LENSRSI: TOWARDS RECURSIVE SELF-IMPROVING OPTICAL LENS DESIGN AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Automatic lens design seeks to construct and optimize optical systems with minimal human intervention. Although numerical optimization and physics-based simulation have automated important parts of this process, agentic systems that can autonomously manage complete optical design workflows remain underexplored, and enabling such systems to improve their own design policies poses a further challenge. We study recursive policy improvement for optical design agents and introduce LENSRSI, a verifier-gated framework that converts simulator-measured state–action–outcome trajectories into validated, inheritable search policies. LENSRSI couples a task-solving loop with a policy-improvement loop: the agent interacts with a physics-based optical simulator, learns which actions work in different optical states, and proposes an updated design policy. Every candidate policy is evaluated independently and inherited only when it improves design quality while preserving promotion-validation performance and physical validity. All comparisons restart from fixed initial prescriptions, so only verified search knowledge is carried across rounds. On the held-out LENSARENAEX benchmark of extreme lens-design tasks, LENSRSI achieves smaller root-mean-square spot radii than non-evolving and limited-update alternatives under matched simulation budgets while maintaining optical validity. Results demonstrate the potential of recursive self-improvement (RSI) to build optical design agents and we hope to inspire further research into RSI for scientific discovery. Codes and datasets will be released on acceptance.

1 INTRODUCTION

Lens design translates system-level imaging requirements into a physical prescription of optical surfaces, materials, and axial separations. These variables interact in a high-dimensional, non-convex space, where improving one property can degrade another. Conventional workflows therefore depend on optical engineers to choose a starting form, define numerical objectives, interpret ray-tracing feedback, and steer iterative refinement (Kingslake & Johnson, 2009). Automatic lens design reduces part of this burden through numerical optimization, database retrieval, and learning-based search (Guo et al., 2019; Zhang et al., 2020; Côté et al., 2021; Gao et al., 2025a; Yang et al., 2024). However, applying these methods to new specifications still requires users to translate design intent into method-specific variables, constraints, and optimization settings.

LENSDESIGNER first took a step toward end-to-end automation by introducing an LLM-driven agent for natural-language lens design (Sun et al., 2026). It established an agentic paradigm for optical lens design in which the agent retrieves relevant starting forms, coordinates lens edits, evaluates candidate prescriptions with a physics-based simulator, and accumulates reusable design heuristics. Built on this pipeline, we study a different question: *Can the agent turn its accumulated design experience into a more effective lens-design search policy?*

Accumulating heuristics does not by itself establish that the resulting agent is stronger. The self-evolving mechanism in LENSDESIGNER reflects on prior trajectories and adds reusable skills, but it does not formulate each revision as an isolated candidate policy that must outperform its parent under matched conditions before inheritance. A revision may fit the cases that produced it while weakening performance on other specifications or compromising physical validity. Reliable self-improvement therefore requires more than remembering successful experience. It requires an ex-

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

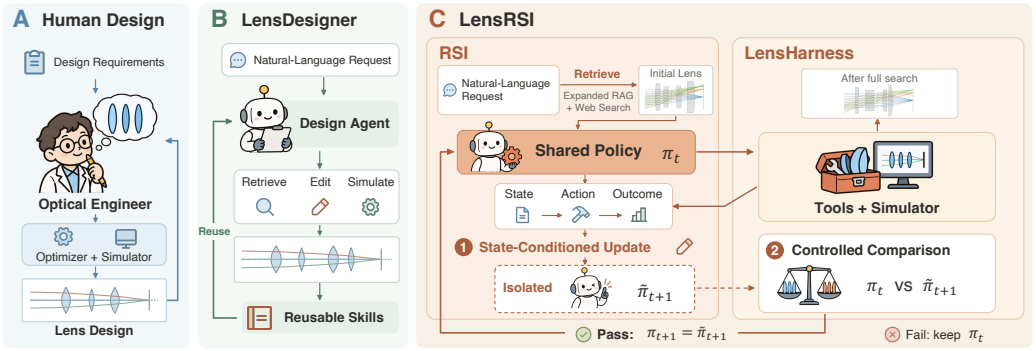

Figure 1: **From human-guided design to verified policy improvement.** LENSDESIGNER reuses design skills, while LENSRSI recursively updates state-conditioned policies verified by LENSHARNESS.

plicit policy lineage together with an independent mechanism that decides which candidate, if any, becomes the parent of the next round.

Lens design provides suitable evidence for this form of improvement. Each agent run produces a trajectory that links an optical state and a lens-editing action to simulator-measured changes in image quality and physical validity. ❶ These trajectories reveal not only whether an action worked, but also the optical conditions under which it worked, which is essential because the same edit can have different effects at different stages of a search. ❷ Moreover, parent and candidate policies can be restarted from identical prescriptions and compared under the same simulation budget. Lens design therefore supplies both the reusable experience and the objective verifier required for recursive self-improvement (RSI), where experience generated by a current policy is used to construct and validate its successor (Good, 1966; Schmidhuber, 2003).

Building on LENSDESIGNER, we introduce LENSRSI, a framework for verified recursive policy improvement in lens design agents (Fig. 1). It retains the natural-language task-solving pipeline while adding a separate policy-improvement loop. ❶ Across development tasks, it converts simulator-verified state–action–outcome trajectories into state-conditioned search preferences that account for immediate and downstream optical improvement. These preferences define an isolated candidate policy rather than directly modifying the active parent. ❷ We further introduce LENSHARNESS, an independent execution and verification environment that restarts parent and candidate policies from identical initial prescriptions, enforces matched simulation budgets, and checks design quality and physical validity. Only a candidate policy that passes this controlled comparison is inherited as the parent policy for the next RSI round.

We evaluate on LENSARENA and our 150-task LENSARENAEX benchmark of challenging, extreme specifications. LENSARENA supplies only promotion-validation feedback, and its trajectories never enter policy memory. LENSARENAEX remains untouched until the final policy is frozen. Under matched budgets, LENSRSI outperforms LENSDESIGNER, automatic lens design methods, non-evolving agents, and controlled variants on both benchmarks, while LENSARENAEX provides the evidence of transfer to unseen specifications.

In summary, this paper makes the following contributions:

- Building on LENSDESIGNER, we formulate lens-agent improvement as explicit recursive policy inheritance and introduce LENSRSI. The framework retains the natural-language task-solving interface while restricting cross-round inheritance to verified search policies.
- We introduce a state-conditioned policy improvement method that converts simulator-verified trajectories into search preferences, together with LENSHARNESS, an independent execution and verification environment for policy evaluation.
- We conduct experiments on the LENSARENA and LENSARENAEX benchmarks, comparing LENSRSI with optimization-based, learning-based, and agentic methods. The results demonstrate

that recursive self-improvement improves lens design performance and transfers to challenging unseen specifications.

2 Related Work

Automatic Lens Design. Automatic lens design translates optical specifications into feasible prescriptions with limited expert supervision. Optimization-based methods search continuous parameters through global-local or coarse-to-fine schemes (Guo et al., 2019; Zhang et al., 2020; Liu et al., 2025; Gao et al., 2025b; Yang et al., 2017; Zhang et al., 2021), while learning-based methods infer design priors or complete systems from lens data (Côté et al., 2019). LensNet predicts and rescales starting forms to a target effective focal length (EFL) (Côté et al., 2021). Quasi-Global Search Optics (QGSO) searches diverse initial systems before jointly optimizing optics and reconstruction networks (Gao et al., 2025a). DeepLens starts from random surfaces and progressively expands the aperture and field of view (FoV) through a curriculum (Yang et al., 2024). Adapting these methods still requires task-specific variables, ranges, schedules, loss weights, and optical expertise. LensDesigner instead integrates retrieval, simulation, and lens refinement in a natural-language agent loop (Sun et al., 2026). Building on it, LensRSI uses simulator-verified trajectories to improve the search policy and independently validates updates before inheritance.

RSI Agents. An RSI agent uses prior runs to modify future behavior, allowing improvements to persist across tasks or generations (Good, 1966; Schmidhuber, 2003). Early systems learned reusable heuristics or ordered solutions (Lenat, 1983; Schmidhuber, 2004), while language agents accumulated feedback and executable skills (Shinn et al., 2023; Wang et al., 2023). The Self-Taught Optimizer (STOP) applied an LLM-based improver to its own code (Zelikman et al., 2023). Recent work optimizes broader agent designs: Automated Design of Agentic Systems (ADAS) programs agents from an archive, AFlow optimizes code-defined workflows, and the Darwin Gödel Machine modifies agent implementations and retains successful descendants (Hu et al., 2025; Zhang et al., 2025; 2026). Auto-research extends this loop to scientific discovery. The AI Scientist automates research workflows (Lu et al., 2026), while AlphaEvolve combines LLM-generated code changes with automated evaluators (Novikov et al., 2025). Such improvement requires objective, reproducible feedback, which is often scarce in scientific tasks (Weng, 2026). In lens design, LensRSI uses simulator measurements and LensHarness to determine which search-policy update becomes active.

3 LensRSI

Before detailing LensRSI, we first formulate the lens-design problem and the recursive policy-improvement objective in Sec. 3.1. We then present the LensRSI–LensHarness pipeline in Sec. 3.2, followed by policy improvement from optical feedback in Sec. 3.3 and LensHarness in Sec. 3.4.

3.1 Formulation

The proposed LensRSI operates at two optimization levels. *(i)* Within each task, an agent searches for a lens prescription that satisfies the requested optical specification. *(ii)* And across tasks, the system updates the search policy that determines which lens edits the agent should attempt.

Lens Design. We focus on refractive lens design under a geometrical-optics model in this work. We describe each task using the core specifications varied in our experiments. These quantities capture the lens structure, angular coverage, image scale, aperture, image quality, and mechanical envelope that are central to lens design. A complete optical specification may also include requirements such as distortion, relative illumination, and telecentricity, and we omit them from the notation because they are not varied as task inputs in our evaluation. We encode the resulting structured specification as

$$x = \left(n_x, \Phi_x, N_x, f_x^{\mathrm{tar}}, r_x^{\mathrm{tar}}, L_x^{\max}, D_x^{\max}\right), \tag{1}$$

where n_x is the requested number of lens elements, Φ_x is the FoV, N_x is the f-number (f/#), and f_x^{tar} is the target EFL. The remaining terms specify the target RMS spot radius, maximum total track, and maximum clear diameter. A task may omit optional mechanical limits. We represent a lens with

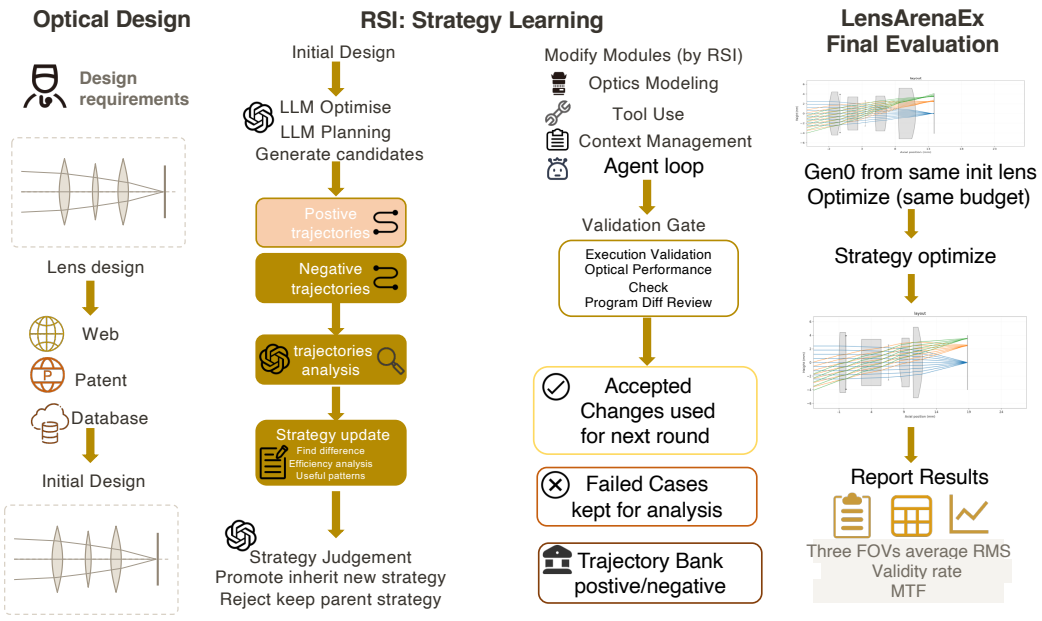

Figure 2: **Overview of our proposed LensRSI.** The agent learns strategy updates from positive and negative design trajectories, admits only validated module changes into the next round, and evaluates the final policy on LensArenaEx.

m optical surfaces as $\theta = \{R_i, d_i, g_i, \kappa_i, \mathbf{a}_i\}_{i=1}^{m}$, where R_i, d_i, g_i, κ_i, and $\mathbf{a}_i$ denote surface radius, axial spacing, material, conic constant, and optional aspheric coefficients.

Given θ, the physics simulator evaluates all prescribed fields and wavelengths. We summarize lens quality with a single optical objective $\mathcal{L}_{\text{opt}}$. Its primary metric is the mean RMS spot radius over three sampled fields, with the worst wavelength retained within each field. MTF at 50, 100, and 200 lp/mm provides a bounded auxiliary signal for contrast preservation and breaks ties between designs with nearly identical RMS (Kingslake & Johnson, 2009). Invalid ray traces or violations of the requested EFL, f/#, and mechanical limits are treated as infeasible. The single-task design problem is

$$\theta^{\star}(x) = \arg\min_{\theta \in \Theta_x} \mathcal{L}_{\text{opt}}(\theta; x). \tag{2}$$

The complete objective and optical constraints are defined in the supplementary material. A design succeeds only when it reaches the requested RMS target and passes all optical checks. We focus on nominal lens design, while production-ready optics also require tolerance analyses for fabrication, assembly, alignment, and environmental variations that are outside our present scope.

Recursive Policy Improvement. Standard automatic lens design optimizes θ for one task. LensRSI instead treats the cross-task search policy as the only recursively inherited object. We write the active policy as $\pi_t = (M_t, C_t)$, where M_t is state-conditioned action memory and C_t contains white-listed search controls. Let Θ_x^0 denote the set of starting prescriptions produced for task x and frozen before recursive improvement. Every parent and candidate run begins from the same Θ_x^0, and no optimized prescription is reused as the starting point of a later round.

We perform N_{RSI} candidate-policy rounds, indexed by $t \in \{0, \ldots, N_{\text{RSI}} - 1\}$. At round t and search step k, the policy selects a bounded lens edit

$$a_k = \pi_t(s_k, x), \qquad (s_{k+1}, \mathbf{m}_{k+1}) = \mathcal{E}(s_k, a_k; x), \tag{3}$$

where s_k contains the current prescription, measured optical state, and search history, a_k edits an allowed optical parameter, and $\mathbf{m}_{k+1}$ contains simulator-measured RMS, MTF, EFL, f/#, geometry, and trace validity. Running the policy from Θ_x^0 under a fixed call budget produces a task-local final prescription $\theta_{\pi_t}(x; \Theta_x^0)$ and a simulator-grounded trajectory. The active parent trajectories collected on the development tasks at round t form $\mathcal{D}_t$, from which the system proposes a candidate policy

$$\widetilde{\pi}_{t+1} = G(\pi_t, \mathcal{D}_t). \tag{4}$$

The candidate cannot activate itself. The fixed LENSHARNESS implements a verifier V that compares the parent and candidate on paired development and promotion-validation tasks under matched simulator budgets. The complete promotion rule is defined in Sec. 3.4. The policy active in the next round is

$$\pi_{t+1} = \begin{cases} \widetilde{\pi}_{t+1}, & V(\pi_t, \widetilde{\pi}_{t+1}) = 1, \\ \pi_t, & \text{otherwise.} \end{cases} \tag{5}$$

This separation distinguishes prescription optimization within a task from policy improvement across tasks. After all N_{RSI} rounds, the final active policy is frozen and evaluated on unseen test tasks that never contribute to $\mathcal{D}_t$ or the promotion decision.

3.2 LENSRSI–LENSHARNESS PIPELINE

Figure 2 organizes our framework into a task-level lens-design loop and a cross-task policy-improvement loop. LENSRSI proposes changes to the search policy, while LENSHARNESS enforces fixed execution and verification rules at both the prescription and policy levels.

Task-Level Optical Design Loop. Given a natural-language request, the agent parses it into a structured specification, as defined in Sec. 3.1. The retrieval stage follows the workflow of optical engineers, who commonly adapt a similar existing lens instead of designing from an arbitrary prescription (Kingslake & Johnson, 2009). Using retrieval-augmented generation (RAG) (Lewis et al., 2020), the agent retrieves multiple starting forms from a heterogeneous pool. This pool combines classical lens designs, our curated patent prescriptions, related systems found through online search, and prescriptions generated by ab initio optimization (Yang et al., 2024). The lens-design space is vast and tightly constrained, so even structurally similar prescriptions rarely satisfy a new combination of EFL, f/#, FoV, and image-quality requirements without substantial modification. Retrieved prescriptions provide only initial structures, and the agent makes the main design decisions during subsequent refinement. For RSI, these starting forms are frozen before policy comparison, so every policy round receives the same initialization.

Starting from these seeds, the agent runs parallel closed-loop refinement branches under the current policy π_t. At each step, the policy selects a structured edit defined by its parameter type, surface location, direction, and step size. The agent then invokes the simulator to trace the edited prescription and obtain feedback on its design quality and physical validity. Based on this feedback, LENSHARNESS either accepts the candidate as the next state or restores the previous prescription. After the branches complete within the assigned simulator budget, LENSHARNESS retains the best valid prescription. It also records accepted and rejected edits with their before-and-after measurements as simulator-grounded trajectories for subsequent cross-task policy improvement.

Cross-Task Agent–Harness Loop. After the active parent completes the development tasks in round t, LENSRSI aggregates its state–action–outcome trajectories to identify search decisions that transfer across tasks. It compiles these records into an isolated candidate policy $\widetilde{\pi}_{t+1}$. The candidate may change the state-conditioned action memory and allowed search controls. The simulator, frozen starting prescriptions, structured specifications, task split, and evaluation budget are held fixed by LENSHARNESS, which runs the parent and candidate independently from the same starting points on paired development and promotion-validation tasks. An accepted candidate becomes the active policy π_{t+1}, while a rejected candidate leaves π_t unchanged. Prescription updates $\theta_k \to \theta_{k+1}$ occur only within an individual task run, whereas $\pi_t \to \pi_{t+1}$ is the sole cross-round recursive update. We will explain candidate policy construction in Sec. 3.3 and its verification and inheritance in Sec. 3.4.

3.3 POLICY IMPROVEMENT FROM OPTICAL FEEDBACK

Optical Actions and Improvement Scoring. Within a lens-design task, the agent modifies a task-specific prescription that cannot be directly reused for another task. However, the measured outcome of each edit can help prioritize similar actions in later searches. We represent each executed edit as

$$z(a_k) = (q_k, u_k, \ell_k, \sigma_k), \tag{6}$$

where q_k is the parameter type, u_k is the edit direction or conic-sign class, ℓ_k is the relative surface region (front, middle, or rear), and σ_k is the search-scale bin. The typed action vocabulary covers radius, axial-spacing, and conic edits. Relative surface regions make these keys comparable across prescriptions with different numbers of surfaces.

The same edit can have different effects at different stages of a search. We therefore pair $z(a_k)$ with a compact optical state ξ_k that records the search phase, the RMS-to-target ratio, the MTF margin, and ray-trace validity. Field- and wavelength-localized measurements from the same simulator call provide additional context without extra optical evaluation.

Each retained action receives a simulator-derived improvement score. Let $J_{\mathrm{RMS}}(\theta; x)$ denote the three-field mean RMS, let $\Delta R_k = 10^3[J_{\mathrm{RMS}}(\theta_k; x) - J_{\mathrm{RMS}}(\widehat{\theta}_{k+1}; x)]$ be its change in micrometers, and let $\Delta M_{k,\nu}$ be the corresponding MTF change at frequency ν. The immediate optical score is

$$g_k = b_k \left[\Delta R_k + \Psi_{\mathrm{MTF}}(\Delta \mathbf{M}_k)\right], \qquad \Psi_{\mathrm{MTF}}(\Delta \mathbf{M}_k) = \mathrm{clip}\left(\sum_{\nu \in \{50,100,200\}} \omega_\nu \Delta M_{k,\nu}, -c, c\right), \tag{7}$$

where b_k indicates that the transition has valid optical measurements and is retained by LensHarness. We use $\boldsymbol{\omega} = (0.5, 0.3, 0.2)$ for MTF50, MTF100, and MTF200. The bounded map Ψ_{MTF} places MTF change on the RMS scoring scale, while cap c restricts it to near-RMS tie breaking and prevents MTF gains from compensating for materially worse RMS.

An action may enable a later improvement even when its immediate score is small or negative. We therefore combine g_k with its discounted downstream outcome G_k:

$$G_k = \sum_{j=k}^{K_x - 1} \gamma^{j-k} g_j + \gamma^{K_x - k} g_x^{\mathrm{term}}, \qquad I_k = \mathrm{clip}\left(\frac{(1-\beta) g_k + \beta G_k}{\delta}, -1, 1\right), \tag{8}$$

where K_x is the number of measured edits and g_x^{term} is the measured correction from the last retained state to the final selected prescription. Rejected or invalid proposals receive zero score, while retained harmful paths can receive a negative score.

Candidate Policy Construction. Across active-parent development trajectories, LensRSI proposes an update to the state-conditioned action score $Q_t(\xi, z)$. For the new observations $\mathcal{B}_t(\xi, z)$, the candidate table is

$$\widetilde{Q}_{t+1}(\xi, z) = \frac{(\alpha + n_t(\xi, z)) Q_t(\xi, z) + \sum_{I \in \mathcal{B}_t(\xi, z)} I}{\alpha + n_t(\xi, z) + |\mathcal{B}_t(\xi, z)|}, \tag{9}$$

where $n_t(\xi, z)$ is the accumulated support and α controls shrinkage toward the inherited score. The framework fixes the state bins, action vocabulary, sparse-state fallback, and writable-control whitelist. A candidate updates only the inherited memory and permitted control values, forming $\widetilde{\pi}_{t+1} = (\widetilde{M}_{t+1}, \widetilde{C}_{t+1})$ from Development evidence. It cannot alter the simulator, evaluation rules, or task splits and remains isolated until verification. If rejected, (M_t, C_t) remains active. During execution, the scores prioritize edits while preserving baseline coverage and condition the allowed optimizer-control proposal.

3.4 LensHarness

We introduce LensHarness, a lens-design-specific execution and verification environment that separates proposing a search-policy update from authorizing its inheritance. It applies fixed execution and evaluation rules outside the candidate policy's writable scope.

Action-Level Optical Verification. LensHarness evaluates each retrieved seed and proposed edit as a complete lens prescription. It treats the current prescription as a no-op candidate, requires valid field traces and specification compliance, and uses three-field mean RMS as the primary selection metric. MTF50, MTF100, and MTF200 provide a bounded auxiliary signal and a tie breaker inside a narrow RMS window. A bounded, physically valid bridge may be retained despite a small immediate RMS increase when it enables a better downstream state. The prescription transition is

$$\theta_{k+1} = \begin{cases} \widehat{\theta}_{k+1}, & b_k = 1, \\ \theta_k, & b_k = 0. \end{cases} \tag{10}$$

The final prescription is also checked against the full structured specification. Every prescription evaluation enters the same budget ledger, including evaluated seed candidates and rejected or invalid

edits. The recorded action, before-and-after measurements, and gate decision form the trajectory used in Sec. 3.3.

Policy-Level Verification and Recursive Inheritance. A candidate policy remains inactive while LENSHARNESS compares it with its parent under matched budgets on paired development tasks and the 120-task LENSARENA promotion-validation benchmark (Sun et al., 2026). LENSARENA is used only for parent–candidate promotion: its trajectories never enter $\mathcal{D}_t$ or policy memory, so it is not held-out evaluation. LENSARENAEX remains inaccessible until the final policy is frozen. Let Δ_{dev} and Δ_{val} denote mean paired RMS differences, computed as candidate minus parent. The promotion decision is

$$V(\pi_t, \widetilde{\pi}_{t+1}) = 1 \quad \Longleftrightarrow \quad \Delta_{\text{dev}} < -\epsilon_{\text{dev}} \ \wedge \Delta_{\text{val}} \leq \epsilon_{\text{val}} \ \wedge G_{\text{opt}} = 1 \ \wedge \ G_{\text{bud}} = 1, \tag{11}$$

where the thresholds and audit rules are fixed before RSI and applied unchanged across rounds. G_{opt} requires specification-valid endpoints and no newly invalid tasks, while G_{bud} enforces the matched-budget audit. Success rate and MTF remain diagnostics rather than hard promotion conditions.

If any condition fails, π_t remains active. Otherwise, the candidate is committed as π_{t+1} and becomes the next parent. This verified policy inheritance is the recursive step in LENSRSI. After N_{RSI} rounds, the final policy is frozen for held-out LENSARENAEX evaluation.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Implementation Details. To ensure reproducibility, we use the open-source Optiland simulator (Harrison, 2026). RAG retrieves five starting prescriptions per task, which the agent refines in parallel before selecting the best valid result. Each method receives a per-task budget of B simulator calls, counting every complete prescription evaluation. Across N_{RSI} rounds, parent and candidate policies are compared from identical initial prescriptions under matched budgets. After RSI completes, saved policies are evaluated on LENSARENAEX to obtain the transfer curve, without influencing policy generation, promotion, or round-count selection. Optimizer settings, execution controls, and detailed budget accounting are provided in the supplementary material.

Benchmarks. The 120-task LENSARENA benchmark serves exclusively as promotion-validation (Sun et al., 2026): aggregate outcomes determine whether a candidate is committed, while task trajectories never enter policy memory. Using empirical evaluation feedback for candidate selection follows prior self-improving agents (Yin et al., 2025; Zhang et al., 2026). Because LENSARENA participates in selection, we report it for transparency rather than as held-out evidence. LENSARENAEX contains 150 more challenging tasks with extreme specifications and remains unseen until the final policy is frozen, providing our held-out test of transfer.

Evaluation Metrics. We report average RMS spot radius (Avg. RMS) and success rate (Success). For each successful design, RMS is measured at normalized field positions 0, 0.7, and 1.0, taking the worst wavelength at each field before averaging. Avg. RMS is averaged over successful designs only, with lower values indicating tighter ray convergence. A task succeeds when its final design passes the optical checks and satisfies $J_{\text{RMS}}(\theta; x) \leq r_x^{\text{tar}}$, the task-specific RMS threshold. Success is the percentage of all tasks meeting these conditions.

Compared Methods. We compare LENSRSI with QGSO, an optimization-based quasi-global search method (Gao et al., 2025a); DeepLens, a learning-based method for ab initio lens design (Yang et al., 2024); and LENSDESIGNER, a prior natural-language optical design agent (Sun et al., 2026). We also include an in-context-learning baseline in which a foundation LLM solves each task independently without inheriting a search policy across tasks.

4.2 RESULTS

Comparison on LENSARENA and LENSARENAEX. Table 1 shows that LENSRSI improves promotion-validation performance on LENSARENA and, more importantly, transfer to held-out LENSARENAEX. With `gpt-5.6-sol`, it reduces Avg. RMS by 20.8% and 44.2%, respectively, relative to the strongest external baseline, LENSDESIGNER with `gemini-3.7-flash`. Success

Table 1: Results on the LENSARENA promotion-validation benchmark and held-out LENSARENAEX benchmark. Best in **bold**.

Method	LENSARENA Avg. RMS (μm) ↓	LENSARENA Success (%) ↑	LENSARENAEX Avg. RMS (μm) ↓	LENSARENAEX Success (%) ↑
[Existing Baselines]				
QGSO	18.2	20.8	485.6	15.3
DeepLens	35.4	1.7	379.7	22.7
[In-Context Learning Baseline]				
`gemini-3.1-pro-preview`	354.4	21.7	1332.5	4.7
LENSDESIGNER				
w/ `gemini-3.7-flash`	9.6	52.5	281.2	44.7
w/ `gemini-3.1-pro-preview`	13.7	45.8	356.9	38.0
LENSRSI (Ours)				
w/ `gemini-3.7-flash`	7.8	76.7	184.4	58.7
w/ `gemini-3.1-pro-preview`	10.4	66.7	293.5	50.7
w/ `kimi-k3`	9.5	74.2	269.6	56.7
w/ `gpt-5.6-sol`	**7.6**	**81.7**	**157.0**	**64.7**
w/ `deepseek-v4-flash`	9.8	70.0	277.8	52.7

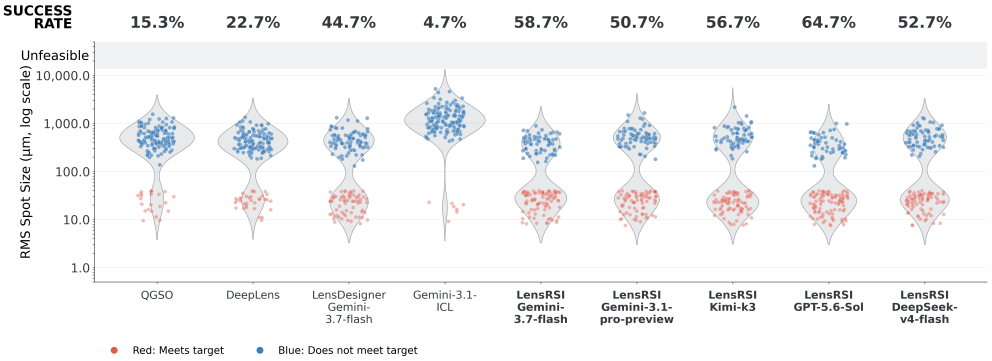

Figure 3: **Distribution of lens design results on LENSARENAEX.** Each point represents the result of an individual task, while the violin shapes summarize the overall distribution for each method. LENSRSI shifts the distributions toward lower RMS spot radii and achieves higher success rates.

rates rise from 52.5% to 81.7% and from 44.7% to 64.7%. With the same `gemini-3.7-flash` backbone, LENSRSI still reduces Avg. RMS by 18.8% and 34.4%. Figure 3 shows that all five LENSRSI configurations outperform the baselines in success rate on held-out LENSARENAEX; the best reaches 64.7%, 20.0 percentage points above the strongest baseline. These results demonstrate improved design quality and transfer to extreme unseen specifications.

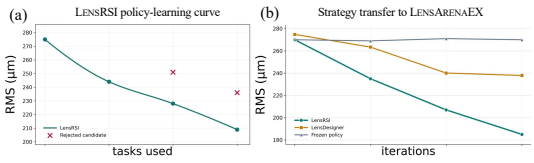

Figure 5: **Policy improvement and transfer.** (a) Verified policy updates. (b) Transfer to LENSARENAEX.

Qualitative Results. Figure 4 compares lens layouts and spot diagrams for a challenging wide-angle task from LENSARENAEX. All methods target the same specifications. LENSRSI achieves an on-axis RMS spot radius of 12.15 μm, whereas QGSO, DeepLens, and LENSDESIGNER all exceed 90 μm. The advantage persists off-axis, with RMS values of 13.10 and 24.88 μm at normalized fields of 0.7 and 1.0, respectively. The spot diagrams show tighter ray concentration in physical coordinates across three fields, illustrating improved imaging quality for this extreme specification.

4.3 ABLATION STUDY

Table 2 evaluates the components of LENSRSI using `gemini-3.7-flash` and matched simulation budgets. *Fixed LensDesigner Skill* retains our task-solving pipeline but uses fixed skills from

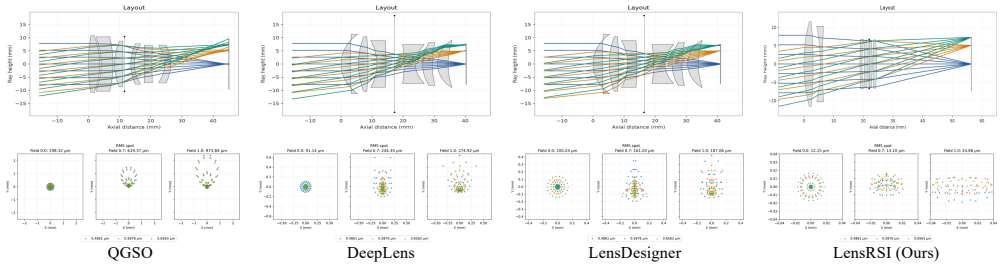

Figure 4: **Visualized results of lens layouts and spot diagrams from the LENSARENAEX benchmark.** All four designs target a 130° FoV, an EFL of 40.83 mm, and an f-number of 2.6. The final RMS spot radius is reported alongside each layout.

Table 2: **Ablation study of the core components of our LENSRSI on both LENSARENA and LENSARENAEX benchmarks.** Best in **Bold**.

	LENSARENA		LENSARENAEX	
Method	Avg. RMS (μm) ↓	Success (%) ↑	Avg. RMS (μm) ↓	Success (%) ↑
[Policy Improvement]				
Non-evolving	17.3	46.7	521.4	29.3
One-step	10.4	67.5	439.7	44.0
State-blind	12.9	54.2	336.2	38.0
Immediate-only	11.7	63.3	276.8	40.7
Memory-only	9.5	69.2	258.6	48.7
Program-only	13.6	59.2	317.4	35.3
Fixed LensDesigner Skill	9.9	60.8	285.8	40.7
[Feedback and Verification]				
Random update	15.1	50.8	364.9	33.3
No promotion gate	10.8	64.2	252.3	46.0
LENSRSI (Full)	**7.8**	**76.7**	**184.4**	**58.7**

LENSDESIGNER without RSI. Compared with this variant, the full method reduces Avg. RMS by 21.2% on LENSARENA and 35.5% on LENSARENAEX, supporting improvement beyond reusing existing skills. State conditioning also matters: relative to *State-blind*, the full method lowers Avg. RMS by 45.2% on LENSARENAEX, while success rises from 38.0% to 58.7%. Finally, removing the promotion gate reduces success on this benchmark from 58.7% to 46.0%. The full method also achieves 26.9% lower Avg. RMS than *No promotion gate*, supporting independent verification before policy inheritance.

Effectiveness of RSI and LENSHARNESS. Figure 5(a) shows a steady decrease in RMS as LENSRSI learns from more design tasks. The rejected candidates marked by red crosses fail to satisfy the promotion criteria. LENSHARNESS rejects these updates rather than passing them to subsequent rounds. Figure 5(b) shows that the policy improvements transfer to LENSARENAEX, where RMS decreases across successive iterations. The frozen policy remains nearly unchanged, indicating that repeated evaluation alone does not improve performance. Although LENSDESIGNER also benefits from accumulating reusable skills, its RMS reduction is smaller and levels off earlier. In contrast, LENSRSI continues to improve, supporting the value of recursive policy updates combined with independent verification.

5 CONCLUSION

We introduce LENSRSI, which learns state-conditioned search policies from optical trajectories and verifies updates through LENSHARNESS. It achieves the lowest Avg. RMS and highest success rates on promotion-validation LENSARENA and transfers to held-out LENSARENAEX. Ablations and learning curves support verified recursive policy improvement, demonstrating the potential of RSI for optical design agents.

Discussion and Limitations. Our method uses predefined optical actions and empirical verification, which cannot guarantee improvement on every unseen task. Evaluation is limited to nominal refractive lens design in simulation, without fabrication tolerances, assembly errors, or physical prototypes. Broader action spaces, tolerance-aware verification, and experimental validation remain future directions.

AI USE STATEMENT

In this work, we used generative AI tools for several research tasks that require disclosure. These tools assisted with refining parts of the conceptual framework, providing feedback on the method and experimental workflow, and implementing the method through code generation, review, and debugging. They also assisted with partitioning the benchmark into development and evaluation subsets.

We did not use generative AI to retrieve or discover related work, identify relevant literature, search for source material, summarize existing literature, or draft sections of the manuscript. We also did not use generative AI to generate experimental measurements, propose or refine scientific hypotheses, formulate mathematical claims, provide ingredients for mathematical proofs, write proofs, conduct qualitative or thematic data analysis, or interpret the reported results. Tasks involving formal theory, mathematical proofs, and qualitative analysis are not applicable to this work.

For tasks with recommended disclosure, generative AI helped format references and improve the grammar and wording of author-written text. We did not use generative AI to create or modify scientific figures.

The authors reviewed all AI-assisted work. We defined the final method, experimental protocol, and data-splitting criteria, reviewed the resulting task records and partitions, inspected and tested every AI-assisted code change, executed the experiments, and independently verified the reported results. We also examined the cited sources and checked all AI-assisted text for technical accuracy, unsupported claims, and formatting errors. No AI-generated output was treated as experimental evidence. The authors take full responsibility for the final text, claims, citations, code, data partitions, and other artifacts in this submission.

ETHICS STATEMENT

This work complies with the ICLR Code of Ethics. It studies automatic lens design in a physics-based simulation environment and involves no human subjects, personal data, or user studies. The generated prescriptions describe nominal optical systems rather than fabrication-ready products. Designs intended for safety-critical or commercial use require independent engineering review, tolerance and manufacturability analyses, and compliance with applicable safety standards. Public patent records and online optical resources are used only to provide design context and starting structures. Any release or deployment must respect applicable licenses, confidentiality requirements, and intellectual-property rights, and we will distribute only materials for which redistribution is permitted.

REPRODUCIBILITY STATEMENT

We describe the task formulation, optical objective, policy update, candidate isolation, two-level verification, and matched-budget accounting in Secs. 3.1–3.4. The supplementary material specifies state discretization and fallback rules, policy-update hyperparameters, writable-control restrictions, promotion thresholds, simulator settings, initialization, data splits, and budget accounting. The code and redistributable data will be released upon acceptance.

REFERENCES

Geoffroi Côté, Jean-François Lalonde, and Simon Thibault. Extrapolating from lens design databases using deep learning. *Optics Express*, 27(20):28279–28292, 2019.

Geoffroi Côté, Jean-François Lalonde, and Simon Thibault. Deep learning-enabled framework for automatic lens design starting point generation. *Optics Express*, 29(3):3841–3854, 2021.

Yao Gao, Qi Jiang, Shaohua Gao, Lei Sun, Kailun Yang, and Kaiwei Wang. Exploring quasi-global solutions to compound lens based computational imaging systems. *IEEE Transactions on Computational Imaging*, 2025a.

Yao Gao, Lei Sun, Shaohua Gao, Qi Jiang, Kailun Yang, Weijian Hu, Xiaolong Qian, Wenyong Li, Luc Van Gool, and Kaiwei Wang. Neuro-inspired automated lens design. *arXiv preprint arXiv:2510.09979*, 2025b.

Irving John Good. Speculations concerning the first ultraintelligent machine. In *Advances in computers*, volume 6, pp. 31–88. Elsevier, 1966.

Dabo Guo, Liang Yin, and Guang Yuan. New automatic optical design method based on combination of particle swarm optimization and least squares. *Optics Express*, 27(12):17027–17040, 2019.

Kramer Harrison. Optiland. `https://github.com/optiland/optiland`, 2026. Open-source optical design software.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, volume 2025, pp. 21344–21377, 2025.

Rudolf Kingslake and R Barry Johnson. *Lens design fundamentals*. academic press, 2009.

Douglas B Lenat. Eurisko: a program that learns new heuristics and domain concepts: the nature of heuristics iii: program design and results. *Artificial intelligence*, 21(1-2):61–98, 1983.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Xiaobing Liu, Xingxiang Zhang, Tianjiao Fu, Kaizhi Wang, Fukun Sun, Tongzheng Bai, and Duo Wang. Global information selectively guided gradient descent for ab initio optical design. *Optics & Laser Technology*, 184:112497, 2025.

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. Towards end-to-end automation of ai research. *Nature*, 651(8107):914–919, 2026.

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.

Jürgen Schmidhuber. Godel machines: Self-referential universal problem solvers making provably optimal self-improvements. *arXiv preprint cs.LO/0309048*, 2003.

Jürgen Schmidhuber. Optimal ordered problem solver. *Machine Learning*, 54(3):211–254, 2004.

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

Lei Sun, Haoran Liang, Dannong Xu, Yao Gao, Yuyu Geng, Jinjin Gu, Kaiwei Wang, Danda Pani Paudel, and Luc Van Gool. LensDesigner: A self-improving agent for optical lens design, 2026. URL `https://Danielxu0208.github.io/LensDesigner/`.

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

Lilian Weng. Harness engineering for self-improvement. *lilianweng.github.io*, July 2026. URL `https://lilianweng.github.io/posts/2026-07-04-harness/`.

Tong Yang, Guo-Fan Jin, and Jun Zhu. Automated design of freeform imaging systems. *Light: Science & Applications*, 6(10):e17081–e17081, 2017.

Xinge Yang, Qiang Fu, and Wolfgang Heidrich. Curriculum learning for ab initio deep learned refractive optics. *Nature communications*, 15(1):6572, 2024.

Xunjian Yin, Xinyi Wang, Liangming Pan, Li Lin, Xiaojun Wan, and William Yang Wang. Gödel agent: A self-referential agent framework for recursively self-improvement. In *Annual Meeting of the Association for Computational Linguistics*, 2025. URL https://api.semanticscholar.org/CorpusID:280017772.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-taught optimizer (stop): Recursively self-improving code generation. *arXiv preprint arXiv:2310.02304*, 2023.

Benqi Zhang, Guofan Jin, and Jun Zhu. Towards automatic freeform optics design: coarse and fine search of the three-mirror solution space. *Light: Science & Applications*, 10(1):65, 2021.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin gödel machine: open-ended evolution of self-improving agents. In *International Conference on Learning Representations*, volume 2026, pp. 104223–104294, 2026.

Jiajun Zhang, Zhaofeng Cen, and Xiaotong Li. Automated design of machine vision lens based on the combination of particle swarm optimization and damped least squares. In *Optical Design and Testing X*, volume 11548, pp. 261–272. SPIE, 2020.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow generation. In *International Conference on Learning Representations*, volume 2025, pp. 34040–34077, 2025.