IRest-VL: Decoupled Multimodal Diagnosis and Programmatic Restoration for Infrared Images

Anonymous authors
Paper under double-blind review

Abstract

Infrared images can suffer from composite degradations, in which multiple noise and blur primitives act in specific orders. Existing end-to-end methods address diagnosis or restoration alone while agents that perform both require multi-round planning and heavyweight tools with coarse or fixed parameters, making them ill-suited to lightweight infrared applications. We introduce IRest-VL, a unified multimodal large language model (MLLM) that diagnoses types, severities, and order while generating a lightweight, standalone restoration program. Building it poses two obstacles: efficiently constructing code supervision in a large mixed discrete–continuous space of tools, program orders, and parameters, and avoiding task interference and the weak compositional generalization that would arise if diagnosis and code were serialized into a single autoregressive target. We address them with experience-accumulated Bayesian optimization (EABO) and a decoupled structured-generation architecture that models degradation through local attributes and directed transitions, rather than serializing diagnosis into restoration-code generation. EABO constructs supervision 27.3× faster than iterative MLLM search, without requiring MLLM calls, while yielding higher downstream restoration SSIM. On synthetic composites applied to real infrared backgrounds, structured diagnosis raises exact diagnosis accuracy on held-out out-of-distribution (OOD) compositions from 7.11% under the direct autoregressive (AR) diagnosis-and-restoration baseline to 28.11%, while maintaining comparable restoration quality.

1 Introduction

Infrared imaging supports sensing in low-light and adverse environments, yet its images may undergo *composite degradation* as multiple noise and blur processes of varying severities act in a degradation order that is not observable from the input. A practical infrared system should therefore identify *what* happened and determine *how* the image should be restored. Infrared systems often operate at the edge, so restoration programs must be lightweight.

Existing methods address only parts of this goal. Diagnosis estimators usually predict one degradation type or level rather than multiple components and their relations (Shiblee et al., 2023; Geng et al., 2020; Kavitha et al., 2023). CNN-, Transformer-, and diffusion-based restorers target one degradation family or an implicit mixture and output an image without explaining the constituent degradations, their severities, or their degradation order (Zhang et al., 2017; Guo et al., 2019; Liang et al., 2021; Zamir et al., 2022; Jiang et al., 2025). Lightweight networks further reduce the cost of one fixed restorer (Liu et al., 2023; Zhang et al., 2024). Vision-language restoration models use degradation-aware representations or descriptions to condition specialized restorers (Luo et al., 2024; Jin et al., 2024). MLLM-based agents further couple degradation perception with restoration-model selection and sequential execution planning (Chen et al., 2024; Zhou et al., 2025). Together, these methods leave open the need for a unified system that connects degradation understanding to executable restoration. For infrared edge deployment, however, the resulting action should

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.

be a lightweight restoration program that is standalone at execution time: once generated, it can process an input image using the registered tools without another MLLM call.

We instead pursue "light tools, rich parameters": one MLLM predicts the degradation components, severities, and underlying degradation order together with image-adaptive restoration code assembled from classical operators and lightweight networks. The program exposes its tools, execution order, and numerical arguments, and is standalone at inference: after one-time generation, it runs using the registered lightweight tools without further MLLM calls. Building this unified model creates two technical challenges. First, image-specific program labels must be discovered by executing candidates in a large conditional mixed discrete–continuous space of tool choices, program orders, and parameters. Exhaustive enumeration is costly, while independent per-image search may revisit configurations that were effective on similar degradation pipelines. Second, naively serializing diagnosis and code into one autoregressive target couples a nearly unique symbolic diagnosis with a functionally non-unique program. Their differences in structure, length, and evaluation criteria can cause task interference. Moreover, whole-pipeline diagnosis may favor memorization of seen compositions over recombination into unseen degradation orders.

We propose IRest-VL to address these two barriers within a shared multimodal model. Offline, EABO uses the tree-structured Parzen estimator (TPE) (Bergstra et al., 2011; Akiba et al., 2019) for the conditional mixed space and reuses verified configurations as budgeted cross-sample warm starts, thereby avoiding a fresh initialization for samples with matching history and improving supervision-construction efficiency without MLLM calls. Online, IRest-VL uses a decoupled structured-generation architecture with a shared visual encoder and two task-specific output paths. A structured diagnosis head predicts local degradation attributes and directed transitions, while a code-only language branch generates the executable restoration program without receiving the diagnosis as a serialized prefix. The shared encoder allows the two branches to benefit from common visual representations, while the task-specific branches prevent the diagnosis target from competing with code generation. Our contributions are threefold: 1) we formulate composite-degradation diagnosis and lightweight programmatic restoration as a unified infrared MLLM task; 2) we propose EABO to reuse verified programs for efficient parameter-rich supervision; and 3) we propose a decoupled architecture, improving OOD generalization and mitigating task competition.

2 Related Work

Degradation diagnosis and end-to-end restoration. Degradation diagnosis estimates noise type, level, or image quality. MobileNoiseNet (Shiblee et al., 2023) classifies noise types, while other estimators classify levels (Geng et al., 2020) or identify noise in natural and medical images (Kavitha et al., 2023). These outputs stop at labels or levels rather than a multi-primitive degradation pipeline. DnCNN (Zhang et al., 2017), CBDNet (Guo et al., 2019), SwinIR (Liang et al., 2021), and Restormer (Zamir et al., 2022) map degraded images to clean ones without degradation-pipeline diagnosis. Infrared designs include wavelet dual-branch diffusion (Jiang et al., 2025), MWR-Net (Qian et al., 2025), and adversarial-learning-based denoising (Yang et al., 2023a). Lightweight image-restoration alternatives include MFDNet (Liu et al., 2023) and SN-Net (Zhang et al., 2024). IRest-VL unifies diagnosis and restoration and treats parameterized callable restorers as composable tools.

Vision-language restoration and MLLM-based agents. Vision-language restorers condition networks on degradation-aware representations or descriptions, including DA-CLIP (Luo et al., 2024) and LLMRA (Jin et al., 2024). MLLM-based restoration agents orchestrate restoration through task and model planning, quality-guided sequential execution, or learned tool-calling policies. RestoreAgent (Chen et al., 2024) and FAPE-IR (Liu et al., 2026a) use language models for restoration planning, Q-Agent (Zhou et al., 2025) selects restoration steps using quality feedback, while Restore-R1 (Lu et al., 2026) and TIR-Agent (Jia et al., 2026) learn tool-calling policies with reinforcement learning. These methods connect degradation understanding to restoration actions, whereas IRest-VL targets a lightweight restoration program for infrared images without extra MLLM calls at execution time.

Multimodal tool learning and search-based supervision. General multimodal tool learners include GPT4Tools (Yang et al., 2023b), LLaVA-Plus (Liu et al., 2024), AdaTooler-V (Wang et al., 2026), ToolVision (Mao et al., 2026), and SpaceTools (Chen et al., 2026). They target generic tool use or task completion rather than infrared composite-degradation restoration, and their supervision construction does not explicitly reuse cross-sample, execution-verified restoration experience. Our search instead optimizes reference restoration in a mixed continuous–discrete space. We use TPE (Bergstra et al., 2011) through Optuna (Akiba et al., 2019) and contribute verified-configuration reuse as budgeted warm starts, not a new acquisition rule.

3 Method

To define and evaluate the joint task, we assume that each degraded infrared image has an underlying degradation pipeline $\mathcal{D}_i = (\mathbf{g}_i, \mathbf{s}_i)$, where $\mathbf{g}_i$ is the degradation-type sequence and $\mathbf{s}_i$ is the aligned degradation-severity sequence. Given a clean image $\mathbf{y}_i$, applying the operations specified by $\mathcal{D}_i$ produces the degraded observation $\mathbf{x}_i$. IRest-VL takes $\mathbf{x}_i$ as input and jointly predicts the pipeline $\hat{\mathcal{D}}_i$ and executable restoration code $\hat{C}_i$. Let $P_{\hat{C}_i}$ be the lightweight restoration program instantiated by $\hat{C}_i$; its output is $\hat{\mathbf{y}}_i = P_{\hat{C}_i}(\mathbf{x}_i)$. The task therefore requires both $\hat{\mathcal{D}}_i$ to match $\mathcal{D}_i$ and $\hat{\mathbf{y}}_i$ to approach $\mathbf{y}_i$.

3.1 EABO for Program Supervision

Empirical motivation. To analyze the limitations of ordinary BO for constructing program supervision, we examine how its selected program orders vary across samples. We construct sample pairs and partition them into different groups as shown in Fig. 1 according to their degradation pipelines. For each group and each pipeline length, we collect pairwise normalized Levenshtein similarities between the BO-selected restoration-program orders and summarize their distribution. As shown in Fig. 1, the same type and severity group reaches .90, .37, and .31 at lengths one, two, and three, respectively, compared with .35 and .28 for the same type but different severity at lengths two and three, and .001, .15, and .18 for different type sequences. These comparisons indicate cross-sample regularity: samples in the same type-and-severity group have more similar selected program orders than samples with different type sequences, while matching only the type sequence retains a smaller similarity advantage. This pattern suggests that retrieving execution-verified programs from similar samples may provide useful warm starts and reduce search effort for a new sample.

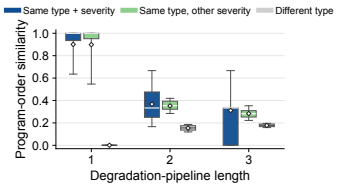

Figure 1: Cross-image reuse analysis.

Program space and optimization objective. Rather than selecting a single fixed restoration network, we search over executable programs. A program of length L is denoted as $P = (o_1, \ldots, o_L)$, where $\ell \in \{1, \ldots, L\}$ indexes an execution step and each call $o_\ell = (t_\ell, \boldsymbol{\theta}_\ell)$ specifies a restoration tool t_ℓ and its argument vector $\boldsymbol{\theta}_\ell$. For an input image $\mathbf{x}$, the program returns the identity, $P(\mathbf{x}) = \mathbf{x}$, when $L = 0$; otherwise, it applies the calls in sequence, $P(\mathbf{x}) = (o_L \circ \cdots \circ o_1)(\mathbf{x})$. The search space consists of eight lightweight callable tools (Appendix C.1), namely six classical operators and two parameter-free, fixed-weight MFDNet tools. The parameterized operators average 2.2 scalar arguments each, and all tools have a mean execution time of 0.07 s, enabling efficient restoration programs that can run standalone. Table 19 verifies the benefit of this rich-parameter design.

For training sample i, let $\mathbf{x}_i$ and $\mathbf{y}_i$ denote the degraded input and clean reference. Because a program can be evaluated only after execution, we measure its quality with the reference-based objective $J(P; \mathbf{x}, \mathbf{y}) = \mathrm{SSIM}(P(\mathbf{x}), \mathbf{y})$ and use $J(P; \mathbf{x}_i, \mathbf{y}_i)$ for sample i. This objective is used only during offline supervision construction. Under a per-image trial budget B, EABO uses Optuna's TPE to explore the conditional mixed discrete–continuous space. No-op samples directly use the identity program P_{id}, and the default budgets are listed in Appendix E. For non-empty pipelines, the annotated degradation types constrain the eligible tools and program length. Each step selects an unprocessed degradation, its associated

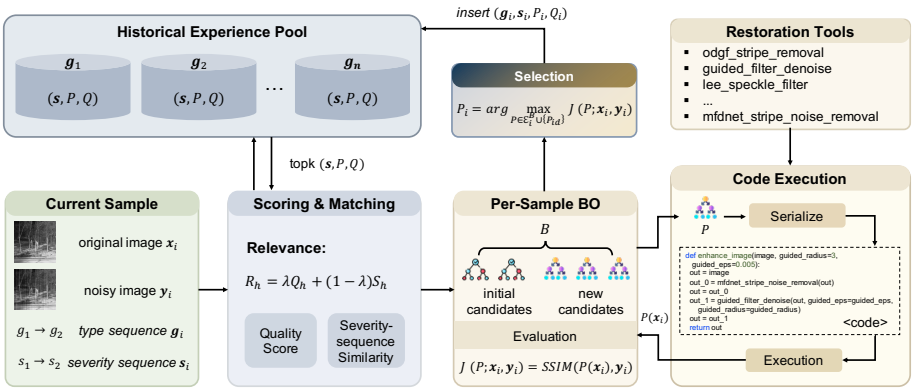

Figure 2: The flowchart of experience-accumulated Bayesian optimization (EABO).

tool, and the tool arguments, with `skip` permitted. Candidates are ranked by executed restoration quality and every selected argument is serialized into the resulting code.

Experience retrieval and warm-start. Fig. 2 summarizes the retrieval and warm-start procedure. EABO retains the same TPE acquisition rule, admissible program space, and total trial budget as ordinary per-image BO; it differs only by initializing the search with executed, quality-verified programs from earlier samples with the same type sequence. Let i index the current sample and $h < i$ a prior sample, with type–severity pairs $(\mathbf{g}_i, \mathbf{s}_i)$ and $(\mathbf{g}_h, \mathbf{s}_h)$. We retrieve only records satisfying $\mathbf{g}_h = \mathbf{g}_i$. Let $K_i = |\mathbf{g}_i|$ be the number of primitives, $k \in \{1, \dots, K_i\}$ their sequence index, P_h the best program retained for sample h, and κ the mapping from low, medium, and severe to $0, 1, 2$. With quality weight $\lambda \in [0, 1]$, the severity-sequence similarity S_h, best-program quality Q_h, and retrieval score R_h are

$$S_h = 1 - \sum_{k=1}^{K_i} |\kappa(s_{i,k}) - \kappa(s_{h,k})| / (2K_i), \quad Q_h = J(P_h; \mathbf{x}_h, \mathbf{y}_h), \quad R_h(\lambda) = \lambda Q_h + (1-\lambda) S_h. \tag{1}$$

Let $\mathcal{A}_i = \{h < i \mid \mathbf{g}_h = \mathbf{g}_i\}$ denote the matching prior-sample indices, and let angle brackets denote an ordered sequence that may contain duplicates. With at most K_{init} warm starts, the matching history and initialization list are

$$\mathcal{H}_i(\mathbf{g}_i) = \langle (\mathbf{s}_h, P_h, Q_h) \rangle_{h \in \mathcal{A}_i}, \qquad \mathcal{I}_i = \text{UniqueTopK}_{K_{\text{init}}}^{\downarrow R_h(\lambda)}(\mathcal{H}_i(\mathbf{g}_i)). \tag{2}$$

This operator ranks $\mathcal{H}_i(\mathbf{g}_i)$ by R_h, removes duplicate parameterized programs, and returns at most K_{init} programs. Changes in tools, order, or arguments define distinct programs. Ties follow insertion order. Thus, $\lambda = 0$ is severity-only and $\lambda = 1$ is quality-only.

Warm starts count toward B, so experience does not increase the search budget. If matching history is empty, the procedure reduces to ordinary TPE. Let $\mathcal{E}_i^{(B)}$ be the candidate programs executed under budget B, and let $\|$ append one record to an ordered history. The best retained program P_i, its quality Q_i, and the bank update are

$$P_i = \arg \max_{P \in \mathcal{E}_i^{(B)} \cup \{P_{\text{id}}\}} J(P; \mathbf{x}_i, \mathbf{y}_i), \; Q_i = J(P_i; \mathbf{x}_i, \mathbf{y}_i), \; \mathcal{H}_{i+1}(\mathbf{g}_i) = \mathcal{H}_i(\mathbf{g}_i) \,\|\, \langle (\mathbf{s}_i, P_i, Q_i) \rangle. \tag{3}$$

Histories for other type sequences remain unchanged. The resulting supervision label is the best observed program under the specified budget and tool space, not a theoretical optimum.

3.2 Decoupled Structured-Generation Architecture

Empirical motivation. Serializing diagnosis followed by restoration code is the most direct unified design, but the first five AR baselines in Table 1 reveal a severe compositional generalization gap. Direct AR Diagnosis and Restoration drops from .6159 full diagnosis accuracy on ID to .0711 on OOD, even though the two splits share the same primitives and severity states and differ only in their compositions. Adding attribute and transition labels to the language output provides a partial clue: placing this reasoning before diagnosis raises OOD accuracy to .0837, whereas placing it after diagnosis gives .0698, suggesting that

Table 1: Comparison of output forms in Direct AR. **D**:diagnosis, **E**:reasoning, **R**:restoration.

Method	ID				OOD			
	Attr F1	Trans F1	Full	SSIM	Attr F1	Trans F1	Full	SSIM
Direct AR D + R	.8488	.5874	.6159	.8566	.6724	.2811	.0711	.7896
Direct AR D	.8577	.6064	.6372	–	.6858	.2959	.0804	–
Direct AR R	–	–	–	.8575	–	–	–	.7954
Direct AR E → D → R	.8516	.5762	.6073	.8568	.7252	.2937	.0837	.7882
Direct AR D → E → R	.8489	.6062	.6286	.8568	.6741	.3142	.0698	.7873
IRest-VL	.8713	.5660	.6193	.8575	.7547	.5601	.2811	.7951

predictions over reusable degradation primitives can help but do not by themselves resolve the gap. This intervention also lengthens the autoregressive target, increasing both training and inference cost. These observations motivate two design requirements: expose attributes and ordered transitions as the basic primitives for compositional diagnosis, while avoiding their explicit serialization through the language decoder.

The first three rows of Table 1 further reveal task interference in the direct AR formulation. Relative to Direct AR Diagnosis and Restoration, removing restoration raises diagnosis accuracy from .6159/.0711 to .6372/.0804 on ID/OOD, while removing diagnosis raises restoration SSIM from .8566/.7896 to .8575/.7954. Thus, serializing the two targets benefits neither task. A data-level mismatch helps explain this behavior: diagnosis requires the annotated degradation pipeline, whereas restoration admits multiple functionally effective programs. Fig. 3 shows that only 22.43% of the best-observed BO programs for multi-degradation inputs exactly reverse the annotated order; primitive-set match is 49.05%, and normalized edit distance is .5043, shown as 50.43 on the figure's ×100 scale. A diagnosis prefix can therefore impose an unreliable order prior on code generation. These observations further motivate us to decouple the two tasks.

Decoupled architecture and task-specific outputs. Fig. 4 visualizes the decoupled structured-generation architecture. It uses a shared visual encoder with two task-specific branches: a structured diagnosis head and a code-only language branch. Let $\mathbf{V}_i$ denote the visual tokens and $\mathbf{m}_i$ their validity mask. The diagnosis head applies masked multi-head self-attention (MHA) followed by residual layer normalization (LN), a feed-forward network (FFN), and masked mean pooling:

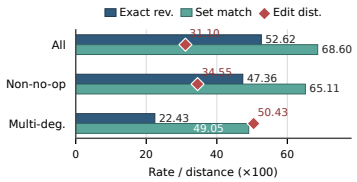

Figure 3: Program-order agreement.

$$\mathbf{U}_i = \mathrm{LN}(\mathbf{V}_i + \mathrm{MHA}(\mathbf{V}_i; \mathbf{m}_i)), \qquad \mathbf{h}_i = \mathrm{Pool}_{\mathbf{m}_i}(\mathrm{LN}(\mathbf{U}_i + \mathrm{FFN}(\mathbf{U}_i))). \tag{4}$$

The pooled representation feeds an attribute head for four states (absent/low/medium/severe) on each of five primitives and a transition head for 20 directed non-self edges. An edge is positive when the pair is adjacent in the annotated type sequence. These local predictions support valid-path reconstruction on unseen compositions. The language decoder receives the image and tool specification and generates only a parsable `enhance_image` function. Diagnosis is not serialized into its context, and only the code span contributes to the language loss.

Training and valid-path decoding. Let $\mathcal{G}$ be the primitive set, $M = |\mathcal{G}|$ its size, $j \in \{1, \ldots, M\}$ a primitive index, and e a directed non-self edge. The diagnosis targets specify one of four attribute states for each primitive and whether each directed edge is present. Let $\mathcal{L}_{\text{code}}$ be the masked code negative log-likelihood, $\mathcal{L}_{\text{attr}}$ multiclass cross-entropy, and $\mathcal{L}_{\text{trans}}$ positive-class-weighted binary cross-entropy with positive weight w_+. With loss weights λ_a and λ_t, the training objective is

$$\mathcal{L} = \mathcal{L}_{\text{code}} + \lambda_a \mathcal{L}_{\text{attr}} + \lambda_t \mathcal{L}_{\text{trans}}. \tag{5}$$

At inference, the attribute and transition heads make independent predictions, which may not form a valid degradation pipeline when combined directly. We therefore enumerate all non-repeating candidate type sequences π, including the empty sequence, up to a maximum length $L_{\max} \leq M$. This constraint prevents repeated primitives and guarantees a valid

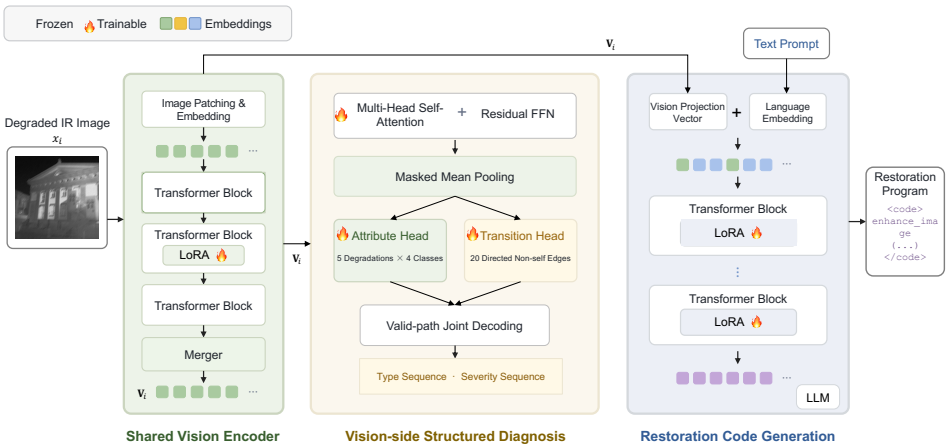

Figure 4: IRest-VL with decoupled structured-generation architecture.

ordered path. Let $E(\pi)$ denote the set of directed adjacent edges in π. For primitive j, let $c_{i,j}(\pi) = 0$ if j is absent from π; otherwise, let it be the most probable non-absent severity predicted for j. Denoting the attribute probability by $p^{\text{attr}}_{i,j}$ and the transition probability by $q^{\text{trans}}_{i,e} = \sigma(z^{\text{trans}}_{i,e})$, we score each candidate with transition weight $\beta \geq 0$:

$$\mathcal{S}_i(\pi) = \sum_{j=1}^{M} \log p^{\text{attr}}_{i,j}(c_{i,j}(\pi)) + \beta \sum_{e \in E(\pi)} \log q^{\text{trans}}_{i,e}, \quad \hat{\pi}_i = \arg\max_{\pi} \mathcal{S}_i(\pi), \tag{6}$$

The predicted severity sequence $\hat{\mathbf{s}}_i$ is obtained from the selected non-absent severity classes along $\hat{\pi}_i$. Although the data contain pipelines of length at most three, the decoder allows $L_{\max} = M = 5$ in the main experiments to avoid imposing the training-template length limit at inference. Appendix F gives the complete decoding procedure.

4 Experiments

4.1 Dataset and Settings

We use 1,503 HM-TIR images (Liu et al., 2026b) and five degradation primitives to form degradation pipelines of length at most three. Train, validation, and Test-ID (in-distribution, ID) use 78 type-sequence templates, whereas Test-OOD uses eight held-out type sequences from the same primitive and severity vocabulary. The resulting split sizes are 105,200 training samples, 750 validation samples, and 1,505 samples in each test split. For cross-dataset program reuse, we additionally use DLS-NUC-100 (He et al., 2018), IR700 (Zou et al., 2021), and LLVIP (Jia et al., 2021) as target image sets. These datasets are not used for training, BO supervision construction, or checkpoint selection. The primary model is Qwen3.5-0.8B with frozen base weights, language- and vision-side low-rank adaptation (LoRA), and a structured diagnosis head built on masked self-attention. We additionally evaluate four other MLLM backbones to test transfer. Detailed data and training protocols are given in Appendices A and E.

4.2 Comparison Methods and Evaluation Metrics

We compare lightweight learned restorers (MFDNet/SN-Net), zero-shot MLLMs, the Direct AR Diagnosis and Restoration baseline, and IRest-VL. We also report two reference oracles: the per-image best single tool and a 400-trial EABO search, both selected with clean test references and therefore not deployable methods. We report exact degradation-pipeline accuracy, pipeline-derived attribute/transition F1, peak signal-to-noise ratio (PSNR), structural similarity (SSIM), and code validity. Failures remain no-op outputs. Metric definitions and comparison conventions are in Appendix G.1.

Table 2: Overall results for infrared composite-degradation diagnosis and programmatic restoration. $F1_a$ and $F1_t$ denote attribute and transition F1.

Category	Method	ID						OOD					
		Diagnosis			Restor.			Diagnosis			Restor.		
		$F1_a$	$F1_t$	Full	PSNR	SSIM	Parse	$F1_a$	$F1_t$	Full	PSNR	SSIM	Parse
No processing	Degraded input	–	–	–	34.08	.6397	–	–	–	–	25.29	.5179	–
Best single tool	guided filter	–	–	–	29.08	.7870	–	–	–	–	27.41	.7264	–
Oracle	Per-image best single tool	–	–	–	37.50	.8454	–	–	–	–	28.25	.7858	–
	400-trial EABO oracle	–	–	–	–	.8608	–	–	–	–	–	.8095	–
Lightweight	MFDNet	–	–	–	31.80	.8305	–	–	–	–	28.48	.7669	–
	SN-Net	–	–	–	32.82	.8653	–	–	–	–	28.68	.7889	–
Zero-shot	GLM-5.2	.1407	.0582	.0000	26.80	.7113	1.000	.2050	.0697	.0007	26.10	.6953	1.000
	Qwen3.8-Max	.2960	.1311	.1050	27.44	.7262	1.000	.3542	.3382	.0551	26.22	.6611	.9953
	GPT-5.6-sol	.2791	.1569	.1037	28.11	.7462	1.000	.3476	.1350	.0133	26.99	.6969	1.000
	Claude Sonnet 5	.2280	.0925	.0252	27.86	.7372	1.000	.2630	.1126	.0073	26.43	.6964	1.000
Direct AR Diagnosis	Qwen3.5-0.8B	.8577	.6064	.6372	–	–	–	.6858	.2959	.0804	–	–	–
Direct AR Diagnosis and Restoration	Qwen3.5-0.8B	.8488	.5874	.6159	37.94	.8566	1.000	.6724	.2811	.0711	28.52	.7896	1.000
	Qwen3.5-9B	.8552	.5910	.6259	37.96	.8574	1.000	.7136	.3363	.1495	28.61	.7933	1.000
	InternVL3.5-8B	.8756	.5803	.6425	37.77	.8564	1.000	.6926	.2913	.0498	28.51	.7915	1.000
	MiMo-VL-7B	.8571	.5934	.6312	37.61	.8576	1.000	.6967	.3085	.0983	28.56	.7916	1.000
	MiniCPM-V-4.6	.9049	.6092	.6764	37.98	.8581	1.000	.7454	.3469	.1754	28.59	.7916	1.000
Our method	**IRest-VL-Qwen3.5-0.8B**	.8713	.5660	.6193	37.96	.8575	1.000	.7547	.5601	.2811	28.61	.7951	1.000
	IRest-VL-Qwen3.5-9B	.8992	.6224	.6731	37.95	.8572	1.000	.7771	.6019	.2791	28.62	.7939	1.000
	IRest-VL-InternVL3.5-8B	.8374	.5706	.5907	37.96	.8574	1.000	.7109	.6179	.2306	28.59	.7949	1.000
	IRest-VL-MiMo-VL-7B	.9033	.6109	.6698	37.98	.8581	1.000	.7762	.6333	.2399	28.65	.7935	1.000
	IRest-VL-MiniCPM-V-4.6	.9138	.5901	.6678	37.97	.8581	1.000	.7944	.5600	.3223	28.67	.7958	1.000

4.3 Overall Results: Diagnosis and Programmatic Restoration

Restoration-only and lightweight baselines. IRest-VL achieves strong restoration with lightweight executable programs, outperforming fixed and learned lightweight alternatives while approaching the search upper bound (Table 2). It improves ID/OOD SSIM over guided filtering from .7870/.7264 to .8575/.7951. It also exceeds the reference-guided per-image best-single-tool oracle on both splits (.8575/.7951 vs. .8454/.7858), showing that tool composition can outperform selecting one tool. Individual-tool results are reported in Appendix H.2. Against lightweight learned restorers, IRest-VL exceeds MFDNet on both splits and is competitive with SN-Net: SN-Net is higher on ID (.8653), while IRest-VL is higher on OOD (.7951 vs. .7889). We omit larger end-to-end restorers because their parameter and deployment budgets are not comparable to the lightweight setting. When resources permit, such models can instead be registered as additional tools. To assess how well the program generator fits the searched supervision, we compare it with the reference-guided 400-trial EABO oracle. Without clean targets at inference, IRest-VL is within .0033/.0144 SSIM of this oracle on ID/OOD (.8608/.8095), recovering most of the restoration quality found by search within the evaluated tool space. Baseline training settings are given in Appendix E.

Zero-shot MLLMs and Direct AR Diagnosis and Restoration. Among zero-shot MLLMs, Qwen3.8-Max gives the best full diagnosis accuracy on both ID and OOD (.1050/.0551), while GPT-5.6-sol gives the best restoration SSIM (.7462/.6969). Both remain substantially below IRest-VL (.6193/.2811 diagnosis and .8575/.7951 SSIM), consistent primarily with their lack of task-specific fine-tuning. Under the matched Qwen3.5-0.8B setting, Direct AR Diagnosis and Restoration and IRest-VL achieve similar ID diagnosis (.6159 vs. .6193) and restoration (.8566 vs. .8575), and remain close on OOD restoration (.7896 vs. .7951). In contrast, IRest-VL raises OOD full diagnosis accuracy from .0711 to .2811. Full diagnosis accuracy requires an exact match to the annotated degradation pipeline, whereas different restoration programs can achieve similar restoration quality. Thus, decoupling substantially improves compositional diagnosis without sacrificing restoration quality. Appendix H.1 breaks down this comparison by pipeline length.

Backbone transfer and capacity trends. Within the Qwen3.5 family, increasing the backbone from 0.8B to 9B raises ID full diagnosis accuracy from .6193 to .6731, but leaves OOD accuracy nearly unchanged (.2811 vs. .2791) and SSIM similar. Across InternVL3.5-8B, MiMo-VL-7B, and MiniCPM-V-4.6, performance is likewise not monotonic in nominal

Table 3: Effect of training-data scale on diagnosis and restoration.

Samples/image	Training samples	Epochs	Diagnosis ID Full	Diagnosis OOD Full	Restoration ID PSNR	Restoration ID SSIM	Restoration OOD PSNR	Restoration OOD SSIM
2×	2,104	100	.4611	.1435	37.53	.8459	28.11	.7787
20×	21,040	10	.5907	.2472	37.89	.8555	28.54	.7911
100×	105,200	2	.6193	.2811	37.96	.8575	28.61	.7951

Table 4: Visual and language LoRA ablation.

Adaptation	Split	Attr F1	Trans F1	Full	PSNR	SSIM	Parse/pass rate
Vision + language	ID	.8713	.5660	.6193	37.96	.8575	1.000/1.000
Vision only	ID	.8086	.5135	.5488	37.54	.8461	1.000/1.000
Language only	ID	.6784	.4427	.4312	36.87	.8497	1.000/1.000
Vision + language	OOD	.7547	.5601	.2811	28.61	.7951	1.000/1.000
Vision only	OOD	.7009	.5423	.2312	28.07	.7645	1.000/.9967
Language only	OOD	.6059	.4456	.1761	28.40	.7845	1.000/1.000

size. Nevertheless, every structured variant improves OOD full diagnosis over its matched Direct AR Diagnosis and Restoration counterpart: the structured models span .2306–.3223, versus .0498–.1754 for Direct AR Diagnosis and Restoration, while OOD SSIM remains tightly grouped at .7935–.7958. These results support transfer across model families, but not a simple scaling law.

Inference and lightweight execution. At an inference batch size of two, decoupling reduces Qwen3.5-0.8B latency from 2.4627 to 1.0309 s/sample and output length from 182 to 86 tokens. Across all five backbones, the reduction is 25.4–58.1%, and generated programs execute in only .0410–.0482 s. Because the executable programs invoke lightweight tools and require no further MLLM call, this decoupled design supports resource-constrained edge deployment. Appendix I gives the complete timing protocol and results. In addition, cross-dataset program-reuse results are reported in Appendix H.3.

4.4 Model Analysis and Ablations

Data-scale effects on diagnosis and restoration. We increase the number of distinct degradation instances and program labels per background while retaining the same clean backgrounds. The 2×, 20×, and 100× settings use EABO with $K_{\text{init}} = 7$ and $\lambda = .7$; their epoch counts keep the total number of presented training examples approximately fixed. Table 3 therefore primarily tests supervision diversity rather than longer training. Diagnosis benefits throughout the scale range: ID full accuracy rises from .4611 to .5907 and .6193, while OOD rises from .1435 to .2472 and .2811. The persistent ID–OOD gap shows that more diverse supervision alleviates but does not remove composition failure. Restoration improves in parallel: ID/OOD PSNR increases from 37.53/28.11 dB at 2× to 37.89/28.54 at 20× and 37.96/28.61 at 100×, while SSIM increases from .8459/.7787 to .8555/.7911 and .8575/.7951. Most restoration gains occur between 2× and 20×. From 20× to 100×, the increments are .07/.07 dB in PSNR and .0020/.0040 in ID/OOD SSIM. Program generation therefore saturates earlier than exact diagnosis under this scaling protocol.

Visual and language LoRA adaptation. Table 4 shows that LoRA branches play different and complementary roles. For diagnosis, visual-side adaptation is more important: removing it (language only) lowers full accuracy from .6193 to .4312 on ID and from .2811 to .1761 on OOD, whereas removing language-side adaptation (vision only) gives the milder drops to .5488 and .2312. Restoration follows a different pattern. Using both LoRA branches yields higher ID SSIM than either single-branch variant (.8575 versus .8461/.8497), while language-side adaptation is especially important for OOD restoration: removing it lowers SSIM to .7645, compared with .7845 when visual-side adaptation is removed. LoRA branches therefore provide complementary capacity for the two outputs, and the residual ID–OOD gap shows that LoRA adaptation alone does not solve compositional generalization.

Table 6: Quality–efficiency comparison of supervision-construction methods on the 2× training set. All construction averages include no-op samples.

Method	Tool calls	MLLM calls	s / sample	Best SSIM	ID Full	OOD Full	ID SSIM	OOD SSIM
LLM iterative search	2.9487	4.3760	43.3289 s	.8516	.4744	.1415	.8353	.7617
MCTS	1.7158	0	2.6058 s	.8384	.4618	.1382	.8261	.7484
EABO	1.7690	0	1.5863 s	.8505	.4551	.1342	.8425	.7775

Table 5: Diagnosis–restoration interaction in IRest-VL.

Setting	ID/OOD Full	ID/OOD SSIM
restoration only	–/–	.8575/.7954
diagnosis+restoration	.6193/.2811	.8575/.7951

Task interaction after decoupling. Table 5 shows that adding the structured diagnosis head to the code-only model leaves ID/OOD SSIM nearly unchanged (.8575/.7954 versus .8575/.7951). Thus, the decoupled architecture introduces no measurable restoration penalty, indicating little interference between diagnosis and restoration. In addition, valid-path decoding sensitivity analysis is reported in Appendix H.4.3.

4.5 Data and Search Ablations

Effectiveness of experience accumulation. We study experience accumulation using the 2× training set (i.e., sampling 2 samples per image). For each trial budget, we construct a supervision set with ordinary BO and another with EABO, then train the same decoupled IRest-VL configuration on each set and evaluate ID and OOD diagnosis and restoration. Increasing the BO trial budget increases the time required to construct the dataset; Fig. 5 therefore plots best-program SSIM and downstream metrics against construction time, allowing comparisons at matched construction times rather than only in trial count. At comparable construction times, EABO gives higher best-program and ID restoration SSIM at all budgets and higher OOD restoration SSIM at most budgets. In contrast, the mean ID/OOD diagnosis accuracies of EABO and vanilla BO are similar. Thus, EABO provides better restoration targets without materially changing diagnosis quality. Detailed sensitivity settings for historical initialization, ranking weight, and parameterization are provided in the appendix.

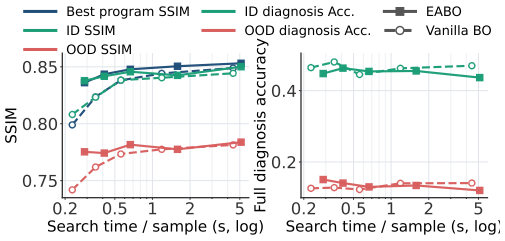

Figure 5: Search-quality versus per-sample construction time. (a) best-program, ID, and OOD SSIM; (b) ID and OOD full diagnosis accuracy.

Search strategies. Table 6 compares EABO, iterative LLM search, and Monte Carlo tree search (MCTS) on the same 2× training set and downstream model. EABO is substantially more efficient than iterative LLM search, achieving a 27.3× speedup without extra MLLM calls. EABO also attains nearly the same best-program SSIM and higher downstream ID/OOD SSIM. Compared with MCTS, EABO is faster and produces higher best-program SSIM. Their ID/OOD diagnosis accuracies are comparable, while EABO gives clearly higher ID/OOD restoration SSIM. Thus, EABO achieves comparable or better supervision quality at substantially lower construction cost and without MLLM token expenditure.

5 Conclusion

IRest-VL unifies composite infrared degradation diagnosis and executable lightweight restoration in one MLLM. EABO efficiently builds parameter-rich supervision, whereas the decoupled architecture improves OOD diagnosis while maintaining comparable restoration quality. The current evaluation is limited to controlled composite degradations within a fixed lightweight tool registry; future work will extend the executable space and test the method on broader real-sensor conditions and target edge hardware.

AI Use Statement

AI assistants were used for language editing, LaTeX formatting, and software-development support. They did not replace the authors' decisions about the research question, experimental design, implementation, analysis, or interpretation. The authors checked the manuscript, code changes, figures, tables, and reported results before submission.

Ethics Statement

This work uses benchmark infrared images and controlled synthetic degradations; it does not collect human-subject data, conduct user studies, or infer sensitive personal attributes. We use the source datasets and pretrained models in accordance with their stated licenses and cite the corresponding sources. We evaluate IRest-VL as an image-restoration and diagnosis system rather than in a clinical or safety-critical setting.

Reproducibility Statement

We specify the data partitions, degradation primitives, tool registry, search objective, baseline protocols, prompts, evaluation metrics, and principal optimization settings in the paper and appendix. The appendix also gives the complete parameterized tool space, valid-path decoding algorithm, type-sequence inventory, and efficiency conventions. Implementation and configuration files will be released upon publication, subject to the licenses of the underlying models and datasets.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. volume 24, 2011.

Haoyu Chen, Wenbo Li, Jinjin Gu, Jingjing Ren, Sixiang Chen, Tian Ye, Renjing Pei, Kaiwen Zhou, Fenglong Song, and Lei Zhu. Restoreagent: Autonomous image restoration agent via multimodal large language models. *NeurIPS*, 37:110643–110666, 2024.

Siyi Chen, Mikaela Angelina Uy, Chan Hee Song, Faisal Ladhak, Adithyavairavan Murali, Qing Qu, Stan Birchfield, Valts Blukis, and Jonathan Tremblay. Spacetools: Tool-augmented spatial reasoning via double interactive rl. In *CVPR*, pp. 37109–37120, 2026.

Luo Geng, Zhao Zicheng, Long Qian, Lv Chun, and Bao Jie. Image noise level classification technique based on image quality assessment. In *IEEE ICPICS*, pp. 651–656, 2020.

Shi Guo, Zifei Yan, Kai Zhang, Wangmeng Zuo, and Lei Zhang. Toward convolutional blind denoising of real photographs. In *CVPR*, pp. 1712–1722, 2019.

Zewei He, Yanpeng Cao, Yafei Dong, Jiangxin Yang, Yanlong Cao, and Christel-Löic Tisse. Single-image-based nonuniformity correction of uncooled long-wave infrared detectors: A deep-learning approach. *Applied optics*, 57(18):D155–D164, 2018.

Guoli Jia, Yisheng Zhang, Haote Hu, Shanxu Zhao, Kaikai Zhao, Long Sun, Xinwei Long, Kai Tian, Che Jiang, Zhaoxiang Liu, et al. Tir-agent: Training an explorative and efficient agent for image restoration. In *ECCV*, pp. 523–540, 2026.

Xinyu Jia, Chuang Zhu, Minzhen Li, Wenqi Tang, and Wenli Zhou. Llvip: A visible-infrared paired dataset for low-light vision. In *ICCV Workshops*, pp. 3489–3497, 2021.

Nanhe Jiang, Yucun Zhang, Qun Li, and Fang Yan. An infrared image denoising method based on wavelet dual-branch diffusion model. *IEEE Transactions on Geoscience and Remote Sensing*, 63:1–13, 2025.

Xiaoyu Jin, Yuan Shi, Bin Xia, and Wenming Yang. Llmra: Multi-modal large language model based restoration assistant. *arXiv preprint arXiv:2401.11401*, 2024.

G Kavitha, Chetana Prakash, Majid Alhomrani, N Pradeep, Abdulhakeem S Alamri, Piyush Kumar Pareek, and Musah Alhassan. Noise estimation and type identification in natural scene and medical images using deep learning approaches. *Contrast Media & Molecular Imaging*, 2023(1):3923667, 2023.

Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *ICCV Workshops*, pp. 1833–1844, 2021.

Jingren Liu, Shuning Xu, Qirui Yang, Yun Wang, Xiangyu Chen, and Zhong Ji. Fape-ir: frequency-aware planning and execution framework for all-in-one image restoration. In *CVPR*, pp. 15561–15573, 2026a.

Jinyuan Liu, Zihang Chen, Zhu Liu, Zhiying Jiang, Long Ma, Xin Fan, and Risheng Liu. Enhancing infrared vision: progressive prompt fusion network and benchmark. volume 38, pp. 96850–96875, 2026b.

Shilong Liu, Hao Cheng, Haotian Liu, Hao Zhang, Feng Li, Tianhe Ren, Xueyan Zou, Jianwei Yang, Hang Su, Jun Zhu, et al. Llava-plus: Learning to use tools for creating multimodal agents. In *ECCV*, pp. 126–142, 2024.

Zhuoqun Liu, Meiguang Jin, Ying Chen, Huaida Liu, Canqian Yang, and Hongkai Xiong. Lightweight network towards real-time image denoising on mobile devices. In *ICIP*, pp. 2270–2274, 2023.

Jianglin Lu, Yuanwei Wu, Ziyi Zhao, Hongcheng Wang, Felix Jimenez, Abrar Majeedi, and Yun Fu. Restore-r1: Efficient image restoration agents via reinforcement learning with multimodal llm perceptual feedback. In *CVPR*, pp. 8629–8639, 2026.

Ziwei Luo, Fredrik K Gustafsson, Zheng Zhao, Jens Sjölund, and Thomas Schön. Controlling vision-language models for multi-task image restoration. In *ICLR*, volume 2024, pp. 16226–16246, 2024.

Delin Mao, Chenghao Sun, Jingwei Song, Chishui Chen, and Linfeng Zhang. Toolvision: Learning when and how to use visual tools with capability-aligned supervision. *arXiv preprint arXiv:2608.08907*, 2026.

Xuanyu Qian, Xuquan Wang, Yujie Xing, Guishuo Yang, Xiong Dun, Zhanshan Wang, and Xinbin Cheng. Mwr-net: an edge-oriented lightweight framework for image restoration in single-lens infrared computational imaging. *Remote Sensing*, 17(17):3005, 2025.

Md Fazle Hasan Shiblee, Md Faiyaj Ahmed Limon, and Md Shahid Iqbal. Deep learning-based novel image noise classification model 'MobileNoiseNet'. In *ICCIT*, pp. 1–6, 2023.

Chaoyang Wang, Kaituo Feng, Dongyang Chen, Zhongyu Wang, Zhixun Li, Sicheng Gao, Meng Meng, Xu Zhou, Manyuan Zhang, Yuzhang Shang, et al. Adatooler-v: Adaptive tool-use for images and videos. In *Findings of ACL*, pp. 18058–18072, 2026.

Pengfei Yang, Heng Wu, Lianglun Cheng, and Shaojuan Luo. Infrared image denoising via adversarial learning with multi-level feature attention network. *Infrared Physics & Technology*, 128:104527, 2023a.

Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. Gpt4tools: Teaching large language model to use tools via self-instruction. *NeurIPS*, 36:71995–72007, 2023b.

Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *CVPR*, pp. 5728–5739, 2022.

Chenxiao Zhang, Xin Deng, Hongpeng Sun, Jingyi Xu, and Mai Xu. Sn-net: Semismooth newton driven lightweight network for real-world image denoising. In *ICIP*, pp. 1424–1430, 2024.

Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

Yingjie Zhou, Jiezhang Cao, Farong Wen, Zicheng Zhang, Yu Zhou, Yue Shi, Xiaohong Liu, Radu Timofte, Luc Van Gool, and Guangtao Zhai. Q-agent: Quality-driven chain-of-thought image restoration agent through robust multimodal large language model. *arXiv preprint arXiv:2504.07148*, 2025.

Yan Zou, Linfei Zhang, Chengqian Liu, Bowen Wang, Yan Hu, and Qian Chen. Super-resolution reconstruction of infrared images based on a convolutional neural network with skip connections. *Optics and Lasers in Engineering*, 146:106717, 2021.

Appendix

Appendix map. The appendix is organized as follows:

- Data Description: degradation primitives, ID/OOD pipeline construction, and data partitions.
- Baseline Protocols: model-comparison and data-construction baselines.
- Tool Space and Search Algorithms: callable tools and the EABO procedure.
- Prompt and Output Format: prompts and tool-message construction.
- Training Configuration: optimization, training, and decoding settings.
- Decoding Algorithm: valid-path structured diagnosis decoding.
- Evaluation Methods: metric definitions and evaluation conventions.
- Additional Results and Ablations: length breakdowns, tool reuse, and sensitivity analyses.
- Efficiency Analysis: inference and restoration-program execution time.
- Qualitative Results: representative diagnosis and restoration examples.

A Data Description

In this section, we describe the experimental data, degradation primitives, ID/OOD pipeline construction, and sample allocation.

A.1 Degradation Primitives and Severity Levels

We use five primitives, i.e., Gaussian noise, speckle noise, Gaussian blur, motion blur, and stripe noise, and allow each pipeline to contain zero, one, two, or three distinct primitives. For each included primitive in a non-empty pipeline, we first assign a low, medium, or severe intensity level, then sample its physical parameters from Table 7, and finally apply the primitives in the prescribed order. We record primitive type, severity, application order, and clean-reference restoration quality as supervision. Importantly, these parameters define the data-generation process and are unrelated to the restoration-tool search space.

Table 7: Synthetic degradation primitives and severity-conditioned parameter ranges. Ranges follow the stated sampling protocol; parameters not tied to severity are shown in all three columns.

Primitive	Parameters	Low	Medium	Severe
Gaussian noise	σ	4.0–7.0	8.0–14.0	15.0–24.0
Speckle noise	σ	0.035–0.065	0.075–0.12	0.13–0.20
Gaussian blur	kernel size; σ	$\{5, 7\}$; 0.8–1.3	$\{9, 11\}$; 1.5–2.4	$\{13, 15\}$; 2.8–4.0
Motion blur	kernel size; angle	$\{7, 9\}$; 0–180°	$\{11, 13\}$; 0–180°	$\{15, 17\}$; 0–180°
Stripe noise	intensity min/max; direction	0.006–0.014 / 0.025–0.045; vertical	0.018–0.035 / 0.050–0.080; vertical	0.040–0.065 / 0.085–0.125; vertical

A.2 ID and OOD Type-Sequence Construction

We create compositional generalization tests while keeping the primitive vocabulary fixed. We enumerate ordered sequences of distinct primitives with lengths zero through three. We retain 78 templates for training, validation, and Test-ID, and hold out eight templates for Test-OOD. The OOD templates therefore contain no new primitive or severity level. They differ only in composition or degradation order. The no-op pipeline is retained in ID but excluded from OOD.

Using the five primitives defined in Appendix A.1, the resulting inventory is summarized in Table 8, which counts all ordered permutations and the ID/OOD allocation at each length. Table 9 lists the eight held-out templates explicitly. Thus, OOD keeps the primitive vocabulary and severity levels fixed while withholding eight type-sequence templates. By construction, it contains only length-two and length-three pipelines.

Table 8: Type-sequence inventory.

Length	All ordered permutations	OOD templates	ID templates
0 (no-op)	1	0	1
1	5	0	5
2	20	4	16
3	60	4	56
Total	86	8	78

Table 9: OOD type-sequence templates.

Length	OOD type sequence
2	`gaussian_blur` → `stripe_noise`
2	`motion_blur` → `stripe_noise`
2	`stripe_noise` → `gaussian_blur`
2	`stripe_noise` → `motion_blur`
3	`stripe_noise` → `gaussian_blur` → `gaussian_noise`
3	`stripe_noise` → `gaussian_blur` → `speckle_noise`
3	`stripe_noise` → `motion_blur` → `gaussian_noise`
3	`stripe_noise` → `motion_blur` → `speckle_noise`

A.3 Data Partitions and Sampling

We sample clean images and degradation pipelines across the training, validation, Test-ID, and Test-OOD splits as follows. For ID, we sample pipeline lengths zero, one, two, and three with probabilities $(0.1, 0.3, 0.3, 0.3)$; for OOD, lengths two and three are sampled equally. We split clean backgrounds into train, validation, and test pools, then sample a template and severity sequence within the assigned split and generate the degraded image with its labels. BO supervision is constructed independently for each split. For each 1,505-sample test split, Test-ID contains 151/451/452/451 samples of lengths 0/1/2/3, whereas Test-OOD contains 0/0/753/752 samples. The pre-degradation image is the clean reference for all evaluations, and history banks never cross split boundaries. The resulting partition sizes and sampling counts are summarized in Table 10.

Table 10: Data partitions and controlled degradation samples.

Split	Clean images	Samples/image	Samples	Templates / role
Train	1,052	100	105,200	78 ID templates / training
Validation	150	5	750	78 ID templates / checkpoint selection
Test-ID	301	5	1,505	78 ID templates / in-distribution
Test-OOD	301	5	1,505	8 held-out templates / composition generalization

B Baseline Protocols

B.1 Model-Comparison Baselines

We now describe the model baselines reported alongside IRest-VL in the main text. All deployable methods are evaluated on the same Test-ID and Test-OOD images. Only the oracle references access the clean target at test time.

No processing. The degraded-input reference applies no restoration and measures the starting image quality on each split.

Single-tool reference. The fixed-tool comparison tests whether composing multiple operations improves over one operator. Each tool is tuned on validation data and then reused for every test image without per-image tuning; guided filtering gives the highest fixed-tool SSIM and is therefore reported in Table 2.

Oracle references. The per-image best-single-tool oracle restricts restoration to one operation. For each test image, it uses the clean target to select the tool and its parameters that maximize SSIM. The 400-trial EABO oracle applies EABO to each test image with

the number of Optuna trials set to 400 and uses clean-target SSIM as the objective. The first oracle isolates the benefit of program composition over a single optimized operation, while the second estimates the quality attainable with a larger search budget. Both require the clean test target and are therefore non-deployable empirical references rather than theoretical upper bounds.

Lightweight restorers. MFDNet and SN-Net are lightweight end-to-end image-restoration networks and receive only the degraded image at test time. MFDNet is trained from scratch on the same 100× training and validation data as the primary model for 50 epochs with mini-batch size 40. Its optimizer and architectural settings follow the published configuration. Because SN-Net is substantially slower to train, we use 9 epochs with mini-batch size 4 and 256×256 patches on the same composite-degradation splits, following its patch-based training protocol.

Zero-shot MLLMs. GLM-5.2, Qwen3.8-Max, GPT-5.6-sol, and Claude Sonnet 5 receive the same degraded image, tool specification, and diagnosis–code output contract without task-specific fine-tuning. This comparison evaluates whether general-purpose MLLMs can directly diagnose composite degradations and generate executable restoration programs.

Direct AR Diagnosis. The Direct AR Diagnosis Qwen3.5-0.8B SFT baseline autoregressively predicts the degradation pipeline without generating a restoration program. It isolates diagnosis behavior without code supervision.

Direct AR Diagnosis and Restoration. The Direct AR Diagnosis and Restoration models serialize diagnosis followed by restoration-code generation. Each model and its IRest-VL counterpart use the same pretrained MLLM initialization, training samples, BO-generated program supervision, epoch schedule, batch size, optimizer, learning rate, weight decay, warm-up ratio, numerical precision, sequence-length limits, LoRA configuration, and checkpoint-evaluation frequency. Both select checkpoints using the same mean of validation full-diagnosis accuracy and restoration SSIM and use the same image and tool specification at inference. Thus, the comparison changes only the output parameterization and associated supervision. Prompt formats are provided in Appendix D.

B.2 Data-Construction Baselines

We compare two search methods for constructing restoration-program supervision: iterative MLLM search and discrete MCTS.

Iterative MLLM search. This baseline alternates program proposal with execution-based parameter optimization. Let N_{round} denote the maximum number of MLLM-guided rounds and B_{Opt} the Optuna budget per round. In each round, an MLLM proposes or revises the restoration tools and their execution order for the current sample using the available-tool specification and prior search feedback. Optuna then uses B_{Opt} trials to optimize the active numerical arguments of that proposal against reference SSIM. The procedure retains the highest-SSIM program observed over at most N_{round} rounds. Thus, the MLLM proposes the discrete program structure, while black-box optimization selects the tool parameters. We set $N_{\text{round}} = 5$ and $B_{\text{Opt}} = 40$.

Discrete MCTS. This baseline represents a restoration program as a path in a Monte Carlo search tree. Let G_{param} denote the number of discrete candidates for each tunable argument and N_{sim} the simulation budget per sample. A node encodes a partial program, and an action extends it with a tool choice and arguments drawn from the G_{param}-value grids. Tree selection balances previously observed rewards with unexplored branches. A completed candidate is executed on the degraded image, scored by reference SSIM, and its reward is propagated back through the visited path. After N_{sim} simulations, the highest-scoring completed program is retained. Unlike iterative MLLM search, this baseline uses no language-model calls, but its parameter resolution is limited by the discrete grid. We set $G_{\text{param}} = 5$ and $N_{\text{sim}} = 50$.

Table 11: Parameterized restoration-tool space and its search variables. Defaults are callable defaults rather than restrictions on BO search.

Tool	Type and mechanism	Associated degradations	Parameter type, range, and default	Parameter semantics
`odgf_stripe_removal`	classical, 1-D guided stripe removal	stripe noise; motion blur	`odgf_half_ws`: int 2–8, step 1, default 4	horizontal half-window; larger values strengthen stripe smoothing but may remove weak texture
`guided_filter_denoise`	classical, edge-preserving guided filtering	all five primitives	`guided_radius`: int 2–9, default 5; `guided_eps`: float .005–.08, default .02	neighborhood radius and edge-preserving regularization; larger `eps` usually smooths more strongly
`lee_speckle_filter`	classical, local-statistics Lee MMSE	speckle noise; Gaussian noise	`lee_filter_size`: odd int 3–9, step 2, default 5; `lee_noise_variance`: float .005–.08, default .03	local window and multiplicative-noise variance estimate; larger variance strengthens smoothing
`kuan_speckle_filter`	classical, local-variation Kuan filter	speckle noise; Gaussian noise	`kuan_filter_size`: odd int 3–9, step 2, default 5; `kuan_nlooks`: float 1–6, default 2	local window and equivalent number of looks; larger `nlooks` assumes weaker speckle
`motion_deblur_wiener`	classical, frequency-domain Wiener deconvolution	motion blur	`motion_length`: odd int 3–17, step 2, default 9; `motion_angle`: float 0–180, default 0; `motion_snr`: float 10–80, default 30	motion-kernel length and angle, and Wiener SNR; a larger SNR can amplify noise and ringing
`mfdnet_gaussian_blur_removal`	fixed-weight lightweight MFDNet	Gaussian blur	no explicit parameter	No parameter; invocation and position are searched.
`mfdnet_stripe_noise_removal`	fixed-weight lightweight MFDNet	stripe noise	no explicit parameter	No parameter; invocation and position are searched.
`guided_layer_detail_enhance`	classical, guided base–detail decomposition	all five primitives	`detail_radius`: int 2–9, default 5; `detail_eps`: float .005–.08, default .02; `detail_gain`: float 1–2.2, default 1.5	base-estimation scale, smoothing, and high-frequency gain; excessive gain may amplify residual noise

C Tool Space and Search Algorithms

We describe the callable restoration-tool space and the EABO algorithm used to construct program supervision.

C.1 Tool Space

The callable restoration space used by offline search and all code-generation baselines contains eight tools: one-dimensional guided stripe removal, guided filtering, Lee and Kuan speckle filters, Wiener motion deblurring, two fixed-weight MFDNet tools, and guided detail enhancement. The tool registry lists each callable tool, its compatible degradation types, its parameter type, range, and default, and the role of each parameter in Table 11.

C.2 EABO Algorithm

Algorithm 1 presents the complete procedure for constructing one executable restoration-program target. Its inputs are the current degraded–clean image pair $(\mathbf{x}_i, \mathbf{y}_i)$, the type and severity sequences $(\mathbf{g}_i, \mathbf{s}_i)$, the matching history $\mathcal{H}_i(\mathbf{g}_i)$, the total trial budget B, the maximum number of historical initializations K_{init}, and the ranking weight λ. The admissible program space is determined by $\mathbf{g}_i$ and the tool registry described in Appendix C.1. We first compute the pipeline length $K_i = |\mathbf{g}_i|$. An empty pipeline directly uses the identity program P_{id}, records its quality, and returns its serialized code.

For a non-empty pipeline, the algorithm ranks records in $\mathcal{H}_i(\mathbf{g}_i)$ by the retrieval score in Eq. 1, which combines severity-sequence similarity S_h and historical restoration quality Q_h. After duplicate complete programs are removed, at most K_{init} programs form the warm-start list $\mathcal{I}_i$. These warm starts are evaluated first and count toward the same total

budget B; TPE uses the remaining trials to propose new programs, together forming the evaluated set $\mathcal{E}_i^{(B)}$. The algorithm then compares all evaluated candidates with P_{id}, retains the highest-SSIM program P_i and its quality Q_i, updates $\mathcal{H}_{i+1}(\mathbf{g}_i)$, and serializes P_i as the code supervision target.

Algorithm 1 EABO for constructing restoration-program supervision.

Require: $(\mathbf{x}_i, \mathbf{y}_i, \mathbf{g}_i, \mathbf{s}_i)$, matching history $\mathcal{H}_i(\mathbf{g}_i)$, budget B, K_{init}, λ

$K_i \leftarrow |\mathbf{g}_i|$
if $K_i = 0$ **then**
  $P_i \leftarrow P_{\text{id}}$; $Q_i \leftarrow J(P_i; \mathbf{x}_i, \mathbf{y}_i)$
  $\mathcal{H}_{i+1}(\mathbf{g}_i) \leftarrow \mathcal{H}_i(\mathbf{g}_i) \parallel \langle(\mathbf{s}_i, P_i, Q_i)\rangle$; **return** $C_i \leftarrow \text{Serialize}(P_i)$
end if
For each $(\mathbf{s}_h, P_h, Q_h) \in \mathcal{H}_i(\mathbf{g}_i)$, compute S_h and $R_h(\lambda)$ from the stored Q_h using Eq. 1.
$\mathcal{I}_i \leftarrow \text{UniqueTopK}_{K_{\text{init}}}^{\downarrow R_h(\lambda)}(\mathcal{H}_i(\mathbf{g}_i))$.
Evaluate $\mathcal{I}_i$ first, then new TPE proposals from the admissible program space, under total budget B, to form $\mathcal{E}_i^{(B)}$.
$P_i \leftarrow \arg\max_{P \in \mathcal{E}_i^{(B)} \cup \{P_{\text{id}}\}} J(P; \mathbf{x}_i, \mathbf{y}_i)$; $Q_i \leftarrow J(P_i; \mathbf{x}_i, \mathbf{y}_i)$.
$\mathcal{H}_{i+1}(\mathbf{g}_i) \leftarrow \mathcal{H}_i(\mathbf{g}_i) \parallel \langle(\mathbf{s}_i, P_i, Q_i)\rangle$.
return $C_i \leftarrow \text{Serialize}(P_i)$.

D Prompt and Output Format

We specify the input/output contract shared by the code-generation experiments and the differences between model prompts. We first specify the common image–tool prompt and required function contract, then give the Direct AR Diagnosis and Restoration and decoupled templates, and finally provide the shared tool block.

D.1 Direct AR Diagnosis and Restoration Prompt

The Direct AR Diagnosis and Restoration baseline is included to test serialized diagnosis–code generation. Its user message contains the degraded image, the diagnosis-and-code instructions, and the tool registry; its assistant target contains a diagnosis block followed by a code block. The standard user template is shown below, where `<TOOL_BLOCK>` denotes the tool specification listed in Table 11.

```
The input image is the degraded infrared image. First diagnose the
degradation pipeline in this infrared image, then generate Python degradation-
removal and enhancement code. Return exactly two sections:
<diagnosis>...</diagnosis> and <code>...</code>, with no markdown,
explanation, or natural-language preface/suffix. The diagnosis section must
contain JSON with degradation_process and degradation_strength arrays containing
the aligned type and severity sequences. If there is no degradation, both
arrays must be empty arrays. Possible degradation types are: gaussian noise,
speckle noise, gaussian blur, motion blur, stripe noise. Possible degradation
strengths are: low, medium, severe. The code must define enhance_image with
image plus every used tool parameter in the function signature. Each tool call
must use explicit keyword arguments that reference the same-named signature
parameters.

<TOOL_BLOCK>
```

D.2 Decoupled Structured-Diagnosis and Code Prompt

The decoupled model removes diagnosis text from the language target so that the language branch learns only executable restoration code. Its user message still contains the same degraded image and tool registry, but its assistant target contains only the code block; structured attributes and transitions are supervised by the separate diagnosis head. The language-side template is:

```
The input image is the degraded infrared image. Generate Python degradation-
removal and enhancement code. Return exactly one section, with no markdown,
diagnosis, reasoning, explanation, or natural-language preface/suffix:
<code>...</code>

The <code> section must define enhance_image with image plus every used tool
parameter in the function signature. Each tool call must use explicit keyword
arguments that reference the same-named signature parameters.

<TOOL_BLOCK>
```

D.3 Shared Tool Block and Message Construction

For reproducibility, we list the exact callable names, signatures, defaults, and admissible parameter ranges. Parameter names in the signatures must be copied as keyword arguments in generated code. The complete registry is shown below:

```
Available infrared image enhancement tools:
- odgf_stripe_removal(image: 'np.ndarray', odgf_half_ws: 'int' = 4) ->
  'np.ndarray': Fast vertical stripe removal using one-dimensional guided
  filtering. Parameter space: odgf_half_ws: integer range [2, 8] step 1.
- guided_filter_denoise(image: 'np.ndarray', guided_radius: 'int' = 5,
  guided_eps: 'float' = 0.02) -> 'np.ndarray': Suppress gaussian_noise while
  preserving major edges. Parameter space: guided_radius: integer range [2, 9]
  step 1; guided_eps: float range [0.005, 0.08].
- lee_speckle_filter(image: 'np.ndarray', lee_filter_size: 'int' = 5,
  lee_noise_variance: 'float' = 0.03) -> 'np.ndarray': Suppress multiplicative
  speckle_noise using a Lee MMSE filter. Parameter space: lee_filter_size:
  integer range [3, 9] step 2; lee_noise_variance: float range [0.005, 0.08].
- kuan_speckle_filter(image: 'np.ndarray', kuan_filter_size: 'int' = 5,
  kuan_nlooks: 'float' = 2.0) -> 'np.ndarray': Suppress speckle_noise using a
  fast Kuan-style adaptive filter. Parameter space: kuan_filter_size: integer
  range [3, 9] step 2; kuan_nlooks: float range [1.0, 6.0].
- motion_deblur_wiener(image: 'np.ndarray', motion_length: 'int' = 9,
  motion_angle: 'float' = 0.0, motion_snr: 'float' = 30.0) -> 'np.ndarray':
  Restore directional motion_blur using Wiener deconvolution. Parameter space:
  motion_length: integer range [3, 17] step 2; motion_angle: float range
  [0.0, 180.0]; motion_snr: float range [10.0, 80.0].
- mfdnet_gaussian_blur_removal(image: 'np.ndarray') -> 'np.ndarray': Remove
  gaussian_blur using a Gaussian-blur-specific MFDNet model. Parameter space:
  no tunable parameters.
- mfdnet_stripe_noise_removal(image: 'np.ndarray') -> 'np.ndarray': Remove
  stripe_noise using a stripe-noise-specific MFDNet model. Parameter space:
  no tunable parameters.
- guided_layer_detail_enhance(image: 'np.ndarray', detail_radius: 'int' = 5,
  detail_eps: 'float' = 0.02, detail_gain: 'float' = 1.5) -> 'np.ndarray':
  Enhance high-frequency detail using a guided base-detail decomposition.
  Parameter space: detail_radius: integer range [2, 9] step 1; detail_eps:
  float range [0.005, 0.08]; detail_gain: float range [1.0, 2.2].
```

E Training Configuration

Table 12 summarizes the offline search, optimization, diagnosis-head, and decoding settings used for reproducibility. Unless noted otherwise, the coupled and decoupled models use the same MLLM backbone, data-construction budget, optimization schedule, and trainable-module configuration; only their output parameterization differs. The default $2\times$ and $100\times$ schedules are summarized in this table, while the complete scaling results and the $20\times$ setting are reported in Table 3.

Checkpoints are selected on the validation split using the task-appropriate criterion: the mean of diagnosis accuracy and restoration SSIM for the full model, and the corresponding

Table 12: Default data-construction, training, and decoding configuration.

Module	Parameter	Default setting
Data construction	BO trials per sample B	40
Data construction	Historical initializations K_{init}	7
Data construction	Experience quality weight λ	.7
Data construction	Severity-sequence-similarity weight $1-\lambda$	.3
Data construction	Historical quality	best SSIM
Training	MLLM backbone	Qwen3.5-0.8B
Training	Epochs ($2\times$ / $100\times$)	40 / 2
Training	Global batch size	32
Training	Learning rate / weight decay	2×10^{-5} / .01
Training	Warm-up ratio / precision	.03 / BF16
Training	Maximum input / new tokens	4096 / 2048
Training	Image tokenized pixels	100,352–401,408
Training	LoRA rank / scale / dropout (vision and language)	16 / 32 / .05
Training	Validation frequency	every 500 steps
Training	Gradient accumulation	2 steps
Diagnosis head	token dimension / attention heads / FFN expansion / dropout	1024 / 8 / 2 / 0
Diagnosis loss	w_+ / λ_a / λ_t	5 / .1 / .03
Decoding	β / $L_{\max}$	1.0 / 5

Algorithm 2 Valid-path decoding for structured degradation diagnosis.

Require: Attribute logits $\mathbf{Z}^{\text{attr}} \in \mathbb{R}^{M\times 4}$, transition logits $\mathbf{z}^{\text{trans}} \in \mathbb{R}^{M(M-1)}$, maximum length $L_{\max}$, transition weight β
Ensure: Type sequence $\hat{\pi}$ and severity sequence $\hat{\mathbf{s}}$
$p_j \leftarrow \text{softmax}(\mathbf{Z}^{\text{attr}}_{j,:})$; $q_{uv} \leftarrow \sigma(z^{\text{trans}}_{uv})$
$\mathcal{C} \leftarrow \left\{\left(\emptyset, \sum_{j=1}^{M} \log p_j[0]\right)\right\}$
for $\ell = 1$ to $L_{\max}$ **do**
  for all non-repeating type sequences $\pi = (\pi_1, \ldots, \pi_\ell)$ **do**
    Assign each $j \notin \pi$ the absent class; for $j \in \pi$, set $c_j(\pi)$ to the most probable non-absent severity
    $S(\pi) \leftarrow \sum_{j=1}^{M} \log p_j[c_j(\pi)] + \beta \sum_{r=1}^{\ell-1} \log q_{\pi_r \pi_{r+1}}$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{(\pi, S(\pi))\}$
  end for
end for
$\hat{\pi} \leftarrow \arg\max_{(\pi,S)\in\mathcal{C}} S$
return $\hat{\pi}$ and the corresponding non-absent severities $\hat{\mathbf{s}}$

task metric for diagnosis-only or code-only ablations. The test split is used only for final evaluation.

F Decoding Algorithm

We convert the independent attribute and transition predictions into one valid degradation pipeline using Algorithm 2, which first converts the logits to probabilities, enumerates structurally valid candidates, scores them using attribute and transition evidence, and returns the highest-scoring path. Let $\mathcal{G}$ denote the five degradation primitives and let $M = |\mathcal{G}| = 5$.

Concretely, the candidate set includes the no-degradation path and all non-repeating type sequences up to the allowed length. For a candidate path, primitives that do not appear in the path are assigned the absent class, while each selected primitive receives its most probable non-absent severity. The path score sums the corresponding attribute log-probabilities and adds the weighted log-probabilities of its adjacent directed transitions. Restricting candidates to non-repeating paths guarantees a valid sequence. The decoder therefore does not combine independently selected attributes and transitions. Algorithm 2 enumerates non-repeating paths up to $L_{\max} = 5$ and selects the path with the highest score defined in Eq. equation 6. The main experiments use $\beta = 1$. The decoding settings, including the transition weight β and maximum path length $L_{\max}$, are listed in Table 12.

G Evaluation Methods

G.1 Evaluation Metrics

All metrics are computed over the complete test split. Failed parses and executions remain in the denominator, and failed restorations use the degraded image.

Full diagnosis accuracy. For N test instances, full diagnosis is correct only when both the type sequence and the aligned severity sequence exactly match the reference:

$$\mathrm{Acc}_{\mathrm{full}} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}[\hat{\mathbf{g}}_i = \mathbf{g}_i \ \wedge \ \hat{\mathbf{s}}_i = \mathbf{s}_i]. \tag{7}$$

Attribute F1. The 15 non-absent primitive–severity labels are evaluated as one-vs-rest binary labels. With ground-truth and predicted labels $b^{\mathrm{attr}}_{i,k}$ and $\hat{b}^{\mathrm{attr}}_{i,k}$, we report global micro-F1, $\mathrm{F1}_a = 2\mathrm{TP}_a/(2\mathrm{TP}_a + \mathrm{FP}_a + \mathrm{FN}_a)$.

Transition F1. The 20 directed transitions between distinct primitives are evaluated as binary labels. Using the same micro-F1 formula with $b^{\mathrm{trans}}_{i,k}$ and $\hat{b}^{\mathrm{trans}}_{i,k}$ gives $\mathrm{F1}_t$.

Code parse rate. A prediction is parseable if the response contains an extractable function named `enhance_image`; the parse rate is the fraction of test responses satisfying this condition.

Code pass rate. A parsed program passes if it executes successfully on the test image; the pass rate is the fraction of all test responses whose extracted code executes successfully.

PSNR. PSNR is computed between the restored and clean uint8 images with peak value 255; zero MSE is assigned 100 dB.

SSIM. SSIM is computed between the restored and clean uint8 images using `data_range=255`.

H Additional Results and Ablations

We further analyze pipeline length, fixed-tool references, cross-dataset reuse, and the sensitivity of the main design choices.

H.1 Breakdown by Degradation-Pipeline Length

We test whether aggregate results hide difficulty that grows with degradation-pipeline length. We group the matched Direct AR Diagnosis and Restoration and IRest-VL predictions by length, report full diagnosis accuracy and restored SSIM for each backbone, and compare the two output designs within each group. ID contains lengths zero, one, two, and three, whereas OOD contains only lengths two and three. Table 13 reports the complete length-specific results.

Diagnosis by length. On ID, the no-op cases at $L = 0$ are nearly perfect (full accuracy .9338–1.0000), while length-one accuracy remains high (.8958–.9446) and gains at lengths two and three are mixed. On OOD, IRest-VL improves every length-two result by .2617–.4196, but all length-three results are at or below .0426, leaving longer unseen compositions unresolved. Length-three diagnosis is difficult because exact matching requires correct predictions for three primitive attributes, two directed transitions, severities, and their order. On held-out OOD compositions, these errors compound across more components, so the valid-path constraint cannot compensate for unreliable local evidence.

Table 13: Full diagnosis accuracy (Full) and restored SSIM by degradation-pipeline length. Each Direct AR Diagnosis and Restoration row serializes diagnosis and code; each IRest-VL row uses structured diagnosis with a separate code-generation path.

MLLM backbone	Output	ID								OOD			
		$L=0$		$L=1$		$L=2$		$L=3$		$L=2$		$L=3$	
		Full	SSIM	Full	SSIM	Full	SSIM	Full	SSIM	Full	SSIM	Full	SSIM
Qwen3.5-0.8B	Direct AR Diagnosis and Restoration	1.0000	1.0000	.8958	.8975	.5619	.8308	.2616	.7936	.1408	.8263	.0013	.7528
	IRest-VL	1.0000	1.0000	.9268	.8971	.5973	.8319	.2062	.7957	.5604	.8287	.0013	.7615
Qwen3.5-9B	Direct AR Diagnosis and Restoration	1.0000	1.0000	.9157	.8980	.5774	.8314	.2594	.7951	.2961	.8304	.0027	.7563
	IRest-VL	1.0000	1.0000	.9379	.8979	.6593	.8318	.3126	.7943	.5578	.8309	.0000	.7568
InternVL3.5-8B	Direct AR Diagnosis and Restoration	.9669	.9994	.9290	.8979	.6128	.8308	.2772	.7928	.0996	.8279	.0000	.7550
	IRest-VL	1.0000	1.0000	.9224	.8981	.5686	.8316	.1441	.7946	.4608	.8288	.0000	.7610
MiMo-VL-7B	Direct AR Diagnosis and Restoration	.9338	1.0000	.9335	.8982	.5907	.8321	.2683	.7952	.1899	.8297	.0066	.7535
	IRest-VL	1.0000	1.0000	.9446	.8983	.6593	.8325	.2949	.7961	.4794	.8323	.0000	.7547
MiniCPM-V-4.6	Direct AR Diagnosis and Restoration	1.0000	1.0000	.9357	.8981	.6504	.8328	.3348	.7959	.3081	.8272	.0426	.7560
	IRest-VL	1.0000	1.0000	.9424	.8981	.6659	.8328	.2838	.7959	.6441	.8313	.0000	.7602

Restoration by length. Restored SSIM decreases consistently as the pipeline grows: the ID ranges are .9994–1.0000, .8971–.8983, .8308–.8328, and .7928–.7961 for lengths zero, one, two, and three, respectively, while OOD decreases from .8263–.8323 at length two to .7528–.7615 at length three. In contrast to diagnosis, Direct AR Diagnosis and Restoration and IRest-VL differ by at most .0087 SSIM in any nonzero-length group. Pipeline length thus determines restoration difficulty more strongly than the output formulation, supporting the conclusion that decoupling improves compositional diagnosis without materially degrading restoration.

H.2 Fixed Single-Tool Results

We use a single-tool reference to measure the benefit of composing multiple restoration operations. For each tool, we first select its arguments with ten Optuna trials on validation data, then freeze that configuration, apply the tool to every ID and OOD test image, and report PSNR and SSIM. The resulting values measure fixed-tool transfer to composite degradations rather than per-test-image tuning. The complete fixed-tool comparison is given in Table 14. The guided filter is the strongest single fixed tool (ID/OOD SSIM: .7870/.7264), likely because its edge-preserving smoothing provides useful responses across heterogeneous noise and blur types rather than being specialized to one degradation primitive.

Table 14: Fixed single-tool restoration results.

Tool	ID PSNR	ID SSIM	OOD PSNR	OOD SSIM
Guided filter	29.08	.7870	27.41	.7264
Kuan filter	28.98	.7792	27.30	.7137
Lee filter	28.89	.7522	26.97	.6796
MFDNet stripe removal	29.79	.7024	26.83	.6366
ODGF stripe removal	28.61	.6917	26.61	.6404
Detail enhancement	28.85	.6289	25.08	.5048
MFDNet deblurring	25.04	.5292	22.65	.3799
Wiener motion deblurring	21.60	.4308	21.96	.4184

H.3 Cross-Dataset Program Reuse

We test whether an executable program searched for one source image can be reused on other infrared images without another search or MLLM call. We first generate one program

for each Test-ID source image, then replay that unchanged program on eight images from each external dataset, and finally compute aggregate and pipeline-length-specific restoration metrics. Table 15 reports aggregate transfer, while Table 16 separates source pipelines by length.

Table 15: Cross-dataset reuse of programs generated once on source images. Each target set contains 12,040 source–target pairs.

Target set	Pairs	Degraded PSNR/SSIM	Restored PSNR/SSIM	Gain PSNR/SSIM
DLS-NUC-100	12,040	35.42/.6241	39.79/.8539	+4.37/+.2298
IR700	12,040	34.14/.6228	37.82/.8291	+3.68/+.2063
LLVIP	12,040	35.24/.6481	40.11/.9099	+4.87/+.2618

Table 16: Cross-dataset reuse by source degradation-pipeline length; each cell is restored SSIM (gain over degraded input).

Target set	Length 1	Length 2	Length 3
DLS-NUC-100	.8828 (+.2053)	.8258 (+.2669)	.8044 (+.2941)
IR700	.8766 (+.1893)	.7993 (+.2411)	.7541 (+.2575)
LLVIP	.9335 (+.2389)	.8961 (+.3076)	.8700 (+.3265)

Tables 15 and 16 show that a program searched for one source image can be replayed on different images under the same controlled degradation pipeline. Restoration improves on all three target datasets, and the gains over the degraded inputs remain positive at every source pipeline length; restored SSIM nevertheless decreases for longer pipelines, especially on DLS-NUC-100 and IR700. This result supports cross-image and cross-dataset reuse of executable restoration programs rather than purely image-specific fitting. In an intended deployment workflow, a representative degraded image can therefore be used to obtain a standalone program for the current degradation state, after which the program can run without another MLLM call.

The evidence is limited to controlled pipelines and EABO-searched source programs; it does not establish transfer to unknown sensor artifacts or arbitrary real imaging chains. A program inferred from one image may also fail to represent the full variation of a system. Extending the conditioning context from one image to multiple images or video frames is a natural direction for improving the stability and coverage of program generation.

H.4 Ablation Results

H.4.1 EABO Initialization Sensitivity

We vary the number of historical programs and the weight assigned to historical quality to measure their effect on retrieval. We first vary K_{init} with $\lambda = 0.7$, then fix $K_{\text{init}} = 7$ and vary λ; all other search settings remain unchanged. All averages include no-op samples. The results for varying K_{init} are reported in Table 17.

Table 17: Effect of historical initialization count on BO supervision quality.

K_{init}	λ	Overall	L0	L1	L2	L3	Gain	Time/sample
0	–	.8442	1.0000	.8979	.8059	.7769	–	1.1865 s
1	.7	.8479	1.0000	.9001	.8124	.7805	+.0037	1.3198 s
3	.7	.8503	1.0000	.9023	.8165	.7821	+.0061	1.5846 s
5	.7	.8507	1.0000	.9029	.8161	.7834	+.0065	1.7136 s
7	.7	.8505	1.0000	.9029	.8162	.7827	+.0063	1.5863 s
9	.7	**.8512**	1.0000	.9033	**.8181**	.7825	**+.0070**	1.8312 s
11	.7	.8501	1.0000	.9015	.8164	.7825	+.0059	1.7260 s

Increasing K_{init} from zero improves overall best-program SSIM from .8442 to .8507 at $K_{\text{init}} = 5$, showing that too few historical initializations leave useful experience unused. The gain then saturates: $K_{\text{init}} = 7$ reaches .8505, while larger values provide no consistent quality

improvement (.8512 at 9 and .8501 at 11) and can increase construction time. Thus, $K_{\text{init}} =$ 5–7 offers a practical balance between supervision quality and search cost.

We next fix $K_{\text{init}} = 7$ and vary λ; all other search settings remain unchanged. The resulting sensitivity is reported in Table 18.

Table 18: Effect of the experience quality weight λ at fixed $K_{\text{init}} = 7$.

K_{init}	λ	Overall	L0	L1	L2	L3	Time/sample
7	.0	.8505	1.0000	.9025	.8166	.7825	1.8478 s
7	.1	.8505	1.0000	.9023	.8169	.7823	1.6686 s
7	.3	.8502	1.0000	.9030	.8151	.7827	1.7457 s
7	.5	**.8508**	1.0000	.9029	**.8174**	.7822	1.7570 s
7	.7	.8505	1.0000	.9029	.8162	.7827	1.5863 s
7	.9	**.8508**	1.0000	**.9032**	.8166	**.7828**	1.8968 s
7	1.0	.8496	1.0000	.9009	.8156	.7821	1.7113 s

The retrieval weight affects supervision quality in a way that favors combining both signals. With $\lambda = 1$, retrieval uses only historical program quality Q_h and the overall best-program SSIM falls to .8496, below the best mixed settings (.8508 at $\lambda = .5$ and .9); ranking history by quality alone is therefore insufficient. At the other extreme, $\lambda = 0$ uses only severity-sequence similarity and reaches .8505, slightly below the strongest intermediate settings. The latter comparison shows that historical program quality still contributes useful information, while the former shows that it should be balanced with severity similarity rather than used in isolation.

H.4.2 Default Arguments versus Parameterized Search

We test whether exposing callable parameters improves supervision over fixed tool defaults. We construct both variants under the same data and trial budget, evaluate the resulting programs on the same samples, and report overall and length-specific average best SSIM together with construction time. Table 19 reports the comparison.

Table 19: Composite comparison of default tool arguments and parameterized EABO.

Setting	Overall	Len. 1	Len. 2	Len. 3	Time/sample
Fixed defaults	.8327	.8768	.7939	.7715	1.4558 s
EABO	**.8505**	**.9029**	**.8162**	**.7827**	1.5863 s
Difference	+.0178	+.0261	+.0223	+.0112	–

Parameterizing the callable tools consistently improves the searched supervision: EABO raises overall best-program SSIM from .8327 with fixed defaults to .8505, with gains of .0261, .0223, and .0112 for pipeline lengths one, two, and three, respectively. The improvement costs only .1305 s per sample (1.4558 s versus 1.5863 s). These results support the rich-parameter design: exposing image-adaptive arguments yields better restoration targets across pipeline lengths at a modest construction-time cost.

H.4.3 Structured-Decoding Sensitivity

We vary the transition weight to measure its effect on valid-path reconstruction. We reuse the same 100× model forward pass, keep the attribute-term weight fixed at one and $L_{\max} =$ 5, vary only β, and report ID/OOD full diagnosis accuracy, attribute F1, and transition F1. The results are given in Table 20.

Table 20: Sensitivity of valid-path decoding to β; $L_{\max} = 5$ is fixed.

β	ID Full	ID $F1_a$	ID $F1_t$	OOD Full	OOD $F1_a$	OOD $F1_t$
.5	**.6206**	**.8726**	**.5701**	.2698	.7531	.5535
1.0	.6193	.8713	.5660	.2811	.7547	.5601
1.5	.6173	.8699	.5644	**.2890**	**.7560**	.5607
2.0	.6113	.8683	.5597	**.2890**	.7547	**.5608**

Table 21: Per-sample inference efficiency on a single NVIDIA L40S GPU at batch size two.

MLLM backbone	Output design	Inference time	Output tokens	Code execution
Qwen3.5-0.8B	Direct AR Diagnosis and Restoration	2.4627 s	182	.0459 s
Qwen3.5-0.8B	Structured diagnosis + decoupled code	1.0309 s	86	.0428 s
Qwen3.5-9B	Direct AR Diagnosis and Restoration	2.3368 s	121	.0432 s
Qwen3.5-9B	Structured diagnosis + decoupled code	1.7439 s	84	.0476 s
InternVL3.5-8B	Direct AR Diagnosis and Restoration	2.2871 s	114	.0424 s
InternVL3.5-8B	Structured diagnosis + decoupled code	1.6736 s	80	.0410 s
MiMo-VL-7B	Direct AR Diagnosis and Restoration	1.8448 s	113	.0471 s
MiMo-VL-7B	Structured diagnosis + decoupled code	1.3621 s	80	.0482 s
MiniCPM-V-4.6	Direct AR Diagnosis and Restoration	1.3855 s	120	.0452 s
MiniCPM-V-4.6	Structured diagnosis + decoupled code	1.0313 s	84	.0427 s

Increasing β shifts valid-path decoding toward transition consistency. On ID, full accuracy and both F1 scores decrease as β increases: full accuracy changes from .6206 to .6113, while $F1_a$ and $F1_t$ change from .8726/.5701 to .8683/.5597. On OOD, full accuracy increases from .2698 to .2890, while $F1_a$ peaks at .7560 for $\beta = 1.5$ and $F1_t$ increases from .5535 to .5608. The ID/OOD F1 pattern indicates that attribute evidence is substantially stronger than transition evidence on ID, whereas this advantage narrows on OOD. A smaller transition weight is therefore preferable for ID, while increasing the transition weight becomes more useful for OOD, where transition structure provides relatively more informative evidence for unseen compositions. We set $\beta = 1.0$ as a balanced default rather than optimizing either split separately, obtaining .6193/.2811 full accuracy on ID/OOD. Because β is used only in diagnosis decoding, it does not change the generated restoration code or the denoising result.

I Efficiency Analysis

We analyze inference and execution efficiency below. All models are evaluated on a single NVIDIA L40S GPU with inference batch size two. We divide total inference time by the 1,505 test samples, report generated-token count, and measure the execution time of the emitted restoration program separately. The resulting values are per-sample averages rather than batch latencies. The resulting measurements are reported in Table 21. Under this common measurement protocol, decoupling reduces mean inference time by 58.1%, 25.4%, 26.8%, 26.2%, and 25.6% across the five MLLM backbones, while code execution remains within .0410–.0482 s. For Qwen3.5-0.8B, fixed-dimensional diagnosis reduces output from 182 to 86 tokens and inference from 2.4627 to 1.0309 s/sample. The direct AR Qwen3.5-0.8B model is slower than the 9B counterpart because it emits 182 versus 121 tokens.

J Qualitative Diagnosis and Restoration Results

Figure 6 gives representative Qwen3.5-9B predictions from Test-ID and Test-OOD. The examples include exact diagnosis on length-two and length-three pipelines, an OOD length-two success, and an OOD length-three diagnosis error. The figure reports the generated tool sequence together with the restored image, illustrating that restoration can remain effective even when the predicted diagnosis is not exact.

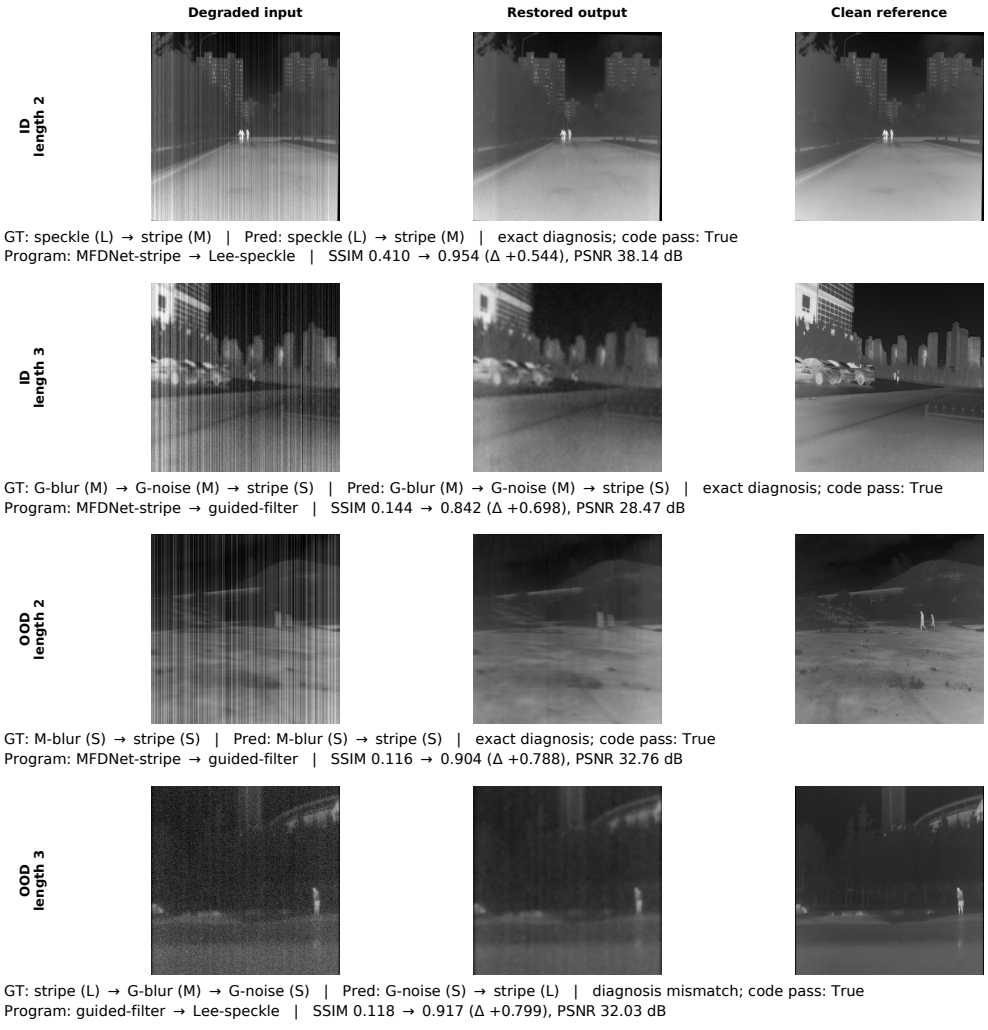

Figure 6: Qualitative diagnosis and programmatic restoration results from the Qwen3.5-9B model. Each row shows the degraded input, restored output, clean reference, and the ground-truth/predicted degradation pipelines with the generated tool sequence. The annotations report SSIM before and after restoration; L/M/S denote low, medium, and severe severity. The examples cover ID and OOD cases, including a length-three OOD diagnosis error.