Technical Report for AI4Math-2026 Track 3: VeraPhys: Verified Ensemble Repair for Multimodal Physics Reasoning

Duc M. Nguyen [1] **Sungahn Ko** [1]

Abstract

Visual physics reasoning requires systems to connect visual perception, physical laws, mathematical derivations, and faithful explanations. Although recent LLMs and VLMs have improved multimodal reasoning, they often struggle when key problem information is contained primarily in images, especially in challenging settings involving handwritten diagrams, equations, or problem statements. This technical report presents our solution for AI4Math-2026 Track 3: SeePhysPro, a multimodal physics reasoning challenge involving diagrams, expressions, and explanations. We develop VeraPhys, **V**erified **E**nsemble **R**ep**A**ir for **Phys**ics, a multi-stage ensemble pipeline that combines Qwen and GPT candidate generation, Python/SymPy-assisted reasoning, Qwen verification, OCR-enhanced visual evidence, and conservative answer repair. On the official `test-mini` evaluation, our approach achieves 70.96% overall accuracy.

1. Introduction

Reasoning has become a central capability for modern language models. In pure-text settings, recent large language models (LLMs) have shown strong performance on mathematical derivation, symbolic manipulation, and step-by-step problem solving (Wei et al., 2022; Wang et al., 2023; Yao et al., 2023a). At the same time, vision-language models (VLMs) have extended this progress to multimodal tasks that require models to read, see, and reason jointly. Despite this progress, a substantial gap remains when the solution depends on visual evidence rather than text alone (Xiang et al., 2026).

Visual physics reasoning is a representative example of this gap. Physics problems often require a system to identify quantities from diagrams, read expressions and labels, connect visual evidence with physical laws, and perform symbolic or numerical derivations. Model performance can degrade sharply when the decisive information appears only in the image, such as a force direction, circuit connection, graph feature, geometric constraint, or handwritten annotation (Xiang et al., 2026).

This technical report describes our solution for SeePhysPro. Building on prior work in multimodal science and mathematical reasoning (Lu et al., 2022; 2024; Yue et al., 2024; 2025), we build a multi-stage ensemble around Qwen and GPT candidate generators, combining diagram-aware prompting, Python/SymPy-assisted calculation, self-consistency sampling, Qwen candidate selection, evidence-based resolution for low-agreement cases, and conservative answer post-processing (Gao et al., 2023; Chen et al., 2023; Yao et al., 2023b). The pipeline separates visual interpretation, physics reasoning, answers selection, and final-format normalization so that errors can be detected and corrected at multiple stages. The main contributions of this report are summarized as follows:

- We present an ensemble pipeline for SeePhysPro that combines tool-assisted problem solving, self-consistency, candidate selection, and evidence-based resolution for multimodal physics answer selection.
- We show that a Qwen-only variant reaches 65.42% accuracy, comparable to GPT-5.5, while integration of GPT candidates and OCR-VL refinements improves the final system to 70.96% accuracy on `test-mini`.

2. Related Work

Multimodal scientific and mathematical reasoning. Recent multimodal benchmarks have exposed a persistent gap between text-only reasoning and reasoning grounded in visual evidence. ScienceQA introduced a large multimodal science question-answering benchmark with explanations, highlighting the role of explicit reasoning traces for scientific questions (Lu et al., 2022). MathVista broadened this direction to mathematical reasoning in visual contexts, com-

[1]Graduate School of Artificial Intelligence, Pohang University of Science and Technology, Pohang, Republic of Korea. Correspondence to: Sungahn Ko <sungahn@postech.ac.kr>.

The 3rd *AI for Math Workshop* at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

bining tasks that require fine-grained visual understanding with compositional mathematical inference (Lu et al., 2024). MMMU further evaluated expert-level multimodal understanding across many disciplines and heterogeneous visual formats (Yue et al., 2024). MMMU-Pro makes this evaluation more stringent by filtering questions answerable by text-only models, augmenting candidate options, and introducing a vision-only setting where questions are embedded in images (Yue et al., 2025). SeePhys Pro is closest to our setting: it studies physics reasoning under modality transfer, where problem information is progressively moved from text into images, and identifies visual variable grounding as a key bottleneck (Xiang et al., 2026). Our work targets this same failure mode in the AI4Math Track 3 setting, where diagrams, labels, handwritten notations, and answer formats must be handled jointly. High-resolution MLLM work addresses a related perception bottleneck by processing images through slices, tiles, or local crops, preserving details that are lost when a large image is compressed into a single fixed-resolution input (Guo et al., 2024; Chen et al., 2024; Ma et al., 2024).

Prompted reasoning and test-time scaling. Chain-of-thought prompting showed that generating intermediate reasoning steps can substantially improve complex reasoning in sufficiently capable language models (Wei et al., 2022). Self-consistency improves on greedy chain-of-thought decoding by sampling multiple reasoning paths and selecting the most consistent answer (Wang et al., 2023). More structured inference-time procedures further treat reasoning as an interactive or search process: ReAct interleaves reasoning with actions (Yao et al., 2023b), while Tree of Thoughts explicitly explores and evaluates multiple intermediate reasoning traces, and formulate LLM's reasoning process as tree search (Yao et al., 2023a). Recent work has also begun to characterize test-time compute as a scaling resource. Snell et al. (2025) show that compute-optimal allocation can outperform simple best-of-N scaling for mathematical reasoning. Wu et al. (2025) study inference scaling laws across majority voting, best-of-N, weighted voting, and tree-search strategies. Zhang et al. (2025) learns how to allocate sampling budgets across inference configurations. These ideas motivate our use of multiple independent samples per example, diverse prompt profiles, and candidate support as test-time evidence. Unlike simple majority voting, however, SeePhysPro answers can be single-choice, multi-choice, numerical, or symbolic, and visually grounded errors often produce high-frequency but wrong candidates. We therefore combine self-consistency with answer normalization, candidate grouping, and conservative VLM-based selection rather than relying only on vote frequency.

Tool-augmented and program-aided reasoning. A complementary line of work improves numerical and symbolic reliability by delegating computation to executable tools. PAL uses language models to generate programs while offloading arithmetic and symbolic execution to a Python runtime (Gao et al., 2023); Program-of-Thought prompting similarly separates reasoning from computation for numerical tasks (Chen et al., 2023). ReAct interleaves reasoning traces with external actions, showing that tool or environment interaction can reduce hallucination and improve interpretability (Yao et al., 2023b). For visual tasks, VISPROG and ViperGPT generate Python-like programs that call vision modules and other routines, enabling training-free compositional visual reasoning (Gupta & Kembhavi, 2023; Surís et al., 2023). Our pipeline follows this tool-augmented philosophy but uses the interpreter mainly as a physics scratchpad for computation, symbolic simplification, unit checks, and option verification, while leaving visual perception to the VLM.

3. Methodology

Let $x_i = (q_i, \mathcal{I}_i)$ denote one test question, where q_i is the text field and $\mathcal{I}_i$ is the associated image set. The system returns $\hat{y}_i = (\hat{a}_i, \hat{\tau}_i)$, consisting of a final answer and a reasoning trace. Figure 1 summarizes the final inference pipeline.

3.1. Candidate Generation

The pipeline uses a set of frozen generator profiles $\mathcal{G}$. Each profile $g \in \mathcal{G}$ specifies a VLM, an instruction prompt, decoding parameters, and an optional local Python/SymPy tool T_g. For sample index r, profile g produces

$$c_{i,g,r} = (a_{i,g,r}, \tau_{i,g,r}) \sim \pi_g(\cdot \mid x_i, p_g, T_g),$$

where $a_{i,g,r}$ is the extracted answer, $\tau_{i,g,r}$ is the reasoning trace, and p_g is the instruction prompt used by profile g. The Qwen profiles include exact-answer reasoning, option-by-option auditing, no-tool reasoning, Python/SymPy-assisted solving, and wide-panel image reruns for difficult level-4 questions. The GPT profiles contribute an $N = 1$ solve without in-context demonstrations and targeted multiple-choice audits. Tool calls are used to carry out local computation, including arithmetic, equation solving, symbolic simplification, unit checks, and option-equivalence tests.

The wide-panel profile is a level-4 perception pass for image-heavy questions. Instead of sending only the original image, it sends the full diagram together with labeled overlapping crops of the same image, up to five image inputs in total. These crops expose small labels, answer-option text, graph axes, circuit connections, and geometry that may be compressed in the full view. The resulting candidate is treated like any other generated answer: it enters the candidate pool and can replace the base answer only through the conservative selection rules below. This is related to high-resolution

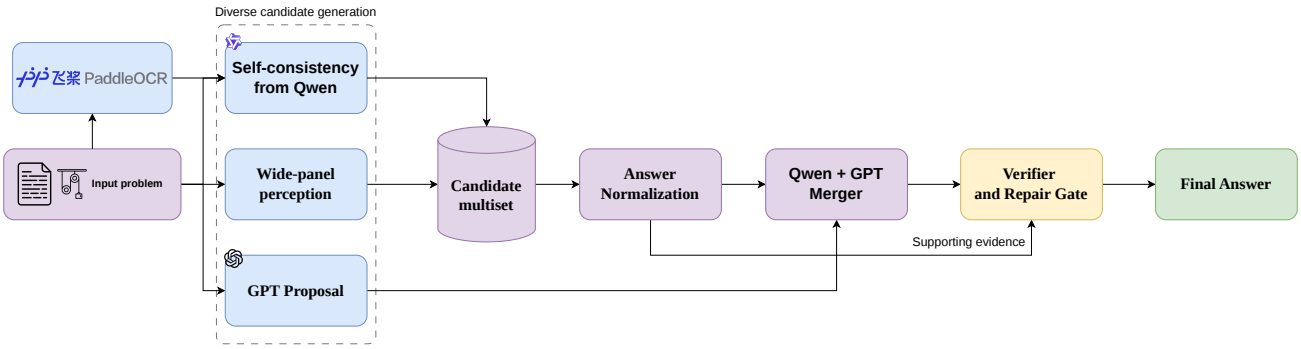

Figure 1. Overview of the final inference pipeline. Diverse VLM and tool-assisted profiles generate candidate answers; normalization converts them into vote keys and support counts; PaddleOCR-VL produces textual interpretation for the image-heavy cases; a mixed Qwen/GPT base is then modified only by conservative verifier and repair gates.

VLM approaches that tile or crop images to preserve local details (Guo et al., 2024; Chen et al., 2024; Ma et al., 2024), but we do not modify the visual encoder; we attach labeled crops as additional image inputs at inference time. We provide an example of this module in Appendix B.

To improve the performance on image-heavy questions, i.e., Level 3 and Level 4, we introduce an OCR module for textual interpretation. For each questions in Level 3 and Level 4, we query PaddleOCR-VL-1.6 (Zhang et al., 2026) through an OpenAI-compatible endpoint to obtain a compact perception sheet from the same image, containing visible problem text, answer choices, symbols, units, graph/-diagram facts, topology, spatial relations, and ambiguous readings.

3.2. Answer Normalization

Let $\mathcal{A}$ be the set of raw extracted answer strings and $\mathcal{T}$ be the set of reasoning traces. For each question x_i, define the candidate index set

$$\mathcal{J}_i = \{(g, r) : g \in \mathcal{G}, r \in R_g, a_{i,g,r} \neq \emptyset\}.$$

The candidate pool is an indexed multiset

$$\mathcal{C}_i = \biguplus_{j \in \mathcal{J}_i} \{c_{i,j}\}, \qquad c_{i,j} = (a_{i,j}, \tau_{i,j}) \in \mathcal{A} \times \mathcal{T},$$

where $j = (g, r)$. We use a multiset rather than an ordinary set because two independent generator calls that produce the same answer should contribute two votes.

Answers are mapped to normalized vote keys by a deterministic function

$$\nu : \mathcal{A} \to \mathcal{K},$$

where $\mathcal{K}$ is the space of normalized answer keys. For a multiple-choice answer, if the stripped and upper-cased string consists only of letters in $\{A, B, C, D\}$ and separators, ν removes separators, removes duplicates, and sorts the letters. Thus BA, A, B, and AB share the same key. For non-choice answers, ν applies only formatting-level normalization: it removes answer prefixes, outer math delimiters, redundant whitespace, and superficial wrappers while preserving the displayed expression or text.

The set of keys observed for question i is

$$\mathcal{K}_i = \{\nu(a_{i,j}) : j \in \mathcal{J}_i\}.$$

For each $\kappa \in \mathcal{K}_i$, define its equivalence class and support as

$$\mathcal{C}_i(\kappa) = \{c_{i,j} \in \mathcal{C}_i : \nu(a_{i,j}) = \kappa\}, \qquad s_i(\kappa) = |\mathcal{C}_i(\kappa)|.$$

When a rule needs a concrete answer string for a key, we choose the representative

$$r_i(\kappa) = \arg\max_{c \in \mathcal{C}_i(\kappa)} Q(c),$$

using a deterministic lexicographic tie-breaker. For $c = (a, \tau)$, define

$$Q(c) = (-\mathbf{1}[\mathsf{susp}(a)], \mathbf{1}[a \neq \emptyset], \mathbf{1}[\mathsf{choice}(a)], \mathbf{1}[|a| \leq 80], \mathbf{1}[\mathsf{expr}(a)], \mathbf{1}[\tau \neq \emptyset], -|a|, -|\tau|).$$

The tuples are compared lexicographically. Here $\mathsf{expr}(a)$ detects a numeric or mathematical expression, and $\mathsf{susp}(a)$ detects placeholder answers such as unsupported "cannot determine" outputs.

3.3. Base Construction

The base answer is constructed from complementary Qwen and GPT branches. The Qwen branch first builds an open-source prediction b_i^Q using $N = 13$ test-time candidate pools, Python/SymPy-assisted symbolic cleanup, and conservative multiple-choice option repair. GPT candidates are then introduced through guarded replacement rules. A GPT $N = 1$ answer may repair low-confidence Qwen outputs only when it is non-empty and valid. For GPT

multiple-choice audits, replacement requires at least three independent candidate sources supporting the proposed option set $s_i(\nu(\tilde{a}_i)) \geq 3$. The result is the mixed-model base $b_i = (a_i^b, \tau_i^b)$.

3.4. Verifier and Repair Gates

The diverse candidate pool is high-recall but noisy, so we let the selector and repair modules propose changes, while deterministic gates accept them only under support and validity checks. Formally, for any candidate u, let a_u denote its answer and τ_u denote its reasoning trace, and write $k(u) = \nu(a_u)$. Thus $k(u)$ is the normalized vote key for candidate u, and $s_i(k(u))$ is the number of generated candidates that support the same answer. Let $\kappa_i^\star = \arg\max_{\kappa \in \mathcal{K}_i} s_i(\kappa)$, and $t_i = r_i(\kappa_i^\star)$. Here t_i is the highest-support candidate. We use $\mathsf{bad}(a)$ for known suspect answers, $\mathsf{choice}(a)$ for valid A–D option sets, and $k(z) \subset k(y)$ for strict option-subset inclusion after normalization. The repair modules V_{sub} and V_{fmt} are VLM-based checks: V_{sub} tries to remove extra multiple-choice letters, while V_{fmt} tries to clean a malformed final answer. Each module returns either a proposed candidate z or $\perp$, which means no repair; V_{sub} additionally returns a confidence label γ_z.

The gates used in Algorithm 2 are

$$
\begin{aligned}
A_{\text{sel}}(b, \tilde{c}, d, \gamma) &\equiv d = \text{replace} \wedge \gamma = \text{high} \\
&\quad \wedge s_i(k(\tilde{c})) \geq 5 \wedge s_i(k(b)) \leq 2 \\
&\quad \wedge k(\tilde{c}) \neq k(b) \wedge U(a_{\tilde{c}}) = 0, \\
A_{\text{susp}}(y) &\equiv \mathsf{bad}(a_y) \wedge s_i(k(t_i)) \geq 5 \\
&\quad \wedge s_i(k(y)) \leq 1, \\
A_{\text{sub}}(y, z, \gamma_z) &\equiv z \neq \perp \wedge \gamma_z = \text{high} \wedge k(z) \subset k(y) \\
&\quad \wedge s_i(k(z)) \geq 6 \wedge s_i(k(y)) \leq 3, \\
B_1(y, z) &\equiv F(y) \wedge s_i(k(z)) \geq 5 \wedge s_i(k(y)) \leq 1, \\
B_2(y, z) &\equiv \mathsf{choice}(a_y) \wedge k(z) \subset k(y) \\
&\quad \wedge s_i(k(z)) \geq 8 \wedge s_i(k(y)) \leq 1, \\
A_{\text{fmt}}(y, z) &\equiv z \neq \perp \wedge \big(B_1(y, z) \vee B_2(y, z)\big),
\end{aligned}
$$

where U marks empty, malformed, or unsupported answers, and $F(y)$ marks clearly corrupted non-choice formatting. In words, a replacement is accepted only when the proposed answer is well-supported, the current answer is weakly supported, and the proposed change is either high-confidence or a narrow formatting/subset repair.

3.5. Overall Procedure

The overall predictor can be written compactly as

$$
\hat{y}_i = R_{\text{fmt}}(R_{\text{sub}}(R_{\text{susp}}(G_{\text{sel}}(B_{\text{GPT+Qwen}}(x_i), \mathcal{C}_i)))) \,,
$$

where $B_{\text{GPT+Qwen}}$ is the guarded mixed-model base, G_{sel} is the Qwen selector gate, R_{susp} is suspect-answer repair, R_{sub} is option-subset repair, and R_{fmt} is final format cleanup.

Algorithm 1 Candidate construction

Require: $x_i, \mathcal{G}, \{R_g\}_{g \in \mathcal{G}}, \nu$
Ensure: Candidate pool $\mathcal{C}_i$, support map s_i

```
C_i ← ∅,   s_i(κ) ← 0 ∀κ
for g ∈ G do
    for r ∈ R_g do
        c_{i,g,r} = (a_{i,g,r}, τ_{i,g,r}) ~ π_g(· | x_i, p_g, T_g)
        if a_{i,g,r} ≠ ∅ then
            κ ← ν(a_{i,g,r})
            C_i ← C_i ⊎ {c_{i,g,r}}
            s_i(κ) ← s_i(κ) + 1
        end if
    end for
end for
return C_i, s_i
```

4. Experiment

4.1. Implementation Details

The main open-source model is Qwen3.6-35B-A3B (Qwen Team, 2026) served through a local OpenAI-compatible endpoint using vLLM (Kwon et al., 2023). Unless explicitly marked as no-tool, Qwen self-consistency branches are tool-assisted: they may call a Python interpreter with SymPy available for symbolic algebra, equation solving, numeric evaluation, and option-equivalence checks. The GPT branch consists of batch GPT-5.5 $N = 1$, which solves without in-context demonstrations and targeted multiple-choice audits. The ordinary VLM solvers receive the current problem text and image; the OCR-VL branch first sends the same image to PaddleOCR-VL, then sends the resulting perception sheet to a text-only solver. Candidate answers are extracted from explicit answer tags, normalized into vote keys, and then passed through the verifier and repair gates described in Algorithms 1 and 2. The code can be found at https://github.com/kurone02/veraphys.

4.2. Evaluation Setup

In our analysis, we observed that some questions across levels share related indices and problem structure, and a simple answer-copy from a previous level to a higher level results in a significant boost in performance. Doing so would defeat the purpose of the challenge, where the goal is to build a robust multimodal reasoning system. Thus, we treat this only as a dataset characteristic: all reported predictions are produced independently for each question, using only that question's prompt, image, and generated candidates. We evaluate on the public `test-mini` split

Algorithm 2 Guarded answer selection and repair

Require: $x_i, b_i = (a_i^b, \tau_i^b), \mathcal{C}_i, s_i, \nu$
Ensure: Final prediction $\hat{y}_i$

$y \leftarrow b_i$
$(d, \tilde{a}, \gamma, \tilde{\tau}) \leftarrow S(x_i, b_i, \mathcal{C}_i)$
$\tilde{c} \leftarrow (\tilde{a}, \tilde{\tau})$
if $A_{\text{sel}}(b_i, \tilde{c}, d, \gamma)$ **then**
  $y \leftarrow \tilde{c}$
end if
if $A_{\text{susp}}(y)$ **then**
  $y \leftarrow t_i$
end if
$(z, \gamma_z) \leftarrow V_{\text{sub}}(x_i, y, \mathcal{C}_i)$
if $A_{\text{sub}}(y, z, \gamma_z)$ **then**
  $y \leftarrow z$
end if
$z \leftarrow V_{\text{fmt}}(x_i, y, \mathcal{C}_i)$
if $A_{\text{fmt}}(y, z)$ **then**
  $y \leftarrow z$
end if
return y

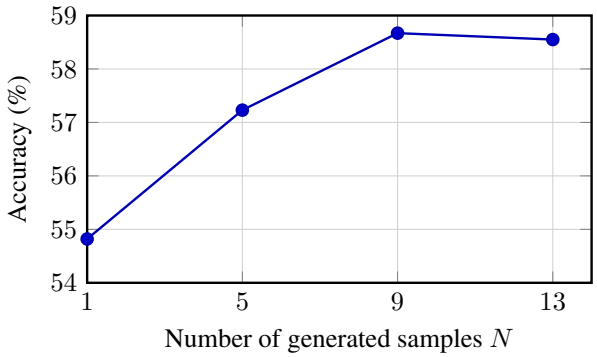

Figure 2. Test-time scaling trend for selected tool-assisted self-consistency budgets.

with 830 questions: 200 questions each from levels 1–4, and 30 questions from the `solve150` split, treated as level 5.

4.3. Test-Time Scaling of Tool-Assisted Self-Consistency

Here N denotes the number of independently generated tool-assisted Qwen responses for each question under the same solver profile. Each response yields one normalized answer candidate, and self-consistency selects the plurality answer among these N candidates. This experiment, therefore, isolates the effect of increasing the same-profile test-time samples, rather than adding new candidate solvers.

Figure 2 reports the effect of increasing the self-consistency budget. Accuracy improves from 54.82% at $N = 1$ to 57.23% at $N = 5$, and then to 58.67% at $N = 9$. Increasing the budget further to $N = 13$ does not help, slightly decreasing to 58.55%. The curve shows that additional same-profile samples help up to a point, but later samples mainly reshuffle ambiguous cases. If several samples share the same wrong diagram interpretation, plurality voting amplifies that error rather than correcting it. The final ensemble pipeline, therefore, does not rely on scaling N alone; it must combine diverse candidate sources with constrained selection and repair to be effective.

4.4. Main Result

Table 1 presents the main results on `test-mini`. VeraPhys achieves the highest overall accuracy among systems evaluated on all five levels, with 70.96% accuracy, corresponding to 589 correct predictions out of 830 questions. This gain is not due to consistently outperforming all baselines at every level: Gemini-3.1-Pro remains slightly stronger on the most visually demanding levels, whereas VeraPhys performs best on levels 1 and 2 and remains competitive on levels 3 and 4. Its overall advantage instead comes from a more balanced level-wise profile. Compared with the GPT-5.5 ($N = 1$) reference, VeraPhys improves overall accuracy from 65.78% to 70.96% and reduces the level-1-to-level-4 transfer gap from 21.00 to 11.50 points, indicating better robustness as more information is transferred from text into visual form.

The remaining weakness is concentrated in levels 3 and 4, where questions often require precise diagram interpretation, fine-grained visual reading, and complete option-set selection. This is consistent with the failure mode observed in 2: repeated samples from a single solver profile may agree on the same incorrect visual interpretation. VeraPhys mitigates this by combining GPT-5.5 outputs with Qwen candidates and applying conservative selection and repair, rather than relying on either pure self-consistency or a single strong generator. The results therefore suggest that the main benefit comes from heterogeneous candidate generation plus guarded aggregation, while perception errors and incomplete option-set reasoning remain the primary bottlenecks.

4.5. Ablation Study

Table 2 ablates the main components of VeraPhys on `test-mini`. Starting from Qwen self-consistency with $N = 5$, adding SymPy-assisted cleanup and option repair gives the largest single gain in the Qwen-only block, improving overall accuracy from 57.23% to 63.73%. The improvement is especially clear on levels 2 to 4, suggesting that many errors are not caused only by weak reasoning, but also by algebraic simplification mistakes, answer-format mismatches, and incomplete multiple-choice option sets. The wide-panel level-4 selector and PaddleOCR-VL further improve visually demanding cases, raising level-4 accuracy from 55.00% to 59.00% and overall accuracy to 65.42%. This shows that targeted visual reruns and OCR-style ev-

Table 1. Main comparison on `test-mini`. Entries are accuracies in percent. All models and methods are evaluated on `test-mini` subset. The Transfer Gap columns follow the modality-transfer metrics of Xiang et al. (2026): $\Delta_S = A_1 - A_2$, $\Delta_V = A_2 - A_3$, $\Delta_R = A_3 - A_4$, and $\Delta_T = A_1 - A_4$, where smaller positive gaps indicate less degradation as information is moved into images. The first 5 results are taken from Xiang et al. (2026).

System	L1	L2	L3	L4	L5	Overall	Δ_S	Δ_V	Δ_R	Δ_T
Gemini-3.1-Pro	71.00	72.00	**66.50**	**66.50**	–	–	-1.00	5.50	**0.00**	4.50
Claude-4.7-Opus	74.00	67.00	56.50	46.50	–	–	7.00	10.50	10.00	27.50
Kimi K2.5	52.00	48.50	46.00	42.00	–	–	3.50	**2.50**	4.00	10.00
Qwen3.5-122B-A10B	47.50	48.00	39.50	42.00	–	–	**-0.50**	8.50	-2.50	5.50
SuperNova	25.50	27.50	20.00	22.00	–	–	-2.00	7.50	-2.00	**3.50**
GPT-5.5	73.00	74.00	60.50	52.00	90.00	65.78	-1.00	13.50	8.50	21.00
VeraPhys	**75.50**	**74.50**	66.00	64.00	**96.67**	**70.96**	1.00	8.50	2.00	11.50

Table 2. Ablation study on `test-mini`. Entries are accuracies in percent. Here Qwen SC denotes tool-assisted Qwen self-consistency with Python/SymPy unless a no-tool branch is named explicitly. Gap columns follow the modality-transfer metrics similar to Table 1.

Variant	L1	L2	L3	L4	L5	Overall	Δ_S	Δ_V	Δ_R	Δ_T
Qwen SC, $N = 5$	64.50	58.00	51.00	49.50	96.67	57.23	6.50	7.00	1.50	15.00
+ SymPy and option repair	70.00	67.00	58.00	55.00	96.67	63.73	3.00	9.00	3.00	15.00
+ Wide-panel L4 selector	69.50	67.50	58.00	57.50	96.67	64.34	2.00	9.50	**0.50**	12.00
+ PaddleOCR-VL-1.6	70.50	67.00	60.50	59.00	96.67	65.42	3.50	6.50	1.50	11.50
GPT-5.5, $N = 1$	73.00	74.00	60.50	52.00	90.00	65.78	-1.00	13.50	8.50	21.00
+ GPT low-conf replace	72.50	75.00	62.00	58.00	96.67	67.95	-2.50	13.00	4.00	14.50
+ GPT choice audit	73.00	75.00	62.50	58.00	96.67	68.19	-2.00	12.50	4.50	15.00
+ Qwen selector	75.00	75.00	61.50	59.00	96.67	68.67	**0.00**	13.50	2.50	16.00
+ Repair gates	**76.00**	**75.00**	62.50	59.00	96.67	69.16	1.00	12.50	3.50	17.00
+ PaddleOCR-VL-1.6	75.50	74.50	**66.00**	**64.00**	**96.67**	**70.96**	1.00	**8.50**	2.00	**11.50**

idence are useful, but their benefits are concentrated on image-heavy examples rather than uniformly distributed across all levels.

GPT-5.5 with $N = 1$ is a strong standalone reference, achieving 65.78% overall accuracy, but it has a large level-1-to-level-4 transfer gap of 21.00 points. Adding GPT candidates to the Qwen pipeline through low-confidence replacement improves the strongest Qwen-only setting from 65.42% to 67.95%, while choice auditing, Qwen-based selection, and repair gates provide smaller but consistent gains. These results indicate that VeraPhys does not simply substitute Qwen with GPT-5.5; instead, it uses GPT outputs as additional candidates under guarded selection and repair.

The final addition of PaddleOCR-VL-1.6 gives the strongest result, improving overall accuracy from 69.16% to 70.96%. The gain is mainly on levels 3 and 4, where accuracy increases from 62.50% to 66.00% and from 59.00% to 64.00%, respectively. It also reduces the total transfer gap (Δ_T) to 11.50 points, the best among the ablated variants. Overall, the ablation supports the design of VeraPhys as a heterogeneous inference-time system: self-consistency improves reasoning diversity, tools and repair modules correct structured answer errors, GPT candidates add strong alternative solutions, and OCR-enhanced visual evidence addresses the remaining perception bottleneck.

5. Future Work

The current system improves accuracy through inference-time ensembling and conservative verification, but several limitations remain. First, the candidate pool is still bounded by the visual interpretations produced by the underlying VLMs. Stronger diagram parsing, OCR, and structured visual grounding could increase candidate diversity in a more meaningful way than simply sampling more answers. Second, the selector is conservative by design, which avoids many regressions but also leaves correct minority candidates unused. Future work could use calibrated uncertainty estimates, option-level truth tables, or executable diagram representations to accept more corrections safely. Finally, the current system uses frozen models only. Training or preference-tuning on visual physics reasoning traces may further improve robustness, especially for levels 3–4 where the remaining errors are dominated by image-grounded multi-step reasoning.

6. Conclusion

We presented an ensemble pipeline for SeePhysPro visual physics reasoning. The system combines Qwen and GPT candidate generation, Python/SymPy-assisted solving, answer normalization, Qwen selection, OCR-enhanced visual evidence, and deterministic repair rules. The main design principle is to improve test-time robustness without training while processing each question independently. On `test-mini`, the final pipeline reaches 70.96% overall accuracy. Ablations show that the largest improvements come from combining heterogeneous Qwen and GPT candidates, while additional guarded repair and OCR-based evidence provide smaller but reliable gains, especially on visually demanding levels. Remaining errors are concentrated in levels 3 and 4, where precise diagram interpretation and complete multi-option selection remain challenging. These results suggest that inference-time ensembling and conservative verification can substantially improve visual physics reasoning without task-specific training, while also highlighting perception and option-set completeness as key directions for future improvement.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Chen, W., Ma, X., Wang, X., and Cohen, W. W. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=YfZ4ZPt8zd.

Chen, Z., Wang, W., Tian, H., Ye, S., Gao, Z., Cui, E., Tong, W., Hu, K., Luo, J., Ma, Z., et al. How far are we to GPT-4V? closing the gap to commercial multimodal models with open-source suites. *arXiv preprint arXiv:2404.16821*, 2024.

Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., and Neubig, G. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10764–10799. PMLR, 2023.

Guo, Z., Xu, R., Yao, Y., Cui, J., Ni, Z., Ge, C., Chua, T.-S., Liu, Z., and Huang, G. Llava-uhd: An lmm perceiving any aspect ratio and high-resolution images. In *Computer Vision - ECCV 2024: 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part LXXXIII*, pp. 390–406, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-3-031-73009-2. doi: 10.1007/978-3-031-73010-8_23. URL https://doi.org/10.1007/978-3-031-73010-8_23.

Gupta, T. and Kembhavi, A. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14953–14962, 2023.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Lu, P., Mishra, S., Xia, T., Qiu, L., Chang, K.-W., Zhu, S.-C., Tafjord, O., Clark, P., and Kalyan, A. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *Advances in Neural Information Processing Systems*, 2022.

Lu, P., Bansal, H., Xia, T., Liu, J., Li, C., Hajishirzi, H., Cheng, H., Chang, K.-W., Galley, M., and Gao, J. MathVista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations*, 2024.

Ma, Y., Wang, Z., Sun, X., Lin, W., Zhou, Q., Ji, J., and Ji, R. INF-LLaVA: Dual-perspective perception for high-resolution multimodal large language model. *arXiv preprint arXiv:2407.16198*, 2024.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, April 2026. URL https://qwen.ai/blog?id=qwen3.6-35b-a3b.

Snell, C. V., Lee, J., Xu, K., and Kumar, A. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Surís, D., Menon, S., and Vondrick, C. ViperGPT: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11854–11864, 2023.

Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in lan-

guage models. In *International Conference on Learning Representations*, 2023.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q. V., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Wu, Y., Sun, Z., Li, S., Welleck, S., and Yang, Y. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=j7DZWSc8qu.

Xiang, K., Zhang, T. J., Liu, Z., Zhou, B., Tang, Y., Yu, J., Lu, J., Huang, S., Li, H., Zhang, L., Liu, K., Zhang, C., Fang, Y., Guo, B., Zhen, H.-L., Tu, D., Huang, Y., and Liang, X. SeePhys Pro: Diagnosing modality transfer and blind-training effects in multimodal RLVR for physics reasoning. *arXiv preprint arXiv:2605.09266*, 2026.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023a.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023b.

Yue, X., Ni, Y., Zhang, K., Zheng, T., Liu, R., Zhang, G., Stevens, S., Jiang, D., Ren, W., Sun, Y., Wei, C., Yu, B., Yuan, R., Sun, R., Yin, M., Zheng, B., Yang, Z., Liu, Y., Huang, W., Sun, H., Su, Y., and Chen, W. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9556–9567, 2024.

Yue, X., Zheng, T., Ni, Y., Wang, Y., Zhang, K., Tong, S., Sun, Y., Yu, B., Zhang, G., Sun, H., Su, Y., Chen, W., and Neubig, G. MMMU-pro: A more robust multi-discipline multimodal understanding benchmark. In Che, W., Nabende, J., Shutova, E., and Pilehvar, M. T. (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15134–15186, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.736. URL https://aclanthology.org/2025.acl-long.736/.

Zhang, K., Zhou, S., Wang, D., Wang, W. Y., and Li, L. Scaling LLM inference efficiently with optimized sample compute allocation. In Chiruzzo, L., Ritter, A., and Wang, L. (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7959–7973, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.404. URL https://aclanthology.org/2025.naacl-long.404/.

Zhang, Z., Liu, H., Liang, S., Zhang, Y., Xiang, Y., Liu, J., Sun, T., Lin, M., Zhang, Y., Zhou, C., Gao, T., Cui, C., Liu, Y., Yu, D., and Ma, Y. PaddleOCR-VL-1.6: Expanding the frontier of document parsing with under-optimized region refinement and progressive post-training. *arXiv preprint arXiv:2606.03264*, 2026. URL https://arxiv.org/abs/2606.03264.

A. Prompt Templates

This appendix gives the main prompt templates used by VeraPhys. Placeholders such as {problem_prompt}, {candidate_block}, and {ocr_notes} are filled independently for each example. The image inputs are attached as multimodal message parts following the text prompt; no candidate prompt receives information from other examples or from reference answers.

```
[System]
You are solving SeePhys Pro physics problems. Think privately and use the available python tool as a calculation
     scratchpad before finalizing when tools are enabled. Use python for equations, numerical checks, unit
     conversions, and option comparisons. Respond finally with <reasoning>...</reasoning> and <answer>...</answer>
      only. The answer must be the final answer only. The reasoning must be concise (1-3 sentences) and support
     the answer. Include units when required. If you are running out of space, end the reasoning early and still
     output <answer>.

[Python/SymPy tool addendum]
Work like a coding agent solving a physics problem: think privately, use the python tool as a scratchpad, then
     give a concise final answer. Deliberately call python before finalizing whenever tools are available. Use it
     to compute numerical values, solve equations, check units, compare answer options, or verify symbolic
     relationships. SymPy is available; import sympy as sp for symbolic algebra, equation solving, simplification,
      dimensional checks, exact fractions/radicals, and option equivalence tests. Print compact intermediate
     results from the script so you can inspect signs, roots, domains, and units before answering. Do not use
     python for web access or external data lookup. If the python tool fails, continue from your own physics
     reasoning. Final output must contain <reasoning>...</reasoning> and <answer>...</answer> only.

[User]
Solve the problem.

{input}
```

Figure 1. Core Qwen tool-assisted solver prompt.

```
[System addendum]
You are an expert multimodal physics solver. Solve the problem independently from the text and image. Be careful
     with diagrams, answer choices, units, multi-select choices, and symbolic equivalence. Final output must
     contain <reasoning>...</reasoning> and <answer>...</answer> only.

[User]
Independent sample #{sample_index}. Use the following solving style:
{variant_instruction}

{optional_description_or_ocr_block}

Problem:
{problem_prompt}

Use the model's private reasoning for the detailed work. In the visible response, give only a concise verification
      and the final answer.

Final output must contain exactly these two tags:
<reasoning>...</reasoning>
<answer>...</answer>
Put only the final answer inside <answer>. For multiple-choice questions, output only the option letter(s), such
     as A or BD. For numeric or symbolic answers, use the simplest exact equivalent expression and include units
     when required. Do not substitute g=9.8 or g=10 unless the problem explicitly gives that value. Prefer exact
     fractions, radicals, pi, and symbols over rounded decimals unless the problem asks for a decimal. Keep <
     reasoning> under 180 words. If space is running out, end the reasoning early and still output <answer>.
```

Figure 2. Self-consistency sample prompt. The profile instruction is selected from the profile list below.

```
direct:
Solve from first principles. Identify the relevant physics law, set up the equations, and compute or simplify the
     requested answer.

option_audit:
If the problem has A-D choices, evaluate each option independently. For multi-select questions, include every true
      option and exclude every false option. Do not choose an option just because it looks familiar.

choice_matrix:
Treat each A-D option as a separate proposition. First determine whether the question asks for the correct option,
      incorrect option, unrelated quantity, ranking, or a multi-select set. Use the image labels before judging
     options. In the concise visible reasoning, state only the truth status of the options you need, then output
     the final sorted option letter(s). If there are no options, solve the numeric or symbolic problem normally.

diagram_first:
First use the image to establish geometry, signs, circuit connections, directions, and labels. Then solve using
```

```
    only those verified diagram facts.

algebra_units:
Emphasize equations, units, dimensions, limiting cases, and sign conventions. Check that the final expression has
    the requested form.

skeptical:
Solve skeptically. Look for traps such as RMS vs peak value, direction signs, omitted options in multi-select
    answers, diagram scale ambiguity, and hidden constraints in the image.

exact_target:
First identify whether the requested answer is option letter(s), a number, or a symbolic expression. For choices,
    verify every option A-D and do not drop true options in multi-select problems. For non-choice answers, keep
    variables symbolic, preserve units, avoid rounded decimals, and never replace g with a number unless the
    problem states one.

exact_target_python:
First identify whether the requested answer is option letter(s), a number, or a symbolic expression. Use Python as
     a scratchpad to verify arithmetic, exact algebra, units, option comparisons, and equation solutions before
    finalizing. For choices, verify every option A-D and do not drop true options in multi-select problems. For
    non-choice answers, keep variables symbolic, preserve units, avoid rounded decimals, and never replace g with
     a number unless the problem states one.

symbolic_python:
First identify the answer target and all relevant symbols from the text and image. Use Python with SymPy before
    finalizing whenever there is algebra, an equation system, an inequality, exact numeric arithmetic, or formula-
    option comparison. In the script, define named symbols with positive/real assumptions where appropriate,
    solve or simplify the derived equations, compare candidate expressions by simplifying their difference or
    ratio, and print the accepted roots/domains. For A-D choices, build a truth table for every option when
    possible; for non-choice answers, keep exact symbolic form and required units.

sympy_final_line:
First identify the answer target and all relevant symbols from the text and image. Before finalizing, run a Python
    /SymPy scratchpad that prints one line beginning with FINAL_ANSWER: followed by the exact answer to copy. Use
     SymPy to solve equations, simplify expressions, compare formula options, and check roots/domains. For A-D
    choices, print a truth table and make FINAL_ANSWER the sorted true option letters. After the tool returns,
    copy the FINAL_ANSWER value exactly unless the tool script plainly encoded the wrong visual/problem facts.

truth_table_exact:
If answer choices A-D are present, solve as an independent truth table. First decide whether the question asks for
     true/correct options, false/incorrect options, a ranking, or a single best option. Then judge each option A,
     B, C, and D separately from the text and image. Do not force a single letter when multiple options are true.
     For formula or numeric options, compare exact expressions and units rather than surface appearance. If there
     are no choices, solve exactly with symbolic variables and units.

stagewise_cot:
Use a structured multimodal reasoning process: first summarize the exact answer target, then write a visual
    interpretation of the diagram, then do the physics/math reasoning or option audit, then conclude. Treat the
    visual interpretation as a stage that can be revised if the equations conflict with labels or geometry. Keep
    the final answer exact and preserve variable labels from the problem.

domain_playbook_python:
First classify the problem into the dominant physics/visual archetype: mechanics, rotation/oscillation, circuits/
    electromagnetism, optics/waves, thermo/fluids, graph/table interpretation, or conceptual option audit. Then
    use the matching checklist: re-read diagram labels and signs, state the answer target, name the governing law,
     define variables with units, check limiting cases and dimensions, and for A-D questions verify every option
    as a proposition before forming the letter set. Use Python/SymPy as a scratchpad for algebra, exact
    simplification, roots/domains, unit conversion, and formula-option equivalence. If the result conflicts with
    a visible label, revise the physical model before finalizing.
```

Figure 3. Qwen self-consistency and tool-use profile instructions.

```
[System]
You are a strict multimodal physics answer verifier. Solve the problem from the current row's original image and
    text only, then compare the same-row candidate answers. Do not use any information from other dataset rows.
    Do not write analysis before the requested XML tags.

[User]
The base answer is from the strongest current pipeline. Candidate answers come from independent same-row attempts.
     Your task is to decide whether the base answer has a concrete error.

Rules:
- First solve the problem from the image/text yourself.
- For A-D or multi-select questions, audit every option independently.
- Candidate frequency is weak evidence; use it only after the physics check.
- Keep the base answer unless you can identify a concrete error in visual reading, target format, option truth
    status, law choice, algebra, arithmetic, sign, units, or requested rounding.
- If the candidates are all unsupported or you are uncertain, keep the base.
```

```
Original problem text:
{problem_prompt}

Base answer:
Answer: {base_answer}
Reasoning excerpt: {base_reasoning}

Same-row candidate answers:
{candidate_block}

Output exactly:
<decision>keep_base|replace</decision>
<answer>final answer only</answer>
<confidence>high|medium|low</confidence>
<base_error>concrete error if replacing, otherwise none</base_error>
<reasoning>concise verification under 120 words</reasoning>
```

Figure 4. Support-based selector prompt used to propose replacements. Deterministic gates decide whether a proposed replacement is accepted.

```
[System]
You are an expert multimodal physics solver and answer selector. You will receive one physics problem and several
     independent test-time candidate answers for that same problem. Solve the problem yourself from the original
     text and image, then choose the candidate that matches your verified solution. If every candidate is wrong,
     provide your verified answer.

[User]
Audit a proposed correction for this multimodal physics problem. The base answer comes from the strongest general
     pipeline. Keep the base answer unless the original text/image and physics clearly show a concrete base error
     and a challenger or corrected answer is better.

Required procedure:
1. Re-read the original image for labels, option wording, signs, axes, geometry, units, and whether the question
     is single-select, multi-select, numeric, or symbolic.
2. Check the base answer first. If it is consistent with the image and physics, output the base answer even if a
     challenger is plausible.
3. Change the answer only when you can identify the exact base mistake and verify the alternative from the problem.
      For A-D choices, audit each option independently before adding or removing letters.
4. Never remove or add option letters merely because a problem might be single-choice, because a shorter answer is
      more common, or because a candidate has more votes. If the base answer is a multi-letter set and a
     challenger is a subset or superset, explicitly prove every changed letter true or false from the original
     image/text and physics. If the target format is ambiguous, keep the base answer.
5. If all listed answers are wrong but the corrected answer is clear, output the corrected answer. If uncertain,
     keep the base answer.

Original problem text:
{problem_prompt}

Base answer:
Source: {base_source}
Answer: {base_answer}
Reasoning excerpt: {base_reasoning}

Challenger answers:
{challenger_block}

Output exactly <answer>...</answer> and <reasoning>...</reasoning>. Put only the final answer inside <answer>.
     Keep reasoning under 100 words.
```

Figure 5. Challenge-verification prompt for low-consensus candidates.

```
[Option extraction system]
You are a strict multiple-choice extraction specialist for multimodal physics problems. Extract the answer target
     and options without solving.

[Option extraction user]
Extract the answer target and answer choices from the problem text and image. Do not solve or judge the options.
     Preserve handwritten text and formulas.

Return exactly these tags:
<target>what the question asks, including whether it asks correct, incorrect, true, false, single-best, multiple
     statements, ranking, numeric, or symbolic</target>
<option_A>full text of option A, or none</option_A>
<option_B>full text of option B, or none</option_B>
<option_C>full text of option C, or none</option_C>
<option_D>full text of option D, or none</option_D>
<non_choice>yes if there are no A-D options, otherwise no</non_choice>
```

```
[Per-option verifier]
Decide whether this option letter should be included in the final answer. This is not always the same as judging
     whether the statement is true: if the question asks for false/incorrect statements, include an option when
     that option is false/incorrect. If the question asks for correct/true statements, include an option when it
     is correct/true. If the question asks for a single best answer, include only if this option is the unique
     best answer.

Required checks:
- Re-read the original image for labels, directions, signs, axes, geometry, circuit topology, graphs, and units.
- Use the extracted option text as a guide, but the original image/text is authoritative.
- Do not decide by candidate frequency or by assuming only one answer.

Original problem text:
{problem_prompt}

Extracted target and options:
{extraction}

Option to verify: {letter}. {option_text}

Output exactly:
<verdict>include|exclude|uncertain</verdict>
<reasoning>one concise option-specific check</reasoning>

[Choice repair]
Audit the A-D choices for this physics problem. The problem may ask for correct statements, incorrect statements,
     true statements, false statements, a single best answer, or a multi-select answer. First determine what kind
     of answer is requested, then decide independently whether each option letter should be included in the final
     answer.

Output exactly:
<target>single_best|multi_correct|multi_incorrect|ranking|other</target>
<A>include|exclude|uncertain</A>
<B>include|exclude|uncertain</B>
<C>include|exclude|uncertain</C>
<D>include|exclude|uncertain</D>
<confidence>high|medium|low</confidence>
<answer>sorted final option letters only, or keep_current if not an A-D choice question</answer>
<reasoning>concise option-specific check under 160 words</reasoning>
```

Figure 6. Multiple-choice audit and repair prompts.

```
[PaddleOCR-VL task prompt]
Create a structured perception sheet for solving this physics problem. Do not solve it. Extract and organize only
     visible information from the image: (1) full problem text and answer choices, preserving option letters A-D,
     symbols, signs, subscripts, superscripts, units, and inequalities; (2) all diagram objects, point labels,
     arrows, axes, graph ticks/endpoints, circuit nodes/branches, component values, angles, distances, directions,
      polarities, and coordinate conventions; (3) topology and spatial relationships, such as which points are
     connected, which surfaces touch, force/field directions, ray paths, and graph shape; (4) uncertain or
     illegible readings with alternatives. Use concise markdown headings: Problem Text, Diagram Facts,
     Relationships, Ambiguities.

[OCR-sheet solver system]
You are an expert text-only physics solver. A specialist visual model has already read the image and produced OCR/
     perception notes. Solve the current problem from the provided problem text and same-row perception notes only.
      Do not use other dataset rows or reference answers.

[OCR-sheet solver user]
Independent OCR-sheet solve #{sample_index}: {profile_name}
Focus: {profile_instruction}

{optional_python_tool_instruction}

Original problem text field:
{problem_prompt}

Structured OCR/perception sheet from the same image:
{ocr_notes}

The perception sheet may contain OCR mistakes. Resolve conflicts using physics consistency and the provided
     problem text. For choice questions, do not assume exactly one option; include every letter required by the
     wording.

Final output must contain exactly these two tags:
<reasoning>...</reasoning>
<answer>...</answer>

[OCR-sheet profiles]
target_python: Identify the requested answer format first. Then set up the equations and call Python/SymPy for
```

```
    algebra, arithmetic, roots, units, or option comparison before finalizing.
option_truth: If A-D options are present, determine whether the question asks for correct or incorrect statements
    and judge each option independently. Include all and only the required letters.
symbolic_exact: Keep the derivation symbolic as long as possible. Compare expressions exactly, preserve signs and
    subscripts, and use dimensions and limiting cases to reject implausible answers.
diagram_facts: Focus on diagram facts from the perception sheet: geometry, labels, axes, directions, circuit
    topology, graph endpoints, and small signs. Solve only after reconciling these facts with the text.
sanity_check: Solve normally, then audit the result with units, signs, monotonicity, boundary cases, and whether
    the final answer format matches the target.
```

Figure 7. PaddleOCR-VL perception prompt and OCR-sheet solver prompt.

```
[System]
You are solving SeePhys Pro multimodal physics problems. Use the text and image as authoritative evidence. Work
    privately and return only <reasoning>...</reasoning> and <answer>...</answer>. The reasoning should be
    concise and the answer tag must contain the final answer only.

[Single-pass solve]
Independent single-shot solve. Use the following solving style:
First identify whether the requested answer is option letter(s), a number, or a symbolic expression. For choices,
    verify every option A-D and do not drop true options in multi-select problems. For non-choice answers, keep
    variables symbolic, preserve units, avoid rounded decimals, and never replace g with a number unless the
    problem states one.

For multiple-choice questions, some questions may have multiple correct answers. Verify each option A-D
    independently and output all correct letters sorted with no separators, for example A, BD, or ACD.

Problem:
{problem_prompt}

Use private reasoning for the detailed work. In the visible response, give only a concise verification and the
    final answer.

Final output must contain exactly these two tags:
<reasoning>...</reasoning>
<answer>...</answer>

[Choice audit repair]
Strict A-D option audit. The current pipeline answer is provided only as a hypothesis; it may be wrong.

Output the final answer first so it cannot be truncated:
<answer>sorted option letters only, e.g. A, BD, or ACD; use NONE only if this is not an A-D option question</
    answer>
<confidence>high|medium|low</confidence>
<A>include|exclude|uncertain</A>
<B>include|exclude|uncertain</B>
<C>include|exclude|uncertain</C>
<D>include|exclude|uncertain</D>
<reasoning>brief option-specific check under 120 words</reasoning>

Rules:
- Do not assume there is exactly one correct option.
- If the problem asks for incorrect/false statements, include the incorrect/false options.
- If it is single-best, include only the unique best option.
- Verify labels, directions, graph axes, circuit topology, signs, and geometry from the image before judging
    options.
- If an option is genuinely uncertain, mark it uncertain and do not include it.

Current pipeline answer: {current_answer}

Problem:
{problem_prompt}
```

Figure 8. GPT-5.5 single-pass solve and choice-audit prompts.

B. Wide-panel Crop

In this section, we provide an example for the wide-panel crop module. Please refer to Figure 3 for more details.

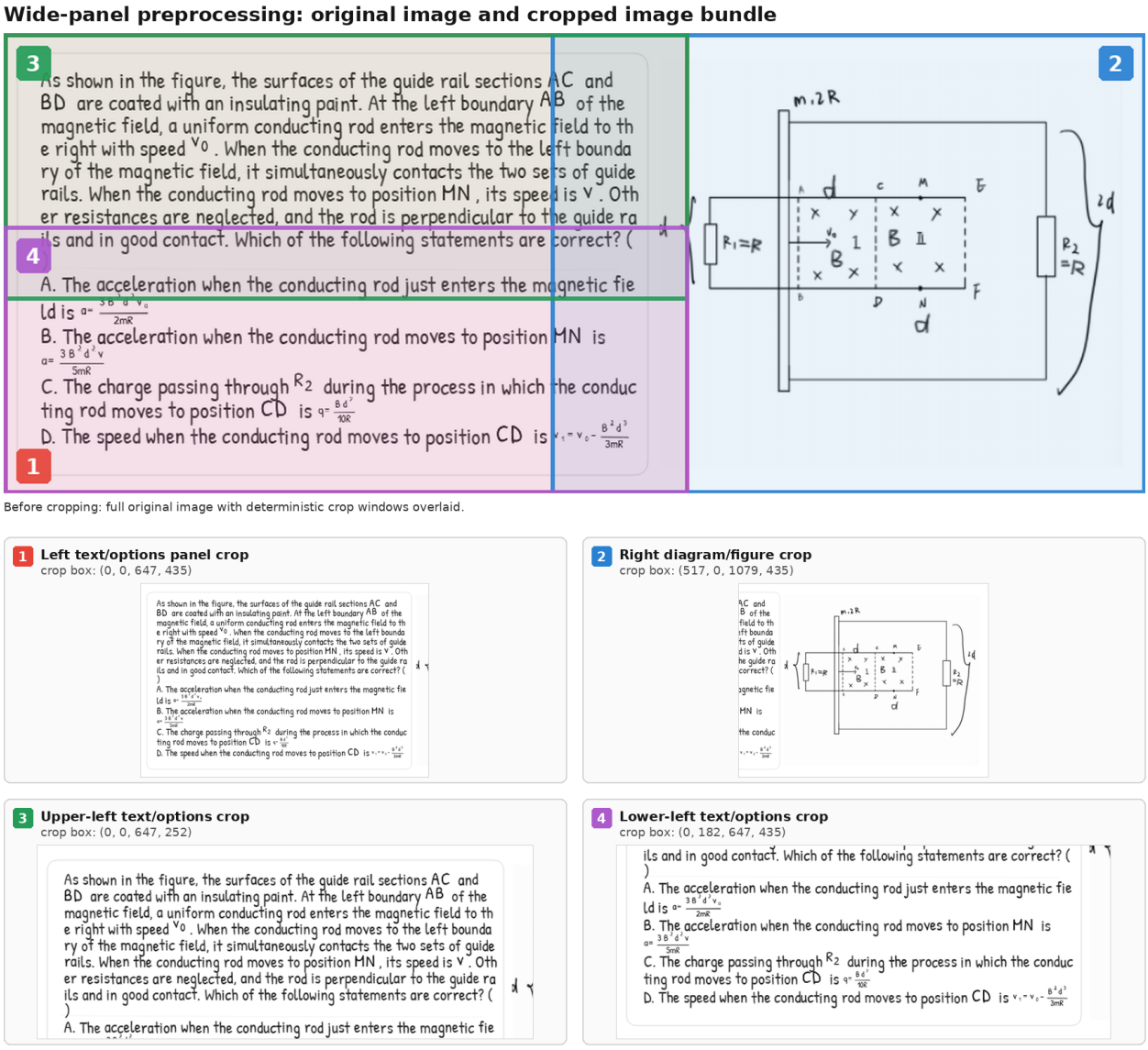

Figure 3. Concrete wide-panel crop example from `level4_testmini_000000`. The top panel shows the original image with deterministic crop windows overlaid. The lower panels show the first four local crops that are sent after the full image when the image budget is five inputs.