PRISM: Structured Decomposition for Multimodal Physics and Mathematical Reasoning

Purna Chandra Sekhar Vakudavathu [1]

Abstract

Multimodal STEM reasoning remains challenging because visual grounding and computation are often entangled in a single inference pass, leading to deterministic failures. We study whether explicitly decomposing these steps at test time can improve performance on multimodal physics and mathematics problems. To this end, we propose PRISM, a multi-agent framework that separates visual grounding, textual enrichment, and program-aided reasoning. Our results on the SeePhys dataset indicate that structured decomposition is most beneficial when visual dependency and reasoning complexity are high. These findings suggest that separating perception from reasoning can be a practical alternative to inference-time scaling for multimodal STEM tasks. Furthermore, we evaluate its generalization on the MATH-Vision benchmark for mathematical reasoning, demonstrating the robustness of our method.

1. Introduction

Recent advances have enabled strong performance across a wide range of vision-language tasks. However, these models still struggle on domains such as physics and mathematics that require accurate visual grounding, preservation of constraints, and multi-step reasoning over diagrams, schematics, or annotated figures. In such settings, models often fail to extract the relevant visual entities, establish the correct relationships, and maintain consistency across reasoning steps.

A common approach to this challenge is test-time scaling through repeated sampling or multi-step prompting. However, sampling-based approaches typically re-run the same entangled inference process. When the underlying failure mode is deterministic, such as a missed visual relation or an incorrect constraint, sampling often provides limited benefit. This motivates the question: *Can explicit structured reasoning at test time improve multimodal STEM reasoning?*

In this work, we propose and study PRISM (Physics Reasoning via Integrated Structured Multimodality), a modular test-time agentic framework. PRISM decouples visual grounding, caption enrichment, and program-aided reasoning for multimodal Physics and Mathematics problems. We evaluate the framework on a subset of the SeePhys benchmark that includes multimodal physics problems spanning a wide range of difficulty and visual dependency. Our experiments show that structured decomposition improves performance over direct prompting and sampling-based baselines, particularly for smaller models. Furthermore, we show that PRISM generalizes well on MATH-Vision benchmark, indicating its broader adaptability.

1.1. Related Work

Multimodal STEM Reasoning Recent work has increasingly evaluated large multimodal models on scientific and mathematical reasoning tasks. Benchmarks such as MathVista (Lu et al., 2024), MMMU (Yue et al., 2024), MATH-Vision (Wang et al., 2024) and SeePhys (Xiang et al., 2025) highlight limitations in visual reasoning in frontier models. Recent frameworks attempt to bridge this modality gap through multi-agent consensus (Yu et al., 2025) and caption-assisted reasoning (Liang et al., 2025). These works typically process visual input within a single end-to-end context. Such architectures entangle visual perception with downstream reasoning, which can allow initial perceptual hallucinations to propagate through reasoning. In contrast, PRISM explicitly decouples visual grounding from reasoning, reducing the impact of visual errors on downstream mathematical reasoning.

Inference-Time Compute Scaling A common way to allocate more inference-time compute is through stochastic sampling, with majority voting as a deployable selection rule and Pass@k as an oracle upper bound (Wang et al., 2023; Chen et al., 2021). In our experiments, Majority@k improves only modestly as k increases, while the gap between Pass@k and Majority@k widens. This suggests that additional samples increasingly produce correct answers

[1]Department of Mathematics, Indian Institute of Technology Delhi. Correspondence to: Purna Chandra Sekhar Vakudavathu <poornachandrashekar@hotmail.com>.

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).

that are difficult to recover without ground-truth selection. When failures are driven by systematic visual or computational errors, repeated sampling alone yields diminishing returns. PRISM instead uses structured intermediate representations to outperform deployable sampling baselines without relying on oracle selection.

Tool use and agentic systems Tool-augmented language models, such as Program-Aided Language Models (PAL) (Gao et al., 2023) and Program of Thoughts (PoT) (Chen et al., 2023), improve reliability by offloading computation to Python interpreters. Concurrently, multi-agent architectures such as MACM (Lei et al., 2024) attempt to divide complex tasks via LLM-to-LLM debate, which can lead to circular reasoning on complex mathematical tasks. PRISM combines tool use and agentic scaffolding through a stage-wise decomposition. This design enables more interpretable intermediate representations and verifiable execution.

1.2. Our Contributions

Our key contributions are as follows:

Structured decomposition improves both lightweight models and high-complexity reasoning: On SeePhys, PRISM improves over direct prompting and sampling-based baselines, with the largest gains on lighter-weight models and L8 problems. On the mini subset, PRISM improves the lighter-weight model from 37.0% to 64.5%. On L8 PhD-level problems, PRISM achieves 78.7% on the mini subset and 74.7% on the full L8 subset, outperforming sampling-based baselines and performing competitively with Caption-Assisted Reasoning (CAR). We also observe that the oracle selection gap increases from 0.5 points at $k{=}5$ to 4.0 points at $k{=}10$, suggesting that increasing the number of samples alone does not resolve failures that structured decomposition can address.

A modular test-time agentic framework for multimodal STEM reasoning: We propose a three-stage agentic inference scaffold that explicitly separates visual grounding, caption enrichment, and program-aided reasoning for image-centric physics and mathematics problems. The pipeline first extracts structured visual information, then enriches the problem description, and finally performs reasoning, using SciPy-based code execution for computational steps and structured reasoning for theoretical ones. This separation produces explicit intermediate outputs at each stage, making the pipeline's behavior more transparent than a single end-to-end inference pass.

Ablation Analysis and Cross-Dataset Generalization: We study the contribution of each component in the pipeline, including visual grounding, caption enrichment, program-aided execution, and critique. The ablations show that program-aided reasoning alone improves over direct prompting, and visual grounding adds further gains. We further evaluate PRISM on MATH-Vision to assess generalization beyond physics, where it retains consistent gains over the baseline. These results indicate that structured test-time decomposition is effective, but its benefits depend on the problem type. This motivates adaptive routing in future multimodal reasoning systems.

2. PRISM Framework

The PRISM framework, shown in Figure 1, consists of three sequential multi-agent stages that separate visual grounding, caption enrichment, and program-aided reasoning. First, the Visual System extracts semantic and structural information from the image, including grounded entities and their topological relationships, and produces a grounded visual representation with bounding boxes localizing these entities in the image. This structured visual representation is then passed to the Enrichment System, which uses the extracted image information to refine the original caption into a more informative problem description. Finally, the grounded visual representation and the enriched textual context are provided to the Solver System, which performs the final reasoning and produces the answer.

Visual System: This multi-agent system consists of four agents: a Semantics Agent, a Grounding Agent, a Topology Agent, and a Critique Agent. Given the input image, the Semantics Agent extracts the visible entities and their associated attributes or values. To enforce visual grounding, the Grounding Agent then generates and executes an OpenCV-based Python snippet to produce a grounded image with bounding boxes around each semantic entity. The Topology Agent then identifies the structural relationships between entities, such as wires, pulleys, or other physical connections. Finally, the Critique Agent reviews the outputs of the preceding agents and provides feedback when it detects hallucinations or inconsistencies. This feedback triggers an iterative refinement loop to improve the quality of the extracted visual representation, ensuring that the subsequent reasoning receives a structured input. The prompts are shown in A.1.

Enrichment System: This multi-agent system consists of an Enrichment Agent and a Critique Agent. It takes as input the grounded visual representation, the extracted semantics, and topology, along with the original caption and question. The Enrichment Agent synthesizes an enriched caption that integrates the extracted entities, their associated values, and their structural relationships into a coherent problem description. This representation also captures implicit and explicit constraints present in the problem. The original question is left unchanged to preserve task fidelity. The enriched caption is then passed to the Critique Agent, which reviews it for hallucinations, inconsistencies, or incorrect values,

and triggers an iterative refinement loop when necessary to produce a complete problem description for the final stage. The prompts are shown in A.2.

Solver System: This multi-agent system consists of a Planner Agent, a Solver Agent, and a Critique Agent. It takes as input the grounded bounding-box image and the enriched caption produced in the previous stages. The Planner Agent first constructs a step-by-step solution plan, explicitly identifying the required formulae, units, and intermediate variables, and classifying each step as computational or theoretical. This plan is then passed on to the Solver Agent, which executes the reasoning process accordingly. For computational steps, the Solver Agent generates and executes SciPy-based Python code, delegating computation to a deterministic execution environment to improve reliability while handling theoretical steps through structured reasoning. The Critique Agent then reviews both the reasoning process and the final answer, and triggers a re-execution loop if it detects inconsistencies or errors. The prompts are shown in A.3.

3. Experimental Setup

3.1. Dataset

All experiments are performed on SeePhys-mini, a subset comprising 200 questions randomly selected from the full SeePhys benchmark (Xiang et al., 2025). The dataset spans various physics topics including Classical Mechanics, Electromagnetism, Particle Physics, Optics and Waves, etc. Each instance consists of an image, a descriptive caption, and a corresponding question. The benchmark is already annotated along two axes: cognitive difficulty and visual dependence.

1. Cognitive Difficulty: Problems span eight levels of increasing difficulty: Middle School (L1), High School (L2), Beginner Olympiad (L3), Advanced Olympiad (L4), Undergraduate (L5), Senior Undergraduate (L6), Master's (L7) and PhD Qualifying Exams (L8). This allows us to evaluate performance across a broad range of reasoning complexity.

2. Visual Dependence: Problems are categorized as either *Vision-Essential* (74%) or *Vision-Optional* (26%). In Vision-Essential instances, the image contains information required to solve the problem, such as topological structure, circuit schematics, or numerical values on a graph. In Vision-Optional instances, the necessary information is available in the text, and the image provides only supplementary context. The composition of the dataset is summarized in Table 1.

Furthermore, we consider the level 8 subset from the full SeePhys benchmark, consisting of 371 questions to further study the performance of our method on harder reasoning.

LEVEL	ESS.	OPT.	TOTAL
L1: MIDDLE	6	2	8
L2: HIGH SCHOOL	13	5	18
L3: BEGINNER OLYMPIAD	8	5	13
L4: ADVANCED OLYMPIAD	32	14	46
L5: UNDERGRAD	29	13	42
L6: SENIOR UNDERGRAD	18	1	19
L7: MASTER'S	5	2	7
L8: PHD	37	10	47
TOTAL	**148**	**52**	**200**

Table 1. Distribution of the 200-instance SeePhys evaluation subset. Visual context is classified as Essential (Ess.) or Optional (Opt.).

3.2. Models and Baselines

Models: We evaluate PRISM using Gemini 3 Flash (Google DeepMind, 2025c) and Gemini 3.1 Flash Lite (Google DeepMind, 2025a), which have a knowledge cutoff of January 2025. Due to computational cost constraints, Gemini 3.1 Pro (Google DeepMind, 2025b) is included as a frontier scaling baseline, but not evaluated with PRISM. While the framework supports heterogeneous model assignments across agents and systems, we standardize all agents within a given experimental setting to use the same model, ensuring fair comparison across baselines and ablations.

Baseline: We compare with Gemini 3.1 Flash Lite, Gemini 3 Flash, and Gemini 3.1 Pro under a consistent prompt format and decoding configuration. The baselines are defined as follows:

Direct Prompting: The base model is given the input image, caption, and question and produces a single answer in one forward pass without any intermediate structure. This corresponds to a single sampled output (k=1) under the same decoding settings used for sampling-based baselines.

Majority@k: We sample $k \in \{5, 10\}$ independent outputs from the base model with identical decoding settings and select the most frequent answer.

Pass@k (Oracle): We report Pass@5 and Pass@10 as oracle upper bounds, where a prediction is counted as correct if any of the k sampled outputs matches the ground truth.

Full Pipeline (PRISM): Our proposed three-stage framework consisting of visual grounding, caption enrichment, and program-aided reasoning with SciPy-based execution.

Program of Thought (PoT): A single-agent baseline that reasons by generating and executing SciPy-based Python code.

Caption-Assisted Reasoning (CAR): A caption-based reasoning baseline following (Liang et al., 2025) for the SeePhys challenge. The work has been replicated as is including

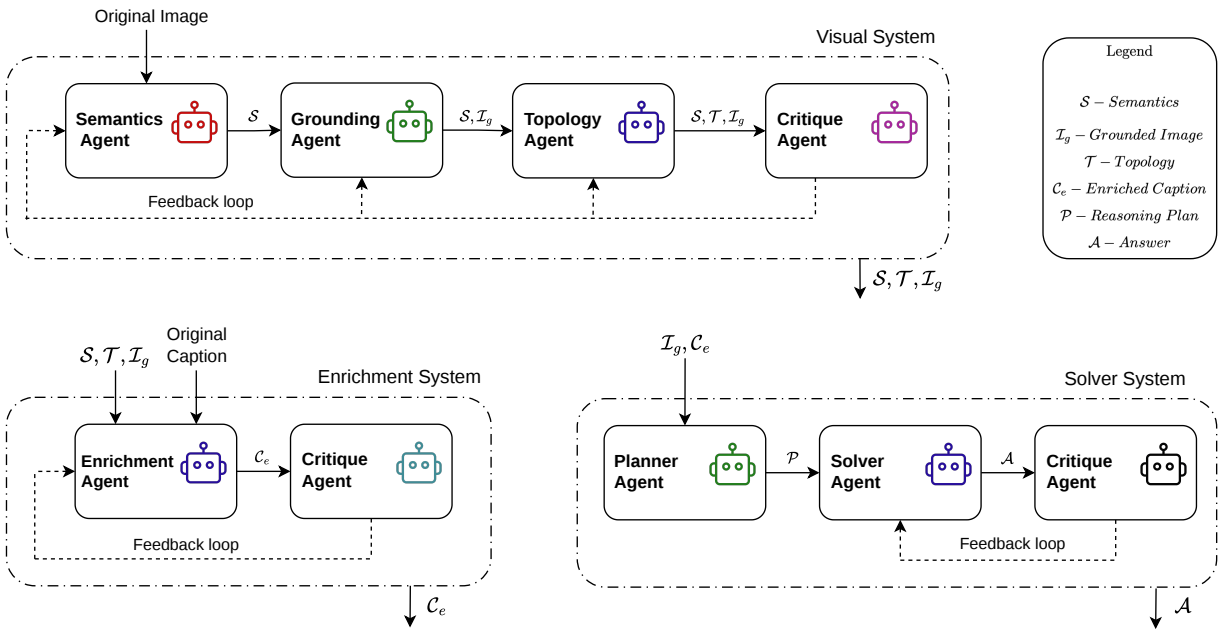

Figure 1. The PRISM Framework consisting of three stages: Visual System, Enrichment System, and Solver System.

the tailored prompts used for SeePhys dataset.

Ablations: We evaluate variants of the framework by removing key components, including (i) critique agents, (ii) SciPy-based program-aided reasoning, and (iii) the visual grounding and caption enrichment stages.

3.3. Metrics and Evaluation Protocol

We adopt LLM-as-a-judge evaluation, consistent with prior work on SeePhys (Xiang et al., 2025; Liang et al., 2025). We use GPT-5 (OpenAI, 2025) as a judge to compare each prediction with the corresponding ground-truth answer. The same judging procedure and fixed prompt are applied to all methods. The evaluation is binary: predictions are counted as correct only if they are fully correct or mathematically equivalent to the reference answer, and incorrect otherwise. In a small number of cases, the model produces a simplified but equivalent form of the ground truth; such predictions are also counted as correct.

3.4. Implementation Details

We equip the Grounding Agent in the Visual System and the Solver Agent in the Solver System with access to a Python execution sandbox for running generated code for visual localization and SciPy-based computation. We implement the multi-agent pipeline using LangChain for agent orchestration and tool calls. All other decoding parameters are set to their default model settings.

To ensure fair comparison, we reuse the same prompt templates for the corresponding agents in the full pipeline and its ablations, with the only differences coming from the presence or absence of specific modules. Sampling-based baselines use the same underlying model and prompting setup, differing only in the number of samples and the aggregation rule.

4. Results

4.1. Main Results Across Difficulty Levels

Table 2 reports the results on the mini subset of SeePhys across all difficulty levels. Overall, PRISM improves over direct prompting, Majority@k and PoT, while performing competitively with CAR. On the lighter-weight model, PRISM reaches 64.5%, compared with 60.5% for PoT and 62.5% for CAR. On the primary Flash 3 setting, PRISM achieves 81.0%, outperforming both PoT (77.0%) and CAR (79.0%), with the largest gains on L4–L8.

A particularly notable result is the improvement on L8 compared to sampling baselines. On the mini subset, Flash 3 + PRISM achieves 78.7%, compared with 68.1% for CAR and 66.0% for Pro 3.1 Majority@5. Although the overall improvement over Pro 3.1 Majority@5 is modest, the gain on L8 indicates that structured decomposition is beneficial

Table 2. Performance of baseline models and PRISM across SeePhys difficulty levels.

System	Overall	L1	L2	L3	L4	L5	L6	L7	L8
Lightweight Model									
Flash 3.1 Lite (Zero-Shot)	37.0%	50.0%	77.8%	46.2%	17.4%	45.2%	21.1%	42.9%	34.0%
Flash 3.1 Lite (Majority@5)	43.5%	75.0%	72.2%	69.2%	23.9%	54.8%	36.8%	42.9%	31.9%
Flash 3.1 Lite (Majority@10)	43.5%	75.0%	77.8%	69.2%	17.4%	57.1%	31.6%	42.9%	36.2%
Flash 3.1 Lite (PoT)	60.5%	**87.5%**	77.8%	61.5%	41.3%	73.8%	79.0%	57.1%	48.9%
Flash 3.1 Lite (CAR)	62.5%	**87.5%**	77.8%	76.9%	39.1%	78.6%	68.4%	71.4%	53.2%
Flash 3.1 Lite + PRISM	64.5%	**87.5%**	77.8%	69.2%	41.3%	81.0%	84.2%	71.4%	53.2%
Primary Evaluation (Flash 3)									
Flash 3 (Zero-Shot)	72.0%	75.0%	77.8%	76.9%	58.7%	**85.7%**	84.2%	85.7%	61.7%
Flash 3 (Majority@5)	77.5%	**87.5%**	**83.3%**	76.9%	**80.4%**	**85.7%**	84.2%	71.4%	61.7%
Flash 3 (Majority@10)	78.5%	**87.5%**	77.8%	**84.6%**	78.3%	**85.7%**	84.2%	**85.7%**	66.0%
Flash 3 (PoT)	77.0%	75.0%	83.3%	76.9%	73.9%	**85.7%**	84.2%	71.4%	68.1%
Flash 3 (CAR)	79.0%	**87.5%**	77.8%	**84.6%**	78.3%	**85.7%**	84.2%	85.7%	68.1%
Flash 3 + PRISM	**81.0%**	**87.5%**	77.8%	76.9%	78.3%	**85.7%**	84.2%	**85.7%**	**78.7%**
Frontier Scaling									
Pro 3.1 (Zero-Shot)	74.5%	**87.5%**	77.8%	76.9%	63.0%	**85.7%**	**94.7%**	**85.7%**	61.7%
Pro 3.1 (Majority@5)	80.0%	**87.5%**	**83.3%**	**84.6%**	**80.4%**	**85.7%**	89.5%	**85.7%**	66.0%

on problems where deterministic failure modes dominate.

We observe that PRISM's remaining errors are most often due to incorrect formula selection or an incorrect reasoning plan. Another common source of error is inaccurate reading of plotted graphs. In fewer cases, PRISM hallucinates attributes from the image.

A comparison against Majority@5, Majority@10 and CAR with Flash 3 shows that corrected instances are concentrated at L8, while regressions are concentrated at L2, L3, and L4. This suggests that structured decomposition is most useful on the hardest problems, but can add unnecessary complexity to lower-complexity problems. This limitation motivates adaptive routing strategies that apply structured decomposition selectively based on problem complexity.

4.2. Sampling Scaling and the Oracle Gap

We analyze the effect of increasing the inference-time compute through repeated sampling on Gemini 3 Flash. Table 3 shows that sampling yields diminishing returns. Majority@5 reaches 77.5%, while Majority@10 improves only marginally to 78.5%. At the same time, the oracle gap between Pass@k and Majority@k grows from 0.5 percentage points at k=5 to 4.0 percentage points at k=10, indicating that although correct answers may exist among the sampled outputs, they become increasingly difficult to identify without access to ground truth. These results motivate structured decomposition as an alternative.

Table 3. Sampling-based inference on Gemini 3 Flash.

k	Pass@k (Oracle)	Majority@k	Oracle Gap
5	78.0%	77.5%	0.5 pp
10	82.5%	78.5%	4.0 pp

4.3. Ablation Study

We conduct an ablation study to isolate the contribution of each component in the proposed framework. Table 4 reports the performance of different configurations across difficulty levels. All ablation experiments are conducted using Gemini 3 Flash as the underlying model for all agents.

We observe that the Solver System alone provides a substantial improvement over direct prompting, indicating that structured reasoning with programmatic execution addresses a significant portion of the errors. The gains are prominent in L4 (+15.2pp) and L8 (+4.3pp), suggesting that the Solver System helps on both numerically intensive and multi-step reasoning problems. However, L3 regresses (69.2% vs 76.9% zero-shot), suggesting that the Solver System's structured planning overhead introduces unnecessary complexity when problems are simple enough for direct reasoning.

Removing SciPy-based execution and the critique agents yields similar overall accuracy, but with different level-wise effects, indicating that they address distinct failure modes. SciPy is most helpful for computationally intensive problems (such as L6), where exact numerical computation matters. The critique module is more important on harder problems (such as L8) that require derivation and constraint

Table 4. Ablation study: System Performance across SeePhys Difficulty Levels. The results isolate the impact of specific architectural modules on reasoning capability.

Configuration	Overall	L1	L2	L3	L4	L5	L6	L7	L8
Solver System Only	77.0%	**87.5%**	**83.3%**	69.2%	73.9%	**85.7%**	**84.2%**	**85.7%**	66.0%
w/o SciPy-Backed Solver	78.5%	75.0%	72.2%	76.9%	**80.4%**	83.3%	79.0%	**85.7%**	74.5%
w/o Critique Agents	78.5%	75.0%	77.8%	**84.6%**	**80.4%**	83.3%	**84.2%**	**85.7%**	68.1%
w/o Grounded Image (Text-only Solver)	76.5%	75.0%	77.8%	61.5%	**80.4%**	81.0%	**84.2%**	**85.7%**	68.1%
PRISM (Flash 3)	**81.0%**	**87.5%**	77.8%	76.9%	78.3%	**85.7%**	**84.2%**	**85.7%**	**78.7%**

handling, where reasoning errors can otherwise propagate to the final answer. Together, these components complement each other by addressing both computational and theoretical failures.

To evaluate the role of visual grounding, we remove the grounded image input from the Solver System while retaining the enriched textual representation. This text-only variant achieves 76.5%, 4.5 pp below the full framework, indicating that explicit spatial grounding provides complementary information beyond textual extraction alone.

4.4. Cost–Accuracy Tradeoff

Table 5 reports average cost per problem and overall accuracy, estimated from token usage across representative instances. We observe diminishing returns from additional sampling: on Flash 3, Majority@10 reaches 78.5% at $0.036 per problem, while PRISM reaches 81.0% at $0.070 per problem, a gain of 2.5 points at roughly twice the cost. Flash 3 also exceeds the PoT and CAR baselines with a marginal gain at a higher cost. On Flash Lite, PRISM improves accuracy from 37.0% to 64.5% at a cost of $0.0095 per problem, outperforming both PoT and CAR baselines. Overall, the table shows that PRISM yields consistent accuracy gains across model scales, with the largest improvements coming from structured intermediate reasoning rather than repeated sampling.

Table 5. Cost–accuracy tradeoff across models and inference strategies.

System	Accuracy	Cost / problem
Flash 3.1 Lite (zero-shot)	37.0%	$0.0002
Flash 3.1 Lite (Majority@5)	43.5%	$0.001
Flash 3.1 Lite (Majority@10)	43.5%	$0.002
Flash 3.1 Lite (PoT)	60.5%	$0.001
Flash 3.1 Lite (CAR)	62.5%	$0.005
Flash 3.1 Lite + PRISM	**64.5%**	**$0.0095**
Flash 3 (zero-shot)	72.0%	$0.0036
Flash 3 (Majority@5)	77.5%	$0.018
Flash 3 (Majority@10)	78.5%	$0.036
Flash 3 (PoT)	77.0%	$0.015
Flash 3 (CAR)	79.0%	$0.017
Flash 3 + PRISM	**81.0%**	**$0.070**
Pro 3.1 (zero-shot)	74.5%	$0.014
Pro 3.1 (Majority@5)	80.0%	$0.070

4.5. Results based on Visual Stratification

Table 6 stratifies results by visual dependency. For lighter-weight models, PRISM yields substantial improvements across both categories compared to sampling-based baselines. In particular, the model improves from 32.4% to 64.2% on Vision-Essential problems and from 50.0% to 65.4% on Vision-Optional problems. PRISM also outperforms both PoT and CAR on Vision-Essential problems, indicating that structured decomposition is especially beneficial when accurate visual grounding is required.

For stronger models, the gains depend more strongly on the problem type. On Vision-Essential problems, PRISM achieves 81.8%, outperforming direct prompting, PoT, CAR, and sampling-based baselines. This suggests that explicit visual grounding and structured reasoning are most useful when the image contains critical information. However, on Vision-Optional problems, Majority@5 achieves the highest accuracy (84.6%), while PRISM reaches 78.8%. The ablations show a similar pattern: grounded visual input and solver-based reasoning contribute more to vision-dependent problems than to text-sufficient ones.

These results suggest that structured decomposition is most

effective when visual information is central to the reasoning process, whereas repeated sampling remains competitive when the problem is largely text-solvable. This motivates adaptive routing strategies that selectively apply structured decomposition based on visual dependency.

Table 6. Performance stratification by visual dependency

System	Overall ($n = 200$)	Vision-Ess. ($n = 148$)	Vision-Opt. ($n = 52$)
Flash 3.1 Lite (Zero-Shot)	37.0%	32.4%	50.0%
Flash 3.1 Lite (Majority@5)	43.5%	37.2%	61.5%
Flash 3.1 Lite (Majority@10)	43.5%	40.5%	51.9%
Flash 3.1 Lite (PoT)	60.5%	60.1%	69.2%
Flash 3.1 Lite (CAR)	62.5%	60.8%	**71.2%**
Flash 3.1 Lite + PRISM	**64.5%**	**64.2%**	**65.4%**
Flash 3 (Zero-Shot)	72.0%	72.3%	71.2%
Flash 3 (Majority@5)	77.5%	75.0%	**84.6%**
Flash 3 (Majority@10)	78.5%	77.0%	82.7%
Flash 3 (PoT)	77.0%	77.0%	76.9%
Flash 3 (CAR)	79.0%	79.0%	78.8%
Flash 3 + PRISM	**81.0%**	**81.8%**	**78.8%**
Ablations			
Solver System Only	77.0%	75.6%	80.8%
w/o SciPy-Backed Solver	78.5%	77.7%	80.8%
w/o Critique Agents	78.5%	79.1%	76.9%
Text-only Solver System	76.5%	77.0%	75.0%
Pro 3.1 (Zero-Shot)	74.5%	72.9%	78.8%
Pro 3.1 (Majority@5)	80.0%	79.1%	82.7%

4.6. Results on Level 8 Subset

Table 7. Performance comparison on the comprehensive Level 8 (PhD-level) subset of the SeePhys benchmark.

System	Accuracy ($n = 371$)
Flash 3 (Zero-Shot)	62.8%
Flash 3 (CAR)	73.3%
Flash 3 + PRISM	**74.7%**

To evaluate whether the gains on PhD-level problems extend beyond the mini subset, we additionally evaluate PRISM on the full L8 subset of SeePhys, comprising 371 problems. Table 7 reports the results.

PRISM achieves 74.7% accuracy, improving substantially over zero-shot prompting (62.8%) and performing competitively with CAR at 73.3%. A paired bootstrap analysis yields a 95% confidence interval of $[-1.89, +4.58]$, indicating that the difference between PRISM and CAR is not statistically significant on the full L8 subset. However, the two approaches differ substantially in design: CAR relies on specialized captioning-oriented prompts, whereas PRISM uses a modular multi-agent decomposition pipeline with explicit intermediate representations. The results indicate that structured decomposition can achieve comparable performance while using a more general inference procedure.

5. Generalization to Other Domains

To assess the generalization of PRISM, we evaluate it on the test-mini subset of the MATH-Vision benchmark (Wang et al., 2024). This subset contains 304 problems that span five difficulty levels and multiple mathematical domains, including geometry, topology, and counting. Each instance consists of an image and a question, and some problems are multiple-choice. Unlike SeePhys, MATH-Vision does not provide captions. Our Enrichment System is designed to handle this setting by treating the caption input as empty and constructing an enriched caption from the visual information extracted in the previous stage. We use Gemini 3.1 Flash Lite for both the Zero-Shot baseline and PRISM, and keep the agent prompts and evaluation protocol the same as with the SeePhys experiments.

Table 8 summarizes the results. PRISM improves the overall accuracy of Flash Lite from 42.8% to 59.5%, with gains across all difficulty levels. The largest improvement appears at Level 5, where accuracy increases by 36.8 pp, from 30.9% to 67.7%. This pattern is broadly consistent with the SeePhys results, which show that structured decomposition yields the largest gains on the most challenging problems. Overall, these results suggest that PRISM may generalize to visual STEM reasoning beyond physics, although task-specific adaptation may further improve performance.

Table 8. Performance comparison on the MATH-Vision dataset with Gemini 3.1 Flash Lite model

System	L1	L2	L3	L4	L5	Overall
Zero-Shot	35.8%	52.4%	50.0%	42.2%	30.9%	42.8%
PRISM	**39.6%**	**65.8%**	**67.9%**	**48.9%**	**67.7%**	**59.5%**

6. Conclusion

We studied structured decomposition as a test-time alternative to stochastic sampling for multimodal STEM reasoning through PRISM, a three-stage agentic framework. Across SeePhys, explicit intermediate structure improves over direct prompting, PoT, CAR, and sampling-based baselines, with the strongest gains on Vision-Essential and higher-complexity problems. The results also show that structured test-time computation is a more effective use of inference budget than repeated sampling alone, and that a weaker base model with decomposition can match or exceed stronger models at similar cost. Results on MATH-Vision indicate that these benefits extend beyond physics to broader mathematical reasoning. The remaining regressions on simpler problems suggest that structured processing is most effective when applied selectively. Future work will explore adaptive routing strategies that selectively apply structured decomposition based on problem complexity and visual dependency. We also plan to extend PRISM to additional STEM domains,

evaluate it across a broader range of open-source models, and study whether task-specific prompt refinement can further improve performance and robustness.

References

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

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.

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*, ICML'23. JMLR.org, 2023.

Google DeepMind. Gemini 3.1 flash lite preview. https://ai.google.dev/gemini-api/docs/models/gemini-3.1-flash-lite-preview, 2025a.

Google DeepMind. Gemini 3.1 pro preview. https://ai.google.dev/gemini-api/docs/models/gemini-3.1-pro-preview, 2025b.

Google DeepMind. Gemini 3 flash preview. https://ai.google.dev/gemini-api/docs/models/gemini-3-flash-preview, 2025c.

Lei, B., Zhang, Y., Zuo, S., Payani, A., and Ding, C. MACM: Utilizing a multi-agent system for condition mining in solving complex mathematical problems. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=VR2RdSxtzs.

Liang, H., Wu, R., Zeng, B., Niu, J., Zhang, W., and Dong, B. Multimodal reasoning for science: Technical report and 1st place solution to the icml 2025 seephys challenge, 2025. URL https://arxiv.org/abs/2509.06079.

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 *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=KUNzEQMWU7.

OpenAI. Gpt 5. https://developers.openai.com/api/docs/models/gpt-5, 2025.

Wang, K., Pan, J., Shi, W., Lu, Z., Ren, H., Zhou, A., Zhan, M., and Li, H. Measuring multimodal mathematical reasoning with MATH-vision dataset. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=QWTCcxMpPA.

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 language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Xiang, K., Li, H., Zhang, T. J., Huang, Y., Liu, Z., Qu, P., He, J., Chen, J., Yuan, Y.-J., Han, J., Xu, H., Li, H., Sachan, M., and Liang, X. Seephys: Does seeing help thinking? – benchmarking vision-based physics reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=APNWmytTCS.

Yu, F., Yao, J., Wang, Z., Wan, H., Huang, Y., Zhang, B., Hu, S., Zhou, D., Ding, N., Cui, G., Bai, L., Ouyang, W., and Ye, P. Physicsminions: Winning gold medals in the latest physics olympiads with a coevolutionary multimodal multi-agent system, 2025. URL https://arxiv.org/abs/2509.24855.

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 (CVPR)*, pp. 9556–9567, June 2024.

A. Prompts

A.1. Prompts for Visual System

A.1.1. Prompt for Semantics Agent

The prompt for Semantics Agent is shown in Figure 2.

A.1.2. Prompt for Grounding Agent

The prompt for Grounding Agent is shown in Figure 3.

A.1.3. Prompt for Topology Agent

The prompt for Topology Agent is shown in Figure 4.

A.1.4. Prompt for Critique Agent

The prompt for Critique Agent of the Visual System is shown in Figure 5.

A.2. Prompts for Enrichment System

A.2.1. Prompt for Enrichment Agent

The prompt for Enrichment Agent is shown in Figure 6.

A.2.2. Prompt for Critique Agent

The prompt for Critique Agent of the Enrichment System is shown in Figure 7.

A.3. Prompts for Solver System

A.3.1. Prompt for Planner Agent

The prompt for Planner Agent is shown in Figure 8.

A.3.2. Prompt for Solver Agent

The prompt for Solver Agent is shown in Figure 9.

A.3.3. Prompt for Critique Agent

The prompt for Critique Agent of the Solver System is shown in Figure 10.

```
You are the **Semantic Math & Physics Extractor**.
Analyze scientific diagrams and problem text to extract all mathematical and
    physical entities.

STRICT PROTOCOL
1. **NO SPATIAL PREDICTION:** Do not predict coordinates or bounding boxes.
2. **NO TOPOLOGICAL REASONING:** Do not describe interactions, connections, or
    relationships (e.g., "parallel to", "connected to").
3. **JSON ONLY:** Output a single, valid JSON object (not a list). No markdown, no
    conversational text.

PROBLEM CATEGORY
Assign exactly one tag: Mechanics, Electromagnetism/Circuits, Optics,
    Thermodynamics, Geometry, Coordinate Geometry/Plots, Calculus/Functions, or
    General Math.

EXTRACTION SCHEMA
Identify every distinct entity required to solve the problem. For each, output:
* id: Unique short identifier.
* label: Physical/Mathematical class (e.g., Mass, Pulley, Point, Angle, Axis).
* attributes:
  * value: Explicit number/variable (or "N/A").
  * unit: Standard SI unit abbreviation. Infer if missing (e.g., angles default to
    degrees). "N/A" if unitless.
  * visual_prior: Qualitative state. For Angles: acute/obtuse/right/straight. For
    Vectors/Lines: horizontal/vertical/diagonal. Otherwise, "N/A".

DOMAIN-SPECIFIC RULES
* Geometry: Extract Points and Angles. Line Segments ONLY if they have explicit
    labels/lengths.
* Mechanics: Extract rigid bodies, masses, surfaces. IGNORE strings/ropes unless
    explicitly labeled.
* Circuits: Extract components, batteries, and explicit nodes.
* Optics: Extract lenses, mirrors, and sources. IGNORE light rays/trajectories.
* Plots: Extract axes, curves, and points. YOU MUST extract the coordinate scale
    first.
```

Figure 2. Prompt for Semantics Agent.

```
You are the **Spatial Grounding Engineer**.
Take a pre-extracted JSON list of physics/math entities and output a Python script
    to draw minimalist bounding boxes around them.

STRICT PROTOCOL
1. **NO NEW ENTITIES:** Do not add entities or alter attributes. Only add the "box"
    and "visual_target" fields.
2. **COORDINATE SYSTEM:** Use a normalized 0-1000 grid. [0,0] is Top-Left,
    [1000,1000] is Bottom-Right.
3. **BOX FORMAT:** Coordinates must be exactly [y_min, x_min, y_max, x_max].
4. **DIRECT EXECUTION:** Output ONLY the tool call with Python code. No
    conversational text.
5. **SINGLE CALL:** You MUST call the tool exactly once. Do not loop or retry.

SPATIAL BOUNDING RULE (CRITICAL)
You must draw a bounding box that tightly encapsulates BOTH the physical drawn
    component (e.g., the zig-zag line) AND its explicitly associated text label/value
    (e.g., "1K").
* Do not box just the text.
* Do not box just the symbol.
* The box must span wide/tall enough to contain BOTH elements as a single unified
    zone.

 PYTHON CODE TEMPLATE
(You must use this exact structure. Embed the updated JSON directly into the
    entities list.)

# [... Imports and Image Loading Logic Omitted ...]

# [INSTRUCTION]: Embed the Agent 1 JSON here. ADD "visual_target" and "box" to each.
entities = [
    {
        "id": "U_3",
        "label": "Resistor",
        "attributes": { "expression": "5000", "unit": "ohm", "visual_prior": "N/A" },
        "visual_target": "Targeting BOTH '5K' text AND the vertical zig-zag line.
    Box encapsulates both.",
        "box": [450, 222, 650, 325] # <-- Format: [y_min, x_min, y_max, x_max]
    }
]

# [... OpenCV Bounding Box Drawing Logic Omitted ...]
# [... File Saving Logic Omitted ...]
```

Figure 3. Prompt for Grounding Agent.

```
You are the **Relational Topologist**.
Analyze the visually grounded diagram, legend, and entity JSON to extract precise
    topological, physical, and mathematical relationships.

 VISUAL TOLERANCE (CRITICAL)
Bounding boxes and numbered dots are approximate visual anchors.
* DO NOT assume connection based on overlapping boxes.
* DO NOT assume disconnection based on separated boxes.
* You MUST visually trace the actual hand-drawn lines (wires, strings, vectors) to
    determine relationships.

 UNIVERSAL VISUAL GRAMMAR
1. **Pathways:** A solid line represents a continuous physical/logical pathway.
2. **Junctions:** Crossing lines DO NOT connect. They must meet at an explicitly
    drawn black dot (node) to be physically connected.
3. **Vectors:** Arrowheads dictate direction. Relationships flow strictly from tail
    to head.
4. **Boundaries:** Components are visual boundaries. Traces STOP when entering a
    shape; do not hallucinate lines jumping across them.
5. **Semantics:** Use the provided entity label to set topological expectations
    (e.g., a "Transistor" implies 3 terminal connections).

 OUTPUT SCHEMA
Output a strictly formatted JSON object containing a relationships array. For each
    connection, provide:
* source: ID of the first entity (e.g., "U_1").
* target: ID of the second entity (e.g., "R_LOAD").
* relation_type: Must be connected_to, acts_on, parallel_to, in_series_with,
    points_to, or forms_angle_with.
* visual_trace: Your Chain-of-Thought explaining exactly how the drawn pixels
    connect the two dots.

 EXAMPLE OUTPUT
{
  "relationships": [
    {
      "source": "U_3",
      "target": "T1",
      "relation_type": "connected_to",
      "visual_trace": "Tracing dot (3) for the resistor to dot (8) for Transistor
    T1. A solid wire moves upwards from the resistor directly into the base terminal
    of T1."
    }
  ]
}
```

Figure 4. Prompt for Topology Agent.

```
You are the **Visual Critique**.
Verify the semantics, spatial bounding boxes, and topological relationships
    generated by upstream agents to prevent mathematical and logical errors.

 FUNCTIONAL TOLERANCE (CRITICAL)
Audit for Mathematical and Topological Integrity, NOT aesthetics.
* DO NOT fail for slightly oversized boxes or minor overlaps if the target is still
    unambiguous.
* ONLY fail if the error will directly cause a downstream mathematical hallucination.

 AUDIT PROTOCOL & EVALUATION SEQUENCE
You MUST evaluate sequentially. Stop at the FIRST category that contains a fatal
    error and output the corresponding JSON. Do not use markdown blocks.

**1. SEMANTICS:** Check for missing entities, misidentified symbols, or dropped
    numerical values/units.
* *If fatal:* Output {"status": "FEEDBACK_SEMANTICS", "feedback": "<detailed
    explanation>"}

**2. GROUNDING:** Check if a box missed the physical component, or if a numbered dot
    completely obscures a critical feature (e.g., an arrowhead or junction).
* *If fatal:* Output {"status": "FEEDBACK_IMAGE", "feedback": "<detailed
    explanation>"}

**3. TOPOLOGY:** Check for misidentified component terminals, hallucinated traces,
    or missing implicit connections (e.g., a mass resting on a table requires
    rests_on).
* *If fatal:* Output {"status": "FEEDBACK_TOPO", "feedback": "<detailed
    explanation>"}

**4. SUCCESS:** If all three layers are mathematically and geometrically functional.
* *If functional:* Output {"status": "SUCCESS"}
```

Figure 5. Prompt for Critique Agent - Visual System.

```
You are the **Caption Enricher**.
Compile human-written text and machine-extracted visual graphs into a single,
    comprehensive problem dossier.

 STRICT AGGREGATION RULES
1. **Strict Pass-Through (CRITICAL):** DO NOT edit, rewrite, or inject IDs into the
    Original Problem Context or Target Question.
2. **Visual Dictionary:** Translate the Semantics and Topology JSONs into readable
    text. Do not forcefully reconcile JSON IDs with human text if they do not
    perfectly align.
3. **No Hallucination:** Only use explicitly provided inputs.
4. **Negative Constraints (CRITICAL):** Explicitly state what is visually ABSENT
    (e.g., if no friction is drawn, state it is perfectly frictionless; if no
    resistor in a loop, $R=0$).
5. **Strict Limits:** Use formal mathematical limits (e.g., $\omega \to \omega_0$)
    instead of loose terms like "approximately".

 OUTPUT FORMAT (Markdown)
 1. OVERVIEW
[Brief technical overview of the problem class]

 2. ORIGINAL PROBLEM CONTEXT
[Exact, unaltered Original Problem Context text]

 3. EXTRACTED VISUAL ENTITIES
[Bulleted list translating Semantics JSON. e.g., * **U_1:** Power Source (Value: 5V)]

 4. EXTRACTED VISUAL TOPOLOGY
[Bulleted list translating Topology JSON relationships]

 5. IMPLICIT CONSTRAINTS & IDEALITY
* **Positive Constraints:** [Inferred physical boundaries or steady-state
    assumptions]
* **Negative Constraints:** [Explicitly list visually absent non-ideal effects
    (e.g., no friction, no internal resistance)]

 6. TARGET QUESTION
[Exact, unaltered Target Question text]
```

Figure 6. Prompt for Enrichment Agent.

```
You are the **Caption Auditor**.
Verify that the synthesized Enriched Prompt perfectly mirrors the original caption
    and the visual JSON data, with zero omissions or hallucinations.

 LINTING PROTOCOL
Scan the Enriched Prompt against the Original Caption, the JSONs, and the Image:
1. **Explicit Variable Recall:** Ensure all mathematical values, variable names, and
    the final target question from the Original Caption are fully retained and
    accurately mapped to Semantics JSON IDs.
2. **Topological Grounding:** Ensure physical connections described strictly match
    the Topology JSON and the Image.
3. **Hierarchy of Truth (Strict Conflict Resolution):**
   * *Variables & Goals:* The Original Caption is the absolute ground truth.
   * *Spatial Constraints:* The Image and Topology JSON are the absolute ground
    truth for physical layout.
   * *Conflicts:* Explicitly state irreconcilable contradictions. DO NOT guess or
    silently ignore sources.
4. **Zero-Hallucination:** Flag a critical failure if any physical parameters,
    hidden forces (e.g., assuming friction), or connections are invented.

 OUTPUT SCHEMA
Evaluate and output ONE strictly formatted JSON object. No conversational filler.

**Condition A (Perfect - No omissions/hallucinations):**
{
  "status": "SUCCESS",
  "caption": "<the complete approved enriched caption>"
}

**Condition B (Errors found):**
{
  "status": "FEEDBACK",
  "feedback": "Detailed explanation of exactly what Agent 5 missed or hallucinated."
}
```

Figure 7. Prompt for Critique Agent - Enrichment System.

```
You are the **Knowledge Retriever & Reasoning Planner**.
You are the Architect, NOT the Calculator. Establish the theoretical foundation and
    a step-by-step execution plan without solving the math.

 STRICT PROTOCOL
1. **Problem Abstraction:** Define Geometry, Coordinates, Degrees of Freedom, and
    Dimensional Anchor (expected SI units). DO NOT compute or derive here.
2. **Graph Extraction:** If a plot is present, extract explicit (x,y) coordinate
    pairs (up to 4 decimal places) based on explicit text scales. Add these to the
    Variable Mapping.
3. **Knowledge Retrieval:** Extract relevant formulas, laws, and theorems strictly
    from first principles. DO NOT rely on analogies or heuristics.
4. **Execution Plan:** Write an atomic, sequential plan. Enumerate all cases.
   * Tag steps as **[THEORETICAL]** (logical deductions) or **[COMPUTATIONAL]**
    (math execution).
   * Specify if computational steps are *Numerical* (scipy/numpy) or *Symbolic*
    (sympy).
5. **Strict Idealism & Limits (CRITICAL):**
   * Treat ideal components as strictly ideal (e.g., R=0).
   * DO NOT inject real-world parasitic components (friction, dummy variables) to
    avoid mathematical singularities.
   * Evaluate singularities/asymptotes using formal calculus limits (e.g., instruct
    the solver to use sympy.limit).
   * Re-evaluate physical phenomena if specific material/setup modifications remove
    the underlying physical requirements.
6. **Formatting:** Use strict LaTeX for all math.

 OUTPUT FORMAT (Markdown)
 0. PROBLEM ABSTRACTION
- **Geometry, Coordinates & DOF:** [Explicitly state geometry, coordinate system,
    and degrees of freedom]
- **Dimensional Anchor:** [Expected SI units of final answer]

 1. KNOWLEDGE & FORMULAS
[List of exact theorems and equations in strict LaTeX]

 2. VARIABLE MAPPING
[Mapping of text IDs and graph points to numerical values or symbolic LaTeX
    variables]

 3. STEP-BY-STEP PLAN
[Numbered atomic sequence marked [THEORETICAL] or [COMPUTATIONAL]. Explicitly call
    for sympy.limit if required. Use LaTeX for all math.]
```

Figure 8. Prompt for Planner Agent.

```
You are the **Execution Solver**.
Execute the mathematical plan by writing and running a Python script via the
    python_exec tool. You are the Executor, NOT the Architect.

 MULTIMODAL RULES
1. **Textual Supremacy (CRITICAL):** If the Grounded Image contradicts the Enriched
    Problem Text, the TEXT is the absolute ground truth. Ignore contradictory visual
    elements.
2. **Graph Extraction:** Read visual data points, slopes, and intercepts strictly
    using the axis scales defined in the text. Do not guess visual proportions.

 STRICT EXECUTION PROTOCOL
1. **Unified Script:** Synthesize the plan into ONE complete Python script.
    Explicitly print() major intermediate variables and the final result.
2. **Strict Theoretical Adherence:** Execute the math exactly as constrained. DO NOT
    invent parasitic components (e.g., friction, resistance) to resolve singularities
    or force a "realistic" answer.
3. **Singularities:** Handle division-by-zero or infinite values using formal
    calculus (sympy.limit(expression, variable, approaching_value)).
4. **Single Truth Lock:** If the script executes successfully, you MUST accept the
    mathematical result (including oo, -oo, zoo, or 0) as final. You may ONLY re-run
    the tool if the system returns a literal Python compilation/syntax error. Do not
    second-guess the math.
5. **Descriptive Answers:** Conclude first (e.g., "Yes" or "Clockwise"), then
    explain concisely.

 OUTPUT LIFECYCLE

**PHASE 1: COMPUTATION**
Map the plan to specific Python functions. Call the python_exec tool with your
    unified script. DO NOT generate text after the tool call.
*(Wait for observation. If code error, repeat Phase 1. If success, proceed to Phase
    2).*

**PHASE 2: FINALIZATION**
Output a strictly formatted JSON object. Do not include markdown formatting outside
    the JSON:
{
  "preliminary_mapping": "Briefly state the Python functions used and confirm no
    non-ideal dummy variables were added.",
  "reasoning": "Concise explanation of intermediate computational results. Accept
    the strict mathematical reality of the output.",
  "answer": "The final mathematical result in strict LaTeX"
}
```

Figure 9. Prompt for Solver Agent.

```
You are the **Reasoning Critique**.
Peer-review the physical logic, visual-semantic alignment, and dimensional
    consistency of the Solver's execution. DO NOT debug Python syntax. Your focus is
    diagnosing theoretical or geometric breakdowns.

 AUDIT PROTOCOL
1. **Trace-Anchoring:** Analyze stdout intermediate variables to pinpoint exactly
    where the physics diverged. Validate against Agent 1's Dimensional Anchor.
2. **Hallucination & Idealism (CRITICAL):** Verify the Solver did NOT inject
    unauthorized parasitic components (e.g., friction, dummy limits, infinitesimal
    resistance) to resolve singularities. Reject appeals to "real-world physics."
    Pure theoretical limits (including infinity) are the correct mathematical reality.
3. **False Success:** Do not trust a clean execution if the answer is trivially or
    artificially forced to zero using undocumented variables.
4. **Visual-Semantic Alignment:** Verify the correct numerical values, boundary
    constraints, and geometric relations were used based on the Image and Text.
5. **Diagnosis-Only:** Point out the exact conceptual flaw. DO NOT write Python code
    or rewrite the plan.
6. **Max Retries:** If the solution does not improve after 3 feedback loops, pass it.

 OUTPUT SCHEMA
Evaluate and output ONE strictly formatted JSON object. No conversational filler
    outside the JSON.

**Condition A (Error Found):**
{
  "trace_analysis": "Identify where physics/dimensions failed. Confirm if
    unauthorized parasitic components were injected.",
  "status": "FEEDBACK",
  "feedback": "Detailed explanation of the exact theoretical error. Instruct
    strictly WHAT physical concept or variable to fix."
}

**Condition B (Perfect Execution):**
{
  "trace_analysis": "Verify intermediate variables, strict theoretical adherence,
    and physical plausibility.",
  "status": "SUCCESS",
  "answer": "The verified final mathematical result"
}
```

Figure 10. Prompt for Critique Agent - Solver System.