AUTOREF: HARNESS OPTIMIZATION FOR AGENTIC MULTI-REFERENCE IMAGE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent image generation models can take multiple reference images as input and combine them into a new image. However, multi-reference image generation remains challenging: models may omit or duplicate subjects from the references, or produce images in which multiple subjects appear unnaturally copied and pasted. Recent work has proposed image generation agents that combine image generation models, reasoning models, and a harness, which is an executable program that specifies how reference images are interpreted, how generation is performed, how outputs are diagnosed, and how the final image is selected. In multi-reference generation, however, references play different roles and outputs must satisfy many criteria at once, such as fidelity to each reference and the naturalness of the whole image, so many parts of the harness could be improved, from how references are processed to how outputs are diagnosed. This makes it hard to predict which changes will improve performance and by how much, and good harnesses difficult to design by hand; indeed, human-written harnesses vary widely in performance. We therefore propose **AutoRef**, which optimizes the harness automatically while keeping both models frozen: a coding agent iteratively rewrites the harness code. AutoRef separates the tasks whose feedback informs proposals from the tasks used to select candidates, and continues the search from a beam of the top-ranked harnesses on the selection tasks. Using this procedure, we discover **AutoRef-Harness**, which improves the open-weight FLUX.2 [klein] 4B from 5.72 to 7.37 (out of 10) on held-out four-reference tasks of the MultiBanana benchmark, matching or exceeding proprietary models including Nano Banana Pro and GPT-Image-1.5. Without re-optimization, the same harness also improves results when the generator, number of references, benchmark, evaluator, or reasoning model differs from those used in the search. We release our code and AutoRef-Harness at https://anonymous.4open.science/r/AutoRef-C506.

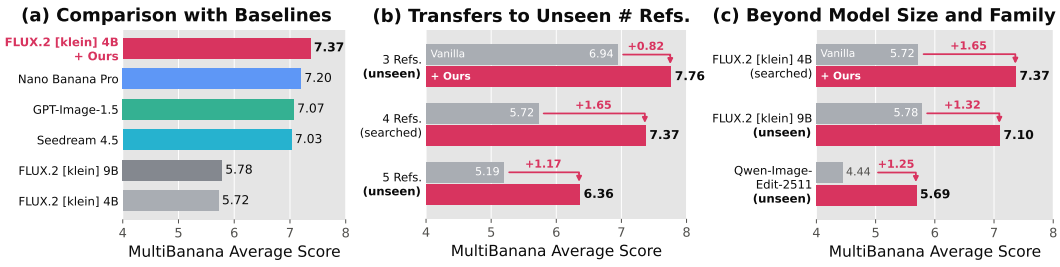

Figure 1: AutoRef-Harness (+ Ours), optimized on four-reference MultiBanana tasks with FLUX.2 [klein] 4B, (**a**) enables FLUX.2 [klein] 4B to match or exceed proprietary models on the held-out test split and transfers unchanged to (**b**) unseen three- and five-reference settings and (**c**) generators of different scales and families.

1 INTRODUCTION

Recent image generation models can take multiple reference images as input and combine them into a new image (Google DeepMind, 2025a;b; OpenAI, 2025a; Wu et al., 2025). This capability, referred to as multi-reference image generation (Wu et al., 2026a; Xia et al., 2026; Zhang et al.,

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.

2026c; Huang et al., 2026b), matters for practical image creation because users can specify people, objects, clothing, backgrounds, and styles using separate images. Such control is directly useful in applications including advertising (Inoue et al., 2023; Morita et al., 2025), virtual try-on (Zhu et al., 2023; Chong et al., 2025; Hu et al., 2026), and content creation (Ruiz et al., 2023; Xu et al., 2026). Yet combining multiple references correctly remains challenging. Models may omit or duplicate subjects from the references, or produce images in which the subjects appear pasted in rather than forming a coherent scene (Xia et al., 2026; Huang et al., 2026b).

Recent work has proposed image generation agents that combine image generation and reasoning models through a harness and iteratively plan, generate, diagnose, and refine (Hao et al., 2023; Yang et al., 2024b; Ma et al., 2025; He et al., 2026b). The harness is executable code that specifies how the frozen models are used: how references are interpreted, prompts are constructed, candidates are generated and evaluated, and the output is selected. In multi-reference image generation, however, a good harness is harder to design than in text-to-image generation: references play different roles (e.g., identity, background, or style), and outputs must satisfy many criteria, including fidelity to each reference and the naturalness of the whole image (Oshima et al., 2026; Huang et al., 2026b). Many parts of the harness could therefore be improved, from which references to provide and in what order to how outputs are checked against each one, yet the effect of each change is hard to predict. Indeed, existing human-written harnesses vary widely in performance (Section 6.4).

Recent work on automatic agent optimization has expanded from prompts and workflows to executable code (Lee et al., 2026; Zhang et al., 2026b; Miyai et al., 2026). These methods, however, have largely been developed for tasks with verifiable rewards such as math (Lee et al., 2026) and coding (Lin et al., 2026; Zhang et al., 2026a), whereas image generation relies on noisy visual evaluation and provides little diagnostic information through scalar scores alone. We therefore propose **AutoRef**, which optimizes harness code while keeping the image generation and reasoning models frozen. To address these challenges, AutoRef separates the tasks used to propose harness updates from those used to select candidates, so that selection does not reuse the examples the proposer sees. It also runs an iterative beam search, in which the top-ranked harnesses on the selection tasks, rather than a harness chosen by the proposer, become the next parents.

AutoRef-Harness, the optimized harness for multi-reference image generation, was discovered by AutoRef on the MultiBanana benchmark (Oshima et al., 2026) using the open-weight FLUX.2 [klein] 4B (Black Forest Labs, 2026). It uses reference-grounded prompting, generates structurally diverse drafts, revises the better draft from explicit complaints, and selects among candidates with failure-aware comparisons. With AutoRef-Harness, FLUX.2 [klein] 4B improves from 5.72 to 7.37 on the four-reference MultiBanana held-out test split, matching or exceeding proprietary models including Nano Banana Pro (Google DeepMind, 2025b) and GPT-Image-1.5 (OpenAI, 2025b) (Figure 1). The same harness also improves results without re-optimization when the generator, number of references, benchmark, evaluator, or reasoning model is changed. We release our code and AutoRef-Harness.

2 Related Work

Multi-Reference Image Generation. Reference-conditioned image generation has evolved from personalized adaptation to specific subjects, as in DreamBooth (Ruiz et al., 2023), toward general-purpose multimodal generation that incorporates multiple reference images. Recent models support flexible multi-reference image generation and editing (Deng et al., 2025; Xia et al., 2026; Wu et al., 2026a; Black Forest Labs, 2026; Wu et al., 2025; Google DeepMind, 2025a;b; 2026; OpenAI, 2025a;b). In parallel, recent work has improved multi-reference image generation by scaling reference-conditioned training data and fine-tuning the underlying models (Zhang et al., 2026c; Huang et al., 2026b). In contrast, MultiBanana (Oshima et al., 2026) shows that simple agentic refinement gives only limited gains on multi-reference tasks, indicating that simply wrapping a strong generator with a fixed agent workflow is insufficient. We therefore optimize the agent harness automatically from task feedback, improving multi-reference generation while keeping the generator frozen.

Automatic Optimization of Agentic Systems. Automatic agent optimization searches over prompts (Zhou et al., 2023; Yang et al., 2024a; Pryzant et al., 2023; Guo et al., 2024), modular pipelines (Khattab et al., 2024; Opsahl-Ong et al., 2024), and agent workflows (Zhuge et al., 2024; Hu et al., 2025; Zhang et al., 2025). Language-based feedback guides revisions (Yuksekgonul et al., 2025; Agrawal et al., 2026), while program evolution extends optimization to executable code and

self-improving agents (Novikov et al., 2025; Lange et al., 2026; Zelikman et al., 2024; Robeyns et al., 2025; Zhang et al., 2026b). Pryzant et al. (2023) and Guo et al. (2024) keep multiple candidates across iterations, and Agrawal et al. (2026) and Khattab et al. (2024) can select them on held-out examples; all of them tune prompts within a fixed program. Meta-Harness (Lee et al., 2026) lets a coding agent read the code, scores, and execution trajectories of prior candidates, choose which one to build on, and revise the harness around a frozen model, scoring candidates on the same tasks that supply this feedback. AutoDesign (Luo et al., 2026) applies harness optimization to academic paper-to-poster generation. Appendix B discusses inference-time scaling for multimodal generation.

3 PRELIMINARIES

Multi-Reference Image Generation. Let $x = (u, \mathcal{I})$ denote a multi-reference image generation task (Wu et al., 2026a; Xia et al., 2026; Huang et al., 2026b), where u is the user prompt (the instruction) and $\mathcal{I} = \{I_1, \ldots, I_m\}$ is the set of reference images. Let M_θ and G_ϕ denote a frozen reasoning model and image generator, respectively. A harness H is an executable program that specifies how these models are used: how references are interpreted, prompts constructed, candidates generated and evaluated, and the output selected. Running the harness yields an image y and an execution trajectory τ:

$$(y, \tau) \sim H(M_\theta, G_\phi, x). \tag{1}$$

Harness Optimization. Our goal is to optimize the harness while keeping the model parameters θ and ϕ frozen (Zhang et al., 2026b; Lin et al., 2026; Miyai et al., 2026). Let p_{task} denote the distribution of multi-reference image generation tasks, and let $R(y, x)$ denote an evaluator that scores the quality of a generated image y for task x. We define the performance of a harness as

$$J(H) = \mathbb{E}_{x \sim p_{\text{task}},\, y \sim H(M_\theta, G_\phi, x)} \left[R(y, x) \right]. \tag{2}$$

The harness optimization objective is therefore

$$H^\star = \arg\max_H J(H). \tag{3}$$

With M_θ and G_ϕ frozen, optimization acts only on the executable code surrounding the models, which allows changes to prompting, generation, evaluation, selection, and control flow.

Harness Search Loop. We approach this optimization problem through iterative code improvement (Zhang et al., 2026b; Lee et al., 2026; Lin et al., 2026). At iteration t, a coding-agent proposer P has access to the current harness H_t and an accumulated search history $\mathcal{L}_t$ of artifacts from previous iterations: harness implementations, evaluation scores, and execution trajectories. The proposer can selectively inspect and search prior artifacts, diagnose failure modes, and decide how to modify the harness. It then proposes an updated harness:

$$H_{t+1} \leftarrow P(H_t, \mathcal{L}_t). \tag{4}$$

The proposed harness is evaluated on a set of search tasks, and its implementation, scores, and trajectories are added to the history $\mathcal{L}_{t+1}$.

4 AUTOREF

We propose **AutoRef**, a method for automatically optimizing harnesses for multi-reference image generation. Existing harness optimization methods (Zhang et al., 2026b; Lee et al., 2026; Miyai et al., 2026) primarily target tasks whose performance can be verified using discrete labels or executable tests. In image generation, however, a visual evaluator must estimate quality; failures are often hard to diagnose from scalar rewards alone, and repeated optimization over a limited set of evaluated examples can overfit to both the search tasks and the evaluator. AutoRef addresses these challenges by (1) separating the tasks used for harness updates from those used for candidate selection, and (2) using beam search that keeps the top-B candidates on the validation tasks as parents for the next iteration. These choices adapt harness optimization to perceptual, non-verifiable image generation tasks. Figure 2 illustrates one iteration; Algorithm 1 (Appendix C) gives the full procedure.

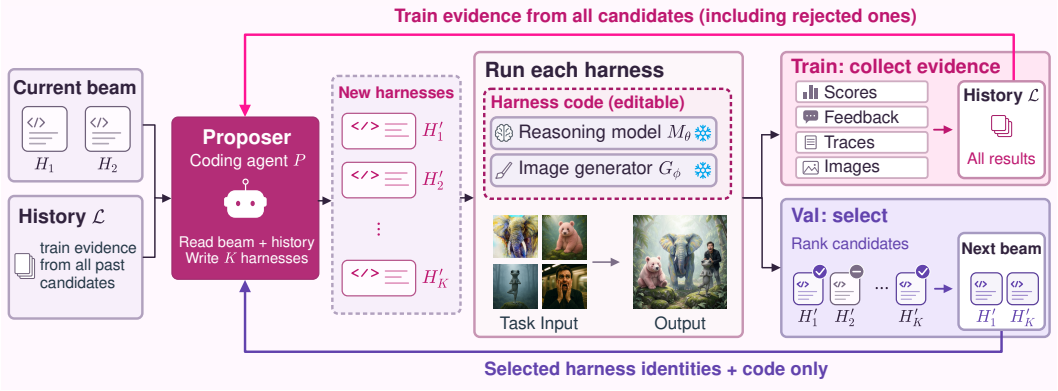

Figure 2: **AutoRef**, one iteration. A harness is a program that calls a frozen reasoning model M_θ and a frozen image generator G_ϕ; the search rewrites this program. The proposer, a coding agent P, reads the current beam and the search history $\mathcal{L}$ and writes $K = 4$ new harnesses as code. On D_{train} (pink), everything from every candidate, rejected ones included — scores, evaluator rationales, execution trajectories, and generated images — enters $\mathcal{L}$, where P may inspect it. D_{val} (purple) only ranks the four and keeps $B = 2$; P is told which two survived but never their scores. The survivors form the next beam; their parents do not compete again.

Task Separation for Proposal and Selection. Directly optimizing against rich but non-verifiable evaluation feedback risks overfitting the harness to both a small set of search tasks and noise in the evaluator (Huang et al., 2026a; Luo et al., 2026). We therefore separate the tasks used to propose harness updates from those used to select among them. We split the search tasks into disjoint sets D_{train} and D_{val}, and write $J_D(H)$ for the mean of $R(y, x)$ over $x \in D$. Evaluations on D_{train} provide feedback for harness improvement: scores, evaluator rationales, execution trajectories, and visual artifacts are added to the search history $\mathcal{L}$ and may be inspected by the proposer. In contrast, D_{val} is used only for candidate selection, and its scores and artifacts are never exposed to the proposer or added to $\mathcal{L}$. Thus, the proposer constructs new harnesses using only training-side feedback, while $J_{D_{\text{val}}}$ selects among them without becoming a direct optimization signal.

Iterative Beam Search. Selecting a single harness at each iteration can commit the search to a lineage favored by stochastic generation or noisy visual evaluation. We therefore maintain a beam of B harnesses. At iteration t, the proposer uses the current beam and accumulated search history to generate K candidate harnesses. Each candidate is evaluated on both D_{train} and D_{val}, and the next beam is formed by the B candidates with the highest $J_{D_{\text{val}}}$. The proposer is told which candidates were selected but not their validation scores, while training-side evidence from all candidates, including unselected ones and their generated images, is preserved in $\mathcal{L}$ for subsequent iterations. In our experiments, we use $B = 2$ and $K = 4$, and initialize the beam with two harnesses: the base generator H_{vanilla} (the generator called once on the user prompt) and GEMS (He et al., 2026b) as H_{init}. In the first iteration, the proposer writes all K candidates from the two initial harnesses; in each later iteration, it writes two candidates from each beam member. The search that produced AutoRef-Harness is traced in Appendix D.

5 The Optimized AutoRef-Harness

AutoRef-Harness is the harness returned by AutoRef (Section 4). It draws three images from G_ϕ and makes all other decisions with M_θ: it generates drafts A and B from two differently structured prompts, keeps the better one, generates draft C from complaints about the winner, and returns the better of the winner and C (Appendix F). Compared with human-written harnesses, it differs in how each step is specialized for multiple references and how the steps are chained: every prompt assigns each requested element to its reference (§5.1); the two drafts differ in prompt structure, not only in sampling (§5.2); complaints name the reference they concern (§5.3); and selection counts hard failures (e.g., a missing reference) before pairwise judgment, and a later draft replaces the incumbent only if it wins under this rule (§5.4). Each component was added in an iteration that raised the validation score, and alternatives like editing the winner in place were dropped (Appendix D).

5.1 Reference-Grounded Prompting

References play different roles (identity, garment, attribute, background, style); a generator that confuses them leaks attributes or drops references. AutoRef-Harness never passes the raw instruction to the generator: M_θ reads the instruction and all references and writes a prompt that assigns each requested element to its reference, excluding unrequested content; later prompts use the same format.

5.2 Structurally Diverse Drafts

A common failure is a pasted-in look: each subject matches its reference, but its lighting, perspective, or colors disagree with the scene. Resampling one prompt rarely fixes this, so the two drafts use different prompt structures. Draft A describes the scene subject by subject. For draft B, M_θ identifies the reference designated as the background or style, and the prompt asks the generator to keep that reference as the canvas and paint the other subjects into it, so that subjects and scene are rendered jointly. If no such reference exists, draft B is a second sample of draft A's prompt.

5.3 Complaint-Directed Revision

M_θ lists up to five concrete complaints about the winner of A and B, each naming the reference it concerns (e.g., wrong identity, attribute from the wrong reference, inconsistent lighting), and rewrites the prompt to address them; draft C is generated from the revised prompt, or by resampling the winner's prompt if there is no complaint.

5.4 Failure-Aware Selection

Candidates are compared in pairs. Each draft is first checked for hard failures (missing reference, extra or duplicated subject, wrong background); the draft with fewer failures wins. On a tie, M_θ lists the differences a strict rater would score and names a winner in both presentation orders; the challenger (draft B, then draft C) must win both. Selection uses M_θ (GPT-5.5), not the evaluator R.

6 Experiments

6.1 Experimental Settings

Benchmarks. We evaluate on MultiBanana (Oshima et al., 2026), a benchmark for multi-reference image generation. We use the 229 tasks with four reference images, split into 48 training tasks, 48 validation tasks, and 133 test tasks, with Qwen3-VL-8B-Instruct (Bai et al., 2025) as the evaluator. We use the training and validation splits for harness optimization, while the held-out test split remains unseen during search. To test generalization across unseen reference counts, we further evaluate on the three- and five-reference settings, randomly sampling 24 tasks per task type (96 per setting).

To evaluate generalization beyond the benchmark and evaluator, we also test on OmniContext (Wu et al., 2026a), which we never use during harness search. We randomly sample 15 tasks from each task type, for 120 tasks in total, and evaluate them using the official GPT-4.1 (OpenAI, 2023) evaluator. With both the benchmark and the evaluator differing from those used in the search, this setting tests whether the learned harness transfers to unseen data distributions and evaluation signals.

Harness Search. We initialize the search with two harnesses: the base FLUX.2 [klein] 4B generator (Black Forest Labs, 2026) and GEMS (He et al., 2026b), an image-generation harness configured with FLUX.2 [klein] 4B as the generator and GPT-5.5 (OpenAI, 2026) as the reasoning model. For AutoRef (Section 4), we use Claude Fable 5.1 as the proposer through the Claude Code CLI (Anthropic, 2025) and run five search iterations. Implementation details and model versions are in Appendix A, and the proposer's prompts are in Appendix E.

Baselines. We compare against a broad set of baselines: proprietary image models including GPT-Image-1.5 (OpenAI, 2025b), Nano Banana Pro (Google DeepMind, 2025b), and Seedream 4.5 (ByteDance Seed, 2025), open image models including OmniGen2 (Wu et al., 2026a), DreamOmni2 (Xia et al., 2026), BAGEL (Deng et al., 2025), FLUX.2 [klein] 4B and 9B (Black Forest Labs, 2026), and Qwen-Image-Edit-2511 (Wu et al., 2025), and agentic or search-based methods

Table 1: MultiBanana (Oshima et al., 2026) with 4 references, held-out test split. Cells are the mean of the five evaluation metrics (1–10) per task type and Avg. the mean of the four. *Gen.* is images drawn per task, a measured mean over the split rather than a nominal maximum. Best open image generators per column in bold, second best underlined. †: AutoRef-Harness discovered with FLUX.2 [klein] 4B on the four-reference MultiBanana setting.

Method	Gen.	Object	Local	Global	Background	**Avg.**
Proprietary Models						
GPT-Image-1.5	1	6.84	8.12	7.22	6.11	7.07
Nano Banana Pro	1	6.73	7.98	7.70	6.39	7.20
Seedream 4.5	1	6.65	7.54	7.83	6.10	7.03
Open Models						
OmniGen2	1	3.56	4.35	3.64	3.43	3.75
DreamOmni2	1	3.38	5.17	3.51	3.28	3.83
BAGEL	1	2.64	4.16	2.76	3.04	3.15
FLUX.2 [klein] 4B	1	5.95	6.11	5.50	5.34	5.72
+ **AutoRef-Harness**	3	<u>7.27</u>	**7.87**	**7.64**	**6.70**	**7.37**
FLUX.2 [klein] 9B	1	6.80	5.64	5.58	5.13	5.78
+ **AutoRef-Harness**†	3	**7.49**	<u>7.30</u>	<u>7.22</u>	<u>6.37</u>	<u>7.10</u>
Qwen-Image-Edit-2511	1	4.05	5.05	4.43	4.23	4.44
+ **AutoRef-Harness**†	3	5.50	6.07	5.79	5.41	5.69

Table 2: MultiBanana with 3 and 5 references. AutoRef-Harness largely preserves its performance gains when transferred to unseen reference counts. Best open image generators per column in bold, second best underlined. †: AutoRef-Harness discovered with FLUX.2 [klein] 4B on the four-reference MultiBanana setting.

		3 references					5 references				
Method	Gen.	Object	Local	Global	Backg.	**Avg.**	Object	Local	Global	Backg.	**Avg.**
Proprietary Models											
GPT-Image-1.5	1	8.42	8.14	7.84	7.03	7.86	5.68	8.48	6.20	6.02	6.60
Nano Banana Pro	1	8.27	8.09	7.62	6.03	7.50	5.53	8.41	7.03	5.88	6.71
Seedream 4.5	1	6.88	8.43	7.68	6.67	7.41	6.02	8.16	6.19	6.04	6.60
Open Models											
OmniGen2	1	4.74	5.78	5.62	4.87	5.25	2.68	4.37	2.73	2.93	3.18
DreamOmni2	1	5.31	6.26	5.20	4.68	5.36	2.01	4.88	3.17	2.42	3.12
BAGEL	1	4.47	4.50	3.69	3.72	4.10	2.20	4.16	2.83	2.08	2.82
FLUX.2 [klein] 4B	1	6.26	7.90	6.83	6.78	6.94	4.04	6.32	5.72	4.70	5.19
+ **AutoRef-Harness**†	3	<u>7.15</u>	**8.68**	<u>7.82</u>	**7.41**	<u>7.76</u>	<u>5.67</u>	7.37	**6.75**	<u>5.65</u>	<u>6.36</u>
FLUX.2 [klein] 9B	1	6.89	7.36	6.69	6.36	6.83	4.84	<u>7.49</u>	5.64	4.45	5.61
+ **AutoRef-Harness**†	3	**8.04**	<u>8.29</u>	**7.87**	<u>7.09</u>	**7.82**	**6.16**	**7.67**	<u>6.12</u>	**5.92**	**6.47**
Qwen-Image-Edit-2511	1	4.56	5.13	5.53	5.03	5.06	1.35	1.83	2.56	1.80	1.89
+ **AutoRef-Harness**†	3	6.87	7.86	6.67	6.39	6.95	1.57	1.69	2.30	1.93	1.87

including Best-of-N (Ma et al., 2025), GEMS (He et al., 2026b), IPR (Oshima et al., 2026), and Idea2Img (Yang et al., 2024b). We also report the harness that Meta-Harness (Lee et al., 2026) converges to under the same budget, generator, reasoning model, and evaluator, so the search algorithm is the only difference between it and AutoRef-Harness.

6.2 Main Results

As shown in Table 1, AutoRef-Harness improves the performance of FLUX.2 [klein] 4B on the four-reference MultiBanana held-out test split. Despite using the relatively small FLUX.2 [klein] 4B as its image generator, the resulting system outperforms all evaluated open models and achieves performance competitive with proprietary models such as GPT-Image-1.5, Nano Banana Pro, and Seedream 4.5. The gains from AutoRef-Harness also transfer beyond the model used during harness optimization: applying the same harness to the larger FLUX.2 [klein] 9B improves its performance, and replacing FLUX.2 with Qwen-Image-Edit-2511 likewise yields a substantial gain. These results indicate that the benefit of the discovered harness is not limited to a particular model scale or generator family. Importantly, we achieve these improvements without updating the image generator

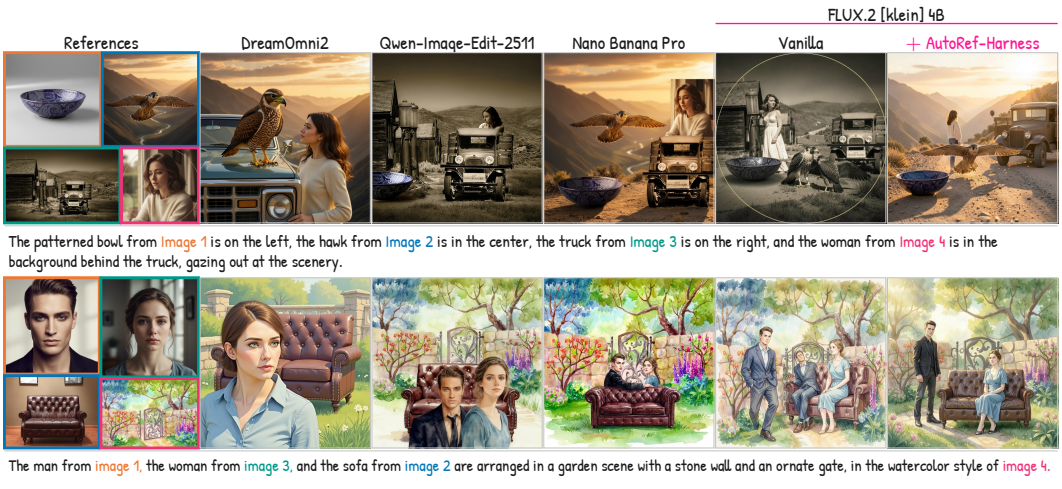

Figure 3: Qualitative comparison on held-out four-reference MultiBanana tasks. Each reference is outlined in a distinct color, and the phrase in the instruction that refers to it is shown in the same color. Baseline models often omit, duplicate, or misplace references, or copy and paste them unnaturally, whereas AutoRef-Harness preserves each reference and integrates all four into a coherent scene.

or reasoning model; we only change the inference-time harness. Per-metric results are reported in Appendix H.1, and the harness further improves Qwen-Image-Edit-2511 after fine-tuning for multi-reference image generation with DyRef (Huang et al., 2026b; Appendix H.3).

Figure 3 illustrates qualitative examples. Baseline models often omit, duplicate, or misplace references, or paste them in unnaturally. For instance, the base FLUX.2 [klein] 4B duplicates the hawk and places the woman in the foreground rather than the background in the first example, and duplicates the man in the second. Nano Banana Pro and Qwen-Image-Edit-2511 instead produce copy-and-paste-like results in the first and second examples, respectively. AutoRef-Harness preserves each reference and naturally integrates it into the requested scene. Appendix J shows more examples.

6.3 Transferability of AutoRef-Harness

Across Reference Counts. AutoRef-Harness is discovered on four-reference MultiBanana tasks but applies to different reference counts without modification. As shown in Table 2, it consistently improves FLUX.2 [klein] 4B on both the unseen three- and five-reference settings. With three references, the resulting system outperforms all open image generators without the harness and remains competitive with proprietary models; the gain also persists in the more challenging five-reference setting. The harness likewise improves the other generators, except Qwen-Image-Edit-2511 with five references, where the generator itself fails and the harness cannot compensate (Appendix I). AutoRef-Harness thus does not rely on the four-reference structure used during harness discovery.

Across Benchmarks and Evaluators. We further evaluate the same AutoRef-Harness on OmniContext, which is never used during harness discovery and is scored by its official GPT-4.1 (OpenAI, 2023) evaluator rather than the Qwen3-VL-8B-Instruct evaluator used for MultiBanana. As shown in Table 3, AutoRef-Harness again improves FLUX.2 [klein] 4B and remains competitive with proprietary image models. Thus, the gains persist under simultaneous changes in both the benchmark distribution and the evaluator, without benchmark-specific modifications to the harness.

Across Reasoning Models. As shown in Section 6.2, AutoRef-Harness transfers across image generators of different scales and families. We next test whether the same harness also transfers across reasoning models. With the open-weight Qwen3-VL-32B (Bai et al., 2025) in place of GPT-5.5 and the harness structure unchanged, AutoRef-Harness improves FLUX.2 [klein] 4B from 5.72 to 6.90 on the four-reference MultiBanana held-out test split (7.37 with GPT-5.5; Figure 4, **Left**). This suggests that the orchestration strategy encoded by AutoRef-Harness is not specific to the proprietary reasoning model used during its discovery.

Table 3: OmniContext (Wu et al., 2026a), unseen during the search and scored by its GPT-4.1 evaluator. Cells are the geometric mean of prompt following and subject consistency (0–10) per task type, and Avg. the mean of the eight; Char.: character, Obj.: object, C.+O.: character and object. Best open image generators per column in bold, second best underlined. †: AutoRef-Harness discovered with FLUX.2 [klein] 4B on the four-reference MultiBanana setting.

Method	Gen.	SINGLE		MULTIPLE			SCENE			**Avg.**
		Char.	Obj.	Char.	Obj.	C.+O.	Char.	Obj.	C.+O.	
Proprietary Models										
GPT-Image-1.5	1	9.56	9.70	9.32	9.46	9.26	9.69	9.39	8.93	9.41
Nano Banana Pro	1	9.63	9.42	9.46	9.19	9.02	9.35	8.39	8.20	9.08
Seedream 4.5	1	9.45	9.50	9.09	9.39	9.09	9.35	8.66	8.23	9.09
Open Models										
OmniGen2	1	8.51	5.73	6.30	6.58	7.71	6.93	6.10	7.03	6.86
DreamOmni2	1	7.81	6.72	4.80	7.07	5.92	5.78	5.63	5.72	6.18
BAGEL	1	6.67	7.09	3.43	6.74	7.08	3.97	4.11	5.63	5.59
FLUX.2 [klein] 4B	1	**9.22**	8.16	7.91	8.21	8.68	9.18	7.47	7.54	8.30
+ **AutoRef-Harness**†	3	8.94	8.99	9.02	8.78	8.59	<u>9.28</u>	**8.89**	<u>8.28</u>	<u>8.85</u>
FLUX.2 [klein] 9B	1	<u>9.17</u>	9.08	8.66	8.14	<u>8.74</u>	**9.35**	8.07	7.47	8.59
+ **AutoRef-Harness**†	3	**9.22**	<u>9.18</u>	**9.19**	**9.32**	**8.82**	**9.35**	8.40	**8.30**	**8.97**
Qwen-Image-Edit-2511	1	9.14	**9.19**	8.66	<u>9.00</u>	8.33	6.97	8.17	8.17	8.45
+ **AutoRef-Harness**†	3	9.12	8.64	<u>9.09</u>	8.62	8.34	8.71	<u>8.82</u>	<u>8.28</u>	8.70

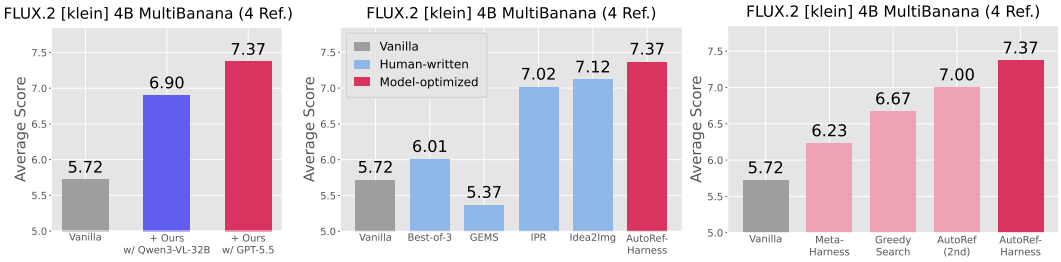

Figure 4: (**Left**) Reasoning-model transfer. AutoRef-Harness (+ Ours) improves FLUX.2 [klein] 4B with both GPT-5.5 and the open-weight Qwen3-VL-32B. (**Middle**) Comparison with human-written harnesses. AutoRef-Harness outperforms Best-of-3, GEMS, IPR, and Idea2Img. (**Right**) Comparison of harness optimization methods. AutoRef discovers stronger harnesses than Meta-Harness and Greedy Search; even its second-best discovered harness outperforms the final harnesses produced by both alternative search procedures.

6.4 Comparison with Harnesses and Harness Search

Comparison with Human-Written Harnesses. We compare AutoRef-Harness with human-written harnesses: Best-of-3 (Ma et al., 2025), GEMS (He et al., 2026b), IPR (Oshima et al., 2026), and Idea2Img (Yang et al., 2024b), which use 3, 2.7, 3, and 9 image generations per task, respectively (details in Appendix G.1). AutoRef-Harness achieves the highest performance on the four-reference MultiBanana held-out test split (Figure 4, **Middle**). Notably, Idea2Img still underperforms despite using three times AutoRef-Harness's generation budget.

Comparison with Harness Optimization Methods. We next compare AutoRef with alternative harness optimization methods (details in Appendix G.2). Meta-Harness (Lee et al., 2026) uses the same tasks for optimization feedback and candidate ranking, and its proposer chooses which candidate to build on from the full search history. Greedy Search adopts AutoRef's train–validation separation but retains only the single best harness per iteration, whereas AutoRef keeps the top-B candidates on D_{val} as parents for the next iteration. Under the same generator, reasoning model, and evaluator, AutoRef yields the strongest final harness on the held-out test split, followed by Greedy Search and Meta-Harness (Figure 4, **Right**). The final harnesses of Meta-Harness and Greedy Search also draw more images per task than AutoRef-Harness (4.2 and 5 vs. 3). Moreover, even AutoRef's second-best harness outperforms both methods' final harnesses. The successive gains from Meta-Harness to

Greedy Search to AutoRef support the value of both train–validation separation and a multi-parent beam when optimizing image-generation harnesses from perceptual evaluation.

6.5 Ablation Study

We ablate one component at a time, replacing a removed draft or revision with another sample from the same prompt and failure-aware selection with the Best-of-N selector (Appendix G.1). Reference-grounded prompting cannot be ablated this way, since all drafts and the revision use grounded prompts; we instead evaluate it alone with a single image (grounding only) and the harness without it (selection only). Table 4 shows that each of the four components contributes: removing structurally diverse drafts, complaint-directed revision, or failure-aware selection lowers the average score, and both grounding only and selection only improve over the generator alone. Grounding only (6.91, one image) approaches IPR (7.02, three images), which also rewrites the prompt from the references, and selection only (6.15) is close to Best-of-3 (6.01), which differs only in its selector; yet the same selector adds 0.44 within the full harness (7.37 vs. 6.93), and no partial combination matches the full harness.

Table 4: Component ablation on the held-out test split. Columns are the components of §5.1–5.4; parentheses give the change from the full harness.

	§5.1	§5.2	§5.3	§5.4	Avg.
Full harness	✓	✓	✓	✓	**7.37**
Leave one out	✓	✗	✓	✓	7.01 (−0.36)
	✓	✓	✗	✓	6.99 (−0.38)
	✓	✓	✓	✗	6.93 (−0.44)
Grounding only	✓	✗	✗	✗	6.91 (−0.46)
Selection only	✗	✗	✗	✓	6.15 (−1.22)
Generator only	✗	✗	✗	✗	5.72 (−1.65)

6.6 Human Evaluation

AutoRef optimizes against an automatic evaluator, so we test whether its gains hold for human raters. Four raters compared FLUX.2 [klein] 4B + AutoRef-Harness with each of four baselines on 50 tasks each from the four-reference held-out test split. For each task, raters see the references, the instruction, and the two outputs in random order, and select the better output, or a tie only if they cannot distinguish the two. As shown in Figure 5, AutoRef-Harness achieves a 70% win rate against its base generator, FLUX.2 [klein] 4B (17% loss), wins more often than it loses against FLUX.2 [klein] 9B (64% vs. 24%) and Seedream 4.5 (63% vs. 27%), and is competitive with Nano Banana Pro (46% vs. 40%).

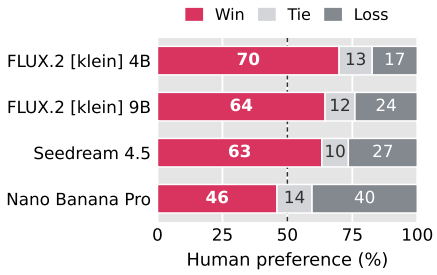

Figure 5: Win, tie, and loss rates (%) of AutoRef-Harness (FLUX.2 [klein] 4B) against each baseline in pairwise human evaluation.

7 Discussion and Limitations

AutoRef-Harness changes only how a frozen generator is used, so it cannot exceed what the generator can produce: when no draft is acceptable, as for Qwen-Image-Edit-2511 at five references, better selection does not help (Appendix I). The search maximizes a single VLM evaluator's score, yet the harness transfers to OmniContext and its GPT-4.1 evaluator. As the evaluator and the proposer are replaceable, AutoRef may benefit from stronger VLMs and coding agents, and richer evaluators could be explored, e.g., combining a VLM with segmentation models (Carion et al., 2026).

8 Conclusion

We introduced AutoRef, a harness optimization method for multi-reference image generation that keeps the image generator and the reasoning model frozen and changes only the program that combines them. AutoRef separates the tasks that inform proposals from those used to select among them, and continues from the top-ranked harnesses on the selection tasks through iterative beam search. The harness it discovers, AutoRef-Harness, makes the open-weight FLUX.2 [klein] 4B competitive with proprietary models on MultiBanana, transfers to unseen reference counts, an unseen benchmark and evaluator, other generators, and another reasoning model, and adds to the gains of fine-tuning. These results suggest that how frozen models are used is itself worth optimizing.

AI USE STATEMENT

In this work, we used generative AI tools for the following tasks: *design or provide feedback on research methodology or experiments, implement methods, assist with translation, and support qualitative and thematic data analysis*. We have not used generative AI tools for the following tasks: *help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, clean and reformat datasets, interpret results*, and the tasks *generate synthetic data sets* and *assist in the writing of proofs* are not applicable to this work. Additionally, we used generative AI tools for the following tasks: *create or modify scientific figures or images, create or edit software code, draft parts of a research paper, and edit a research paper to improve readability*. We have reviewed all AI-assisted work. Two authors verified and tested LLM-generated code and writing for correctness. We take responsibility for the final content, including text, claims, or artifacts produced with generative AI.

ETHICS STATEMENT

Our method optimizes only the agent harness and does not update the underlying reasoning or image generation models. Thus, it does not introduce new behaviors through model-weight modification, although it can change how existing model capabilities are composed. As with image generation more broadly, improved multi-reference image generation may be misused to create misleading or unauthorized synthetic content. The harness also inherits limitations and biases from the underlying models and evaluators. We therefore encourage responsible use of the released code and harness.

REPRODUCIBILITY STATEMENT

We release the AutoRef implementation and AutoRef-Harness at an anonymized URL (https://anonymous.4open.science/r/AutoRef-C506), together with the data splits and the pinned model revisions. We conduct all evaluations on publicly available benchmarks. The main paper describes the harness optimization procedure, model configurations, data splits, and evaluation protocol, while the Appendix provides additional evaluation details and complete results.

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, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. Gepa: Reflective prompt evolution can outperform reinforcement learning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 8479–8565, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/0e9e708b6f48e14fd0ac29e167413f76-Paper-Conference.pdf.

Anthropic. Claude code. https://claude.com/product/claude-code, 2025. Accessed: 2026-09-18.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Black Forest Labs. FLUX.2 [klein]. https://bfl.ai/models/flux-2-klein, 2026. Accessed: 2026-09-24.

ByteDance Seed. Seedream 4.5. https://seed.bytedance.com/en/seedream4_5, 2025. Accessed: 2026-09-23.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, RISHI HAZRA, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollar, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. SAM 3: Segment anything with concepts. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=r35clVtGzw.

Sixiang Chen, Zhaohu Xing, Tian Ye, Xinyu Geng, Yunlong Lin, Jianyu Lai, Xuanhua He, Fuxiang Zhai, Jialin Gao, and Lei Zhu. Genevolve: Self-evolving image generation agents via tool-orchestrated visual experience distillation. *arXiv preprint arXiv:2605.21605*, 2026.

Zheng Chong, Xiao Dong, Haoxiang Li, shiyue Zhang, Wenqing Zhang, Hanqing Zhao, xujie zhang, Dongmei Jiang, and Xiaodan Liang. CatVTON: Concatenation is all you need for virtual try-on with diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=jt1h2dnmng.

Siddhartha Datta, Alexander Ku, Deepak Ramachandran, and Peter Anderson. Prompt expansion for adaptive text-to-image generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3449–3476, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.189. URL https://aclanthology.org/2024.acl-long.189/.

Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, Guang Shi, and Haoqi Fan. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025.

Hiroki Furuta, Heiga Zen, Dale Schuurmans, Aleksandra Faust, Yutaka Matsuo, Percy Liang, and Sherry Yang. Improving dynamic object interactions in text-to-video generation with ai feedback. *arXiv preprint arXiv:2412.02617*, 2024.

Google DeepMind. Nano banana: Gemini 2.5 flash image model. https://developers.googleblog.com/en/introducing-gemini-2-5-flash-image/, 2025a. Accessed: 2025-10-31.

Google DeepMind. Nano banana pro. https://deepmind.google/models/gemini-image/pro/, 2025b. Accessed: 2025-11-27.

Google DeepMind. Nano banana 2: Combining pro capabilities with lightning-fast speed. https://blog.google/innovation-and-ai/technology/ai/nano-banana-2/, 2026. Accessed: 2026-4-30.

Rogerio Guimaraes and Pietro Perona. Inference-time scaling of diffusion models via progressive seed pruning. *arXiv preprint arxiv:2607.21591*, 2026.

Litao Guo, Xinli Xu, Luozhou Wang, Jiantao Lin, Jinsong Zhou, Zixin Zhang, Bolan Su, and Yingcong Chen. Comfymind: Toward general-purpose generation via tree-based planning and reactive feedback. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 45128–45164. Curran Associates, Inc., 2025. doi: 10.52202/085713-1503. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/40168e00bf87869c5d153e934d8a3602-Paper-Conference.pdf.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 34133–34156, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/9156b0f6dfa9bbd18c79cc459ef5d61c-Paper-Conference.pdf.

Yaru Hao, Zewen Chi, Li Dong, and Furu Wei. Optimizing prompts for text-to-image generation. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 66923–66939. Curran Associates, Inc., 2023. doi: 10.52202/075280-2923. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/d346d91999074dd8d6073d4c3b13733b-Paper-Conference.pdf.

Haoran He, Jiajun Liang, Xintao Wang, Pengfei Wan, Kun Gai, and Ling Pan. Scaling image and video generation via test-time evolutionary search, 2026a. URL https://openreview.net/forum?id=CFlOUNWsaP.

Zefeng He, Siyuan Huang, Xiaoye Qu, Yafu Li, Tong Zhu, Yu Cheng, and Yang Yang. GEMS: Agent-native multimodal generation with memory and skills. *arXiv preprint arXiv:2603.28088*, 2026b.

Junyao Hu, Zhongwei Cheng, Waikeung Wong, and Xingxing Zou. Garments2look: A multi-reference dataset for high-fidelity outfit-level virtual try-on with clothing and accessories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 21344–21377, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/36b7acf6f6010652b3f2a433774a66fe-Paper-Conference.pdf.

Chengsong Huang, Zifeng Wang, Rujun Han, Jun Yan, Yanfei Chen, Zoey CuiZhu, Ke Jiang, Peng Xia, Han Yu, Yufan Zhuang, Yifei Ming, Jiaqi Pan, Bhavana Dalvi Mishra, Jiaxin Huang, Burak Gokturk, Tomas Pfister, and Chen-Yu Lee. Envharness: Awakening static worlds for agent learning, 2026a. URL https://arxiv.org/abs/2608.19880.

Wenwang Huang, Yusen Fu, Junjie Wang, Mengfei Huang, Yulin Li, Gan Liu, Jing Cai, Yancheng He, and Zhuotao Tian. Scaling multi-reference image generation with dynamic reward optimization. In *The 19th European Conference on Computer Vision*, 2026b.

Naoto Inoue, Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. LayoutDM: Discrete Diffusion Model for Controllable Layout Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10167–10176, 2023.

Kaixun Jiang, Yuzheng Wang, Junjie Zhou, Pandeng Li, Zhihang Liu, Chen-Wei Xie, Zhaoyu Chen, Yun Zheng, and Wenqiang Zhang. Genagent: Scaling text-to-image generation via agentic multimodal reasoning. *The 19th European Conference on Computer Vision*, 2026.

Jaemin Jung, Kyeongha Rho, Inkyu Shin, and Joon Son Chung. Inference-time scaling for joint audio-video generation. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=MHNFjjm5nO.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan A, Saiful Haq, Ashutosh Sharma, Thomas Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. Dspy: Compiling declarative language model calls into state-of-the-art pipelines. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 54928–54958, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/f1cf02ce09757f57c3b93c0db83181e0-Paper-Conference.pdf.

Sunwoo Kim, Minkyu Kim, and Dongmin Park. Test-time alignment of diffusion models without reward over-optimization. In *The Thirteenth International Conference on Learning Representations*, 2025.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pp. 22199–22213, 2022.

V Kovalev, A Kuvshinov, A Buzovkin, D Pokidov, and D Timonin. CRAFT: Continuous reasoning and agentic feedback tuning for multimodal text-to-image generation. *arXiv preprint arXiv:2512.20362*, 2025.

Robert Lange, Yuki Imajuku, and Edoardo Cetin. Shinkaevolve: Towards open-ended and sample-efficient program evolution. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 74026–74078, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/7886b9bafe76c52fd568db10ff9772df-Paper-Conference.pdf.

Yoonho Lee, Roshen Sanjay Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=tmbOUyFx3R.

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, 2026. URL https://arxiv.org/abs/2604.25850.

Yaxin Luo, Haobin Jiang, Jialv Zou, Xu Huang, Wenhao Yan, Haodong Li, Zhengrong Yue, Jing Li, Xiaofu Chen, Xiaohan Zhao, et al. Autodesign: Meta-harness optimization for long-horizon agentic design. *arXiv preprint arXiv:2608.13560*, 2026.

Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, and Saining Xie. Scaling inference time compute for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2523–2534, June 2025.

Meta. Introducing muse image: Image generation built for your world. Meta Newsroom, July 2026. URL https://about.fb.com/news/2026/07/introducing-muse-image-meta-ai/. Accessed: 2026-09-15.

Atsuyuki Miyai, Kiyoharu Aizawa, and Toshihiko Yamasaki. Task-coevolve: Efficient harness optimization via adaptive validation task selection. *arXiv preprint arxiv:2608.20169*, 2026.

Ryugo Morita, Stanislav Frolov, Brian Bernhard Moser, Takahiro Shirakawa, Ko Watanabe, Andreas Dengel, and Jinjia Zhou. Tkg-dm: Training-free chroma key content generation diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13031–13040, June 2025.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Ku Onoda, Paavo Parmas, Hiroki Furuta, Soichiro Nishimori, Yuta Oshima, Shohei Taniguchi, and Yutaka Matsuo. Multi-axis max@k reinforcement learning for representative diversity in text-to-image generation, 2026. URL https://arxiv.org/abs/2607.14962.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

OpenAI. Gpt-4o image generation. https://openai.com/index/introducing-4o-image-generation/, 2025a. Accessed: 2025-10-31.

OpenAI. The new chatgpt images is here. https://openai.com/index/new-chatgpt-images-is-here/, 2025b. Accessed: 2026-4-30.

OpenAI. Introducing GPT-5.5. https://openai.com/index/introducing-gpt-5-5/, 2026. Accessed: 2026-09-24.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9340–9366, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.525. URL https://aclanthology.org/2024.emnlp-main.525/.

Yuta Oshima, Masahiro Suzuki, Yutaka Matsuo, and Hiroki Furuta. Inference-time text-to-video alignment with diffusion latent beam search. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=c9EAmyYPOv.

Yuta Oshima, Daiki Miyake, Kohsei Matsutani, Yusuke Iwasawa, Masahiro Suzuki, Yutaka Matsuo, and Hiroki Furuta. Multibanana: A challenging benchmark for multi-reference text-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 448–460, June 2026.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7957–7968, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.494. URL https://aclanthology.org/2023.emnlp-main.494/.

Xiangyan Qu, Zhenlong Yuan, Jing Tang, Rui Chen, Datao Tang, Meng Yu, Lei Sun, Yancheng Bai, Xiangxiang Chu, Gaopeng Gou, Gang Xiong, and Yujun Cai. From scale to speed: Adaptive test-time scaling for image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23272–23282, June 2026.

Ruchit Rawal, Reza Shirkavand, Sayak Paul, Yuxin Wen, Heng Huang, Yizheng Chen, Tom Goldstein, and Gowthami Somepalli. Flash-bon: Instant drafts for inference-time scaling in diffusion models. In *The 19th European Conference on Computer Vision*, 2026.

Maxime Robeyns, Martin Szummer, and Laurence Aitchison. A Self-Improving Coding Agent. In *ICLR 2025 Workshop on Scaling Self-Improving Foundation Models*, 2025. URL https://openreview.net/forum?id=rShJCyLsOr.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22500–22510, June 2023.

Shreshth Saini, Neil Birkbeck, Yilin Wang, Balu Adsumilli, and Alan C. Bovik. Cachedsearch: Training-free cached exploration for test-time search in video diffusion, 2026. URL https://arxiv.org/abs/2607.23159.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 38154–38180. Curran Associates, Inc., 2023. doi: 10.52202/075280-1657. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/77c33e6a367922d003ff102ffb92b658-Paper-Conference.pdf.

Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=Jp988ELppQ.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Xingchen Wan, Han Zhou, Ruoxi Sun, Hootan Nakhost, Ke Jiang, Rajarishi Sinha, and Sercan Ö Arık. Maestro: Self-improving text-to-image generation via agent orchestration. *arXiv preprint arXiv:2509.10704*, 2025.

Zhenyu Wang, Aoxue Li, Zhenguo Li, and Xihui Liu. Genartist: Multimodal llm as an agent for unified image generation and editing. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 128374–128395. Curran Associates, Inc., 2024. doi: 10.52202/079017-4077. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/e7c786024ca718f2487712bfe9f51030-Paper-Conference.pdf.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-image technical report, 2025. URL https://arxiv.org/abs/2508.02324.

Chenyuan Wu, Jiahao Wang, Pengfei Zheng, Ruiran Yan, Shitao Xiao, Xin Luo, Yueze Wang, Wanli Li, Xiyan Jiang, Yexin Liu, et al. Omnigen2: Towards instruction-aligned multimodal generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21964–21975, 2026a.

Meiqi Wu, Jiashu Zhu, Xiaokun Feng, Chubin Chen, Chen Zhu, Bingze Song, Fangyuan Mao, Jiahong Wu, Xiangxiang Chu, and Kaiqi Huang. Imagerysearch: Adaptive test-time search for video generation beyond semantic dependency constraints. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(13):10700–10708, Mar. 2026b. doi: 10.1609/aaai.v40i13.38044. URL https://ojs.aaai.org/index.php/AAAI/article/view/38044.

Bin Xia, bohao peng, Yuechen Zhang, Junjia Huang, Jiyang Liu, Jingyao Li, Haoru Tan, Sitong Wu, Chengyao Wang, Yitong Wang, Bei Yu, and Jiaya Jia. Dreamomni2: Multimodal instruction-based generation and editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 29275–29284, June 2026.

Ruihang Xu, Dewei Zhou, Fan Ma, and Yi Yang. Contextgen: Contextual layout anchoring for identity-consistent multi-instance generation. In *The Fourteenth International Conference on Learning Representations*, 2026.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 12028–12068, 2024a. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/3339f19c5fcee3ad74502947a32be9e6-Paper-Conference.pdf.

Zhengyuan Yang, Jianfeng Wang, Linjie Li, Kevin Lin, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. Idea2img: Iterative self-refinement with gpt-4v for automatic image design and generation. In *European conference on computer vision*, pp. 167–184. Springer, 2024b.

Mingde Yao, Zhiyuan You, King-Man Tam, Menglu Wang, and Tianfan Xue. PhotoAgent: Exploratory Visual Aesthetic Planning with Large Vision Models. In *International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/63474.

Po-Hung Yeh, Kuang-Huei Lee, and Jun cheng Chen. Training-free diffusion model alignment with sampling demons. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tfemquulED.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639(8055):609–616, 2025. doi: 10.1038/s41586-025-08661-4. URL https://www.nature.com/articles/s41586-025-08661-4.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-Taught Optimizer (STOP): Recursively Self-Improving Code Generation. In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=46Zgqo4QIU.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-harness: Harnesses that improve themselves, 2026a. URL https://arxiv.org/abs/2606.09498.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin gödel machine: Open-ended evolution of self-improving agents. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 104223–104294, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/aa5f5e6eb6f613ec412f1d948dfa21a5-Paper-Conference.pdf.

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 Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 34040–34077, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5492ecbce4439401798dcd2c90be94cd-Paper-Conference.pdf.

Jingxu Zhang, Daneul Kim, Yueming Pan, Dong Chen, Kai Qiu, Yang Liu, Yifan Yang, Qi Dai, Xiaoyan Sun, and Chong Luo. Rcedit-500k: Reference completion for image-conditioned image editing. In *European Conference on Computer Vision (ECCV)*, 2026c.

XiangCheng Zhang, Haowei Lin, Haotian Ye, James Zou, Jianzhu Ma, Yitao Liang, and Yilun Du. Inference-time scaling of diffusion models through classical search. In *The Fourteenth International Conference on Learning Representations*, 2026d. URL https://openreview.net/forum?id=b7Ftp6U78i.

Jiahao Zhao, Xiaomin Yu, Zhongxiang Sun, Fengwei Teng, Chengwei Qin, Xiaobin Hu, Jun Xu, and Shuicheng Yan. Toolartist: Tool-using unified multimodal models for agentic image generation. *arXiv preprint arXiv:2608.04436*, 2026a.

Zengqun Zhao, Ziquan Liu, Yu Cao, Shaogang Gong, Zhensong Zhang, Jifei Song, Jiankang Deng, and Ioannis Patras. Latsearch: Latent reward-guided search for faster inference-time scaling in video diffusion. In *European Conference on Computer Vision (ECCV)*, 2026b.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large Language Models Are Human-Level Prompt Engineers. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=92gvk82DE-.

Luyang Zhu, Dawei Yang, Tyler Zhu, Fitsum Reda, William Chan, Chitwan Saharia, Mohammad Norouzi, and Ira Kemelmacher-Shlizerman. Tryondiffusion: A tale of two unets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4606–4615, June 2023.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62743–62767. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/zhuge24a.html.

APPENDIX

A IMPLEMENTATION DETAILS

The proposer is Claude Fable 5.1, run through the Claude Code CLI with the `Read`, `Glob`, `Grep`, `Write`, `Edit`, and `Bash` tools. It receives only text at the start of each session and loads images from the filesystem when it needs to inspect them. The search runs for five iterations, and the search and all evaluations run on NVIDIA RTX 6000 Ada GPUs. Table 5 lists the models and their identifiers. The reasoning model is a dated API snapshot rather than a moving alias, open-weight models are pinned to fixed revisions in the released code, and the evaluator decodes greedily. The proprietary models are called through their APIs with all references and the instruction, a square output, and otherwise default settings: GPT-Image-1.5 through the image edit endpoint at 1024×1024, Nano Banana Pro through the Gemini API with a 1:1 aspect ratio, and Seedream 4.5 at its native 2048×2048, downsampled to 1024×1024 before evaluation. Code is available at `https://anonymous.4open.science/r/AutoRef-C506`.

Table 5: Model versions used in this work. Identifiers are Hugging Face repositories or API model names. Exact revisions are pinned in the released code.

Model	Identifier
Open image models	
OmniGen2 (Wu et al., 2026a)	`OmniGen2/OmniGen2`
DreamOmni2 (Xia et al., 2026)	`xiabs/DreamOmni2`
BAGEL (Deng et al., 2025)	`ByteDance-Seed/BAGEL-7B-MoT`
FLUX.2 [klein] 4B (Black Forest Labs, 2026)	`black-forest-labs/FLUX.2-klein-4B`
FLUX.2 [klein] 9B (Black Forest Labs, 2026)	`black-forest-labs/FLUX.2-klein-9B`
Qwen-Image-Edit-2511 (Wu et al., 2025)	`Qwen/Qwen-Image-Edit-2511`
DyRef (Huang et al., 2026b)	`Weistrass/Qwen-Image-Edit-2511-DyRef`
Proprietary image models	
GPT-Image-1.5 (OpenAI, 2025b)	`gpt-image-1.5`
Nano Banana Pro (Google DeepMind, 2025b)	`gemini-3-pro-image`
Seedream 4.5 (ByteDance Seed, 2025)	`seedream-4-5-251128`
Reasoning models	
GPT-5.5 (OpenAI, 2026)	`gpt-5.5-2026-04-23`
Qwen3-VL-32B-Instruct (Bai et al., 2025)	`Qwen/Qwen3-VL-32B-Instruct`
Evaluator (MultiBanana)	
Qwen3-VL-8B-Instruct (Bai et al., 2025)	`Qwen/Qwen3-VL-8B-Instruct`
Evaluator (OmniContext)	
GPT-4.1	`gpt-4.1`
Proposer	
Claude Fable 5.1 via Claude Code (Anthropic, 2025)	`claude-fable-5-1`

B EXTENDED RELATED WORK

Test-Time Scaling for Multimodal Generation. Test-time scaling (TTS), which improves model capabilities by allocating additional computation at inference time, has its roots in the development of reasoning in large language models (Kojima et al., 2022; Snell et al., 2025). This paradigm has recently been extended to image and video generation, where a growing body of work improves generation quality and alignment with human preferences (Furuta et al., 2024; Onoda et al., 2026) by scaling inference-time computation without updating model parameters (Yeh et al., 2025; Kim et al., 2025; Ma et al., 2025; Singhal et al., 2025; Oshima et al., 2025; Zhang et al., 2026d; He et al., 2026a). Beyond uniformly increasing computation for all inputs, adaptive TTS dynamically allocates computation budgets according to input difficulty or intermediate evaluations, aiming to achieve more efficient search (Wu et al., 2026b; Zhao et al., 2026b; Guimaraes & Perona, 2026; Rawal et al., 2026; Saini et al., 2026; Jung et al., 2026), with applications spanning image editing (Qu et al., 2026) and joint audio-video generation (Jung et al., 2026). From a broader perspective, self-improving agents

and agentic refinement, which iteratively alternate between generation, evaluation, and revision, can also be viewed as a form of TTS that leverages additional inference-time computation to improve solutions (Hao et al., 2023; Wang et al., 2024; Yao et al., 2026). In this view, our work does not merely increase the search performed for each instance; it automatically optimizes the agent harness to find a more effective and computationally efficient inference procedure.

Image Generation and Editing Agents. Image generation agents combine prompt adaptation (Hao et al., 2023; Datta et al., 2024), tool orchestration (Shen et al., 2023; Wang et al., 2024; Guo et al., 2025), and visual feedback (Yang et al., 2024b; Wan et al., 2025; Kovalev et al., 2025) to improve model outputs. GEMS (He et al., 2026b) integrates iterative generation with trajectory memory and reusable skills. Related systems use tree search for multi-step editing (Yao et al., 2026) or combine tool use and self-refinement with multi-reference composition (Meta, 2026). Beyond runtime refinement, reusable agent policies can be learned through reinforcement learning or experience distillation (Jiang et al., 2026; Chen et al., 2026), with some approaches jointly training reasoning, tool use, and native image generation (Zhao et al., 2026a). Unlike these human-written or trained agents, AutoRef searches over the harness code itself while keeping all models frozen.

C The AutoRef Algorithm

Algorithm 1 AutoRef: Harness Optimization for Multi-Reference Image Generation

Require: Frozen reasoning model M_θ, image generator G_ϕ, evaluator R, proposer P
Require: Training set D_{train}, validation set D_{val}, iterations T, beam width B, candidates per iteration K
Ensure: Final beam $\mathcal{B}_T$ and search history $\mathcal{L}_T$
1: $\mathcal{B}_0 \leftarrow \{H_{\text{vanilla}}, H_{\text{init}}\}$
2: Run each $H \in \mathcal{B}_0$ on D_{train} and collect training evidence
3: Initialize $\mathcal{L}_0$ with the harnesses and their training evidence
4: **for** $t = 0, \ldots, T-1$ **do**
5: P inspects $\mathcal{B}_t$ and $\mathcal{L}_t$ and proposes $\mathcal{C}_{t+1} = \{H_{t+1}^{(k)}\}_{k=1}^{K}$ ▷ Propose candidate harnesses
6: **for** $k = 1, \ldots, K$ **do**
7: Run $H_{t+1}^{(k)}$ on D_{train} and collect evidence $\mathcal{E}_{t+1}^{(k)}$ ▷ Collect proposer-visible training evidence
8: **end for**
9: **for** $k = 1, \ldots, K$ **do**
10: Evaluate $J_{D_{\text{val}}}(H_{t+1}^{(k)})$
11: **end for**
12: $\mathcal{B}_{t+1} \leftarrow \text{TopB}_{H \in \mathcal{C}_{t+1}} J_{D_{\text{val}}}(H)$ ▷ Select on the hidden validation set
13: Append all candidate harnesses, training evidence, and selected candidate identities to $\mathcal{L}_{t+1}$; do not expose validation scores or artifacts
14: **end for**
15: **return** $(\mathcal{B}_T, \mathcal{L}_T)$

Algorithm 1 gives the full search loop of Section 4. The search starts with the base generator H_{vanilla} and the initial harness H_{init}, both run on D_{train}, so the first history already contains their outputs (lines 1–3). At each iteration, the proposer reads the current beam and the history and writes K new harnesses as code (line 5). Each candidate is run on D_{train} (line 7), and its code, per-task scores, evaluator rationales, execution trajectories, and generated images are added to the search history for later iterations (line 13). Each candidate is then scored on D_{val} (line 10), and the B candidates with the highest $J_{D_{\text{val}}}$ form the next beam (line 12). The history records which candidates were kept, but not their validation scores or outputs (line 13): the proposer learns which directions survived without seeing the data that decided it. With $T = 5$, $K = 4$, and $B = 2$, the search evaluates 20 candidates, and AutoRef-Harness is the member of $\mathcal{B}_T$ with the highest $J_{D_{\text{val}}}$.

D How the Harness Changed Across Iterations

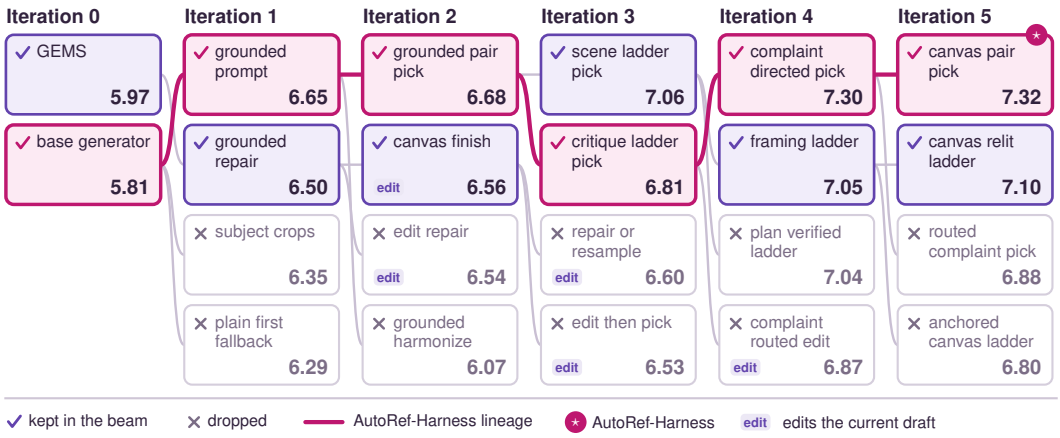

Figure 6: **The AutoRef search that produced AutoRef-Harness.** The two initial harnesses and the 20 candidates written over five iterations, each with its $J_{D_{\text{val}}}$ score; each candidate has an edge to its parent. Checked boxes were kept in the beam; the pink path is the lineage that ends in AutoRef-Harness ($\star$), and *edit* marks candidates that repair a draft by editing it. Candidate names are those given by the proposer.

Figure 6 shows the complete search: two initial harnesses (the base generator and GEMS) and five iterations of $K = 4$ candidates with a beam of $B = 2$.

How AutoRef-Harness was assembled. The lineage of AutoRef-Harness (the pink path) acquired the components of Section 5 over the five iterations; scores in Figure 6 are $J_{D_{\text{val}}}$. Iteration 1 replaced the raw instruction with a prompt written by the reasoning model that states what each reference contributes, still with one image per task (reference-grounded prompting, §5.1; 5.81 $\rightarrow$ 6.65). Iteration 2 drew two drafts from two differently framed grounded prompts and chose between them by a pairwise comparison run in both presentation orders (6.68). Iteration 3 drew both drafts from a scene-first prompt and added a third draft generated from a prompt revised for the current winner's weakest criterion (6.81); the budget has remained at three images per task since. Iteration 4 based this revision on concrete complaints checked against the references and applied a hard-failure check to every draft before the pairwise comparison (complaint-directed revision and failure-aware selection, §5.3–5.4; 7.30). Iteration 5 restored structural diversity: one of the two drafts became canvas-anchored, with the reference that sets the background or style passed to the generator first (structurally diverse drafts, §5.2; 7.32).

Directions that were not retained. The *edit* candidates repair the current draft by passing it to the generator as the first image, followed by the references and an instruction to change only the failing element. This direction entered at iteration 2 (edit repair, 6.54; canvas finish, 6.56, which also re-renders the draft under a single light). Its descendants reached 6.53 and 6.60, and the direction left the beam at iteration 3, and a later edit-based variant on the main lineage (complaint routed edit, 6.87) was also dropped. The proposer's own analyses on the training tasks identify the causes: edits left the flagged failure in place on 8 of 12 flagged tasks, and the single-light re-render lowered the score of the draft it was applied to (6.16 $\rightarrow$ 5.91). Other single-step variants were dropped at once: cropping small subjects from their references (subject crops, 6.35), generating from the raw instruction and grounding only after a failure (plain first fallback, 6.29), and re-rendering the grounded draft under one light (grounded harmonize, 6.07). A second lineage branched off at iteration 3 (scene ladder pick, 7.06, three drafts from one scene-first prompt) and then wrote the scene as a structured plan from which several framings of the prompt were rendered (framing ladder, 7.05; plan verified ladder, 7.04). It remained in the beam until the end and, at iteration 5, also arrived at canvas-anchored drafts (canvas relit ladder, 7.10), but did not reach AutoRef-Harness.

Comparison with other search procedures. Figure 7 shows the best $J_{D_{\text{val}}}$ reached at each iteration by AutoRef and by the two search baselines of Appendix G.2. After five iterations, AutoRef reaches 7.32, Greedy Search 7.15, and Meta-Harness 6.74.

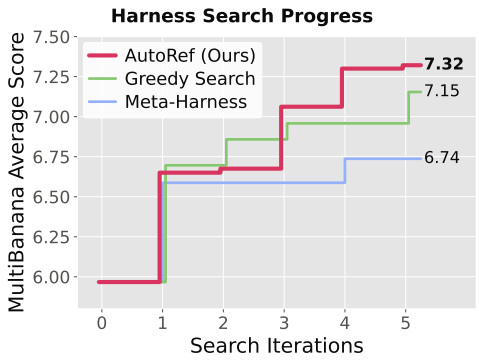

Figure 7: **Search progress on D_{val}.** Best validation score found so far at each iteration by AutoRef, Greedy Search, and Meta-Harness, with the same proposer, models, and evaluator. Meta-Harness does not select on D_{val}; its candidates are scored on it only for this figure.

E Prompts for Proposer Coding Agent

The proposer is a coding agent that starts a new session, with no memory of earlier sessions, at every iteration. It receives two prompts: a system prompt that is fixed across iterations and provided as a Claude Code (Anthropic, 2025) skill, and an iteration prompt that specifies the iteration, the number of tasks each candidate is evaluated on, the current beam, and the log files it may read. It gets everything else by reading files. In the excerpts below, split names are replaced by our notation.

Information available to the proposer. The proposer runs in an isolated container. It can read the code, scores, evaluator rationales, execution trajectories, and generated images of every earlier candidate on D_{train}, and the names of the candidates kept in the beam. Evaluation results on D_{val} and on the held-out test split are stored outside the container and are never exposed.

System prompt. Most of the system prompt describes the task and the harness interface and is adapted from the proposer skill of Meta-Harness (Lee et al., 2026). We reproduce the parts that shape the search: the two model calls available to a harness, and the rules that determine what counts as a valid candidate. In the excerpt, a harness is a Python class whose `run` method maps a task to an output image, `ctx.think` calls the reasoning model, `ctx.generate` calls the image generator, and `mean_generations` is the reported number of images per task.

Proposer System Prompt (excerpt)

[... the task, and the harness interface ...]

`ctx` gives you, and nothing else:
- `ctx.think(prompt, images=[...]) -> str` – the frozen LLM (gpt-5.5). `images` are bytes or paths; `<image>` placeholders in the prompt take them in order.
- `ctx.generate(prompt, images=[...]) -> bytes` – the frozen image model. `images` are bytes or paths the model conditions on. Call it as many times as your mechanism needs; every call is counted and reported as `mean_generations`.
- `ctx.calls` – the running count of every call this run has made.

Anti-parameter-tuning rules

The most common failure mode is creating harnesses that are just parameter variants of existing ones. Check `evolution_summary.jsonl` for what's been tried – sweeps (how many rounds, how many questions, how many references to pass) almost always regress or tie.

Good candidates change a fundamental mechanism:
- A new verification design (e.g. graded checks instead of yes/no, or checks that compare against a specific reference)
- A new refinement architecture (e.g. separate what must be kept from what must change)
- A new control flow (e.g. let the first verification decide how many rounds to run)
- A new rule for what to return

Bad candidates just tune numbers. If `run()` is identical to an existing harness except for constants, it is a parameter variant. Rewrite with a genuinely novel mechanism.

Combining harnesses is valid. Take the verification from A and the refinement from B.

Anti-overfitting rules

- **No task-specific hints.** Do not hardcode knowledge about particular prompts or subjects.
- **Never mention the benchmark name** in harness code, prompts, or comments.
- **General patterns are OK.** "Check the subject before the background" applies broadly.
- `ctx` is your only access to the models.

Iteration prompt. This is the only prompt that changes between iterations. It specifies the iteration, the number of tasks per candidate, earlier iterations' logs, the current beam, and the number of candidates per beam member. The example below is from iteration 5; paths are placeholders.

Iteration Prompt (iteration 5)

Run iteration 5 of the harness evolution loop for task 'multibanana'. Each candidate will be benchmarked on 48 tasks of D_{train}.

Run directories
All logs and results for this run are under `<logs>/`.
- `<logs>/evolution_summary.jsonl` - past results
- `<logs>/frontier.json` - frontier
- `<logs>/runs/<harness>.json/harness_result.json` - per-harness scores, tool counts, worst tasks
- `<logs>/runs/<harness>.json/trace/<id>.json` - every tool call, per task
- `<logs>/reports/` - post-eval reports
- Harness files live in `<harnesses>/`
- Write `pending_eval.json` to: `<logs>/pending_eval.json`
- `<logs>/adoption.jsonl` - which candidates were kept as base harnesses, and which were not
- `<logs>/beam.json` - the current base harnesses

Base harnesses
The current base harnesses are `complaint_directed_pick`, `framing_ladder` (files in `<harnesses>/`). Write exactly 4 candidates: 2 that build on each base. Each candidate's `base_harness` is the exact name of its base. Which candidates are kept as bases for the next iteration is decided on tasks outside D_{train} that this session cannot see: the 2 candidates that do best there replace the current bases, so a gain has to hold on unseen tasks to be kept.

F THE FLOW OF AUTOREF-HARNESS



Figure 8: **Execution trajectory of AutoRef-Harness on a held-out four-reference task.** Steps 1–5 show the input, the identification of the scene reference, the two drafts, failure-aware selection, and complaint-directed revision. Quoted text with a blue bar is a prompt sent to the reasoning model M_θ, and the text below it is the model's response; boxes labeled "image model" mark calls to the generator G_ϕ. Prompts and responses are abridged but not otherwise modified. Each reference's frame color matches that of its number in the text.

Figure 8 shows how AutoRef-Harness processes one task. The reasoning model first infers that image 2, the poodle in a library, sets the scene (step 2). It then writes two structurally different prompts (step 3): a scene-first prompt that describes the setting and then each subject (draft A), and a canvas-anchored prompt that passes image 2 first and places the other subjects in it (draft B). Both drafts pass the failure check, and draft B is preferred in both presentation orders (step 4). The reasoning model lists complaints about draft B, mainly that the violinist in image 4 does not match

her reference and is not in the foreground, and its revised prompt produces draft C (step 5). Draft C does not beat draft B under the same rule, so draft B is returned (evaluator score 8.6/10).

G BASELINES

G.1 HUMAN-WRITTEN HARNESS BASELINES

All four human-written baselines use the same generator and reasoning model as AutoRef-Harness; adaptations to the multi-reference setting are noted per method.

Best-of-N (Ma et al., 2025). N images are sampled independently from the original instruction, and the reasoning model picks one in one call showing the references and candidates, using a priority-ordered rubric (every requested subject present, fidelity to each reference, correct background, consistent lighting, realism). We use $N = 3$ throughout, matching AutoRef-Harness's three images.

GEMS (He et al., 2026b). An agentic loop with skills and memory. The instruction is routed to a matching skill and decomposed into yes/no requirement questions; each round generates an image, checks every question against it, stops if all pass, and otherwise summarizes the round into memory and rewrites the prompt from the accumulated history. The image satisfying the most questions is returned. We use the published prompts and the default four rounds, and show the verifier the references beside the image, since the original loop verifies single-image text-to-image outputs.

IPR (Oshima et al., 2026). Iterative Prompt Refinement, the agentic baseline proposed with MultiBanana. Over three steps, each step generates from the current prompt, and a planner refines the prompt from the references and the image just generated:

$$y^{t+1} = \mathrm{Gen}(u^t, \mathcal{I}), \qquad u^{t+1} = \mathrm{Plan}(u^t, \mathcal{I}, y^{t+1}).$$

The generator never sees earlier images, and the last image is returned. No code was released, so we re-implement it from its equations and prompts, with our reasoning model as the planner.

Idea2Img (Yang et al., 2024b). Iterative self-refinement in which a multimodal model drafts several prompts, selects the best image, and writes feedback that, with a memory of earlier prompts and feedback, guides the next round. We keep the official budget of three prompts over three rounds with a final selection among round winners (nine images per task). As this triples AutoRef-Harness's budget, we also report a budget-matched variant: three prompts, one round, no feedback.

G.2 HARNESS SEARCH BASELINES

Meta-Harness (Lee et al., 2026). Our implementation of the Meta-Harness protocol, run with the same proposer, models, and evaluator as AutoRef. The proposer sees the full history of earlier candidates — their code, scores, execution trajectories, and images — and the candidates are ranked on the same tasks whose feedback the proposer reads, so there is no separate selection split. It has no explicit parents: the proposer chooses which candidate to build on, and the best candidate on these tasks is reported. We run it for five iterations, as for AutoRef (the original runs 20–40 iterations).

Greedy Search. AutoRef with a beam of one. Proposal and selection use separate splits as in AutoRef (D_{train} for feedback, D_{val} for selection), but only the single best candidate on D_{val} is kept at each iteration and becomes the parent of all candidates in the next iteration.

H Further Results

H.1 Detailed MultiBanana Results

The Qwen3-VL-8B-Instruct (Bai et al., 2025) evaluator of MultiBanana scores each image on five criteria (instruction alignment, reference consistency, background–subject match, physical realism, and visual quality), and the main text reports their mean. Tables 6–8 report each criterion for four, three, and five references, averaged over task types; per-type averages are in Tables 1 and 2.

Table 6: MultiBanana with 4 references, held-out test split (n=133), by evaluation metric. Each metric is averaged over the four task types (object, local, global, background), and Avg. is the mean of the five metrics, as in the Avg. column of Tables 1 and 2. *Gen.* is images drawn per task. Inst.: instruction alignment; Ref.: reference consistency; BG: background–subject match; Real.: physical realism; Qual.: visual quality. Best open image generators per column in bold, second best underlined.

Method	Gen.	Inst.	Ref.	BG	Real.	Qual.	Avg.
GPT-Image-1.5	1	6.58	7.63	6.49	6.79	7.88	7.07
Nano Banana Pro	1	6.94	7.74	6.62	6.78	7.94	7.20
Seedream 4.5	1	6.62	7.78	6.38	6.73	7.66	7.03
OmniGen2	1	3.08	3.36	3.16	3.87	5.26	3.75
DreamOmni2	1	3.36	3.57	3.10	3.91	5.22	3.83
BAGEL	1	2.83	3.13	2.66	3.22	3.92	3.15
FLUX.2 [klein] 4B	1	5.25	5.86	5.04	5.64	6.83	5.72
+ Best-of-3 (Ma et al., 2025)	3	5.37	6.08	5.50	6.08	7.02	6.01
+ GEMS (He et al., 2026b)	2.7	4.92	5.43	4.74	5.38	6.39	5.37
+ IPR (Oshima et al., 2026)	3	6.42	7.08	6.67	7.10	7.84	7.02
+ Idea2Img (Yang et al., 2024b)	9	<u>6.50</u>	**7.39**	6.71	7.00	7.98	<u>7.12</u>
+ Idea2Img, budget-matched	3	5.89	6.85	5.90	6.24	7.20	6.42
+ Meta-Harness (Lee et al., 2026)	4.2	5.64	6.22	5.84	6.29	7.18	6.23
+ Greedy Search	5	5.90	6.82	6.31	6.72	7.58	6.67
+ AutoRef (2nd)	3	6.15	6.75	<u>6.88</u>	7.37	7.85	7.00
+ **AutoRef-Harness**	3	**6.57**	<u>7.15</u>	**7.20**	**7.77**	**8.16**	**7.37**
+ **AutoRef-Harness**, Qwen3-VL-32B	3	6.22	6.96	6.46	7.14	7.70	6.90
FLUX.2 [klein] 9B	1	5.19	5.96	5.28	5.80	6.70	5.78
+ **AutoRef-Harness**	3	6.48	6.85	6.71	<u>7.46</u>	<u>7.99</u>	7.10
Qwen-Image-Edit-2511	1	3.95	4.63	3.66	4.28	5.69	4.44
+ **AutoRef-Harness**	3	5.11	5.78	5.15	5.82	6.59	5.69
+ DyRef (Huang et al., 2026b)	1	4.77	4.89	4.23	4.81	5.86	4.91
+ DyRef + **AutoRef-Harness**	3	5.68	5.66	5.22	6.12	6.83	5.90

H.2 Component Ablation by Task Type

Table 9 gives the per-type scores behind Table 4. Removing any one of the three components lowers the score on every task type, and without reference-grounded prompting (selection only) the score falls below all three leave-one-out rows on every task type.

H.3 Compatibility with Fine-Tuning

We ask whether harness optimization remains useful when the underlying image generator is already optimized for multi-reference image generation. DyRef (Huang et al., 2026b) improves Qwen-Image-Edit-2511 through supervised fine-tuning followed by reward optimization. As shown in Figure 9, AutoRef-Harness applied to the original Qwen-Image-Edit-2511 achieves 5.69, outperforming DyRef alone at 4.91. Applying the same harness to the DyRef-trained model further improves performance to 5.90. These results suggest that harness optimization and model-weight optimization provide complementary gains and can be combined.

Table 7: MultiBanana with 3 references (n=96), by evaluation metric. Columns as in Table 6.

Method	Gen.	Inst.	Ref.	BG	Real.	Qual.	Avg.
GPT-Image-1.5	1	7.22	8.31	7.56	7.84	8.35	7.86
Nano Banana Pro	1	6.79	7.91	7.15	7.68	7.99	7.50
Seedream 4.5	1	6.95	8.47	6.58	7.06	8.00	7.41
OmniGen2	1	4.55	5.15	4.51	5.46	6.58	5.25
DreamOmni2	1	4.75	5.20	4.67	5.56	6.64	5.36
BAGEL	1	3.77	4.14	3.36	4.16	5.05	4.10
FLUX.2 [klein] 4B	1	6.03	6.96	6.59	7.31	7.82	6.94
+ Best-of-3	3	6.14	7.46	6.51	6.88	7.80	6.96
+ **AutoRef-Harness**	3	6.99	7.69	7.74	8.03	8.38	7.76
+ **AutoRef-Harness**, Qwen3-VL-32B	3	6.71	**7.71**	7.27	7.84	8.41	7.59
FLUX.2 [klein] 9B	1	6.18	7.27	6.31	6.74	7.63	6.83
+ **AutoRef-Harness**	3	**7.05**	7.66	**7.75**	**8.21**	**8.45**	**7.82**
Qwen-Image-Edit-2511	1	4.53	5.48	4.10	4.95	6.25	5.06
+ **AutoRef-Harness**	3	6.25	7.16	6.48	7.05	7.79	6.95

Table 8: MultiBanana with 5 references (n=96), by evaluation metric. Columns as in Table 6.

Method	Gen.	Inst.	Ref.	BG	Real.	Qual.	Avg.
GPT-Image-1.5	1	6.73	7.45	5.65	5.96	7.20	6.60
Nano Banana Pro	1	6.82	7.50	5.66	6.21	7.36	6.71
Seedream 4.5	1	6.64	7.36	5.80	6.02	7.19	6.60
OmniGen2	1	2.84	2.66	2.59	3.20	4.60	3.18
DreamOmni2	1	2.54	3.10	2.70	3.17	4.08	3.12
BAGEL	1	3.00	2.73	2.17	2.61	3.57	2.82
FLUX.2 [klein] 4B	1	5.01	5.22	4.38	4.94	6.43	5.19
+ Best-of-3	3	5.42	5.39	4.54	5.19	6.40	5.39
+ **AutoRef-Harness**	3	6.35	6.24	5.55	**6.29**	**7.36**	6.36
+ **AutoRef-Harness**, Qwen3-VL-32B	3	5.72	6.14	5.07	5.81	6.79	5.91
FLUX.2 [klein] 9B	1	5.65	5.79	4.72	5.31	6.56	5.61
+ **AutoRef-Harness**	3	**6.59**	**6.91**	**5.59**	6.08	7.16	**6.47**
Qwen-Image-Edit-2511	1	1.74	2.04	1.55	1.85	2.24	1.89
+ **AutoRef-Harness**	3	1.72	2.02	1.45	1.70	2.47	1.87

Table 9: Component ablation on the held-out test split, by task type. Rows as in Table 4; Avg. is the mean over the four task types, as reported there. All rows draw three images per task except Grounding only and Generator only, which draw one.

	Object	Local	Global	Background	Avg.
Full harness	7.27	7.87	7.64	6.70	**7.37**
– diverse drafts	7.07	7.37	7.14	6.47	7.01
– complaint revision	7.24	7.01	7.33	6.38	6.99
– failure-aware selection	6.79	7.39	7.55	6.00	6.93
Grounding only	7.21	6.72	7.21	6.50	6.91
Selection only	6.11	6.78	6.14	5.56	6.15
Generator only	5.95	6.11	5.50	5.34	5.72

H.4 Detailed OmniContext Results

Tables 10 and 11 provide the full OmniContext breakdown behind Table 3, reporting prompt following (PF), subject consistency (SC), and their geometric mean for each task type.

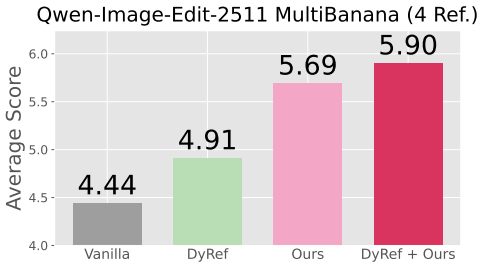

Figure 9: **Harness optimization is compatible with fine-tuning.** On Qwen-Image-Edit-2511, AutoRef-Harness outperforms DyRef alone, and applying it to the DyRef-trained model yields a further improvement. Ours denotes AutoRef-Harness.

Table 10: OmniContext, SINGLE and MULTIPLE task types, 15 tasks each: prompt following, subject consistency, and their geometric mean behind Table 3. *Gen.* is images drawn per task. Best open image generators per column in bold, second best underlined.

		SINGLE						MULTIPLE								
		Character			Object			Character			Object			Char. + Obj.		
Method	Gen.	PF	SC	Overall	PF	SC	Overall	PF	SC	Overall	PF	SC	Overall	PF	SC	Overall
GPT-Image-1.5	1	9.80	9.33	9.56	9.80	9.60	9.70	9.67	9.00	9.32	9.67	9.27	9.46	9.27	9.27	9.26
Nano Banana Pro	1	9.53	9.73	9.63	9.53	9.33	9.42	9.73	9.20	9.46	9.40	9.00	9.19	9.00	9.07	9.02
Seedream 4.5	1	9.60	9.33	9.45	9.47	9.60	9.50	9.20	9.13	9.09	9.67	9.13	9.39	9.13	9.07	9.09
OmniGen2	1	8.20	8.93	8.51	6.87	6.40	5.73	7.13	5.87	6.30	7.67	5.87	6.58	7.80	7.67	7.71
DreamOmni2	1	7.53	8.33	7.81	7.20	6.60	6.72	5.07	5.13	4.80	7.00	7.40	7.07	6.73	5.53	5.92
BAGEL	1	8.20	6.40	6.67	6.67	8.53	7.09	4.27	3.07	3.43	7.07	6.80	6.74	7.20	7.20	7.08
FLUX.2 [klein] 4B	1	9.40	9.07	9.22	8.27	8.87	8.16	8.00	7.93	7.91	8.87	7.73	8.21	8.47	**8.93**	8.68
+ Best-of-3	3	9.40	**9.13**	**9.26**	8.60	8.20	7.97	8.73	8.73	8.71	8.87	8.87	8.81	8.73	8.87	8.79
+ **AutoRef-Harness**	3	9.33	8.60	8.94	9.20	**9.00**	8.99	9.33	8.73	9.02	9.00	8.60	8.78	8.53	8.67	8.59
FLUX.2 [klein] 9B	1	9.40	9.00	9.17	9.33	8.93	9.08	9.00	8.40	8.66	8.73	7.73	8.14	**8.87**	8.67	8.74
+ **AutoRef-Harness**	3	**9.47**	9.00	9.22	9.53	8.87	9.18	**9.40**	**9.00**	**9.19**	**9.73**	**8.93**	**9.32**	8.80	8.87	**8.82**
Qwen-Image-Edit-2511	1	9.40	8.93	9.14	**9.73**	8.73	**9.19**	8.87	8.53	8.66	9.60	8.47	9.00	8.20	8.53	8.33
+ **AutoRef-Harness**	3	9.27	9.00	9.12	9.13	8.47	8.64	9.33	8.87	9.09	9.13	8.20	8.62	8.47	8.27	8.34

Table 11: OmniContext, SCENE task types. Columns as in Table 10.

		SCENE								
		Character			Object			Char. + Obj.		
Method	Gen.	PF	SC	Overall	PF	SC	Overall	PF	SC	Overall
GPT-Image-1.5	1	10.00	9.40	9.69	9.53	9.27	9.39	8.80	9.13	8.93
Nano Banana Pro	1	9.73	9.00	9.35	8.07	8.87	8.39	7.93	8.53	8.20
Seedream 4.5	1	9.87	8.87	9.35	8.60	8.80	8.66	8.13	8.47	8.23
OmniGen2	1	7.27	6.80	6.93	6.47	5.87	6.10	7.53	6.73	7.03
DreamOmni2	1	6.40	5.40	5.78	6.40	5.07	5.63	6.20	5.40	5.72
BAGEL	1	4.87	4.00	3.97	4.13	4.20	4.11	5.73	5.67	5.63
FLUX.2 [klein] 4B	1	9.67	8.73	9.18	7.40	7.67	7.47	7.40	7.73	7.54
+ Best-of-3	3	9.73	**8.93**	9.32	8.73	8.33	8.49	7.93	7.80	7.74
+ **AutoRef-Harness**	3	9.73	8.87	9.28	9.00	**8.80**	**8.89**	8.27	**8.33**	8.28
FLUX.2 [klein] 9B	1	**9.87**	8.87	**9.35**	7.87	8.47	8.07	7.27	7.80	7.47
+ **AutoRef-Harness**	3	**9.87**	8.87	**9.35**	8.80	8.07	8.40	**8.60**	8.20	**8.30**
Qwen-Image-Edit-2511	1	7.67	6.60	6.97	8.40	8.07	8.17	8.53	7.87	8.17
+ **AutoRef-Harness**	3	9.07	8.40	8.71	**9.27**	8.40	8.82	8.40	8.20	8.28

I WHERE THE HARNESS CANNOT HELP

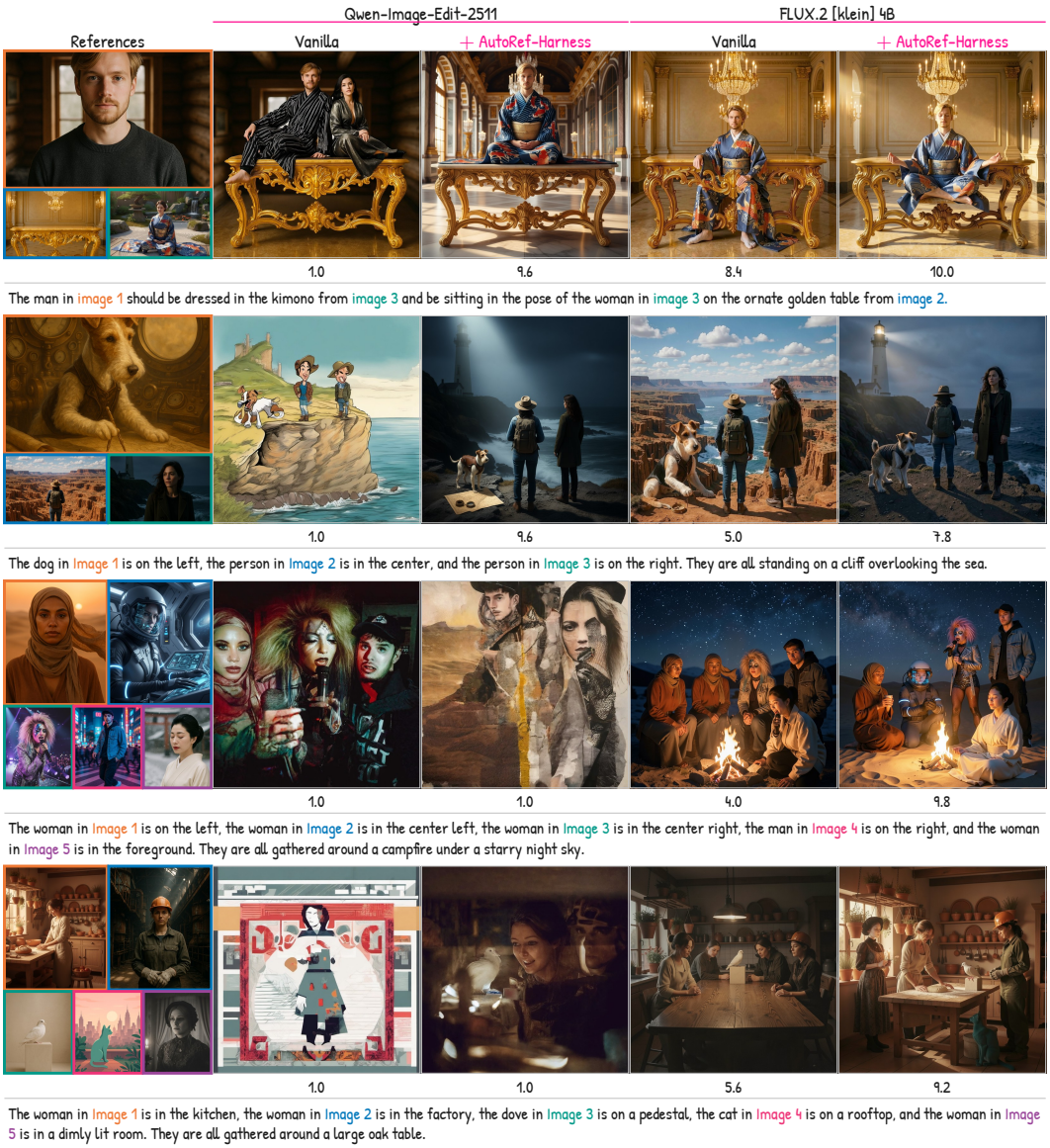

Figure 10: **AutoRef-Harness helps Qwen-Image-Edit-2511 at three references but not at five.** Outputs of Qwen-Image-Edit-2511 and FLUX.2 [klein] 4B, each without and with AutoRef-Harness, on three-reference (top two rows) and five-reference (bottom two rows) tasks. The number below each output is its evaluator score (1–10).

AutoRef-Harness does not change the generator; it can only return one of the images the frozen generator produces. Qwen-Image-Edit-2511 exposes this limit (Figure 10): the harness raises its score from 5.06 to 6.95 at three references and from 4.44 to 5.69 at four, but not at five (from 1.89 to 1.87; Tables 1 and 2). All harness steps still run at five references, but the generator rarely produces an acceptable image: the hard-failure check (§5.4) flags all three drafts on 87 of 96 tasks (91%), versus 9 of 96 (9%) for FLUX.2 [klein] 4B at five references and 7 of 96 (7%) for Qwen-Image-Edit-2511 at three. When no candidate is acceptable, better selection cannot help; overcoming this limit likely requires reducing how many references the generator must compose at once.

J Further Qualitative Comparisons

All figures in this section follow the layout of Figure 3: the references on the left, each framed in a distinct color that also marks its number in the instruction (when the instruction numbers the references), the outputs of five methods, and the full instruction below. Figures 11–13 show three held-out four-reference tasks each, for object composition, local attribute transfer, and background and global style; Figures 14 and 15 show tasks with three and five references, counts not used during the search; and Figure 16 shows OmniContext. None of these tasks was seen during the search. They were selected among the tasks with the largest score gap between the base generator and AutoRef-Harness, so they illustrate the failures the harness removes and are not a random sample; aggregate results are in Appendix H. In all settings, the harness removes the same kinds of failure: a requested reference is dropped or used in the wrong role.

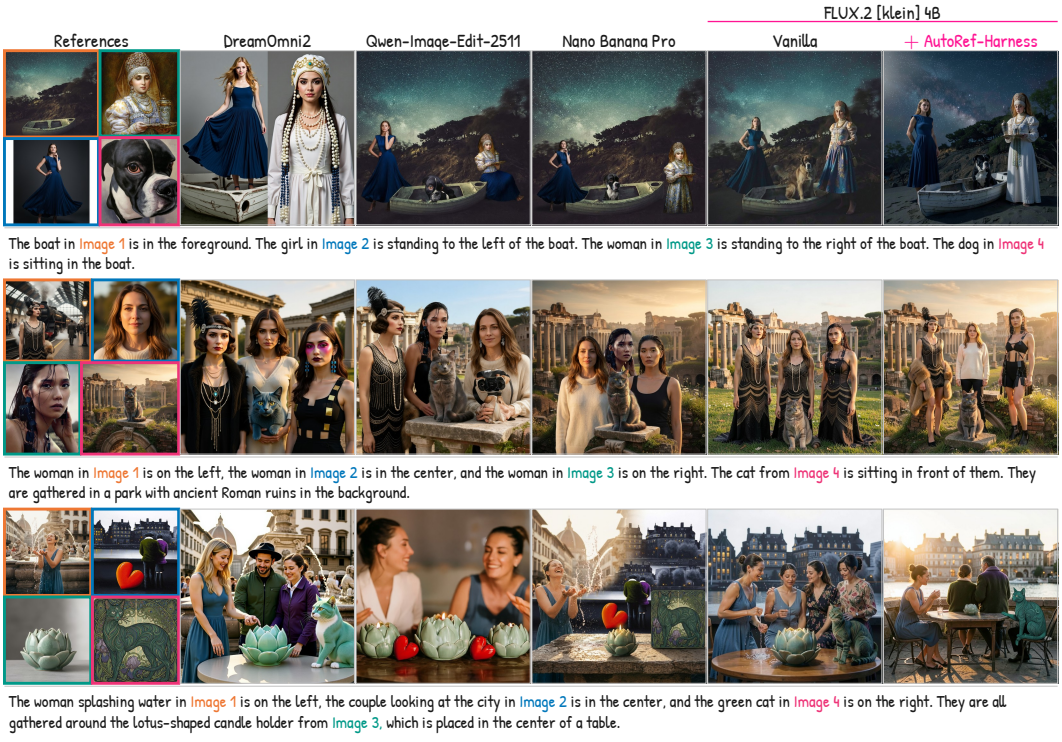

Figure 11: **Object composition.** Held-out four-reference tasks that compose all subjects into one scene.

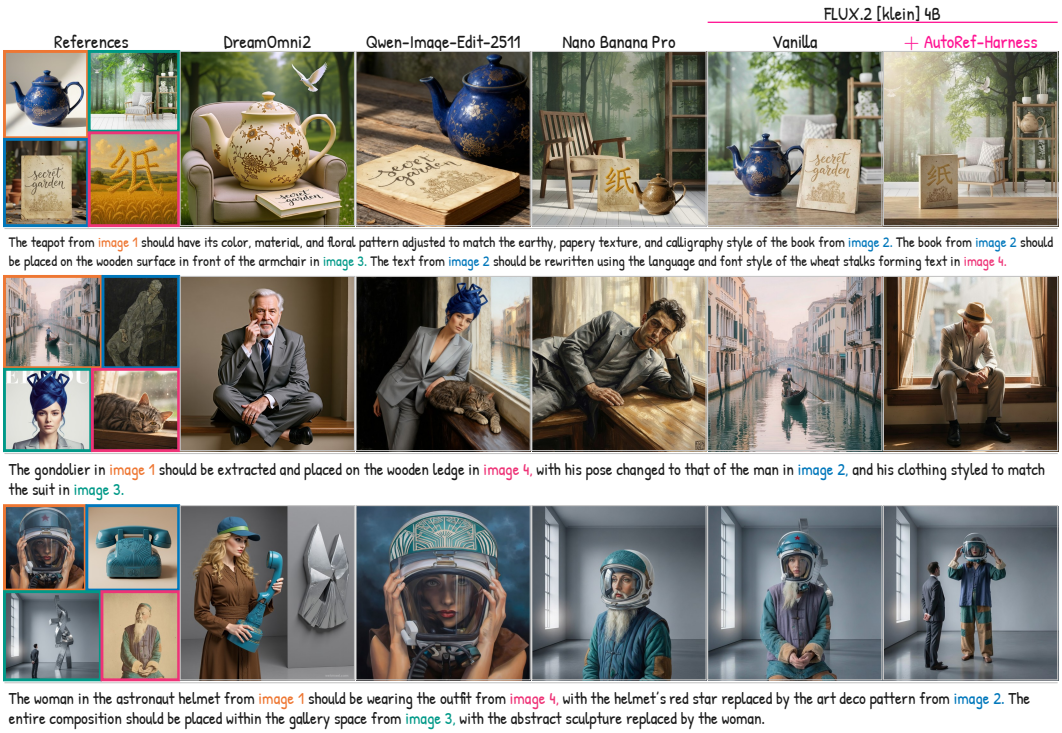

Figure 12: **Local attribute transfer.** Held-out four-reference tasks in which an attribute of one reference (e.g., a pose, a garment, a texture, or a text style) is applied to another.

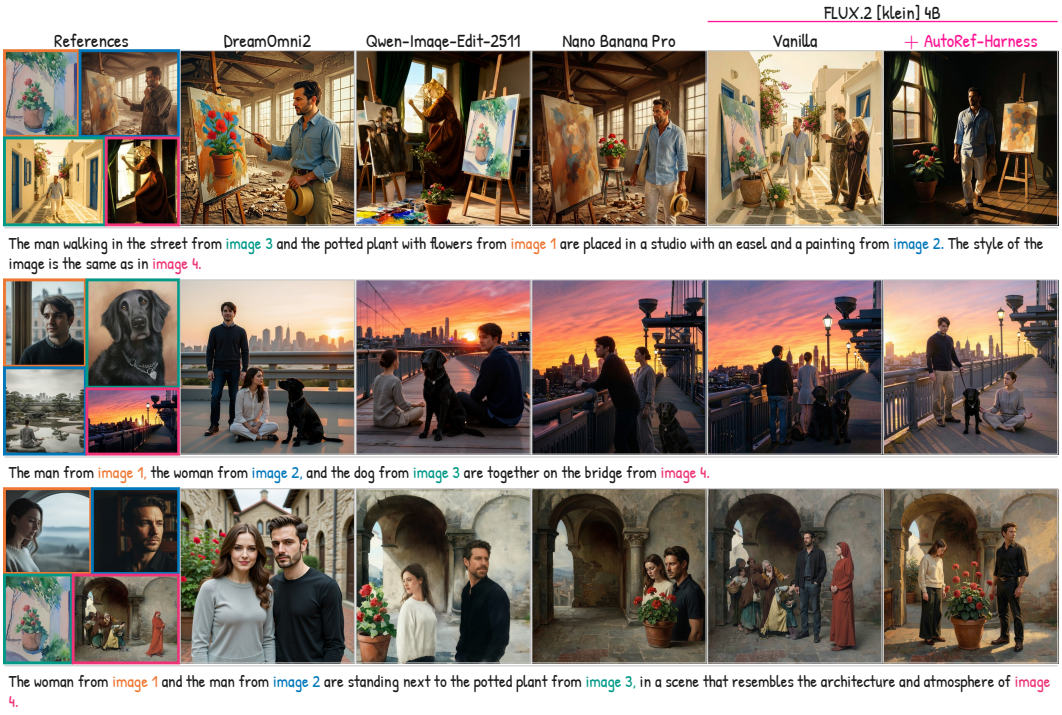

Figure 13: **Background and global style.** Held-out four-reference tasks in which one reference sets the background or the style of the whole image.

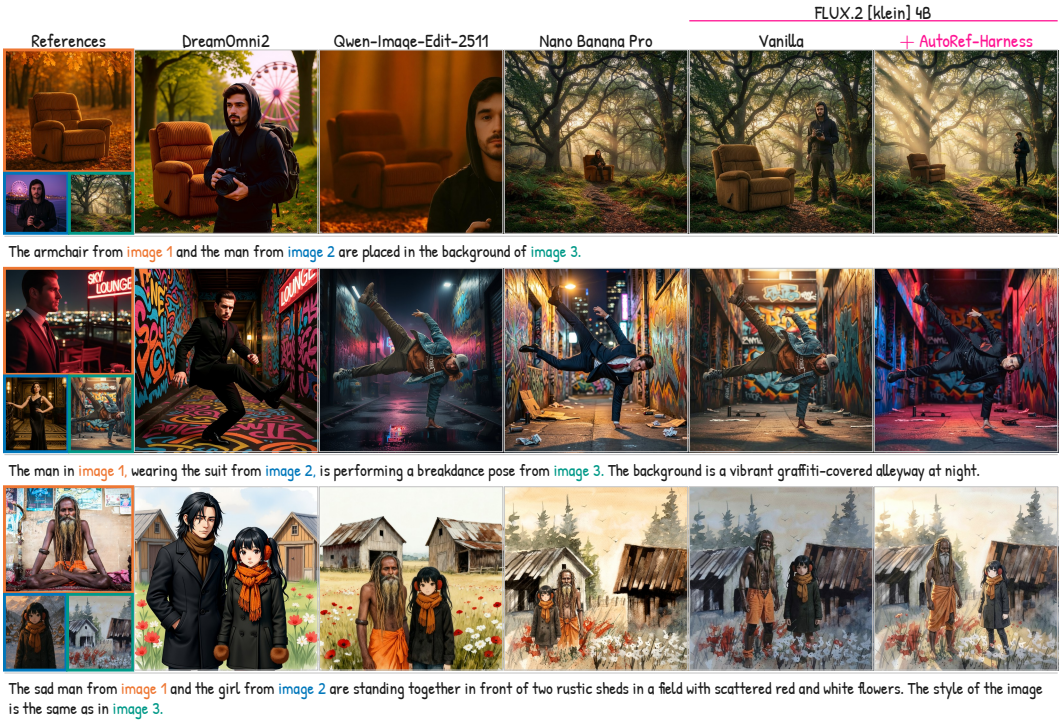

Figure 14: **Three references.** MultiBanana tasks with three references, a count not used during the search.

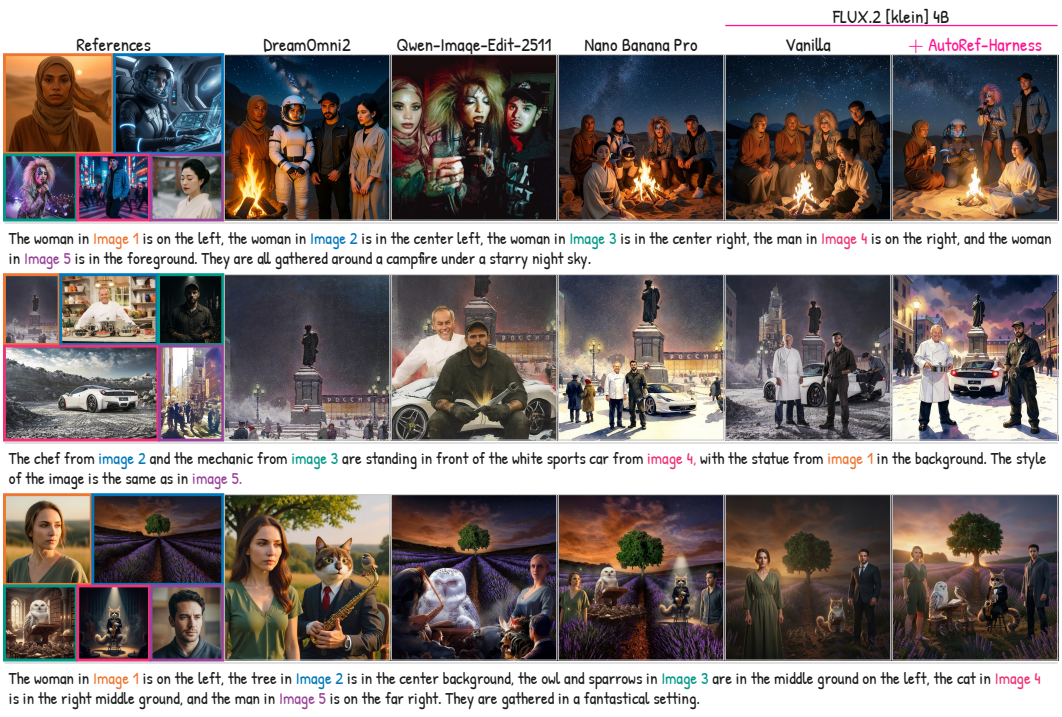

Figure 15: **Five references.** MultiBanana tasks with five references, a count not used during the search.

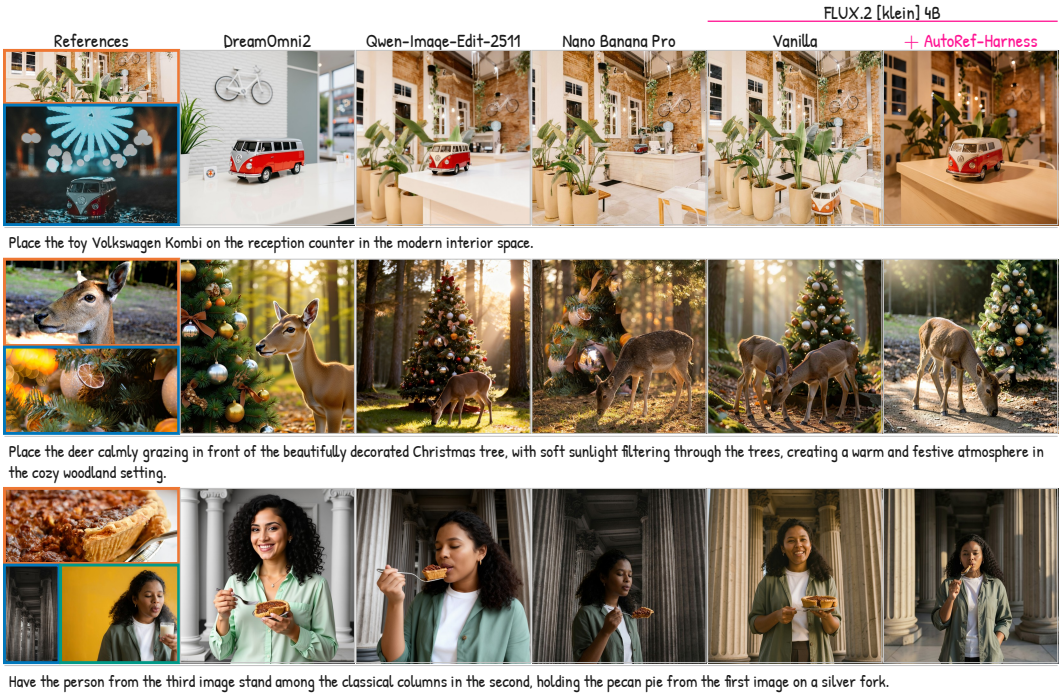

Figure 16: **OmniContext.** Tasks with two or three references, from a benchmark not used during the search.