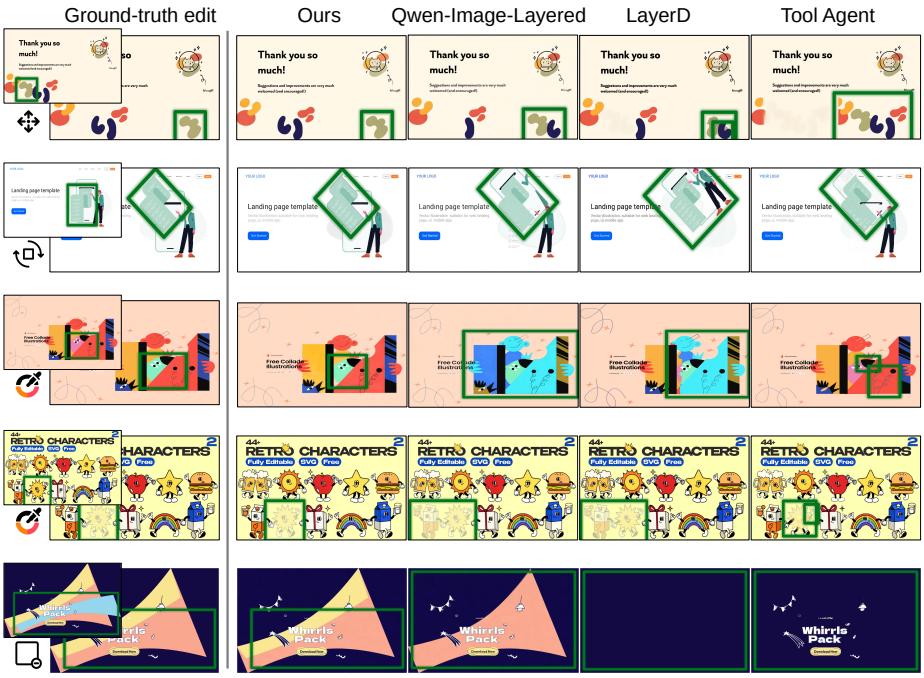


Fig. 5: Example of edit instructions performed on editable formats produced by each baseline. Ground-truth edit pairs are the original image (top-left) and the edited image. Target edit areas are highlighted in green. Best viewed digitally.

4.4 Reconstruction Accuracy

We evaluate visual reconstruction accuracy on both the Figma design dataset and the Crello dataset. For each input raster image, each method produces an editable output in our target format, which we render back to a raster image. We compare the rendered reconstruction to the input using complementary metrics that capture object-level appearance (mean L1 error), global visual fidelity (PSNR and LPIPS [43]), and layout fidelity (panoptic quality (PQ) [14] and detection F1), reported in Tables 1 and 2.

On Figma, our method is best across all metrics, improving object appearance, global fidelity, and layout. The Tool Agent baseline shows much weaker global fidelity than other approaches, supporting our design choice of structured hierarchical decomposition with graceful verification to prevent error accumulation in long, linear tool chains. Qwen-Image-Layered [41] attains strong global similarity but lags in layout, consistent with layer decompositions that match overall appearance while mis-segmenting or mis-ordering elements. Qualitative results are presented in Figure 6, where baseline approaches often trade off between plausible hierarchical layouts and precise object segmentation, whereas our

Table 1: Reconstruction accuracy measured on the Figma dataset.

Method	Object-level	Global-level		Layout	
	L1↓	PSNR↑	LPIPS↓	PQ↑	F1↑
VTracer [6]	0.0977	20.487	0.1917	24.64	0.309
LayerD [30]	0.0704	16.141	0.3381	30.09	0.350
Qwen-Image-Layered [41]	0.0493	26.192	0.1073	35.37	0.429
Tool Agent [39]	0.0493	13.923	0.3869	45.33	0.527
Ours	**0.0431**	**26.286**	**0.0883**	**45.37**	**0.535**

Table 2: Reconstruction accuracy measured on the Crello dataset [37]

Method	Object-level	Global-level		Layout	
	L1↓	PSNR↑	LPIPS↓	PQ↑	F1↑
VTracer [6]	0.0953	18.805	0.2919	19.55	0.243
LayerD [30]	0.0938	14.452	0.4585	30.98	0.369
Qwen-Image-Layered [41]	0.0506	**26.419**	**0.0985**	40.19	0.496
Tool Agent [39]	0.0767	12.011	0.5674	44.16	0.527
Ours	**0.0465**	23.525	0.1249	**49.57**	**0.587**

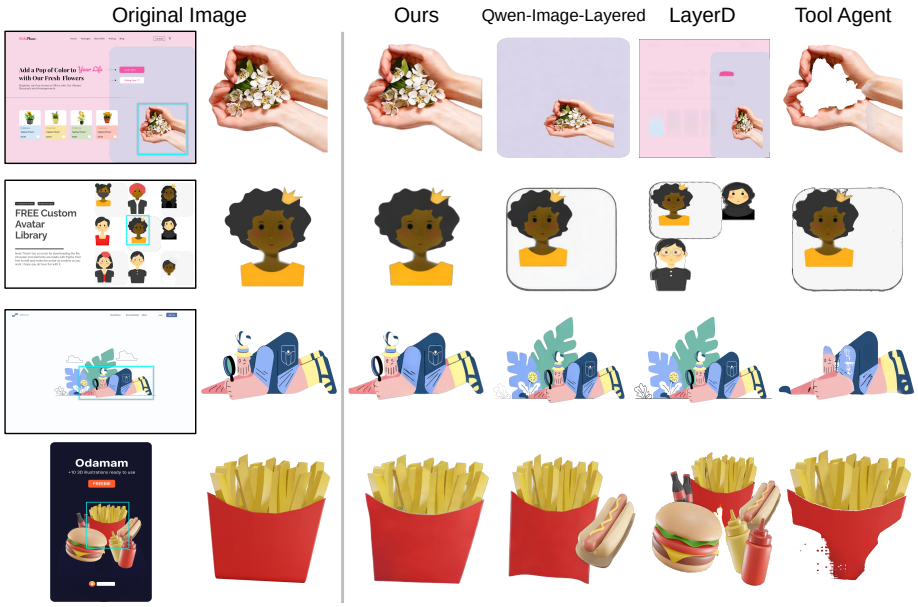


Fig. 6: Decomposition accuracy comparison against baseline approaches. Previous approaches often fail to decompose multiple objects with high accuracy.

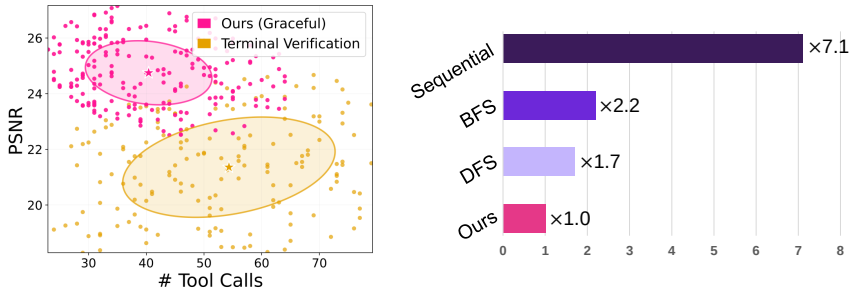
method preserves both, yielding decompositions that are structurally coherent and cleanly editable.

On Crello, trends are similar. LayerD and Qwen-Image-Layered improve due to closer training distribution, but our approach still achieves the strongest layout fidelity while remaining competitive on object-level and global metrics, without explicit training on the Crello dataset.

5 Analysis

5.1 Inference Cost, Variance, and Parallelism

Agentic systems must account for cost, since each planning step and tool call consumes time and tokens, and one might expect that adding verification would only slow the pipeline down. In contrast, our results show that many small local verifications are cheaper than a few large terminal checks, because validating each structured decomposition step and repairing errors immediately avoids long error cascades that would otherwise force expensive restarts. In Figure 7(a), we plot time against accuracy (*i.e.*, PSNR) for individual runs, and we find that our method is faster and more accurate than a sparse terminal verification, suggesting that keeping every step safe is often the shortest path through the search space. In addition, we find that our execution episodes have lower variance in the number of tool calls, indicating stable executions with minimal waste of tool calls.



(a) Verification Cost-Accuracy Tradeoff (b) Runtime from Parallel Execution

Fig. 7: (a) Verification cost accuracy tradeoff. Graceful verification is faster, more accurate, and has lower variance than terminal verification. (b) Speedup from parallel tree expansion. Independent node expansions enable concurrent execution, yielding up to a 7.1× speedup over serial tool use.

Furthermore, tool using agents are often slow for practical use because decisions and tool calls are typically executed in a single serial trajectory [16,17,35,36]. In contrast, our reconstruction is organized as a tree, and each expansion depends only on its parent node and lineal history rather than on sibling states, so multiple frontier nodes can be expanded in parallel without concurrency conflicts. As a result, our approach runs 7.1 times faster compared to a serial tool

execution, as shown in Figure 7 (b). More broadly, our analysis reveals that formulating reconstruction as tree expansion enables speedups in both a technical sense, since independent nodes can be scheduled concurrently with minimal shared state, and a theoretical sense, since the dependency structure reduces the critical path length from the number of expansions to the depth of the hierarchy.

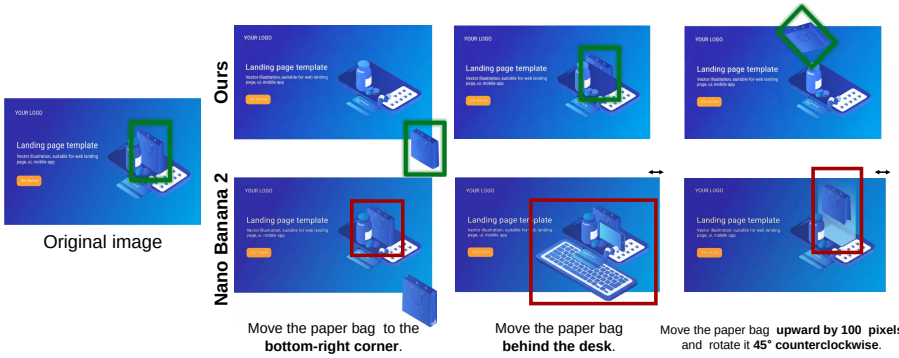


Fig. 8: Nano Banana 2 comparison on spatial edits. For the same move and rotate instructions, Nano Banana 2 often changes unintended regions and change image sizes.

5.2 Why Editable Formats Still Matter

Image editing models [1, 11] are rapidly becoming more capable and can often produce visually plausible single step edits, which makes them appealing as a general solution for design iteration. Our comparison in Figure 8 shows that this flexibility does not necessarily indicate accuracy, as Nano Banana 2, a state-of-the-art image editing service, struggles with detailed spatial adjustments and also change aspect ratios even when the instruction is local. This highlights why recovering an explicit editable representation remains important, because a layer hierarchy with object identities and parameters supports deterministic layout changes, stable composition, and repeatable multi-step edits without unintended global warping.

5.3 Limitations

A single notion of a good editable format is hard to define, since it depends on the intended edit and the level of control a designer needs. For example, vectorization can be unnecessary when the goal is a layout level adjustment, while fine grained paths matter when editing icons or shapes. Since we do not train to match the exact granularity of the Figma or Crello annotations, our reconstructions do not always align one to one with their layer splits, even when the rendered appearance is correct.

At the same time, not being tied to a fixed training distribution gives our system the flexibility to tune to any level of granularity. Since reconstruction

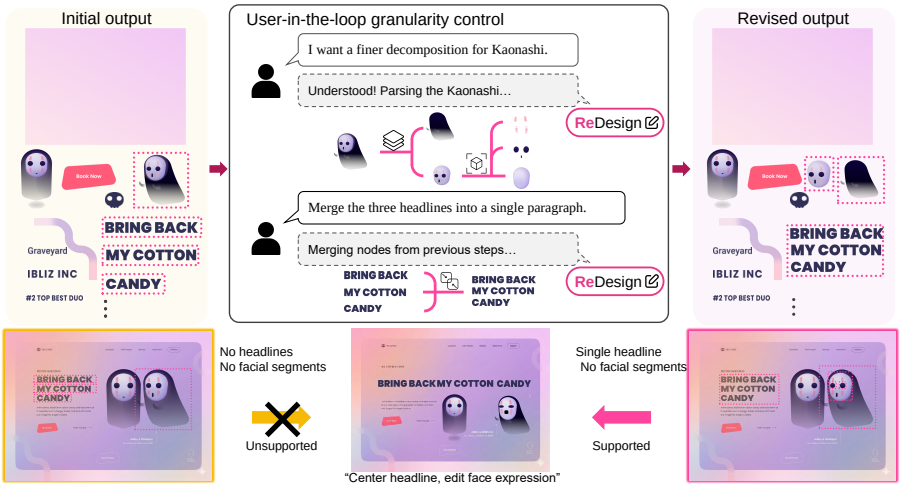


Fig. 9: Our method offers the flexibility to adjust editing granularity through a user-in-the-loop revision.

is agentic, users can request a finer or coarser decomposition through prompting, and the system can continue expanding the tree to match the desired edit intent, as shown in Figure 9. Moreover, the hierarchy naturally exposes higher level groups first, so complex vector paths can remain encapsulated and only be revealed when a user explicitly needs low level geometric control.

6 Conclusion

In this paper, we introduce an agentic framework that reconstructs editable design files from raster images by growing a layer hierarchy with specialized tools and enforcing graceful verification at every node expansion, so errors are detected and repaired locally instead of accumulating into hard failures. Across our new benchmark of raw Figma files with 14,796 edit replay instructions, ReDesign delivers strong visual fidelity while achieving the highest editability across layout, color, and text edits, and our analysis shows that step level verification and tree structured execution can be faster, more accurate, and more reliable than terminal checking and serial tool chains.

More broadly, our results suggest a practical direction for agentic systems, that imposing structural constraints on how an agent represents progress can simultaneously improve controllability, accuracy, and efficiency, because failure becomes local and repairable, and executions are predictable and thus scalable. In the perspective of designing workflows, despite the ever so growing capabilities of raster-based image editing, our method points to a future where editable representations remain essential, since they provide precise, real time control through explicit objects, parameters, and hierarchy, enabling users to make targeted changes with predictable behavior rather than relying on fragile pixel level edits.

Acknowledgment

This work was supported by Institute for Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (RS-2019-II190075, Artificial Intelligence Graduate School Program(KAIST)). This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. RS-2025-00555621). This work was supported by the Technological Innovation R&D Program (RS-2024-00438252) funded by the Ministry of SMEs and Startups(MSS, Korea).

References

1. Batifol, S., Blattmann, A., Boesel, F., Consul, S., Diagne, C., Dockhorn, T., English, J., English, Z., Esser, P., Kulal, S., et al.: Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. arXiv preprint (2025) 3, 4, 13
2. Carlier, A., Danelljan, M., Alahi, A., Timofte, R.: Deepsvg: A hierarchical generative network for vector graphics animation. NeurIPS (2020) 3
3. Chen, H., Xu, X., Li, W., Ren, J., Ye, T., Liu, S., Chen, Y.C., Zhu, L., Wang, X.: Posta: A go-to framework for customized artistic poster generation (2025) 3
4. Chen, J., Wang, Z., Zhao, N., Zhang, L., Liu, D., Yang, J., Chen, Q.: Rethinking layered graphic design generation with a top-down approach. In: ICCV (2025) 2, 3
5. Chen, J., Jiang, H., Wang, Y., Wu, K., Li, J., Zhang, C., Yanai, K., Chen, D., Yuan, Y.: Prismlayers: Open data for high-quality multi-layer transparent image generative models. arXiv preprint (2025) 3
6. Cortex, V.: Vtracer (2023), https://www.visioncortex.org/vtracer-docs 3, 6, 8, 11
7. Cui, C., Sun, T., Lin, M., Gao, T., Zhang, Y., Liu, J., Wang, X., Zhang, Z., Zhou, C., Liu, H., Zhang, Y., Lv, W., Huang, K., Zhang, Y., Zhang, J., Zhang, J., Liu, Y., Yu, D., Ma, Y.: Paddleocr 3.0 technical report (2025) 2, 5
8. Figma, I.: Figma community. https://www.figma.com/community 3
9. Fontanella, A., Tudosiu, P.D., Yang, Y., Zhang, S., Parisot, S.: Generating compositional scenes via text-to-image rgba instance generation. NeurIPS (2024) 3
10. Google: Gemini 3 (2025), https://blog.google/products-and-platforms/products/gemini/gemini-3/ 5
11. Google DeepMind: Nano banana: Ai image editing tool (2025) 3, 4, 13
12. Huang, R., Cai, K., Han, J., Liang, X., Pei, R., Lu, G., Xu, S., Zhang, W., Xu, H.: Layerdiff: Exploring text-guided multi-layered composable image synthesis via layer-collaborative diffusion model. In: ECCV. Springer (2024) 3
13. Kang, K., Sim, G., Kim, G., Kim, D., Nam, S., Cho, S.: Layeringdiff: Layered image synthesis via generation, then disassembly with generative knowledge. arXiv preprint (2025) 3
14. Kirillov, A., He, K., Girshick, R., Rother, C., Dollár, P.: Panoptic segmentation. In: CVPR (2019) 10
15. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., et al.: Segment anything. In: ICCV (2023) 6

16. Li, K., Zhang, Z., Yin, H., Ye, R., Zhao, Y., Zhang, L., Ou, L., Zhang, D., Wu, X., Wu, J., et al.: Websailor-v2: Bridging the chasm to proprietary agents via synthetic data and scalable reinforcement learning. arXiv preprint (2025) 12
17. Li, Z., Guan, X., Zhang, B., Huang, S., Zhou, H., Lai, S., Yan, M., Jiang, Y., Xie, P., Huang, F., et al.: Webweaver: Structuring web-scale evidence with dynamic outlines for open-ended deep research. arXiv preprint (2025) 12
18. Liu, C., Song, Y., Wang, H., Shou, M.Z.: Omnipsd: Layered psd generation with diffusion transformer. arXiv preprint (2025) 2, 3
19. Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Li, C., Yang, J., Su, H., Zhu, J., et al.: Grounding dino: Marrying dino with grounded pre-training for open-set object detection. ECCV (2024) 5
20. Liu, Y.L., Lai, W.S., Yang, M.H., Chuang, Y.Y., Huang, J.B.: Learning to see through obstructions. In: CVPR (2020) 3
21. Pu, Y., Zhao, Y., Tang, Z., Yin, R., Ye, H., Yuan, Y., Chen, D., Bao, J., Zhang, S., Wang, Y., et al.: Art: Anonymous region transformer for variable multi-layer transparent image generation. In: CVPR (2025) 3
22. Qi, Z., Ma, M., Xu, J., Zhang, L.L., Yang, F., Yang, M.: Mutual reasoning makes smaller llms stronger problem-solvers. ICLR (2025) 4
23. Rodriguez, J.A., Agarwal, S., Laradji, I.H., Rodriguez, P., Vazquez, D., Pal, C., Pedersoli, M.: Starvector: Generating scalable vector graphics code from images. CVPR (2025) 2, 3, 8
24. Samet, H., Tamminen, M.: Efficient component labeling of images of arbitrary dimension represented by linear bintrees. IEEE TPAMI **10**(4), 579–586 (2002) 5
25. Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., Scialom, T.: Toolformer: Language models can teach themselves to use tools. NeurIPS (2023) 4
26. Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., Yao, S.: Reflexion: Language agents with verbal reinforcement learning. NeurIPS (2023) 2, 4
27. Singh, A., Fry, A., Perelman, A., Tart, A., Ganesh, A., El-Kishky, A., McLaughlin, A., Low, A., Ostrow, A., Ananthram, A., et al.: Openai gpt-5 system card. arXiv preprint (2025) 4
28. Song, Y., Chen, D., Shou, M.Z.: Layertracer: Cognitive-aligned layered svg synthesis via diffusion transformer. In: ICCV (2025) 3
29. Suvorov, R., Logacheva, E., Mashikhin, A., Remizova, A., Ashukha, A., Silvestrov, A., Kong, N., Goka, H., Park, K., Lempitsky, V.: Resolution-robust large mask inpainting with fourier convolutions. In: IEEE/CVF Winter Conference on Applications of Computer Vision (2022) 5, 6
30. Suzuki, T., Liu, K.J., Inoue, N., Yamaguchi, K.: Layerd: Decomposing raster graphic designs into layers. In: ICCV. pp. 17783–17792 (2025) 2, 3, 8, 9, 11
31. Team, G., Anil, R., Borgeaud, S., Alayrac, J.B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A.M., Hauth, A., Millican, K., et al.: Gemini: a family of highly capable multimodal models. arXiv preprint (2023) 4
32. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE TIP (2004) 9
33. Wang, Z., Zhao, H., Zhou, Q., Lu, X., Li, X., Song, Y.: Diffdecompose: Layer-wise decomposition of alpha-composited images via diffusion transformers. arXiv preprint (2025) 3
34. WhatFontIs: WhatFontIs – font identification API (2024), https://www.whatfontis.com/ 6

35. Wu, J., Li, B., Fang, R., Yin, W., Zhang, L., Tao, Z., Zhang, D., Xi, Z., Fu, G., Jiang, Y., et al.: Webdancer: Towards autonomous information seeking agency. NeurIPS (2025) 12
36. Wu, J., Yin, W., Jiang, Y., Wang, Z., Xi, Z., Fang, R., Zhang, L., He, Y., Zhou, D., Xie, P., et al.: Webwalker: Benchmarking llms in web traversal. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 10290–10305 (2025) 12
37. Yamaguchi, K.: Canvasvae: Learning to generate vector graphic documents. In: ICCV (2021) 7, 8, 11
38. Yang, Y., Cheng, W., Chen, S., Zeng, X., Yin, F., Zhang, J., Wang, L., Yu, G., Ma, X., Jiang, Y.G.: Omnisvg: A unified scalable vector graphics generation model. NeurIPS (2025) 2, 3, 8
39. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K.R., Cao, Y.: React: Synergizing reasoning and acting in language models. In: ICLR (2023) 4, 8, 11
40. Ye, M., Zhang, J., Liu, J., Liu, C., Yin, B., Liu, C., Du, B., Tao, D.: Hi-sam: Marrying segment anything model for hierarchical text segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence (2024) 5
41. Yin, S., Zhang, Z., Tang, Z., Gao, K., Xu, X., Yan, K., Li, J., Chen, Y., Chen, Y., Shum, H.Y., et al.: Qwen-image-layered: Towards inherent editability via layer decomposition. arXiv preprint (2025) 2, 3, 5, 8, 9, 10, 11
42. Zhang, L., Agrawala, M.: Transparent image layer diffusion using latent transparency. ACM TOG (2024) 3
43. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR (2018) 10
44. Zhang, X., Zhao, W., Lu, X., Chien, J.: Text2layer: Layered image generation using latent diffusion model. arXiv preprint arXiv:2307.09781 (2023) 3
45. Zhang, Z., Cheng, Y., Hong, D., Yang, M., Shi, G., Ma, L., Zhang, H., Shao, J., Wu, X.: Creatiposter: Towards editable and controllable multi-layer graphic design generation. arXiv preprint (2025) 3
46. Zhao, J., Zhou, S., Wang, Z., Yang, P., Loy, C.C.: Objectclear: Complete object removal via object-effect attention. In: CVPR (2026) 6