GThinker: Towards General Multimodal Reasoning via Cue-Guided Rethinking

Supplementary Material

In the supplementary material, we give a more detailed illustration of our method and experiments due to the page limit in the main text. We first give the reproduction claim in Section 1 and detail the data construction in Section 2. Then, we introduce the evaluation setting and training setting of our method in Section 3 and Section 4, respectively. Furthermore, we provide the results of more baseline models in Section 5, including different 7B-level models and larger 32B models, with an insightful discussion. Also, we perform quantitative analysis on the Visual Inertia problem in Section 6. Meanwhile, we introduce more about the reward function we design for the open-ended question in Section 7. Finally, we provide a brief discussion on the possible limitations.

1. Reproduction

In this paper, we construct a high-quality dataset leveraging several advanced multimodal models like GPT-4o, o1, and o3. Though we have detailed the data construction pipeline in the main text and also in the supplementary material to make it fully reproducible, it's still costly. Therefore, we open-source the constructed data, the model, and the training framework for reproduction and advancement in multimodal reasoning, which can be found at https://github.com/jefferyZhan/GThinker.

2. GThinker-11K Construction

To support the training of GThinker, we have designed a scalable data generation pipeline to construct the GThinker-11K data as we have concluded in §3.2 and §3.3, respectively. In this section, we systematically introduce the data construction process, including the 7K cold start data, as depicted in Figure 6, and 4K RL data.

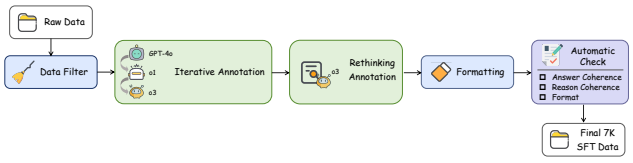

Figure 6. Full Iterative Multimodal Annotation Pipeline.

2.1. Data Preparations

Though several datasets are constructed to enhance multimodal reasoning capabilities in MLLMs [60–62] spanning diverse domains, they often present challenges such as high knowledge dependency, limited visual cues, or limited reasoning level. To extend the multimodal reasoning to general scenarios beyond knowledge-intensive math and science problems, we empirically find that the M^{3}CoT dataset provides a well-established data baseline for multimodal reasoning across domains. It details how to collect data across science, mathematics, and general scenarios with commonsense, and ensure the visual reliance and reasoning complexity with final manual checking. Building on baseline, we apply a two-step filtering process to ensure data quality: (1) we discard entries with corrupted or missing images, and (2) we verify the remaining samples' compliance with closed-source model usage policies using GPT-4o, resulting in 7,358 high-quality samples. We illustrated the data composition in Table 6.

Table 6. Data composition of 7K Cold Start data of GThinker-11K.

Type	Vol.	Source
Science	5266	KiloGram[23], ScienceQA [33], M^{3}CoT [7]
Mathematics	621	TabMWP [34], Math [17]
Commonsense	1471	Sherlock [18](Questions generated by M^{3}CoT)

2.2. Iterative Annotation

To generate high-quality reasoning paths and visual cues, we propose a iterative multimodal annotation methodology that leverages multiple leading MLLMs, such as OpenAI's o-series, for end-to-end reasoning path generation. Prior approaches [56, 61, 62] rely on multi-step pipelines to generate captions first and then utilize the reasoning LLMs to produce the final CoT annotations. Such methods can be limited by the quality of the captions and lose the specific features of multimodal reasoning. Instead, our method directly uses leading multimodal models and can produce higher-quality and more coherent multimodal long-chain reasoning paths with richer step-by-step visual cues and stronger logical deductions.

Specifically, as shown in Figure 4 and Figure 6, we draw the insight that different models offer complementary strengths [62] and design an iterative refinement strategy. We first input the question-image pair with the hinted answer with the user role into GPT-4o, which has been widely proven to be an advanced multimodal model, and get the initial annotations following the format defined in the system prompt shown in Figure 7. Then, we include the CoT annotation generated by GPT-4o as the context input for the next model, as indicated by the prompt shown in Fig-

Prompt 1: System Prompt

A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think><answer> answer here </answer>. In the reasoning process enclosed within <think> </think>, each specific visual cue is enclosed within <vcues_*>...</vcues_*>, where * indicates the index of the specific cue. Before concluding the final answer, pause for a quick consistency check: verify whether the visual cues support the reasoning and whether each step logically follows from what is seen. If correct, conclude the answer; otherwise, revise the visual cues and reasoning, then conclude.

Figure 7. System Prompt

ure 8. We organize the user input as defined in the "Input you will receive" part of Figure 8, and iteratively update the previous reasoning response. Leveraging the iterative multimodal annotation prompt as the fixed system prompt, we can guide a stronger multimodal reasoning model to generate a better reasoning chain with relevant visual cues identified and labeled with fewer missing and more convincing logic than the previous one. Meanwhile, for better instruction following and generation quality, we leverage the context learning capabilities of existing advanced MLLMs by inserting 2 hand-crafted task samples into the prompt. We iterate this process twice with the leading o1 and o3 models, respectively. This three-stage process significantly improves the precision and depth of final thinking annotations by leveraging the diverse capabilities of each model.

2.3. Rethinking Annotation

With the positive, high-quality reasoning annotation data, we further extend our process to generate the reasoning annotations with the cue-rethinking process using the same collected data to support our designed judge-guided selective training. Though previous methods [67, 68] have explored asking the leading MLLMs to generate negative visual annotations, such a manually crafted process may introduce hallucination, in which the false paths are not relevant to the image. To capture real mistake patterns, we first sample natural but wrong responses from different models [1, 58] with the thinking system prompt, *i.e.* without the visual cue clarification, for the training data. Then, we retain one response for each sample with flawed and mistake-relevant visual cues. We achieve this by asking o3 to compare the gold reasoning path with the sampled wrong responses and return the qualified one. This enables the sampled responses to contain missing or uncertain and misleading visual cues and faulty inferences, which is indicated as error pre-judgement. While positive samples provide a reference for correction, the variability in natural language expression requires a more nuanced approach. To this end, we employ the advanced reasoning capabilities of o3, using carefully designed prompts shown in Figure 9. For visual cue correction, each initial cue is explicitly linked to its corrected counterpart, followed by the revised deduction, ensuring the data remains structured and easy to parse. Meanwhile, for the previously generated correct annotations, we also utilize this prompt and the o3 model to insert a transitional sentence. However, this sentence indicates successful verification, thereby bypassing any further cue rethinking. We accomplish this annotation by replacing the italicized text, the previous short rationale, and the example within the prompt.

2.4. Formatting

After all annotations are completed, we use *regex* tool of Python to parse and verify the response format. We first check whether the format <think>*</think><answer> $*$ </answer> is satisfied and also <vcues_*> </vcues_*> is closed. Then, we utilize the GPT-4o to fix the samples with incorrect format structure. Finally, we remove the <aha> indicator for the rethinking transitional sentence.

2.5. Automatic Verification

With the formatted annotated data with and without cue-rethinking process, we perform automatic checks targeted at three critical aspects to ensure high data quality, helped by annotation-excluded Gemini 2.5 Pro [9], as illustrated in Figure 6. First, for format validation, we re-ensure that for each annotation, the positive reasoning path ends with a concluded answer, and the visual cues can be parsed. Second, for answer consistency, the annotated answers are parsed and cross-checked against the ground truth. Third, for reasoning coherence, we input the image, QA pair, and annotated reasoning into Gemini 2.5 Pro to evaluate logical alignment between visual cues and reasoning with the designed prompt shown in Figure 10, flagging any contradictions. Samples with identified issues are reprocessed through the relevant correction steps in our pipeline. Samples with identified issues are reprocessed through the relevant correction steps in our pipeline.

Prompt 2: Iterative Multimodal Annotation Prompt

You are a Checker-&-Corrector-&-Annotator of multimodal chain-of-thought answers.

Input you will receive (always in this order)
1. The multi-choice question with the corresponding image.
2. The true answer label (e.g. "B").
3. A short, human-annotated rationale for that true answer.
4. The model's PREVIOUS reasoning response, formatted exactly as

<think> . . . model's chain-of-thought (CoT). . . </think>
<answer> . . . model's final letter or text answer. . . </answer>

• Inside the <think>. . . </think> block, visual cues that the model claims to use are wrapped as <vcues_1> . . . </vcues_1>, <vcues_2> . . . </vcues_2>, etc.

Your task:
A. Verify the correctness of the previous model's answer and reasoning against the given image, true answer and human rationale.
B. If the model's final answer is already correct, keep the answer part.
C. If the answer is correct but some visual cues or reasoning steps are wrong or missing, fix the wrong cues / steps and append the NECESSARY cues/steps according to your knowledge.
D. If the answer is wrong, repair the erroneous cues / logic so that the corrected reasoning leads to the true answer.
E. Preserve structure, ordering and tags as possible—modify ONLY what is necessary for correctness and clarity.
F. Keep all tag syntax unchanged (<think> . . . </think>, <answer> . . . </answer>, <vcues_*> . . . </vcues_*>) so the output can be parsed automatically.

Output format
Return ONE corrected response, nothing else, in exactly the same two-tag layout:

<think>
. . . corrected chain-of-thought with fixed <vcues_*></vcues_*>. . .
</think>
<answer>
. . . single correct choice or textual answer. . .
</answer>

Additional rules
• If you remove an incorrect visual cue, replace it with the correct cue and keep the numbering consistent.
• Never fabricate content outside the scope of the provided information.
• Be concise—do not add redundant and repeated explanations beyond what is needed for a logically sound, correct solution.

Examples
• *Example 1*
• *Example 2*

Figure 8. Prompt for iterative multimodal annotation

Prompt 3: Cue-Rethinking Annotation Prompt

You are a Visual Reasoning Corrector and Annotator. Process the input <Model_Infer> with these rules:
%The user input is a dict including the image, question-answer pair, error pre-judgement(true or visual cue error), the to-be-annotated model inferred response, and also the verified rationale. Each of them is wrapped with the key like <Model_Infer> and <image>.

1. **Response Segmentation**:
- Remove the answer conclusion part in the model.
- Then, wrap the model's entire thought process in <think></think>.

2. **Visual Cues Annotation**:
- Within the <think> section, identify specific visual cue phrases (not entire paragraphs) and annotate each one with a tag in the format <vcues_*></vcues_*>, starting numbering from 1 (i.e. <vcues_1>, <vcues_2>, . . .).

3. **Visual Cues Reasoning Error Diagnosis and Correction**:
3.0. *All the data to be processed now concern errors in visual cues and may also include reasoning errors based on visual cues. These reasoning errors may include issues such as insufficient knowledge, over-analysis, etc.*
3.1. **During this process, do not revise the model's previous entire original thought after annotation**
3.2. Before the closing </think> tag, and insert a generated transitional sentence wrapped with <aha></aha> that conveys a message similar in meaning to: "Let's check each visual cue and corresponding reasoning before giving the final answer. *Generate the error type based on the Error Pre-judgement: Some visual cues seem to need adjustment.*" (The exact wording can vary as long as the idea is the same.)
3.3. On the next line immediately after this transitional sentence, for each visual cue annotated (using <vcues_*></vcues_*>) and their corresponding reasoning parts before <aha>, compare them with :
- The verified rationale (<rationale>)
- Your understanding of image
Then, after </aha>, update the corrected reasoning based on the visual cues if required. If necessary, replicate the relevant part from the original <vcues_*></vcues_*> tag alongside the revised reasoning.
3.4. After completing the reasoning corrections, perform a logical verification of the reasoning after the </aha> part
3.5. Append the final correct answer wrapped with <answer></answer>, i.e. <answer><Correct Answer></answer>, in the next line after the </think>, ensuring that the final answer is adjusted correctly.

4. **Output Constraints**:
- Preserve the original reasoning structure as much as possible.
- **Do not include similar phrases like "based on the rationale", "The reasoning should focus", "aligns with the rationale", "the model", because the processed content is used for the model training instead of a third-person view**
- Ensure that all annotations (<think>, <answer>, <vcues_*>, <aha>) are properly formatted and inserted in the correct locations.

Example 1:
...

Figure 9. Rethinking Annotation Prompt

To finalize the quality control of the designed pipeline, we manually review a randomly selected 15% subset of the final dataset and confirm that our pipeline reliably produces high-quality annotations, which ensures scalability.

2.6. Reinforcement Learning Data Construction

We first collect data from a broader range of sources, including [38, 60, 61] to ensure the generalization and obtain

Prompt 4: Verification Prompt

You are given a multiple-choice question with options and the image, the correct answer, and a generated response in the following format:

<think>thinking process here</think>
<answer>answer choice</answer>

You should align the answer choice in <answer></answer> with the choice content in the question, and then check whether the reasoning in <think>...<think> logically supports the answer choice content.

If the thinking process leads to that answer choice, output 1. Otherwise, output 0 and explain why it does not lead to the answer.

Figure 10. Verification Prompt

Table 7. RL data composition.

Type	Volume	Sources
Math	748	GeoQA+[4], TabMWP[34]
Science	1557	ScienceQA[33], PISC[27]
General	1719	GQA[20], CLEVR[25], COCO[30]

13K data after deduplication and sampling based on the categories. We follow [7] to divide the collected data into three primary categories, including math, science, and general reasoning. In different primary types of tasks, the image types vary, such as doc and table images in the science reasoning task. Instead of directly employing these data, we adopt the proposed offline balanced sampling methodology to cluster and curate 4K samples. We illustrate the composition of the final 4K data for the main result and the sources in Table 7.

3. Evaluation Settings

All evaluations are conducted on a single node equipped with 8 NVIDIA H100 GPUs. For M^3CoT, we follow each model's official settings and prompts and use VLMEvalKit [11] for fair evaluation. For other benchmarks, we use the results reported in their original papers. For RL-enhanced reasoning models, which primarily focus on math and science domains, we follow their released models and evaluation guidelines to conduct testing. The evaluation focuses on multimodal reasoning across general, mathematical, and scientific scenarios:

- **M^3CoT:** A challenging benchmark that spans science, commonsense, and math domains, with each example verified to require multi-step reasoning. M^3CoT uniquely validates that samples require multi-step reasoning, tightly coupled with key visual cues, whereas others may skew towards recognition or text logic. Therefore, we primarily use this benchmark to comprehensively evaluate models' multimodal reasoning capabilities across diverse scenarios.
- **General scenario benchmarks:** MMStar [5] and RealWorldQA [59]. These benchmarks focus on general and realistic scenarios, including parts of understanding-based reasoning tasks, and are used to evaluate multimodal reasoning capabilities. We also include HallusionBench [14] to evaluate models' capability in mitigating visual hallucinations and illusions, which closely aligns with our focus.
- **Science and math scenario benchmarks:** We use MMMU-Pro [66], which covers multiple scientific subjects, to evaluate multimodal reasoning in scientific contexts. For math-specific evaluation, we adopt the widely used MathVista [35] and MathVision [52]benchmarks.

4. Training Details

4.1. System Prompt

For the training and evaluation of the GThinker, we utilize the same system prompt to wrap the conversation, as shown in Figure 7.

4.2. Hyper-parameters

We have illustrated the key hyper-parameters in §4.1. In this section, we provide more information about the hyper-parameters used in our experiment. For the cold start stage, we use the training code of [69] and list the primary hyper-parameters in Table 8. For the DAPO, we utilize the EasyR1 framework and mainly use the default setting for training and list the modified ones in Table 9.

Table 8. Hyper-parameters for cold start stage

Name	Value
precision	bf16
max_seq_length	4096
warmup_ratio	0.1
max_pixels	12845056
min_pixels	316
learning rate	5e-6
batch size	128

Table 9. Hyper-parameters for DAPO

Name	Value
max_promp_length	15000
max_response_length	4096
global_batch_size	64
rollout_batch_size	64
max_pixels	4194304
min_pixels	262144
weight_decay	1e-2

Table 10. Experiments on scaled and other baseline models

Model	**M^{3}CoT**	**MMStar**	**MMMU Pro**	**MathVista**
Valley2-7B	63.5	59.5	31.2	63.2
GThinker-Valley2-7B	74.8	62.1	38.7	65.5
Ovis2-8B	62.9	62.9	36.8	69.5
GThinker-Ovis2-8B	74.2	63.5	37.4	72.4
Qwen2.5-VL-32B	72.7	68.8	44.0	72.9
GThinker-Qwen2.5-VL-32B	81.9	69.6	50.8	73.9

5. Experiments on More Baseline Models

To rigorously evaluate the generalizability of GThinker, we conduct experiments on the latest 7B leading models on the closed-source opencompass leaderboard, Valley2-7B and Ovis2-8B, and also on a larger scale Qwen2.5-VL-32B. We rerun the whole training process for each model, and specifically, we collect the response for each sample using the to-be-trained model and conduct the judge-guided selective training. We evaluate these three models under the same setting on four representative benchmarks, including comprehensive M^{3}CoT, general reasoning MMStar, knowledge-focused challenging reasoning MMMU Pro, and math reasoning MathVista.

As shown in Table 10, GThinker consistently improves the performance on four benchmarks of all three models of different base models and of different scales, confirming its broad applicability. For models at the 7B scale, the magnitude of performance improvement varies according to their initial capabilities. Following GThinker training, Valley2-7B and Ovis2-8B achieve gains of 10.3 and 11.3 points on M^{3}CoT, and 2.3 and 2.9 points on MathVista, respectively. Notably, Valley2-7B, which initially exhibits relatively weaker capabilities in knowledge and general scene reasoning, demonstrates substantial improvements. It reaches scores of 38.7 on MMMU Pro and 62.1 on MMStar, achieving parity with the performance of more advanced Ovis2-8B. These results indicate that our proposed reasoning paradigm, together with the training framework, effectively enhances the general reasoning capabilities of models with varying proficiency levels, successfully bridging existing capability gaps and demonstrating strong generalizability. Regarding the 32B model Qwen2.5-VL-32B, we observe an average improvement of 4.45 % across the four benchmarks. While the gains on MMStar and MathVista are relatively modest, attributed to the model's already robust baseline in mathematics and general scene reasoning, leaving limited room for improvement with the current data scale, the significant boosts on MMMU Pro and M3CoT are driven by synergistic factors. Primarily, GThinker's visual-cue-based reasoning approach enables the model to locate visual evidence more accurately, effectively eliciting the 32B model's extensive internal knowledge and reasoning capacity. Furthermore, this performance is bolstered by the superior visual perception capabilities of the Qwen2.5-VL architecture compared to other models [68, 69], which allows for more precise identification of visual cues. Collectively, the results across different base models and larger parameter scales further validate the effectiveness and robust generalizability of the proposed GThinker method.

6. More analysis on Visual Inertia

Based on our empirical observations, the Visual Inertia problem primarily manifests in three ways. The first is the omission of key visual cues. Overly relying on prior knowledge, the model is prone to overlooking critical visual details that are essential to the specific scene or question. The second is the misinterpretation of visual cues. The model tends to hallucinate or describe visual elements based on common-sense priors. Rather than grounding its interpretation strictly in the actual image content, it makes assumptions before fully comprehending the visual prompt. The third is perceptual ambiguity. The model sometimes strug-

gles to commit to a single interpretation and generates conflicting possibilities, which misguides the subsequent reasoning steps. Our method effectively addresses these issues through the proposed cue rethinking pattern. By re-evaluating the visual cues, our approach recovers missing cues and corrects misinterpretations. Moreover, this process helps verify uncertain observations to eliminate ambiguity, thereby significantly alleviating the Visual Inertia problem in current visual reasoning tasks.

To explicitly demonstrate the effectiveness of our method against Visual Inertia, we further conduct a quantitative experiment. As the quantitative analysis on Visual Inertia shown in Table 11, 71.6% of failures stem from Visual Inertia for the baseline. While existing methods like MM-Eureka fail to address this, GThinker utilizes the Cue-Rethinking mechanism to explicitly reason based on visual cues and revise them when necessary, resulting in a 34% reduction in the Visual Inertia error rate, further verified by Human-in-the-loop for fixed samples. This targeted reduction confirms the objective visual cue accuracy and that our gains are causally attributable to mitigating Visual Inertia.

Table 11. Quantitative analysis on Visual Inertia, judging with GPT-4o on M^3CoT.

Model	Visual Inertia	
	# Samples	Proportion among Error Samples
Qwen2.5-VL-7B	576	71.6%
MM-Eureka-7B	556	71.3%
GThinker-7B	203	47.3%

7. Verification on the Reward Design

As illustrated in §3.3, we design a reward combining the exact matching with edit distance for the open-ended question after applying the response prompt. Inspired by the success of the similarity calculation in OCR[16], we replace the reward model for open-ended questions in RLVR training with the edit distance. Specifically, for the comparatively short answer, which we define as less than 5 words, we use exact matching and use the edit distance instead. For the reward computation based on the edit distance, we follow the equation below:

$$r_{edit}(\mathbf{o}_{response}, \mathbf{o}_{gt}) = \begin{cases} 1 - r, & r \leq (\tau = 0.6) \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where $r = 1 - \frac{d_{\text{edit}}(\mathbf{o}_{response}, \mathbf{o}_{gt})}{\max(|\mathbf{o}_{response}|, |\mathbf{o}_{gt}|)}$. To verify the effectiveness of this design, we track the reward value of open-ended samples applying this reward function with an ablation experiment training using only open-ended samples. As shown in Figure 11, the reward function we designed is capable of providing robust and consistent rewards to guide model optimization, thereby validating the effectiveness of our method.

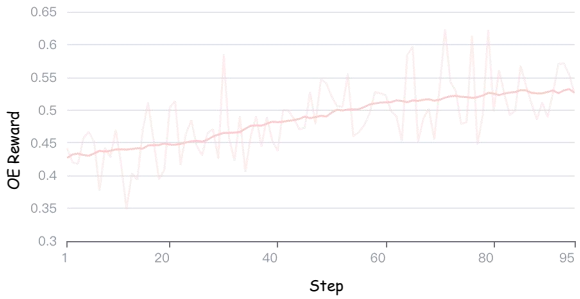

Figure 11. Reward curve of the open-ended reward function.

8. Limitations

A primary challenge in this field is the scarcity of open-source, visually dependent multimodal reasoning datasets, which constrains the scalability of data-intensive approaches. Furthermore, our pipeline leverages advanced MLLMs (e.g., Gemini 2.5 Pro) for both Chain-of-Thought annotation and as judge models to guide training. While relying on model generations can potentially introduce inherent biases or errors compared to gold-standard human labels, this trade-off is essential for scalability, given the prohibitive costs of large-scale manual annotation. To address this, we mitigate potential inaccuracies through our iterative multimodal annotation strategy and automatic verification mechanisms. Human spot-checks have further validated the high fidelity of our synthesized data. Moreover, our extensive experiments demonstrate substantial performance gains with minimal sensitivity to potential judge model variance, confirming the robustness of our approach. Future work will continue to explore superior methods for constructing high-quality reasoning datasets and leveraging stronger foundation models.