See Less, See Right: Bi-directional Perceptual Shaping For Multimodal Reasoning

Supplementary Material

6. Implementation Details

6.1. Data Statistics

We provide a quantitative breakdown of the data generation pipeline described in the main paper. The construction of our training dataset involves a rigorous filtering and synthesis process to ensure the quality and difficulty of the reasoning tasks. The statistics for each stage are summarized in Table 5 and detailed below:

- **Stage 1: Sampling and Reformulation.** We initially randomly sampled 50,000 raw chart-code pairs from the ECD [50] dataset. After the *Question Reformulation and Validation* phase, where the LLM arbitrator (GPT-5-mini) converted open-ended questions into verified multiple-choice formats, approximately 30K valid samples were retained.
- **Stage 2: Difficulty Filtering.** To ensure the model learns from non-trivial examples, we filtered the dataset using the base model (Qwen2.5-VL-7B-Instruct). Approximately 10K "easy" samples that answered correctly in all 8 rollouts, were discarded, leaving roughly 20K challenging samples.
- **Stage 3: Code Editing.** In this final stage, we performed programmatic editing to generate visual counterparts. We successfully generated the Evidence-Ablated View (I_{abl}) for 13K samples. Within this subset, we further successfully synthesized the Evidence-Preserving View (I_{pres}) for approximately 7K instances.

Consequently, the final high-quality dataset used for BiPS training comprises **13K samples**.

Table 5. **Statistics of the Data Generation Pipeline.** The table tracks the number of samples retained after each processing stage.

Pipeline Stage	Sample Count
Initial Sampling (from ECD)	50K
After Reformulation & Validation	∼30K
After Difficulty Filtering	∼20K
Final Training Set (Success in I_{abl} Generation)	**∼13K**
– subset containing I_{pres}	*∼7K*

6.2. Training Details

We detail the hyperparameter configurations for RL training in Table 6. Specifically, we employ the AdamW optimizer with a learning rate of 1×10^{-6} and keep the vision tower unfrozen. The reward is composed of 0.1 for correct formatting and 0.9 for the correct prediction.

Table 6. Hyperparameters for Reinforcement Learning.

Hyperparameter	Value
Batch Size	256
Learning Rate	1×10^{-6}
Optimizer	AdamW
Freeze Vision Tower	False
Max Response Length	2,048
KL Divergence Coefficient	0.01
Rollout Number	8
Temperature	0.85

7. Additional Results

7.1. Case Study

As shown in Figure 6, beyond in-domain chart benchmarks, we further evaluate cross-domain transfer on visual counting. The baseline fails due to incomplete object-level reasoning, whereas our model explicitly tracks and subtracts objects to arrive at the correct answer. The stronger performance in this setting indicates improved cross-domain generalization.

Figure 7 illustrates an example of an evidence-preserving view and an evidence-ablated view. This case clearly shows that simple operations such as cropping are insufficient for generating the two views, since the relevant visual evidence can be fine-grained and the associated meta-information may be sparse in the image. Moreover, the evidence-preserving view must retain all visual elements related to the question, including the necessary meta-information, whereas the evidence-ablated view provides a more precise, fine-grained modification that removes only the key visual element (e.g., the target line).

7.2. Comparison with Standard GRPO

To verify that the improvements stem from our specific perceptual shaping pipeline rather than merely applying RL to the data, we compare **BiPS-General** against a baseline trained with standard GRPO. This baseline is fine-tuned on the exact same combined dataset (programmatic samples + ViRL39k) but treats all data uniformly without the proposed two-stage perceptual shaping curriculum.

Question: Subtract all big red rubber cylinders. Subtract all blue objects. How many objects are left?

Qwen2.5VL-7B-Instruct:
There are no big red rubber cylinders in the image. There are two blue objects: a blue cube and a blue reflective sphere. Subtracting these two blue objects leaves us with 6 objects.
Final value: 6

BiPS:
<think> To solve the problem, we need to follow these steps:
1. Identify all the objects in the image. 2. Subtract all big red rubber cylinders. Since there are no big red rubber cylinders in the image, this step does not change the count of objects. 3. Subtract all blue objects. The blue objects in the image are a blue cube and a blue sphere. After removing the blue objects, we are left with the following objects: - Yellow cylinder - Two gray spheres - Red cube - Purple sphere - Red cylinder - Silver sphere Counting these, we have 7 objects left. </think>
\boxed{7}

Figure 6. **Cross-domain case on visual counting.** The baseline fails due to incomplete object reasoning, whereas BiPS correctly tracks and subtracts objects to obtain the right answer.

Table 7. Comparison between standard GRPO and our BiPS-General on the same data mixture. Our method consistently outperforms the standard GRPO baseline across all benchmarks.

Model	CharXiv	ChartQAPro	ChartMuseum	EvoChart	MathVista	MathVision	MathVerse	MMStar
Qwen2.5-VL-7B	42.5	36.6	26.8	52.0	68.2	25.2	41.1	62.1
GRPO	45.4	50.2	32.9	68.0	74.3	27.3	42.6	64.6
BiPS-General	**50.6**	**51.8**	**34.0**	**68.7**	**75.0**	**28.6**	**45.3**	**65.7**

Analysis. As shown in Table 7, while standard GRPO yields substantial gains over the base model, **BiPS-General** consistently outperforms it across all benchmarks. Notably, on complex chart reasoning tasks like CharXiv, our method surpasses the standard GRPO baseline by a significant margin (**+5.2**). This performance gap highlights a critical insight: simply optimizing reasoning via RL is insufficient if the underlying visual grounding is flawed. By explicitly shaping the model's perception through our programmatic curriculum before the general reasoning stage, BiPS ensures that the RL process operates on high-fidelity visual signals, thereby amplifying the effectiveness of the optimization.

7.3. Qwen3-VL-Thinking Results and Discussion

We further evaluate BiPS in the thinking-mode setting by fine-tuning Qwen3-VL-8B-Thinking [1] on 13K chart samples, with hyperparameters following the same configuration as described in Section 4.1 and Section 6. We still obtain significant improvements on charts and generalization to non-chart domains, which shows that BiPS provides perceptual improvements that complement the model's strong reasoning capabilities.

Setting. We use temperature = 1.0, top-p = 0.95, top-k = 20 and presence penalty = 1.2 for the fine-tuned model by default. For the base Qwen3-VL-8B-Thinking model, we keep temperature, top-p, and top-k unchanged, and sweep a small, pre-specified grid of presence penalties: 1.2 (matching the fine-tuned setting) and 1.5 (following the technical report [1]). This controlled sweep accounts for decoding sensitivity and maintains consistency with prior evaluations, and we report the best score within this fixed grid for the base model. The prompts follow those used in Qwen3-VL. For MMStar, we remove the duplicate instruction "Please select the correct answer from the options above." when the question already includes the hint: "Please answer the question and provide the correct option letter, e.g., A, B, C, D, at the end." We find that RL fine-tuning on chart samples negatively impacts optical character recognition (OCR) performance on text. This effect can be partially alleviated by adding a system prompt that biases the model toward OCR: "You are a vision-language model. For image-based problems, always prioritize accurate reading of all visible text, symbols, and numbers." We adopt this prompt for MathVerse, where a subset of examples present the question directly within the image. We use the same prompt across all models for a fair comparison.

Discussion. While BiPS delivers consistently strong in-domain gains and robust OOD generalization on Qwen2.5-VL-7B, its behavior on the more advanced reasoning model Qwen3-VL-8B-Thinking reflects a more challenging gen-

Table 8. Effect of BiPS on Qwen3-VL-8B-Thinking

Model	CharXiv	ChartQAPro	ChartMuseum	Evochart	MathVista	MathVision	MathVerse	MMStar
Qwen3-VL-8B-Thinking	53.0	54.1	40.4	72.2	**81.0**	62.2	77.2	75.3
GRPO	54.3	54.6	43.0	74.0	80.4	60.8	76.2	75.3
Ours	**58.1**	**56.8**	**44.1**	**75.5**	80.4	**63.9**	**77.4**	**76.3**

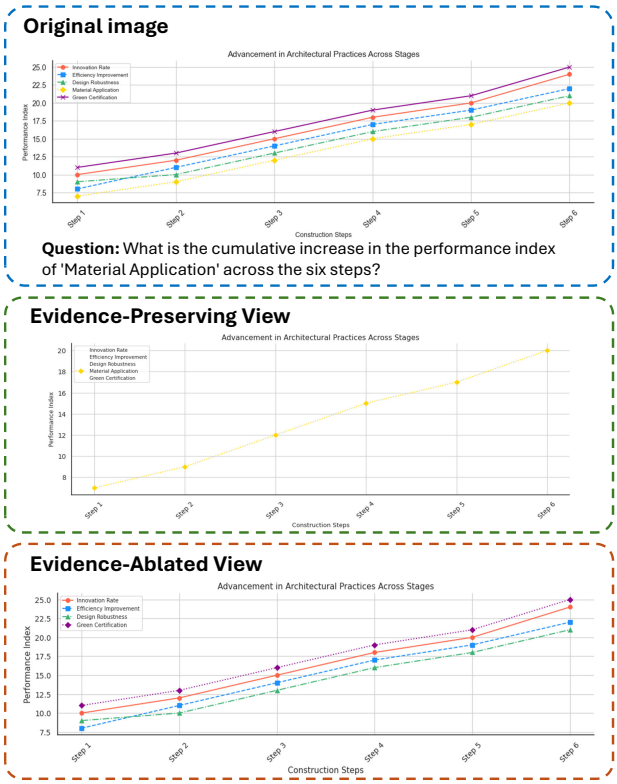

Figure 7. Evidence-Preserving and Evidence-Ablated views.

eralization setting. BiPS continues to yield substantial in-domain improvements, while largely maintaining cross-domain performance with moderate OOD gains on most benchmarks [20]. This trend is expected: BiPS is trained exclusively on chart data, and transferring fine-grained perceptual supervision to diverse visual domains becomes inherently harder under strong reasoning priors. Nevertheless, BiPS preserves cross-domain performance, indicating that the learned visual grounding signal remains transferable, although its magnitude is naturally bounded when the target domains differ substantially from the training distribution.

Extensibility. Beyond chart-centric training, BiPS naturally generalizes as a transferable perceptual supervision mechanism. While chart data alone already yields consistent cross-domain improvements, stronger and more uniform gains can be expected by extending BiPS to multi-domain training, for example by 1) mixing heterogeneous data, where non-chart domains are optimized with standard GRPO objectives, or 2) constructing bidirectional views across multiple domains and jointly optimizing them under the BiPS framework. BiPS can be extended to non-chart domains through construction of training views. For example, for natural images, recent visual chain-of-thought pipelines [37] suggest that segmentation tools such as SAM can automatically generate semantic masks, enabling edited views without human annotation. Similarly, procedural domains like Mazes [21] offer precise rendering control, allowing exact synthesis of counterfactual views similar to ours. We leave a systematic exploration of such hybrid and multi-domain settings to future work.

8. Prompts

Question Reformulation and Validation

- **System:** You are an expert in data analysis and question generation.
  Task: You will analyze a chart-related question-answer pair for correctness and potentially rewrite it as a multiple-choice question.
  Given:
  - Chart metadata and code snippets
  - A problem/question about the chart
  - An answer to that problem

  Your task:
  1. **Analyze correctness**: Determine if both the question and answer are factually correct based on the chart data.
  2. **Generate output**:
     - If correct: Rewrite as a multiple-choice question with 3–4 options.
     - If incorrect: Explain the error(s) without rewriting.
     - If uncertain: Explain what information is missing or unclear.

  Guidelines:
  - Ensure options are plausible but only one is correct.
  - Include at least 3 options, preferably 4.

- Distractors should reflect realistic misconceptions.
- Keep questions clear and unambiguous.
- Use data directly from provided metadata/-code.

- **User: Chart Metadata**: {code}
  Original Problem: {question}
  Provided Answer: {answer}
  Please analyze the correctness of this question-answer pair and generate the appropriate output according to the format specified.

Evidence-Preserving View

- **System:** You are an expert in chart code editing and data visualization. You will receive chart code (Matplotlib/Seaborn/Plotly/Altair) and a question. Your goal is to **minimize edits** while removing irrelevant elements and **preserving layout**: figure size, subplot grid, spacing, suptitle, legend order/length, trace order, color assignment, and axis links.
  Editing Principles:
  - Preserve all layout structure; if hiding content, keep axes and series positions.
  - Never change plotting library; only minimal imports for placeholders are allowed.
  - Keep legend/trace counts; for removed series use placeholders (e.g., NaNs, transparent marks, or "legendonly").
  - Be careful not to let the model derive the answer directly from the remaining elements; keep the necessary distractors.
  - Maintain axis limits when feasible to avoid scale drift.
- **Decision Rules:**
  - **Subplot-specific questions:** Keep only referenced subplots; blank others but preserve axes.
  - **Legend/category-specific questions:** Keep only mentioned categories; others become placeholders.
  - **Series/trace-specific questions:** Keep only targeted lines/bars/points; blank others.
  - **Global comparison or vague questions:** Do not edit.
  - If uncertain, set `should_edit = false`.
- **Post-edit requirements:**
  - Subplot grid unchanged; all axes preserved.
  - Legend length and order unchanged.
  - Placeholders inserted for every removed subplot/series.
  - Axis ranges preserved when appropriate.
  - Output must be **JSON only**.
- **User: Chart Code**: {code}
  Question: {problem_str}
  Please determine whether the chart code should be edited to remove irrelevant elements according to the rules above.

Evidence-Ablated View

- **System:** You are an expert in chart code obfuscation for evaluation/red-teaming. Given chart code (Matplotlib/Seaborn/Plotly/Altair) and a question, your task is to make the question **unanswerable** by removing/blanking decisive evidence while **preserving layout**.
  Objectives (priority order):
  1. Ensure **unanswerability**: blank all chart elements that allow a definitive answer.
  2. **Preserve layout**: keep figure size, subplot grid, spacing, legend structure, series order, and color assignments.
  3. **Minimize edits**: hide or blank evidence without refactoring or adding new content.
- **Decisive Evidence (to be blanked):**
  - Any subplot targeted or compared by the question.
  - Legend/categories mentioned or implied by the question/options.
  - Series/traces/marks revealing values, trends, peaks, ranks, or comparisons.
  - Numeric cues: labels, annotations, thresholds, reference lines.
  - Axes information that allows inference once geometry is hidden.
  - If unsure, treat as decisive (favor over-blanking).
- **Blanking Tactics (by library):**
  - **Matplotlib**: Replace data with NaNs, set invisible while keeping legend handles, or use dummy `Line2D`. For bars/scatter: empty/-NaN arrays or alpha=0. For entire subplots: `ax.cla(); ax.set_axis_off()`.
  - **Plotly**: Keep trace but hide geometry via `visible='legendonly'` or empty `x/y` while keeping `showlegend=True`.
  - **Altair**: Keep encodings and legend domain; blank via opacity=0 or empty filters.
- **Decision Rules:**

  - **Options provided**: blank all option-referenced elements.
  - **Comparisons/ranking/extremes**: blank all compared candidates.
  - **Single-target lookup**: blank the target's geometry and any revealing annotation.
  - **Global comparisons**: blank decisive evidence across all involved candidates.
  - **Trend/correlation**: blank scatter points and trend/regression lines.
  - **Threshold questions**: blank values and relevant threshold lines.
- **Post-edit Requirements:**
  - Subplot grid unchanged; axes preserved.
  - Legend length/order preserved (dummy placeholders allowed).
  - Placeholders inserted for all blanked series/-subplots.
  - Axis ranges preserved when applicable; code must run.
  - Remaining visuals must not allow a human to answer the question.
- **User: Chart Code**: {code}
  Question: {problem_str}
  Your task: Make the question **unanswerable** by blanking all decisive evidence while preserving layout.