Cube Bench: A Benchmark for Spatial Visual Reasoning in MLLMs

Dhruv Anand   Ehsan Shareghi
Department of Data Science and AI, Monash University
dana0001@student.monash.edu

Abstract

This research introduces Cube Bench, a dynamic Rubik's cube-generation framework and a set of benchmark tasks. It breaks competence into five skills: (i) reading the cube correctly from image and text, (ii) choosing the best next move, (iii) predicting before acting whether a candidate move will help, do nothing, or hurt, (iv) keeping control over many steps and recovering from mistakes, and (v) self-checking to revise mistakes. Using the same scrambles, a shared prompt/parse setup, and a straightforward "moves-to-go" score, we compare recent MLLMs side-by-side, depth by depth. Testing seven recent MLLMs on Cube Bench reveals a persistent gap between strong one-step performance and reliable solving: accuracy drops sharply with depth; early halts trigger error cascades; good perception does not guarantee selection/control. We also observe a pronounced closed-source versus open-source gap: the best closed model leads in both static and long-horizon reasoning, while open-source models cluster near chance; yet even closed-source models degrade at higher depths, struggling to maintain high accuracy. Reflection yields modest gains with occasional overthinking. Cube Bench offers a compact, transparent yardstick for spatial visual reasoning.[1]

1 Introduction

Multimodal large language models (MLLMs) promise perception-to-action reasoning; yet, most evaluations remain *static* i.e., single-turn recognition or QA on fixed inputs, so outputs never feed back into subsequent states (Li et al., 2024). Flagship vision–language tests such as VQAv2 exemplify this one-shot format, which lacks feedback coupling and long-horizon credit assignment (Goyal et al., 2017). Interactive agent suites expose the resulting gap: even models that excel on static tasks often falter over long horizons. For example, in WebArena the best GPT-4–based agent achieves 14.41% end-to-end success versus 78.24% for humans, underscoring brittleness under feedback (Zhou et al., 2024). VisualWebArena likewise highlights that most prior suites target text-only agents and introduce visually grounded web tasks to evaluate multimodal agents in realistic interfaces (Koh et al., 2024). To avoid the confounds of open-ended websites (e.g., DOM drift, tool failures, grading noise) while directly testing closed-loop behavior, we adopt a compact, deterministic environment with oracle progress signals that isolate (i) pre-action evaluation, (ii) one-step selection, and (iii) stepwise adherence under feedback.

Building on this design goal, we introduce **Cube Bench**, a simple, reproducible benchmark that: (1) measures the gap between one-step and multi-step solving across scramble depth and prompt format; (2) quantifies recovery after the model makes a mistake; (3) tests the "think-ahead" ability by asking the model to predict if a candidate move will help, do nothing, or hurt before acting; (4) establishes a basic perception floor, i.e., checking that the model can correctly read the cube and align image and text, linking it to later control; and (5) evaluates self-reflection ability. Rubik's Cube offers a small, composable action set with rich long-horizon behavior; each move deterministically updates the state, enabling interactive step-by-step (or closed-loop) evaluation that adapts to the model's actions.

Our evaluation invokes the following research questions (RQ):

RQ1 How big is the gap between one-step recognition and true closed-loop solving, and how does that gap change with scramble depth (higher puzzle complexity) and input modality?

RQ2 After making a mistake, how often do models recover, and how does recovery change as

[1] https://github.com/AnyPlease/cube-bench

puzzles becomes more complex?

RQ3 Can models "mentally" simulate a move and predict whether it helps before acting?

RQ4 Do models reliably “see” and align image and text in the first place, and does that perceptual grounding predict later stability?

RQ5 Does self-reflection improve choices without leaking answers, and what is the trade-off (error fixes vs. overthinking)?

The contributions from our research are as follows:

- **Cube bench**: a dynamic, reproducible benchmark to produce scrambles, consistent prompts and parsers, and evaluation metrics.
- We find a persistent gap between strong one-step accuracy and reliable closed-loop control, which widens with scramble depth, high single-turn performance does not translate into stable multi-step solving.
- We observe depth-sensitive degradation across models: even the strongest degrade at higher depths, while open-source models cluster near chance on the hardest settings.
- We show that reflection helps selectively, delivering modest, label-safe gains in some conditions; however, but it occasionally induces overthinking that harms downstream choices.
- We confirm that perception is necessary but not sufficient; accurate cube reading and verification alone do not guarantee good action selection and control under compounding errors.
- We uncover surprising limitations between open-source and closed-source “thinking” models on the same tasks, quantify the gap, and show where each side leads or lags.

In the rest of the paper, we outline related work on multimodal evaluation and interactive agent benchmarks (§2), present Cube Bench and the VirtualCube generator (§3), and detail our experimental methodology (§4), including the model suite, and task protocols with scoring and fairness controls (Face Reconstruction, Cross-Modal Verification, Move Prediction, Reflection, Closed-Loop Step-by-Step, Causal Move-Effect, and Learning Curve). We then report results (§5): basic capabilities (§5.1), the effect of reflection interventions (§5.2), closed-loop adherence vs. depth and modality (§5.3), and recovery dynamics (§5.4).

2 Related work

2.1 Static perception benchmarks

A large share of multimodal evaluation remains static: single-turn recognition or QA over fixed inputs, where the model’s output does not affect subsequent state (Li et al., 2024). Canonical vision–language datasets exemplify this format: ImageNet and COCO emphasize perception via classification, detection, and captioning on isolated inputs (Russakovsky et al., 2015; Lin et al., 2014); VQAv2 frames one-shot question answering over images (Goyal et al., 2017); CLEVR targets compositional reasoning with synthetic scenes (Johnson et al., 2017); GQA extends to real-image relational reasoning (Hudson and Manning, 2019). Specialized benchmarks follow a similar single-turn design, e.g., TextCaps for OCR-heavy captioning, ChartQA for questions about data graphics, and ScienceQA for multimodal science items (Sidorov et al., 2020; Masry et al., 2022; Lu et al., 2022). While these suites are invaluable for perception and short-form reasoning, they lack feedback coupling and long-horizon credit assignment; static accuracy can therefore overestimate a model’s capacity for planning, recovery, and closed-loop control. This motivates controlled settings where progress is measured step-by-step and where perception, preaction evaluation, and action selection can be disentangled.

2.2 Interactive & long-horizon benchmarks

Interactive benchmarks make the static–dynamic gap concrete: when models must plan, act, and observe consequences over many steps, performance drops sharply despite strong single-turn scores. In **WebArena**, a realistic, reproducible suite of multistep website tasks, the best GPT-4-based agent achieves only **14.41%** end-to-end success, versus **78.24%** for humans (Zhou et al., 2024). **VisualWebArena** extends this to visually grounded interfaces and documents the persistent limitations of both text-only and multimodal agents in realistic web tasks (Koh et al., 2024). Results of this kind point to a recurring pattern: small early mistakes cascade, recovery is rare, and tool use degrades over long horizons.

At the same time, open-ended web environments introduce confounds that complicate diagnosis. Websites and DOM structures can change between runs; tools and APIs fail intermittently; instructions and grading can be ambiguous, making it hard to attribute errors to perception, evaluation, or action selection. Recent analyses (e.g., more rigorous re-evaluations and safety/reliability–focused suites) also highlight how non-stationarity and transient web unreliability stress current agents beyond simple completion rates (Zhou et al., 2024; Koh et al., 2024). Taken together, this motivates a *compact, deterministic, and oracle-scored* alternative where progress is measured step-by-step and sources of failure can be isolated. Our cube-based setting provides such control: it separates (i) *think-before-acting* (predicting move effects), (ii) *one-step selection* (choosing the best move), and (iii) *closed-loop adherence* (following a teacher over multiple steps), yielding transparent, reproducible signals for long-horizon reasoning under feedback.

2.3 Rubik's Cube in LLM/VLM and planning research

Rubik's Cube has long served as a structured testbed for search and planning, with exact shortest-path results ("God's Number" in the half-turn metric is 20 moves) and classical solvers that provide oracle distances (Rokicki et al., 2014; Korf, 1997). Deep RL + search systems, such as *DeepCubeA*, demonstrate scalable learning-to-search on the cube and related combinatorial puzzles, reinforcing the value of oracle metrics for stepwise progress (Agostinelli et al., 2019).

More recently, several VLM/LLM-centric efforts use the cube for embodied, multi-step manipulation. *CubeRobot* proposes a VLM with a dual-loop "VisionCoT" architecture and introduces *CubeCoT* as a cube-focused dataset and evaluation (Wang et al., 2025b). *VisionCube* adds multi-view 3D cues and memory for long-horizon planning, also reporting results on *CubeCoT* (Wang et al., 2025a). *Auto-RubikAI* composes a VLM, an LLM, and a symbolic knowledge base to plan and execute restoration sequences (Fan and Yuan, 2025). Earlier, OpenAI's Dactyl showcased sim-to-real dexterity by solving the cube with a robot hand, illustrating hardware and actuation confounds typical of embodied setups (OpenAI et al., 2019).

These systems highlight contemporary interest in cubes as a spatial-reasoning domain, but they mix perception, solving, 3D reconstruction, tooling, and actuation, making it hard to attribute errors. In contrast, our benchmark keeps the environment compact, deterministic, and oracle-scored: we disentangle (i) pre-action evaluation, (ii) one-step selection, and (iii) closed-loop adherence, yielding clean, reproducible signals for long-horizon reasoning under feedback.

3 Cube Bench

Contemporary spatial-reasoning benchmarks primarily test static scene understanding. The **Cube Bench** instead targets sequential spatial reasoning and algorithmic control. Rubik's Cube provides compact and visually rich states and ground-truth progress/ optimization (distance-to solution; Face twist /Half-Turn), allowing us to isolate perception from planning and control under feedback.

3.1 Benchmark Tasks

Models can caption, classify, and chat about pixels. But can they *use* what they see and read to act coherently over time? Cube Bench presents this through a compact, rule-bound world where progress and optimality are objectives (distance-to-solution; Face-Turn / Half-Turn). Each step isolates a different capability, allowing us to determine whether failures arise from seeing, thinking, or doing, and at what computational cost.

From seeing to knowing (RQ4: perception & grounding). We begin with vision as the floor: reconstruct the visible 3×3 face from a rendered view and verify that the image and text agree. This anchors local perception and cross-modal grounding before any control, so downstream errors can be attributed to planning rather than misreading the state. (Tasks: *Cube Face Recognition*, §4.3.1 & *Cross-Modal Verification*, §4.3.2).

From knowing to choosing (RQ1: competence gap in static vs. closed-loop). Next, we ask the static question: given four candidate moves, which one most reduces the distance-to-solved? This single-turn MCQ (*Move Prediction*, §4.3.3) provides an upper bound on policy selection without compounding errors. Comparing this to closed-loop behavior below reveals the central gap between one-shot recognition/choice and sequential adherence.

From choosing to anticipating (RQ3: pre-action simulation). Before allowing models to act, we test whether they can predict the effect of each candidate move, specifically whether it would shrink, leave unchanged, or increase the remaining moves

Term	Notation	Definition / Use
Scramble depth	d	Number of moves used to generate the start state under the chosen metric.
Distance-to-solved	$d(s)$	Minimal moves from state s to solved under that metric (computed via IDA* with pattern databases (Korf, 1997)).
Half-turn metric	HTM / FTM	Quarter- and half-turns both count as one move. Used interchangeably in the literature; baseline metric here.
Quarter-turn metric	QTM	Quarter-turn counts as one move; half-turn counts as two. Used only when stated explicitly.
Teacher plan	π^{teach}	Inverse of the scramble (guaranteed solve, not necessarily optimal). Used for teacher-adherence and baselines.
Optimal move	$a^{\star}$	Any move that reduces $d(s)$ by exactly 1 (ties possible).

Table 1: **Terminology and notation.** Unless noted, the same metric is used for scrambling and evaluation.

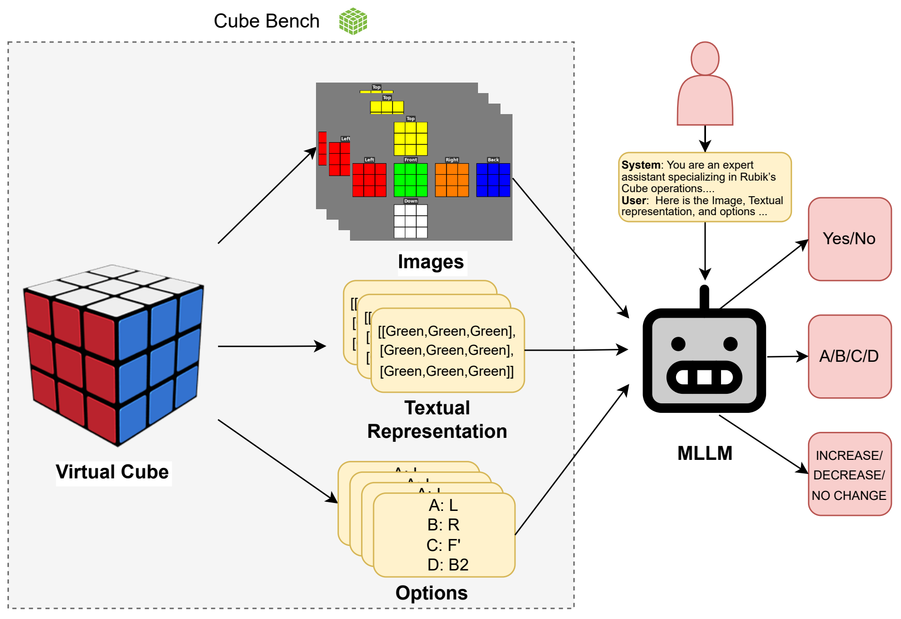
Cube Bench
Images
[[Green,Green,Green],
[Green,Green,Green],
[Green,Green,Green]]
Textual Representation
A: L
B: R
C: F'
D: B2
Options
Virtual Cube
System: You are an expert assistant specializing in Rubik's Cube operations....
User: Here is the Image, Textual representation, and options ...
MLLM
Yes/No
A/B/C/D
INCREASE/ DECREASE/ NO CHANGE

Figure 1: **Cube Bench dataflow.** A *VirtualCube* state is emitted as (i) a rendered image, (ii) a normalized textual serialization, and (iii) a fairness-aware four-option move set (A–D). The MLLM consumes the required modalities, user prompt and returns task-specific judgments: *Yes/No* (Verification), *A/B/C/D* (Move Prediction / Closed-Loop), or `DECREASE / NO_CHANGE / INCREASE` (Causal Move-Effect).

to solve, without executing it (*Causal Move-Effect*, §4.3.6). This isolates the *Evaluation* step in a simple decision loop we call **TEA**: *Transition* (list the possible next moves), *Evaluation* (estimate each move's impact on progress), and *Argmin* (choose the move with the best predicted outcome).

From anticipating to acting (RQ1: optimality under depth & modality). We then open the loop. From a scramble of depth d, the model selects one move at a time; the environment applies it and returns the new state (*Step-By-Step*, §4.3.5). We score teacher-adherence and oracle progress under fixed HTM/FTM, stratified by depth and prompt modality, to see how quickly compounding errors erode optimality, and which promptings (image+text vs. text-only) close the gap.

From acting to recovering (RQ2: in-context recovery). Finally, Competent agents make mistakes and recover. We therefore, measure whether behavior rebounds after an error and how the chance of recovery changes with scramble depth (*Learning Curve*, §4.3.7). Summaries such as solve-rate distributions, median attempts (on solved cases), and first-error indices reveal whether the policy replans or spirals.

3.2 Dataset as Generator: The `VirtualCube` Environment

Instead of shipping a fixed corpus, **Cube Bench** generates each item on demand using a lightweight

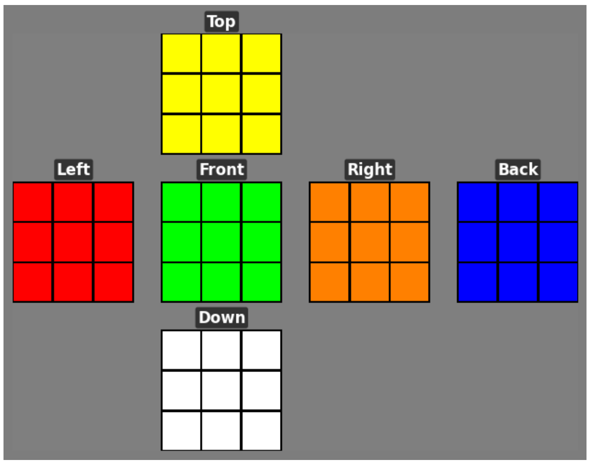

Figure 2: An example of rendered `scrambled image` from the dataset.

simulator, `VirtualCube` (Fig. 1). This choice prevents memorization of a small set of scenes, keeps storage cost negligible, and gives us fine-grained control over difficulty and modality while remaining fully reproducible.

Design. (i) *Controllability*: set scramble depth d, move metric (HTM/FTM), camera/view, lighting, and visual perturbations (e.g., occlusion, brightness, recolor); (ii) *Faithful ground truth*: Present an oracle capable of providing exact measurements of distance-to-solution by employing IDA* search in conjunction with pattern heuristics, as well as implementing the teacher plan, which constitutes the inverse of the scramble; (iii) *Fair sampling*: when MCQs are required, draw four options (A–D) via a fairness-aware sampler that balances progress/no-change/regress classes and avoids degenerate choices; (iv) *Reproducibility*: every sample is identified by $(d, \text{seed}, \text{variant})$ and regenerated from a logged config.

What it emits. For each seed, `VirtualCube` produces: (1) a rendered image of the cube from a canonical viewpoint for cube net (e.g., Figure 2); (2) a normalized textual serialization of the visible or full state (e.g., 3×3 face tokens); (3) an optional A–D option set for MCQ tasks. These streams feed the task heads used throughout the suite: *Verification* (Yes/No), *Move Prediction / Closed-Loop* (A–D), and *Causal Move-Effect* (DECREASE/NO CHANGE/INCREASE). Oracle distances provide labels and acceptance criteria (teacher-adherence and/or any optimal-progress move under the chosen metric).

Name	Vendor	License
Gemma 3-27B	Google	✔
GLM-4.5V	ZhipuAI	✔
Llama 4-Scout-17B:16E	Meta	✔
Qwen2.5-VL-7B-Instruct	Alibaba	✔
Qwen2.5-VL-32B-Instruct	Alibaba	✔
Qwen3-VL-30B-A3B-Thinking	Alibaba	✔
Gemini 2.5 Pro	Google	✗

Table 2: LLMs used in the experiments. ✔= open-source, ✗= closed-source.

4 Experimental Methodology

All experiments were run on NVIDIA A100 GPUs (80 GB). We used Python 3.11.13 and vLLM (Kwon et al., 2023) v0.11.0 as the inference engine, wrapped by our Cube Bench evaluation harness. Unless otherwise stated, inference used bfloat16 (bf16) precision and request-level batching of 100 prompts. Decoding was deterministic (temperature = 0.0, i.e., greedy) for every task *except* the reflection variants, where we enabled stochastic decoding, as specified in their respective subsections.

Each task section shows the exact prompt we used, how we rendered the inputs (image/text), and any decoding limits (such as the maximum tokens). To keep things repeatable, we fix the random seed for each run, record the engine details (model name, vLLM version, precision, batch size), and keep the prompts and answer choices the same within a task unless we state otherwise. We report results by combining all cases for that task (questions, steps, or episodes, as appropriate). Each section also lists the metrics used and the number of examples we tested.

4.1 Ethics & Data Privacy

No human participants, animals, or personally identifiable information were involved. All inputs are procedurally generated cube images/text states from `VirtualCube` and model outputs from released LLMs used under their licenses. Institutional ethics approval was therefore *not required*. We followed university research data-management practices for storage, access, and retention; experiment logs contain only non-identifiable seeds and aggregate metrics. Any third-party assets were used within their stated licenses (see Table 2).

4.2 Models Evaluated

We evaluated seven multimodal LLMs that span open-weight releases and API-only services (Table 2). The open-weight group consisted of GLM-4.5V (Team et al., 2025b), Qwen2.5-VL-7B/32B-Instruct (Bai et al., 2025) and Qwen3-VL-30B-A3B-Thinking (Team, 2025), Gemma 3-27B (Team et al., 2025a) and Llama 4-Scout-17B:16E (Meta AI, 2025). The closed-source consisted of Gemini 2.5 Pro (Google, 2025). These choices reflect widely used, recent multimodal families with public weights (open-weight) or stable cloud endpoints (API), and include both dense and Mixture-of-Expert variants.

4.3 Task Suite

The Cube Bench consists of rich and diverse tests, which are designed to evaluate MLLMs Visual and Reasoning capability.

4.3.1 Cube Face Recognition

Given a single cube image, the model must reconstruct the visible face as a 3×3 grid of discrete text tokens. This isolates local perception/compositional grounding before any planning or control, providing an upper bound for vision-dependent tasks (e.g., verification (§4.3.2), move prediction (§4.3.3)) and addressing RQ4.

Setup. We present a single rendered image of a Rubik's Cube from a canonical viewpoint with the FRONT face fully visible (plus adjacent edge context). The model must return a 3×3 face matrix in row-major order over a fixed vocabulary $\mathcal{V}$. We accept either color tokens {`W`,`Y`,`G`,`B`,`O`,`R`} or face tokens {`U`,`D`,`F`,`B`,`L`,`R`}; outputs are normalized via a fixed mapping.[2] Any response that does not yield a valid 3×3 matrix over $\mathcal{V}$ after normalization is recorded as a parse violation; free-form text outside the matrix is ignored for scoring.

Scoring. Primary: *Element-wise Accuracy* (per-sticker correctness, averaged over the 3×3 grid). Secondary: *Matrix Accuracy* (exact match of the full 3×3). We additionally report results stratified by scramble depth $d \in \{1, 2, 3\}$, noting that recognition should be depth-insensitive.

Notes. See Definition (§A.1.1) and Generation & Fairness Controls (§A.1.2) in Appendix A, and see §B.1 in Appendix B for a detailed prompt for this task.

Recognition serves as a vision floor; large gaps here indicate perception issues rather than reasoning or control deficits.

4.3.2 Cross-Modal Verification

Given a rendered *cube net* and a textual hypothesis about the FRONT face, the model must answer **Yes/No** correctly. This probes *cross-modal grounding and consistency* before any control, catching hallucinations or weak visual grounding that would otherwise contaminate downstream move selection, and addresses RQ4.

Setup. *Image:* one rendered cube net with canonical layout and face labels; all stickers are fully visible (no occlusion). *Text:* a front-face hypothesis instantiated from a template. In the simplest (and default) instantiation used in our runs, the hypothesis is a full 3×3 grid for the FRONT face (i.e., "these nine stickers, in this order, are the front face"). We also support richer templates (single-cell claims, row/column constraints, color counts, full-matrix assertions), but unless stated otherwise we evaluate the full-face equality case. *Response format:* a normalized Boolean string: `Answer: Yes` or `Answer: No`. Free-form text is ignored for scoring.

Scoring. Primary: *Balanced Accuracy* neutralizes any uneven Yes/No prior (when the prior is enforced to 50/50, this equals accuracy). We treat "Yes" as the positive class:

$$\text{Balanced Accuracy} = \tfrac{1}{2}\,(\text{TPR} + \text{TNR}).$$

$$\text{TPR} = \Pr(\hat{y} = \text{Yes} \mid y = \text{Yes}) = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

$$\text{TNR} = \Pr(\hat{y} = \text{No} \mid y = \text{No}) = \frac{\text{TN}}{\text{TN} + \text{FP}}.$$

where TP is True Positive, TN is True Negative, FP is False Positive, and FN is False Negative.

Secondary metrics: *Per-template accuracy* (if multiple prompt templates are used), *Parse rate* (share of predictions that are exactly `Yes` or `No`), and *Label bias* (mean predicted "Yes" rate vs. the true prior). Unless noted otherwise, results are reported at scramble depth $d{=}5$. at $d{=}1$, the front face can be monochromatic, so we default to $d{=}5$ to better test visual recognition in a multi-color setting. Verification is expected to be mostly depth-insensitive[3].

[2]Normalization: e.g., `white` $\mapsto$ `W`; case-insensitive; punctuation ignored.

[3]Ideally, verification is depth insensitive for a good vision-language model

Notes. See Definition (§A.2.1) and Generation & Fairness Controls (§A.2.2) in Appendix A.

Verification is a minimal, high-precision gate on cross-modal grounding: failures here indicate perception/alignment issues, not planning. We, therefore, place it before move-selection or closed-loop tests.

4.3.3 Move Prediction (Static MCQs)

At depth $d{=}1$, the model sees four candidate moves (A–D) and must pick the one that most reduces the remaining steps to solve ("moves-to-go") under a fixed scoring rule (ties are broken by a preset rule i.,e., lexicographically options). This isolates the TEA loop's selection step, *Argmin* (the "A" in *Transition* → *Evaluation* → *Argmin*), as a single, non-compounding decision and targets RQ1: the gap between static pick-the-best performance and closed-loop adherence. We run matched prompts in three modes (see Appendix B.3): Image + Text, Image-only, and Text-only, to measure reliance on each modality and any benefit from fusing them. We reuse this static MCQ test for guided reflection (§4.3.4, answering RQ5), keeping prompts, rendering, parsing rules, and scoring identical for fair comparison.

This protocol addresses RQ1 by quantifying single-step prediction in a static setting and using it as the baseline for comparison with closed-loop solving.

Setup. *State.* A one-move scramble $s = T(s^\star, a_s)$ from solved $s^\star$ rendered by `VirtualCube` with fixed camera/colors. *Options.* Four labeled actions $\{a_A, a_B, a_C, a_D\}$ where exactly one is optimal at $d{=}1$ (the gold). Distractors are sampled from remaining legal moves. Options are shuffled per item. *Prompt variants (matched triplets per state).* (M1) $I{+}S$: rendered image I plus a canonical text serialization S of the front face; (M2) I only; (M3) S only. *Response format.* A single normalized choice `ANSWER: A|B|C|D`. Robust parsers accept `<ANSWER>X</ANSWER>`, `ANSWER: X`, or `<X>`; all others are parse-violations.

Scoring. *Primary:* Top-1 accuracy per modality. *Compliance:* Parse rate (number of valid A–D tag).

Notes. See Definition (§A.3.1) and Generation & Fairness Controls (§A.3.2) in Appendix A.

4.3.4 Reflection Guided Reasoning

Single-step multiple-choice questions (MCQs) can fail due to superficial heuristics or slight parsing mistakes. This study examines whether a structured process of *draft* → *reflection* → *re-answer* (Renze and Guven, 2024) enhances the quality of choices in visual reasoning *without* updating parameters or risking dataset leakage. The evaluation focuses on whether reflection interventions (both guided and redacted) significantly improve the selection of optimal answers without leaking any answers. This approach isolates the impact of post-hoc self-correction on identical items, quantifies the frequency of correcting incorrect answers, and assesses any "overthinking" that might cause regressions on answers that were previously correct, addressing RQ5.

Setup. Items are generated (using §4.3.3) with a scrambled depth of 1 (d=1). Unless stated, temperature= 0.0 and the default prompt modality is **mixed** (image+text), see §B.3.1 in Appendix B. Reflection templates are predetermined with two options: Guided–Unredacted and Guided–Redacted (default). The re-answer prompt ensures a single normalized choice and a fixed tie-break order (A > B > C > D) for unclear reflections.

Protocol. For each item i: (1) **Draft**: query once and record the initial choice $\hat{y}_i \in \{A, B, C, D\}$. (2) **Reflection**: present the same state and options plus the model's prior choice (gold masked in the redacted condition) and collect free-form reflection r_i. (3) **Re-answer**: re-ask the same MCQ, injecting r_i, and require a strict one-line response. No finetuning or memory carries across items.

Conditions. **Guided–Unredacted**: reflection receives the gold option (upper-bound oracle guidance or leaky reflections). **Guided–Redacted**: gold masked (measures structured reflection benefit without leakage). We provide an analysis of each sample (both correct and incorrect) to evaluate the model's tendency to overthink and its rate of error correction.

Scoring. **Initial Acc**: accuracy before reflection (from §4.3.3). **Final Acc (All)**: accuracy after re-answering (all samples). **Gain (Δ)**: Net gain in accuracy after reflection (Final Acc. - Initial Acc.) **Error-Fix Rate (EFR)**: the fraction of previously incorrect samples that were corrected. **Overthink Rate (OTR)**: the fraction of previously correct samples that have been flipped to wrong (N/A in Wrong-Only).

Compliance. Re-answers not matching the strict format are marked as parse violations and are scored as incorrect.

Notes. See Definition (§A.4.1) and Generation & Fairness Controls (§A.4.2) in Appendix A.

4.3.5 Closed-Loop Step-By-Step

From a scramble of depth d, the model selects the next move at each step from a four-option MCQ (A–D) while the environment immediately *executes* the choice. The teacher plan is the inverse of the scramble; a step is marked correct if the choice matches the teacher or any oracle-optimal move that strictly reduces distance-to-solved under a fixed turn metric (HTM/FTM) for the run. Episodes terminate on the first non-progressing action or a parse failure.

This protocol measures compounding errors and stability under feedback rather than one-shot guessing, and we use it to directly address RQ1 by quantifying the gap between static recognition and closed-loop solving across depths. It also serves as the common baseline for RQ2 (the recovery-focused *Learning Curve*, §4.3.7 variant permits progress after near-misses).

Setup. Depth $d \in \{1, 2, 3, 4, 5\}$; N episodes with a deterministic per-sample seed i. Each episode initializes `VirtualCube` at depth d and fixes a single distance metric (HTM/FTM) for the run. The teacher plan is the inverse scramble derived from the seed. We use a fixed stepwise MCQ prompt template and a rendering config (camera/view, variant). At each step t, the current state is rendered and a four-option legal move set $\{a_A, a_B, a_C, a_D\}$ is formed by including the teacher move and sampling three unique distractors; options A–D are uniformly shuffled. Responses are expected as normalized tokens `A|B|C|D` (optional `IDK`); robust parsers also accept `<ANSWER>X</ANSWER>`, `ANSWER: X`, or `<X>`.

Scoring. We report two metrics: *Teacher Adherence (TA%)* and *Perfect Solve Rate (Perfect-Solve%)*. A step is correct iff the model's move equals the teacher move a_t^{teach}. Non-teacher moves that still reduce distance may be applied to continue the episode but do not count toward TA%. Episodes end at the first non-progress or a parse failure, and we use *unconditional denominators*: halted/missing later steps are scored as incorrect. Let N be the number of episodes and d_i the scramble depth of episode i.

$$\text{TA\%} \;=\; \frac{100}{N}\sum_{i=1}^{N}\frac{1}{d_i}\sum_{t=1}^{d_i}\{\text{correct}_{i,t}\},$$

with non-compliant outputs (non-A–D, parse fails) scored as incorrect.

Perfect-Solves%. Share of episodes with *all* steps correct under the same criterion; any early halt, non-progress, parse failure, or non-compliant output breaks perfection:

$$\text{Perfect-Solves\%} \;=\; \frac{100}{N}\sum_{i=1}^{N}\Big\{\prod_{t=1}^{d_i}\text{correct}_{i,t}\Big\}.$$

Notes. See Definition (§A.5.1) and Generation & Fairness Controls (§A.5.2) in Appendix A.

Abstention (Selective Control). To probe calibrated non-answers without inflating headline scores, we allow an explicit `IDK` output but keep our primary metrics unchanged: for TA% and Perfect%, `IDK` is scored as incorrect (no credit), and results are reported with unconditional denominators. In other words, abstention can prevent a bad move, but it does not boost adherence.

We additionally study selective behavior in the Appendix: models may abstain (`IDK`) when uncertain, and we report *coverage* (answered/total), *selective accuracy* (correct/answered), and *APA (Abstention-Penalized Accuracy)* with weight λ for `IDK`. Implementation follows our runner: `IDK` is detected via a strict tag or common variants; the default policy is `teacher_on_abstain` (apply the teacher move to preserve episode length), with `skip_item` as a sensitivity variant. Unless noted, λ=0.25 and, when used, confidence thresholds control when `IDK` is permitted. Formal definitions and full confusion/coverage table appear in App. §A.6.

4.3.6 Causal Move-Effect

Before acting, the model must *evaluate* each candidate move's local consequence on distance-to solved: DECREASE (closer), NO CHANGE (same), or INCREASE (farther). This isolates the TEA loop's Evaluation step, separating state evaluation from policy selection, and complements closed-loop by avoiding compounding errors while still requiring forward simulation. It addresses RQ3 by testing whether models can accurately forecast the direction of change in a task-aligned objective induced by a candidate action prior to execution.

Setup. A cube state s is sampled by `VirtualCube`. We construct four candidate moves $a_A, \dots, a_D$ with a fairness-aware sampler. The prompt provides a textual serialization S of s plus face-center colors (for color→face mapping). The model must return four tags, one per option, using the strict format *<A> DECREASE|NO_CHANGE|INCREASE </ A>*. Robust parsing accepts light variants (e.g., `A: DECREASE`); missing tags are scored as errors and recorded as `MISSING` in the confusion matrix. No image is used for scoring (prompt instructs TEXT-only reasoning to isolate modality preference).

Scoring. (i) *Micro-Accuracy* p_o over the four labels; (ii) *Macro-F1* averaged over {DECREASE, NO_CHANGE, INCREASE}; (iii) *Cohen's* κ (chance-corrected):

$$\pi_i = \Pr(\text{gold} = i), \quad q_i = \Pr(\text{pred} = i)$$

$$p_e = \sum_i \pi_i q_i, \quad \kappa = \frac{p_o - p_e}{1 - p_e}.$$

We compute p_o from the confusion-matrix diagonal, and p_e from the gold priors π and the model mix q (also logged as `expected_dot`). Intuition: κ=1 perfect agreement; κ=0 chance; κ<0 worse than chance. Parse failures (`MISSING`) count as false negatives for the gold class.

Notes. See Definition (§A.7.1) and Generation & Fairness Controls (§A.7.2) in Appendix A.

4.3.7 Learning Curve (closed-loop MCQ)

This test probes whether a model can improve (or at least persist) across short interactive trajectories when each decision is posed as a four-option MCQ. We repeatedly ask for the next move (A–D) on a cube scrambled to depth d, applying an *accept-progress* policy so episodes can recover from near-misses rather than collapsing on the first error (cf. §4.3.5). It addresses RQ2, which examines the models' capacity to correct errors.

Setup. Depth d, N episodes, deterministic per-sample seed i, fixed prompt template, renderer config, and `max_attempts`. Each episode starts from a depth-d scramble; the teacher plan is the inverse scramble. At each attempt we form a fresh MCQ from the current state, query (image+text unless specified), parse `<ANSWER>X</ANSWER>` or `ANSWER: X` (else *parse-fail*), apply the control policy, and stop on solve or on `max_attempts`.

Scoring. Let there be N episodes and, for episode i, $T_i \in \mathbb{N} \cup \{\infty\}$ the number of attempts used to solve ($T_i = \infty$ if unsolved). With $\mathbb{1}\{\cdot\}$ the indicator:

$$\text{SR} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\{T_i < \infty\},$$

$$P(1) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\{T_i = 1\},$$

$$P(\leq 3) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\{T_i \leq \min(3, \texttt{max\_attps})\}.$$

$$\text{Med@Solved} = \text{median}\{\, T_i : T_i < \infty \,\},$$

$$\text{Avgz@All} = \frac{1}{N}\sum_{i=1}^{N} \min\{T_i, \texttt{max\_attempts}\}.$$

We report 95% confidence intervals for SR using the Wilson score interval; $P(1)$ and $P(\leq 3)$ use the unconditional denominator N (unsolved contribute 0).

Notes. See Definition (§A.8.1) and Generation & Fairness Controls (§A.8.2) in Appendix A.

5 Experiments & Results

Protocol consistency. Unless otherwise stated, all experiments use the same prompt templates (see Appendix B) and a fixed sample size of $N = 100$ items per condition. This consistency holds across all tests in §5 (depths 1–5, modalities, and any intervention variants), with the item sets drawn once and reused across models to enable paired comparisons. Strict A–D formatting is required for MCQ outputs; any non-compliant response is counted as an error.

5.1 Basic Capabilities: Recognition, Verification & Move Prediction

We begin by establishing a vision and one-step decision floor before any closed-loop control. **Face reconstruction** (§4.3.1) measures per-sticker and full-matrix fidelity; **cross-modal verification** (§4.3.2) checks image ↔ text consistency with balanced accuracy; and **static move prediction** (§4.3.3) evaluates one-step optimal-move selection across Image + Text, Image-only, and Text-only prompts. Together these isolate perception and single-turn choice, against which later sequential adherence can be compared.

Model	Element-wise Acc. (%)			Matrix Acc. (%)			Verification (%) @ $d = 5$		
	$d = 1$	$d = 2$	$d = 3$	$d = 1$	$d = 2$	$d = 3$	Bal.	Parse	Bias
Gemma 3-27B	61.9	47.2	36.60	31.00	8.00	3.00	58.00	100.00	70.00
GLM-4.5V	100.00	99.70	98.70	100.00	98.00	96.00	95.00	100.00	55.00
Llama 4-Scout-17B:16E	70.60	41.00	33.20	36.00	7.00	3.00	47.00	100.00	93.00
Qwen 2.5-VL-7B-Instruct	76.20	70.60	62.00	40.00	22.00	13.00	71.30	98.00	57.10
Qwen 2.5-VL-32B-Instruct	84.00	78.80	71.30	68.00	43.00	19.00	81.00	100.00	65.00
Qwen3-VL-30B-A3B-Thinking	98.10	95.40	86.70	94.00	78.00	60.00	84.00	100.00	66.00
Gemini 2.5 Pro	100.00	98.80	96.80	100.00	90.00	84.00	100.00	100.00	52.00

Table 3: Left: *Cube Face Reconstruction* reported as element-wise (per-sticker) accuracy and exact 3×3 matrix accuracy at depths $d \in \{1, 2, 3\}$. Right: *Cross-modal Yes/No Verification* at fixed depth $d = 5$ with **Bal**=$(\mathrm{TPR} + \mathrm{TNR})/2$, **Parse**=% outputs matching `Answer: Yes|No`, and **Bias**=$|\text{Yes rate} - 0.5|$ (lower is better).

Headline patterns: (i) high performers sustain strong reconstruction at shallow depths, with degradation concentrated in exact 3×3 layout at d=3; (ii) verification balanced accuracy varies by family despite near-zero parse violations; and (iii) on one-move MCQs, top models lead across modalities (e.g., Gemini 2.5 Pro: 96% Image+Text, 92% Image-only, 80% Text-only), while smaller open-weights cluster near 20–35%.

5.1.1 Cube Face Reconstruction

Using the method from §4.3.1 (element-wise vs. matrix accuracy; parse-violation rate) and prompt in §B.1 in Appendix B, we observe depth-linked layout fragility despite largely stable per-sticker recognition. For example, in Table 3, by d=3 Gemini 2.5 Pro retains high per-sticker fidelity (96.8%) with some drop in exact 3×3 matches (84.0%), while Qwen 2.5-VL-32B falls more sharply (71.3% / 19.0%), indicating weaker global spatial coherence. Overall, all models show widening gaps between element-wise and matrix accuracy as depth increases, aligning with small edge misplacement compounding into full-face mismatches.

Takeaway. Depth primarily impacts *spatial coherence* beyond mere sticker recognition; while element-wise accuracy remains high, it is the precise 3×3 matches that are affected. Robust frontier models maintain performance at d=3, yet mid-range and smaller open-source configurations exhibit significant matrix drop, notwithstanding adequate sticker scores, thereby indicating weak global binding. This verifies Face Reconstruction as the *vision floor*: deficiencies in this context pertain to perceptual/grounding issues rather than planning. Consequently, we condition Verification (§5.1.2), Move Prediction (§5.1.3), and Closed-Loop (§5.3 & §5.4) processes on passing this test, and assess both element-wise and matrix accuracy to clearly differentiate token recognition from spatial arrangement.

5.1.2 Cross-Modal Verification

Using the method from §4.3.2 and the prompt in §B.2 in Appendix B, we tested the model's ability to verify a cube's orientation. The model was given an image of an unfolded cube (a net) and a hypothesis for the FRONT face, and it had to answer *Yes* or *No* to confirm if the hypothesis was correct. We measured performance using *Balanced Accuracy*, which provides a fair score by neutralizing any potential imbalance in the *Yes*/*No* class distribution. We also reported the *parse rate* (how often the model provided a valid *Yes/No* answer) and *label bias* (its tendency to favor "Yes"). While this verification task should ideally be insensitive to scramble depth, we used a fixed depth of $d = 5$. This ensures the FRONT face was sufficiently mixed (i.e., less monochromatic), making the visual confirmation non-trivial. To further isolate the reasoning task, our experiments used a balanced 50/50 prior for correct answers and only varied the phrasing (surface forms) to prevent the model from relying on lexical shortcuts.

Models with flawless parsing still differ in their discrimination ability. For example, Qwen3-VL-30B-A3B-Thinking reaches Bal. 84.00%, Qwen 2.5-VL-32B hits 81.00%, and Qwen 2.5-VL-7B achieves 71.30%. GLM-4.5V attains 95.00%, while Gemini 2.5 Pro was perfect in our runs (Bal. 100.00%). Although parse rates are ≈100% for these top models, a residual label bias persists (ranging from 52–93%). For instance, Llama 4-Scout-17B:16E exhibits a high label bias of 93%, indicating it largely favours "Yes" and suggesting a tendency to default to "Yes" when uncertain. See Table 3 (right) for the per-model breakdown.

Takeaway. Verification acts as a high-precision gate on cross-modal grounding: failures here reflect perception/alignment issues rather than planning. We therefore place it before move selection or closed-loop tests and use per-template breakdowns when applicable to localize weaknesses.

5.1.3 Optimal Move Prediction (Static MCQ)

Using the method established in §4.3.3 and prompts (image + text, image-only, text-only) in §B.3 in Appendix B, we probe single-turn competence by asking the model to choose the unique action (A–D) that most reduces the distance-to-solved for a one-move scramble (d=1). This isolates the Argmin/selection step without closed-loop compounding. We run matched prompt variants i.e., Image + Text, Image-only, and Text-only, per state to quantify modality reliance. Responses must normalize to a single A–D tag; we also track the parse-violation rate. Options are fairness-shuffled, so positions A–D are approximately uniform over the set. The primary metric is top-1 accuracy per modality.

From Table 4, Gemini 2.5 Pro tops all three modalities (Image+Text 96%, Image-only 92%, Text-only 80%) with perfect A–D parsing, showing that both perception and the final Argmin selection (A from TEA loop) step are strong. In contrast, most open-weights sit around 20–35% and show uneven modality behavior. Gemma-3-27B is particularly weak on Image-only (27% with 100% parsing), which is consistent with its poor face reconstruction and low verification from §5.1.1 & §5.1.2. Qwen-2.5-VL-7B performs better on Text-only (26%) than on Image-only (18%) and even Image+Text (19%). Its modest visual reconstruction indicates that adding a noisy visual stream negatively affects fusion. Qwen-2.5-VL-32B stays 26–28% across modalities with perfect parsing despite stronger perception and verification, pointing to selection limits. Qwen3-VL-30B-A3B-Thinking shows good verification but fragile format compliance (Text-only parsing 77%) that depresses its MCQ. GLM-4.5V is the clearest decoupling case: near-perfect reconstruction and strong verification, yet low MCQ (31% Image+Text; 21% Image-only), again implicating the selection/ranking step rather than perception.

Takeawys. (i) Static MCQ exposes a real selection gap: several models with decent reconstruction/verification (§5.1.1 & §5.1.2) still hover near chance because they mis-rank candidates or violate the A–D format; (ii) Fusion isn't free when vision is the bottleneck (e.g., Gemma-3, Qwen-7B); Image+Text accuracy can underperform text-only accuracies (e.g., Llama 4); (iii) Strict parsing matters, non-A–D outputs directly convert to errors and can mask underlying competence; (iv) Only Gemini 2.5 Pro presents a consistent story across tasks, high perceptual fidelity (reconstruction), strong cross-modal grounding (verification), and accurate selection (MCQ) while others are capped either by perception (Gemma-3) or by the Argmin step (GLM-4.5V, Qwen-32B).

5.2 Reasoning Interventions Improve MCQ Accuracy

We evaluate a draft $\rightarrow$ reflection $\rightarrow$ re-answer pipeline (§4.3.4) on the same MCQ items as Table 4 under two regimes. In the *Unredacted* (leaky; see Appendix B.4.1 for a detailed unredacted reflection prompt) setting, every model improves significantly, see Table 5, with GLM4.5v and Llama 4-Scout-17B:16E saturating at 100% (+69 and +82 points, respectively), and Qwen models gaining +42 to +50 points. This confirms that targeted reasoning can recover many initial mistakes when the label is leaked.

The *Redacted* (no-leak; see Appendix B.4.2 for a detailed redacted reflection prompt) setting is more revealing. The Qwen family shows consistent, meaningful gains of **+11–14** points (EFR $\approx$ 28–36%), albeit with sizable OTR (50–65%). Llama 4-Scout-17B:16E improves modestly (+6; EFR 20%, OTR 40%). Gemma-3-27B shows no net benefit (0), and GLM4.5v degrades (**–15**) due to high overthinking (OTR 66.7% vs. EFR 6.2%). These outcomes emphasize that label-free reflection can help, but is not uniformly positive, and that the EFR/OTR balance is decisive.

Takeaways. (1) **Reflection helps, but the leaky view inflates it.** Unredacted results are upper bounds (even reaching 100%); real-world, no-leak gains are smaller, (2) **Mind the EFR–OTR tradeoff.** Net improvement hinges on fixing wrong answers without flipping correct ones. GLM4.5v's –15 points under Redacted is explained by very high OTR (66.7%) overwhelming limited EFR (6.2%), (3) **Benefits vary by model.** Qwen models consistently gain (+11–14); Llama 4-Scout gains slightly (+6); Gemma-3 shows none; some models can *overthink* and regress without labels, and (4) **Reasoning $\neq$ perception.** These interventions pri-

	Image + Text		Image		Text	
Model	**Acc. (%)**	**Parse (%)**	**Acc. (%)**	**Parse (%)**	**Acc. (%)**	**Parse (%)**
Gemma 3-27B	33.00	100.00	27.00	100.00	31.00	100.00
GLM-4.5V	31.00	98.00	21.00	98.00	28.00	96.00
Llama 4-Scout-17B:16E	18.00	100.00	16.00	100.00	25.00	100.00
Qwen 2.5-VL-7B-Instruct	19.00	100.00	18.00	100.00	26.00	100.00
Qwen 2.5-VL-32B-Instruct	28.00	100.00	27.00	100.00	26.00	100.00
Qwen3-VL-30B-A3B-Thinking	23.00	86.00	19.00	98.00	32.00	77.00
Gemini 2.5 Pro	96.00	100.00	92.00	100.00	80.00	100.00

Table 4: **SolveMovesTest (1 move from solved).** Top-1 accuracy and format compliance (Parse: % outputs matching `ANSWER: [A-D]` or `<ANSWER>...</ANSWER>`) across prompt modalities. All runs use the same number of items per modality for comparability.

		Guided (Unredacted)				Guided (Redacted)			
Model	**Initial Acc. (%)**	**Final Acc. (%) ↑**	**Δ ↑**	**EFR (%) ↑**	**OTR (%) ↓**	**Final Acc. (%) ↑**	**Δ ↑**	**EFR (%) ↑**	**OTR (%) ↓**
Gemma 3-27B	22.00	81.00	+59.00	81.70	20.70	22.00	+0.00	22.00	78.00
GLM4.5v	31.00	100.00	+69.00	100.00	0.00	16.00	-15.00	6.20	66.70
Llama 4-Scout-17B:16E	18.00	100.00	+82.00	100.00	0.00	24.00	+6.00	20.00	40.00
Qwen 2.5-VL-7B-Inst.	17.00	59.00	+42.00	59.00	41.20	31.00	+14.00	36.00	65.00
Qwen 2.5-VL-32B-Inst.	24.00	70.00	+46.00	71.80	36.43	35.00	+11.00	34.71	64.00
Qwen3-VL-A3B-Thinking	23.00	72.00	+50.00	71.80	27.34	34.00	+11.00	27.81	50.00

Table 5: Reflection-guided re-answering on Optimal Move Prediction involves two methods: "Guided (Unredacted)" with correct options during reflection for an oracle upper bound, and "Guided (Redacted)" without labels as the default. Initial Acc. is accuracy before reflection (see Table 4); Final Acc. is after one reflection pass; Delta is the absolute change; EFR is the corrected share of initially wrong items; OTR is the share of correct items flipped to wrong. All runs use Image+Text modality, zero temperature, deterministic option shuffling, and strict A–D parsing.

marily strengthen selection on static items; they do not repair weak visual grounding or format compliance (§5.1.1, §5.1.2).

5.3 Closed-loop control

Having tested **Reconstruction** (§5.1.1), **Verification** (§5.1.2), and **Reflection** (§5.2), we now ask whether those gains survive *feedback*: after each choice the cube state changes, so errors can compound and recovery matters. We therefore evaluate MLLMs under closed-loop control, using two complementary probes mapped to our TEA framing (Transition $\rightarrow$ Evaluation $\rightarrow$ Argmin):

(1) Optimal Adherence (closed-loop next-move). At each step the model must pick the next action and stay on the teacher trajectory; success requires sustained, stepwise adherence under state feedback. We report *Teacher-Adherence (TA%)*, and *Perfect-Solves%* (episodes with TA $=100\%$).

(2) Move-Effect (evaluate-before-you-act within the loop). Before acting, the model labels each candidate (A–D) as DECREASE/NO CHANGE/INCREASE in distance-to-solved, exposing whether its internal evaluation supports good control. We report *Micro-Accuracy*, *Macro-F1*, and *Cohen's* κ with $p_e = q \cdot \pi$ to discount prior-chasing. A fairness-aware sampler balances classes and option letters to avoid exploitable priors.

Unless stated otherwise, protocols and metrics follow §4.3.5 & §4.3.6, with the text as ground truth (images shown only for reference).

5.3.1 Optimal-Move Adherence vs Depth

Applying the methodology established in §4.3.5 and the prompt in §B.5 in Appendix B, we proceed as follows: we run the step-by-step protocol from scrambles of depth $d \in \{1, \dots, 5\}$. At each step, the model selects A–D; the environment executes the choice immediately. A step is correct if it matches the teacher plan (inverse scramble) or any oracle-optimal move that strictly reduces the distance-to-solved under a fixed turn metric. Episodes halt at the first non-progressing action or failure to parse. We report Teacher-Adherence (TA%: correct steps per episode divided by d) and Perfect-Solves (% of episodes completed), using unconditional denominators so early halts count as errors.

Models degrade with depth at varying rates. From Fig. 3, Gemini 2.5 Pro starts strong in the green band at d=1, then declines with errors (from 90 to 10 TA%). Open-weights cluster in the red band by $d \geq 3$: Qwen-2.5-VL-7B

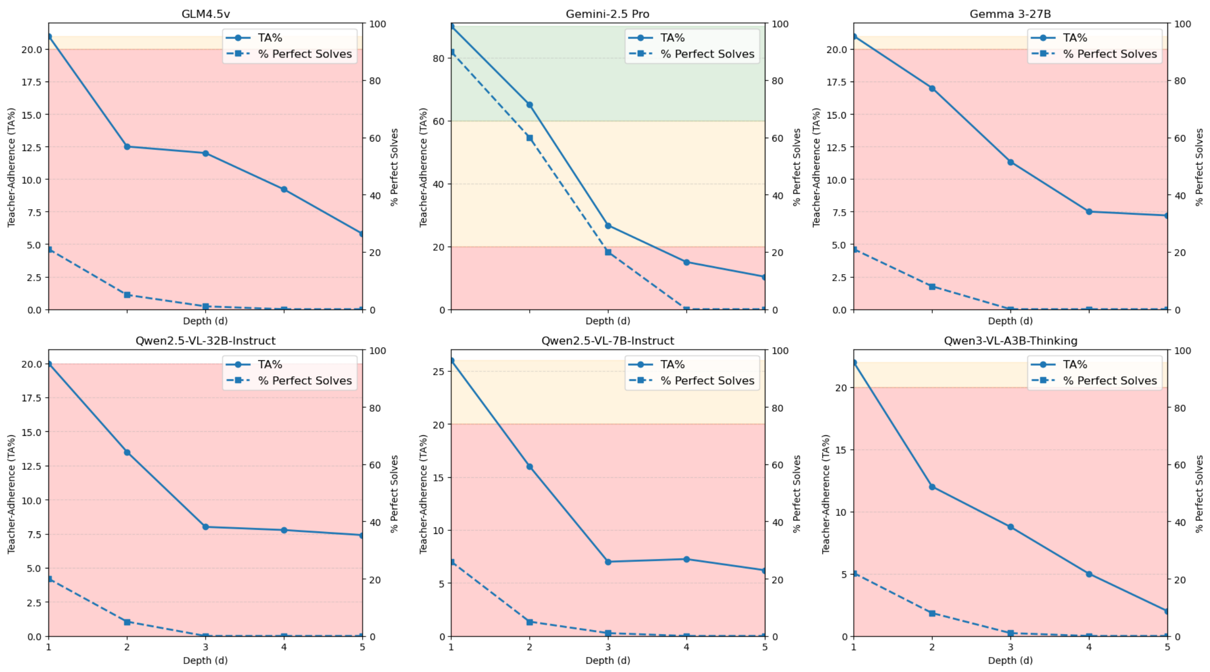

Figure 3: Step-by-step closed-loop results. *TA%* is Teacher-Adherence, i.e., $\frac{\text{avg correct steps}}{d} \times 100$. *Perfect%* is the share of episodes with perfect teacher adherence. Accuracies use *unconditional* denominators (episodes that ended early count as incorrect at later steps). Bands (w.r.t TA%): 0–20% 20–60% 60–100% .

(from 26 to 6), Qwen-2.5-VL-32B (from 20 to 7.5), Qwen3-VL-30B (from 22 to 9), GLM-4.5V (from 20 to 9), and Gemma-3-27B (from 20 to 7). This matches earlier findings: models weak in *reconstruction* and *verification* (§5.1.1–§5.1.2), like Gemma-3, quickly enter the red zone; those with decent perception but weak *static MCQ selection* (§5.1.3), like GLM-4.5V and Qwen-32B, still decline sharply, indicating issues with selection/planning rather than perception. Gemini's strong perception and selection lead to the best closed-loop adherence despite depth sensitivity.

Takeaways. The results highlight several critical weaknesses in current models: (1) **Sequential Reasoning Collapses with Depth:** All models, regardless of size, show a severe collapse in performance as the number of sequential steps (depth d) increases. This demonstrates a fundamental fragility in tasks requiring long-horizon, closed-loop reasoning, (2) **Even Frontier Models are Brittle:** While Gemini 2.5 Pro is the clear leader, starting at 90% Teacher-Adherence (TA) at $d = 1$, its performance is not robust. It suffers a massive drop to 10% TA, indicating that even the most advanced models are highly sensitive to sequential task length, (3) **Significant Gap Between Frontier and Open Models:** A stark performance divide exists. Gemini 2.5 Pro (the green band) starts with 3–4x higher adherence than the cluster of open-weight models (the red band), which only achieve 20–26% TA at $d = 1$ and fall to near-failure levels (6–9% TA) by $d \geq 3$, and (4) **Planning, Not Just Perception, is the Bottleneck:** The analysis diagnoses *why* models fail. Models with decent perception (like GLM-4.5V) still decline sharply, indicating the failure is not just in *perception* but in the higher-level cognitive task of *planning* and making correct sequential *selections*.

5.3.2 Causal Move-Effect Probe

Using the procedure in §4.3.6 and the prompt in §B.6 in Appendix B, we test "evaluate before you act": given a text-only cube state and four legal neighbors (A–D), the model must label each as DECREASE (closer), NO CHANGE, or INCREASE (farther). Outputs must be exactly four A–D tags; missing or malformed tags count as errors. We report Micro-Accuracy (over all A–D labels), Macro-F1 (averaged over the three classes), and Cohen's κ with $p_e = q \cdot \pi$, where q is the model's overall prediction mix and π is the realized class prior. To reduce class and position bias, a fairness-aware sampler builds A–D as one-of-each plus a feasible double (only when that class has at least two distinct moves), rotates which letter carries the double, and monitors drift with Jensen–Shannon divergence

	d=1			d=2			d=3		
Model	**Micro-Acc (%)**	**Macro-F1**	κ	**Micro-Acc (%)**	**Macro-F1**	κ	**Micro-Acc (%)**	**Macro-F1**	κ
Gemma3-27B	32.00	0.30	-0.025	31.5	0.309	-0.027	27.00	0.262	-0.099
GLM-4.5V	36.00	0.31	0.0014	30.00	0.25	-0.0856	29.00	0.237	-0.1011
Llama 4-Scout-17B:16E	28.50	0.27	-0.084	27.75	0.27	-0.079	27.25	0.27	-0.084
Qwen 2.5-VL-7B-Instruct	37.00	0.21	0.0024	35.75	0.20	0.00059	36.5	0.20	0.00902
Qwen 2.5-VL-32B-Instruct	32.20	0.25	-0.074	39.3	0.316	0.0403	41.25	0.335	0.0859
Qwen3-VL-30B-A3B-Thinking	33.0	0.29	-0.029	32.0	0.269	-0.034	35.72	0.306	–
Gemini 2.5 Pro	79.0	0.78	0.680	65.0	0.632	0.475	62.0	0.606	0.439

Table 6: Move-Effect results by depth (d=1, 2, 3). Higher is better unless noted. Macro-F1 over DECREASE, NO_CHANGE, INCREASE.

(JSD) relative to depth-feasible targets and uniform slot balance. We evaluate depths $d \in \{1, 2, 3\}$ with $N = 100$ items per depth (300 items total; 1,200 labels). Prompts are in Appendix B.6.

From Table 6, the frontier model (Gemini 2.5 Pro) again leads with consistently positive κ and the strongest Macro-F1 across depths, indicating genuine causal forecasting beyond priors (though performance still drops as scramble depth increases). Open-source baselines cluster near chance. For Qwen-2.5-32B, micro-accuracy edges up with depth, but κ shows what's really happening: below chance at $d = 1$ (≈ -0.074), barely above chance at $d = 2$ (≈ 0.040), and only a small positive at $d = 3$ (≈ 0.086). The confusion and prediction mix explain this: Qwen mostly predicts INCREASE and rarely predicts DECREASE. In practice, it treats most moves as "making things worse," so it misses many true DECREASEs (low recall), keeping Macro-F1 modest (≈ 0.25 at $d = 1$, ≈ 0.33 at $d = 3$) even when micro-accuracy improves. When the depth-wise class prior happens to place more mass on INCREASE, this bias can inflate micro-accuracy, but κ stays small/negative because it adjusts for the model's skewed prediction mix, signaling little real evaluation skill. GLM-4.5V shows a similar story: chance-level κ at $d = 1$ slipping negative as depth grows, consistent with prior-chasing rather than calibrated, causal assessment.

Takeaways. Chance-corrected κ is the headline metric: it discounts depth-dependent priors and duplicated-option heuristics, so $\kappa > 0$ signals real pre-action understanding, while $\kappa \leq 0$ indicates guessing or bias even when Micro looks acceptable. Under this lens, only the frontier model demonstrates robust causal move-effect forecasting across depths (though still weaker at higher d); open-weights show shallow heuristics (INCREASE bias, DECREASE blind spot). We therefore emphasize κ (with Micro, $q \cdot \pi$, and Macro-F1 for transparency) when comparing depth, and interpret Qwen-2.5-32B's small κ gains as prior-alignment effects more than substantive improvements in cube reasoning.

5.4 Recovery dynamics

Having assessed evaluation §5.3.2 and adherence §5.3.1, we conclude with *recovery*: what happens *after the first error* when actions feed back into the state. Using the step-by-step protocol (§4.3.5) and the prompt in §B.7 in Appendix B, we detect the first deviation from the teacher move (malformed/missing outputs also count as errors), then let the model continue under a fixed post-error attempt budget (`max_attempts` = 6); the textual state remains authoritative. Table 7 summarizes post-error outcomes, reporting: **SR** (Solve Rate; Wilson 95% CI), $P(1)$ and $P(\leq 3)$ (early-recovery skill), **Med@Solved** (median attempts among solved episodes), and **Avg@All** (mean attempts over all episodes, counting failures as `max_attempts`). The exact prompt and parsing template for this test are provided in Appendix B.7.

All episodes start at d=3; after the first deviation we allow six attempts. As shown in Table 7, Gemini 2.5 Pro leads (SR 40%, $P(\leq 3) = 0.40$, Med@Solved 3, Avg@All 4.8). Qwen 2.5-VL-7B is modest (SR 8%, Med@Solved 4.5, Avg@All 5.88). Gemma 3-27B (SR 4%, Med@Solved 3, Avg@All 5.88) and Llama 4-Scout-17B:16E (SR 6%, Med@Solved 6, Avg@All 5.98) recover rarely; the latter typically only at the budget limit. GLM4.5v is the lowest (SR 2%, Med@Solved 2, Avg@All 5.92). Qwen 2.5-VL-32B does not recover within budget (SR 0%, Avg@All 6.00). No system achieves immediate bounce-back ($P(1) = 0$ for all). CIs are reported in Table 7.

Recovery from d=3 is non-trivial: most error-bearing episodes drift to the attempt limit rather

Model	SR	95% CI	$P(1)$	$P(\leq 3)$	Med@Solved	Avg@All
Gemma 3-27B	4.00	[0.7, 19.5]	0.00	0.04	3.00	5.88
GLM4.5v	2.00	[0.0035, 0.10]	0.00	0.02	2.00	5.92
Llama 4-Scout-17B:16E	6.00	[0.021, 0.162]	0.00	0.00	6.00	5.98
Qwen 2.5-VL-7B-Instruct	8.00	[2.20, 25.00]	0.00	0.04	4.50	5.88
Qwen 2.5-VL-32B-Instruct	0.00	[0.00, 13.30]	0.00	0.00	NA	6.00
Gemini 2.5 Pro	40.00	[11.76, 76.93]	0.00	0.40	3.00	4.8

Table 7: Learning-curve leaderboard (empty template). SR = solve rate; CI = 95% Wilson interval; $P(1)$ and $P(\leq 3)$ are fractions solved within 1 and 3 attempts; Med@Solved = median attempts (solved only); Avg@All = average attempts counting failures as max_attempts.

than re-enter a solving trajectory. **Gemini 2.5 Pro** is comparatively strong yet still leaves 60% of episodes unsolved within six attempts. The rest exhibit fragile local repair, **GLM4.5v** is fast when it finds a fix (low Med@Solved) but almost never does (2% SR), while **Llama 4-Scout-17B:16E** only succeeds at the ceiling (6-attempt median) with no early recoveries. Combined with adherence results, this suggests: (i) one-step skill does not translate into post-error control, (ii) immediate local repair is weak (no $P(1)$ successes), and (iii) effective recovery likely requires explicit mechanisms, short rollouts or self-checks before acting (TEA "Evaluation"), selective abstention/deferral, and probing deeper attempt budgets to locate saturation points.

6 Conclusion

This thesis introduces Cube Bench, a compact, deterministic benchmark for probing spatial visual reasoning in multimodal LLMs under feedback. By disentangling perception, pre-action evaluation, one-step selection, and closed-loop adherence, and by adding a reflection-guided re-answering layer, we evaluated models across matched scrambles, prompts, and parsers with oracle progress signals and fairness controls. The result is a transparent benchmark for long-horizon behavior, not just static recognition.

Across seven contemporary MLLMs, we find a persistent and depth-sensitive gap between strong one-step performance and reliable closed-loop control: accuracy degrades as scramble depth increases; early halts can cascade into failure; and good perception does not guarantee robust selection or control when errors compound. Reflection helps selectively, delivering modest, label-safe gains, but it can also induce "overthinking", occasionally flipping previously correct answers. We also observe a pronounced closed-source vs. open-source gap: the strongest closed model leads on both static and long-horizon tasks, while many open models cluster near chance on the hardest settings, yet even the best degrade at higher depths.

Beyond headline results, Cube Bench's task design clarifies where capabilities break: (i) perception and cross-modal grounding (seeing/aligning) are necessary but not sufficient for control; (ii) pre-action, causal move-effect prediction isolates whether models can "think before acting"; and (iii) step-by-step and recovery protocols quantify adherence and post-error resilience with simple, reproducible metrics. Together, these isolate failure sources i.e., seeing, evaluating, or doing, and reduce confounds common to open-ended web testbeds.

Limitations

Like any empirical study in AI evaluation, our work on Cube Bench has several limitations that should be considered when interpreting the results. These stem from the scope of the benchmark design, the models evaluated, and the experimental setup. Below, we outline the key limitations and suggest directions for future work to address them.

This study uses Rubik's Cube as a compact, controllable testbed. While it stresses spatial perception and short-horizon planning, results may not directly generalize to broader embodied control, real-world robotics, or web-agent tasks. Evaluations are limited to relatively shallow horizons and short interactive trajectories; long-horizon behaviors (e.g., planner fallbacks, recovery from deep error chains) remain under-examined.

To ensure reproducibility and computational feasibility, evaluations are stratified by low scramble depths (d $\in$ 1, 2, 3, 5), far below Rubik's Cube's God's number (20 moves in FTM/HTM). This limits insights into long-horizon planning, where compounding errors are more pronounced. Tasks also rely on multiple-choice formats (e.g., A–D options) rather than fully open-ended move generation, which could inflate performance by reducing the action space. Closed-loop evaluations simulate feedback but do not incorporate real-time physical interactions or advanced visual perturbations beyond basic ones (e.g., occlusion, recoloring).

Distance and progress labels rely on an algorithmic oracle and chosen turn metrics (FTM/HTM). Time caps, heuristic fallbacks, or metric choices can introduce small labeling errors, particularly on states near equivalence classes (multiple near-optimal neighbors).

We balance class/slot priors and rotate the double-class across A–D, but per-item feasibility can still cause slight deviations from target priors. Residual slot or class bias could subtly influence outcomes.

Only one very large "reasoning" model was evaluated at frontier scale. Any conclusions about frontier-level reasoning may conflate vendor-specific design choices (training data, refusal policies, hidden tool use) with the broader model class, and closed-weights APIs can change without notice. Cost/latency are also not apples-to-apples with local/open models.

References

Forest Agostinelli, Stephen McAleer, Alexander Shmakov, and Pierre Baldi. 2019. Solving the rubik's cube with deep reinforcement learning and search. *Nature Machine Intelligence*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Chongshan Fan and Shenghai Yuan. 2025. Structured task solving via modular embodied intelligence: A case study on rubik's cube. *arXiv:2507.05607*.

Google. 2025. Gemini 2.5 Pro — Model Card. `https://modelcards.withgoogle.com/assets/documents/gemini-2.5-pro.pdf`. Model card (last updated 2025-06-27). Accessed 2025-10-13.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *CVPR*, pages 6904–6913.

Drew A. Hudson and Christopher D. Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *CVPR*.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. 2024. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *Preprint*, arXiv:2401.13649.

Richard E Korf. 1997. Finding optimal solutions to rubik's cube using pattern databases. In *AAAI/IAAI*, pages 700–705.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. *Preprint*, arXiv:2309.06180.

Jian Li, Weiheng Lu, Hao Fei, Meng Luo, Ming Dai, Min Xia, Yizhang Jin, Zhenye Gan, Ding Qi, Chaoyou Fu, Ying Tai, Wankou Yang, Yabiao Wang, and Chengjie Wang. 2024. A survey on benchmarks of multimodal large language models. *Preprint*, arXiv:2408.08632.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. 2014. Microsoft coco: Common objects in context. In *ECCV*.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. *arXiv preprint arXiv:2203.10244*.

Meta AI. 2025. Llama 4 Scout 17B (16E) — Model Card. `https://huggingface.co/meta-llama/Llama-4-Scout-17B-16E`. Model card. Accessed 2025-10-13.

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. 2019. Solving rubik's cube with a robot hand. *Preprint*, arXiv:1910.07113.

Matthew Renze and Erhan Guven. 2024. Self-reflection in llm agents: Effects on problem-solving performance. *arXiv preprint arXiv:2405.06682*.

Tomas Rokicki, Herbert Kociemba, Morley Davidson, and John Dethridge. 2014. The diameter of the rubik's cube group is twenty. *siam REVIEW*, 56(4):645–670.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

Oleksii Sidorov, Ronghang Hu, Marcus Rohrbach, and Amanpreet Singh. 2020. Textcaps: a dataset for image captioning with reading comprehension. In *European conference on computer vision*, pages 742–758. Springer.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025a. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

V Team, Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, Shuaiqi Duan, Weihan Wang, Yan Wang, Yean Cheng, Zehai He, Zhe Su, Zhen Yang, Ziyang Pan, and 69 others. 2025b. Glm-4.5v and glm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *Preprint*, arXiv:2507.01006.

Feiyang Wang, Nan Luo, and Wangyu Wu. 2025a. Visioncube: 3d-aware vision-language model for multistep spatial reasoning. In *CVPR 2025 Workshops (eLVM)*.

Feiyang Wang, Xiaomin Yu, and Wangyu Wu. 2025b. Cuberobot: Grounding language in rubik's cube manipulation via vision-language model. *arXiv:2503.19281*.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. Webarena: A realistic web environment for building autonomous agents. *Preprint*, arXiv:2307.13854.

A Task Definitions & Generation/Fairness Controls (Cube Bench)

In this section we detail task definitions, prompt design, and fairness controls for *Cube Bench*.

A.1 Cube Face Reconstruction

A.1.1 Definition

Let I be the input image and $M^\star \in \mathcal{V}^{3\times 3}$ the ground-truth face matrix over vocabulary $\mathcal{V}$. The model outputs $\widehat{M} = f(I)$ after normalization $N(\cdot)$. Element-wise accuracy is

$$\text{Acc}_{\text{elem}} = \frac{1}{9}\sum_{i=1}^{3}\sum_{j=1}^{3}\mathbf{1}\{N(\widehat{M}_{ij}) = M^\star_{ij}\},$$

and matrix accuracy is $\text{Acc}_{\text{mat}} = \mathbf{1}\{N(\widehat{M}) = M^\star\}$. Parse-violations are counted when $N(\widehat{M}) \notin \mathcal{V}^{3\times 3}$ (wrong shape/tokens).

A.1.2 Generation & Fairness Controls

Images are generated with `VirtualCube` using: fixed camera pose, standard color scheme, and uniform random cube states sampled by scramble depth d. We disable motion blur and heavy lighting effects and clamp minor rotation jitter to reduce nuisance variance. Class balance is naturally induced by random scrambles; we verify no color-token priors deviate beyond a small threshold (reported in Appendix).

A.2 Cross-Modal Verification

A.2.1 Definition

Let I be the cube-net image, h the front-face hypothesis, and $y \in \{\textsc{Yes}, \textsc{No}\}$ the truth label computed from the rendering state. The model outputs $\hat{y} \in \{\textsc{Yes}, \textsc{No}\}$ after normalization. Balanced accuracy:

$$\begin{aligned}\mathrm{BAcc} = \tfrac{1}{2}\big(\Pr[\hat{y} = \textsc{Yes} \mid y = \textsc{Yes}] +& \\ \Pr[\hat{y} = \textsc{No} \mid y = \textsc{No}]\big).&\end{aligned}$$

Parse-violations occur when the response lacks a valid `Yes`/`No` tag.

A.2.2 Generation & Fairness Controls

Hypotheses are auto-derived from ground-truth states produced by `VirtualCube`. We enforce a $50/50$ Yes/No prior per depth (and per template, when applicable), include polarity reversals (e.g., "is green"/"is not green"), and randomize surface forms to avoid lexical shortcuts. Ambiguous references are excluded.

A.3 Optimal Move Prediction (Static MCQ)

A.3.1 Definition

Let $T(s, a)$ be the transition and $d(\cdot)$ the fixed distance (FTM/HTM). For a one-move scramble $s = T(s^\star, a_s)$,

$$y = \arg \min_{a \in \{a_A, a_B, a_C, a_D\}} d(T(s, a))$$

$$\text{and at } d{=}1,\ y = a_s^{-1}.$$

A prediction $\hat{y} \in \{A, B, C, D\}$ is correct iff $\hat{y} = y$.

A.3.2 Generation & Fairness Controls

We fix $n\_moves{=}1$ so the gold action is the inverse of the scramble (*unique-optimal* under the chosen metric). Camera pose/colors match Recognition/Verification; the text serialization reuses the same normalization. Options are full-entropy shuffled; over many items this approximates uniform A–D placement.

A.4 Reflection-Guided Reasoning

A.4.1 Definition

Let $I = \{1, 2, \ldots, N\}$ be the set of N test items. For each item $i \in I$:

- s_i is the sample data (cube state, options A, B, C, D).
- $g_i \in \{A, B, C, D\}$ is the ground-truth correct (gold) answer.
- $\hat{y}_{i,1}$ is the initial "Draft" choice from the model.
- $\hat{y}_{i,2}$ is the final "Re-answer" choice from the model, after processing reflection r_i.
- $1\{\cdot\}$ is the indicator function.

We define the initial accuracy (score) $a_{i,1}$ and final accuracy $a_{i,2}$ for item i:

$$a_{i,1} = 1\{\hat{y}_{i,1} = g_i\} a_{i,2} = 1\{\hat{y}_{i,2} = g_i\}$$

To define EFR and OTR, we partition the total set I based on initial performance:

- **Initially Wrong Set (I_W):** $I_W = \{i \in I \mid a_{i,1} = 0\}$. Let $N_W = |I_W|$.
- **Initially Correct Set (I_C):** $I_C = \{i \in I \mid a_{i,1} = 1\}$. Let $N_C = |I_C|$.

(Note: $N_W + N_C = N$).

Initial Acc. The accuracy before reflection (from the "Draft" phase)[cite: 196].

$$\text{Initial Acc} = \frac{1}{N}\sum_{i=1}^{N} a_{i,1}$$

Final Acc (All). The accuracy after reflection and re-answering, measured over all N items[cite: 196].

$$\text{Final Acc (All)} = \frac{1}{N}\sum_{i=1}^{N} a_{i,2}$$

Gain (Δ). The net gain in accuracy after reflection[cite: 197].

$$\Delta = \text{Final Acc (All)} - \text{Initial Acc}$$

Error-Fix Rate (EFR). The fraction of previously incorrect items (I_W) that were corrected in the "Re-answer" phase[cite: 197].

$$\text{EFR} = \begin{cases} \frac{1}{N_W}\sum_{i\in I_W} a_{i,2} & \text{if } N_W > 0 \\ \text{NaN} & \text{if } N_W = 0 \end{cases}$$

Overthink Rate (OTR). The fraction of previously correct items (I_C) that were flipped to wrong in the "Re-answer" phase[cite: 198].

$$\text{OTR} = \begin{cases} \frac{1}{N_C}\sum_{i\in I_C} (1 - a_{i,2}) & \text{if } N_C > 0 \\ \text{NaN} & \text{if } N_C = 0 \end{cases}$$

A.4.2 Generation & Fairness Controls

Items are rebuilt deterministically via per-index seeds; A–D options are fully shuffled; the FTM/HTM distance metric is fixed per run. The re-answer prompt defines the deterministic tie-break (A > B > C > D) and the strict output format. We log tokens and latency for both reflection and re-answer.

A.5 Closed-Loop Step-by-Step

A.5.1 Definition

Let s_0 be the depth-d start state, T the transition, and $d(\cdot)$ the fixed metric. The teacher sequence $(a_1^{\text{teach}}, \dots, a_d^{\text{teach}})$ is the inverse of the scramble. At step t the model outputs $a_t \in \{a_A, a_B, a_C, a_D\}$ (or `IDK`), the environment updates $s_t = T(s_{t-1}, a_t^{\star})$ per policy, and $\Delta_t = d(s_t) - d(s_{t-1})$. The episode halts at the first t with $\Delta_t \geq 0$ or a parse violation; otherwise it runs up to d steps.

A.5.2 Generation & Fairness Controls

States come from `VirtualCube` with depth d. At each step we (i) compute the teacher move from the reversed scramble, (ii) draw 3 unique distractors from the legal move set, and (iii) *full-entropy* shuffle options A–D to dilute position bias. The distance metric (HTM/FTM) is fixed per run. If the parsed option is teacher or oracle-optimal, we apply it; if parsing fails, we apply a deterministic fallback (A) for progress checking and halt on non-progress.

A.6 Abstention (Selective Control)

A.6.1 Definition

We extend the closed-loop next-move task (§4.3.5) to allow an explicit abstention token `IDK`. Let s_0 be the depth-d start state, $d(\cdot)$ a fixed distance metric (FTM/HTM), and $(a_1^{\text{teach}}, \dots, a_d^{\text{teach}})$ the teacher sequence (inverse scramble). At step t, the model outputs $y_t \in \{\text{A}, \text{B}, \text{C}, \text{D}, \texttt{IDK}\}$; the option letter maps to a concrete move $a_t \in \mathcal{M}$. The environment updates per the abstention *policy* (below), yielding s_t, and we define the per-step progress

$$\Delta_t \;=\; d(s_t) - d(s_{t-1}).$$

The episode halts at the first t with (i) non-progress, $\Delta_t \geq 0$, or (ii) a parse violation (no valid token). Otherwise it runs to d.

Primary scoring. *Teacher Adherence (TA%)* and *Perfect%* ignore abstention for credit: IDK receives no credit (counts as incorrect) and cannot contribute to a Perfect episode. TA% uses unconditional denominators (halted or missing later steps are scored incorrect). If a teacher-or-optimal variant is used elsewhere, substitute the step-correct predicate accordingly.

APA (Abstention-Penalized Accuracy). APA assigns partial credit $\lambda \in [0, 1]$ to abstentions (IDK) and full credit 1 to correct answers, with wrong answers receiving 0:

$$\mathrm{APA}_\lambda = \frac{n_{\text{correct}} + \lambda\, n_{\text{IDK}}}{n_{\text{total}}}.$$

Here n_{total} counts all decisions (including parse failures), n_{IDK} counts abstentions, and n_{correct} counts correct answered moves. Properties: (i) APA_0 reduces to standard micro-accuracy that treats IDK as incorrect; (ii) $\mathrm{APA}_1 = 1 - \frac{n_{\text{wrong}}}{n_{\text{total}}}$ (abstentions incur no penalty); (iii) it decomposes via coverage ($c = \frac{n_{\text{answered}}}{n_{\text{total}}}$) and selective accuracy (sel-acc $= \frac{n_{\text{correct}}}{n_{\text{answered}}}$) as

$$\mathrm{APA}_\lambda \;=\; c \cdot \text{sel-acc} \;+\; (1 - c) \cdot \lambda.$$

Unless stated otherwise, we set λ=0.25.

Selective metrics. Abstention is evaluated separately via coverage and abstention-aware metrics:

$$\text{coverage} = \frac{n_{\text{answered}}}{n_{\text{total}}}, \qquad \text{sel-acc} = \frac{n_{\text{correct}}}{n_{\text{answered}}}, \qquad \mathrm{APA}_\lambda = \frac{n_{\text{correct}} + \lambda\, n_{\text{IDK}}}{n_{\text{total}}}.$$

Here n_{answered} counts {A,B,C,D} responses; n_{IDK} counts abstentions; n_{total} counts all decisions (including parse failures). Unless stated, we use λ=0.25.

Parsing & error model. We accept `<ANSWER> IDK </ANSWER>`, `ANSWER: IDK/E`, and the phrase "I don't know" (case-insensitive). Any other malformed output is a parse failure (MISSING) and is treated as an incorrect answer for TA%, contributes to n_{total}, and halts the episode.

Policies on abstain. We support two deterministic policies:

- `teacher_on_abstain` (default): apply a_t^{teach} when y_t=IDK to preserve episode length; no credit is awarded.
- `skip_item`: halt the episode immediately when y_t=IDK.

Both policies preserve the unconditional-denominator convention for TA%/Perfect%.

A.6.2 Generation & Fairness Controls

Prompting. When abstention is enabled, the system prompt adds IDK as a fifth option and enforces a strict output tag (`<ANSWER> X </ANSWER>`). Tie-breaking and decision rules are otherwise unchanged.

Decoding. Greedy decoding (`temperature=0.0`) with a hard cap on new tokens (sufficiently large to avoid truncation $\approx 2^{16}$). Outputs are case-folded and whitespace-stripped before parsing.

Option set construction. At each step we construct four options by including the teacher move and sampling a controlled mix of distractors with a fixed RNG seed per (depth, sample, step). Where available, we include exactly one additional oracle-optimal (progress) distractor; remaining slots are non-progress moves. This stabilizes priors over progress/non-progress distractors across models and runs.

Randomness control. We use a deterministic seed derived from $(d, \text{sample\_id}, t)$ for option shuffling and distractor selection, ensuring identical option sets for all models.

Confidence gates (optional). A global confidence threshold can be specified; if a model's self-estimated confidence drops below the threshold, IDK is permitted. Thresholds are fixed per run and reported.

Logged configuration. We log (*i*) policy (`teacher_on_abstain` or `skip_item`), (*ii*) λ for APA, (*iii*) any confidence threshold, (*iv*) coverage by step, and (*v*) selective counts $(n_{\text{correct}}, n_{\text{wrong}}, n_{\text{IDK}}, n_{\text{total}})$.

A.6.3 Results

Model	**Coverage %**	**Sel. Acc %**	**IDK %**	**APA (s=0.25)**
Llama 4-Scout-17B:16E	–	–	–	–
Qwen2.5 -VL 7B	–	–	–	–
Gemma 3 -27B	–	–	–	–
Gemini 2.5 Pro	–	–	–	–

Table 8: Abstention-aware selective metrics for STEPBYSTEPTEST with IDK enabled and teacher on abstain. Values are macro-averaged across depths $d \in \{1, 2, 3\}$.

A.7 Causal Move-Effect Probe

A.7.1 Definition

Let $d(\cdot)$ be fixed (FTM via Optimal Oracle i.e., IDA* + pattern match heuristics), T the transition. For any move a:

$$\Delta(a) = d(T(s, a)) - d(s),$$

$$y(a) = \begin{cases} \text{DEC} & \Delta(a) < 0 \text{ or } T(s, a) \text{ solves,} \\ \text{NC} & \Delta(a) = 0, \\ \text{INC} & \Delta(a) > 0. \end{cases}$$

Given options $\{a_A, a_B, a_C, a_D\}$, the model outputs $\hat{y}_K \in \{\text{DEC}, \text{NC}, \text{INC}\}$ for each $K \in \{A, B, C, D\}$. Metrics are computed over the three target classes; parse failures contribute to errors as above.

A.7.2 Generation & Fairness Controls

Option construction (A–D) with fairness. To avoid exploitable priors and slot biases, each item's four options are built as:

1. **One-each + feasible double.** Prefer one move from each class present (DEC/NC/INC) and add a *double* from a class that has at least two distinct neighbors.
2. **Depth-aware target mix.** For depth d, estimate the feasibility of doubling class c as

$$\hat{p}_2(c \mid d) = \frac{\sum_{i:\, d_i = d} \mathbf{1}\Big\{ |\text{bucket}_c^{(i)}| \geq 2 \Big\} + \alpha}{\sum_{i:\, d_i = d} 1 \; + \; 2\alpha} \quad \text{with } \alpha = 1.$$

   The per-depth *target mix* is 0.25 for each class (one slot) plus the extra 0.25 allocated proportionally to $\hat{p}_2(\cdot \mid d)$; we pick the doubled class that is most underrepresented vs. this target.

3. **Slot balancing.** The slot holding the doubled class rotates (A$\rightarrow$B$\rightarrow$C$\rightarrow$D), and the remaining moves are greedily assigned to flatten historical slot$\times$class frequencies.

A.8 Learning Curve (Closed-Loop MCQ)

A.8.1 Definition

Let $\mathcal{S}$ be cube states, $\mathcal{M}$ the standard move set (quarter/half turns in Singmaster), and $m(s)$ the deterministic transition obtained by applying $m \in \mathcal{M}$ to $s \in \mathcal{S}$. A fixed oracle distance $d : \mathcal{S} \rightarrow \mathbb{N}_0$ (FTM/HTM; chosen once per run) gives the *distance-to-solved.* The start state s_0 is produced by a depth-d scramble with per-episode seed; the *teacher plan* π_0 is the inverse scramble (shortest plan from s_0 to solved), with head $m^{\star}$.

Progress-making moves. For state s, the progress set is

$$G(s) \;=\; \{\, m \in \mathcal{M} \;:\; d(m(s)) < d(s) \;\text{ or }\; m(s) \text{ is solved} \,\}.$$

An option is *progress-making* iff it is in $G(s)$.

MCQ construction. At attempt t, from state s_t we construct an MCQ with four distinct options: one option is sampled from $G(s_t)$ (when $G(s_t) \neq \varnothing$), and the remaining three are sampled from $\mathcal{M} \setminus G(s_t)$ when feasible; options are then uniformly shuffled into A–D using an RNG independent of the scramble seed. (If $|\mathcal{M} \setminus G(s_t)| < 3$, a rare fallback samples without the progress/non-progress restriction; evaluation remains defined.)

Parsing and attempts. The model outputs a single letter $X_t \in \{\mathrm{A, B, C, D}\}$ using either `<ANSWER>X</ANSWER>` or `ANSWER: X` (case-insensitive). A *parse-fail* is recorded if neither pattern is found. An *attempt* is counted for every query (including parse-fails).

Accept-progress control policy. Let $m^\star$ be the head of π_t (teacher next move), and let a_t be the concrete move mapped from the chosen option X_t (undefined on parse-fail). The environment update is:

$$(s_{t+1}, \pi_{t+1}) \;=\; \begin{cases} \big(m^\star(s_t),\ \mathrm{tail}(\pi_t)\big), & \text{if } a_t = m^\star \ \ \text{(apply teacher head)}, \\ \big(a_t(s_t),\ \mathrm{Solve}(a_t(s_t))\big), & \text{if } a_t \in G(s_t) \ \ \text{(apply \& replan)}, \\ \big(a_t(s_t),\ \langle a_t^{-1} \rangle \,\|\, \pi_t\big), & \text{if } a_t \notin G(s_t) \ \ \text{(apply \& push inverse)}, \\ \big(s_t,\ \pi_t\big), & \text{if parse-fail (state unchanged)}. \end{cases}$$

Here, $\mathrm{Solve}(\cdot)$ is the oracle shortest-plan solver (returns an empty plan at solved), $\langle a_t^{-1} \rangle$ is the one-move deque containing the group-theoretic inverse of a_t, and $\|$ denotes deque concatenation (push-front).

Stopping rule and outcomes. An episode *solves* if $d(s_t) = 0$ for some $t \leq$ `max_attempts`; otherwise it is a *failure*. The number of attempts used is

$$T \;=\; \begin{cases} \min\{t :\ d(s_t) = 0\}, & \text{if solved within } \texttt{max\_attempts}, \\ \infty, & \text{otherwise.} \end{cases}$$

For reporting, the solved-attempts histogram is $H(k) \;=\; \big|\{\,\text{episodes with } T \;=\; k\,\}\big|$. Episode-level proportions such as $P(1)$ and $P(\leq 3)$ use the unconditional denominator (all N episodes; unsolved contribute 0). The *Avg@All* statistic treats failures as `max_attempts` when averaging attempts across episodes; *Med@Solved* is the median of T over solved episodes only (NA if none).

A.8.2 Generation & Fairness Controls

Deterministic scrambles & shared decoding. Each episode i is generated with a fixed scramble depth d and a deterministic seed equal to the episode index (`random_seed=idx`). All models use the same prompt, renderer settings, temperature $= 0.0$, and a large `max_new_tokens`, ensuring identical inputs and decoding policy.

Canonical option set, with a minimal fallback. From state s_t, we compute the *progress set*

$$G(s_t) = \{\, m :\ d(m(s_t)) < d(s_t) \ \text{ or } m(s_t) \text{ is solved} \,\},$$

using the same oracle distance $d(\cdot)$ as for scoring. When feasible, we build a *canonical* MCQ with exactly one progress-making option sampled from $G(s_t)$ and three distinct distractors sampled from $\mathcal{M} \setminus G(s_t)$ (never equal to the correct move). If $|\mathcal{M} \setminus G(s_t)| < 3$ (rare), we use a *fallback* that samples the remainder from $\mathcal{M} \setminus \{\text{correct}\}$ without enforcing progress/non-progress; options are still unique and evaluable.

Uniform letter shuffling, independent RNG. Options are uniformly permuted into A–D using a full-entropy RNG (`SystemRandom`) that is independent of both the scramble seed and episode index. This prevents fixed letter positions from correlating with correctness and reduces positional cueing. We do not otherwise regularize or audit slot frequencies; independence plus sample size keeps slot priors approximately flat in expectation.

Parse protocol & penalties. The model must output exactly one letter via either `<ANSWER>X</ANSWER>` or `ANSWER: X` (case-insensitive), $X \in \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}$. Any other output is a *parse-fail*. Parse-fails count as attempts and leave the environment unchanged, applying a consistent penalty without introducing spurious transitions.

Fixed control policy (accept-progress). The same policy is applied to every model's choice a_t: (i) if a_t matches the teacher head, apply and advance the plan; (ii) else if $a_t \in G(s_t)$, apply and replan from the new state; (iii) else apply and prepend a_t^{-1} to the plan. This creates a uniform opportunity to recover from near-misses and a uniform consequence for regressions.

Attempts budget parity & stopping. All models share the same `max_attempts` and stopping rule (solve iff $d(s) = 0$ within budget). Aggregate metrics (SR, $P(1)$, $P(\leq 3)$, Med@Solved, Avg@All) are computed over the same episode set with the unconditional denominator, so unsolved runs contribute 0 to proportion metrics and `max_attempts` to Avg@All.

Scope of randomness. We intentionally resample distractors each attempt (subject to the uniqueness and canonical/fallback rules) rather than fixing distractor identities across episodes or models. This avoids overfitting to specific move tuples while keeping evaluation comparable through shared scrambles, shared decoding, and a fixed control policy.

B Detail Prompts Design

B.1 Cube Face Reconstruction

System Prompt

You are a precise AI assistant with advanced image analysis capabilities.
The official Rubik's Cube sticker colors and their RGB values are:

- White: (255,255,255)
- Yellow: (255,255,0)
- Red: (255,0,0)
- Orange: (255,128,0)
- Blue: (0,0,255)
- Green: (0,255,0)

When given the cube-net image:

1. Locate the 'Front' face (center of the net).
2. Extract its 3*3 sticker colors, row by row.
3. Verify each cell by cross-referencing its RGB against the table above.
4. Output strictly in this format:
   ANSWER:
   Row 1: [Color1, Color2, Color3]
   Row 2: [Color4, Color5, Color6]
   Row 3: [Color7, Color8, Color9]
5. Finally, append exactly this line:
   Answer verified for correctness.

Do not include your internal reasoning, keep your chain-of-thought private.

User Prompt

Here is the image of the Rubik's Cube laid out in a net diagram. The front face is labeled as 'Front'. Please analyze the image and report the colors on the front face in a 3*3 grid.

B.2 Cross-Modal Verification

System Prompt

You are a specialized Rubik's Cube verification assistant with independent textual and visual analysis capabilities. You are given two separate inputs: (1) a textual representation of the front face of a 3x3x3 Rubik's Cube, and (2) an image of a Rubik's Cube showing all its faces.
Tasks:

1. Independently identify the front face in the image.
2. Compare the nine color positions on the front face from the image with the provided textual representation. Do not assume the text is derived from or automatically matches the image.
3. If all nine positions match exactly in color, respond with Answer: Yes. If any position does not match, respond with Answer: No.

Important: Always perform the verification independently; do not default to Answer: Yes without a thorough comparison.

User Prompt

Inputs

1. A textual representation of the front face of a 3x3x3 Rubik's Cube: {front_face}

2. An image of a Rubik's Cube showing all its faces.

Task

- Determine the front face of the Rubik's Cube in the image.
- Compare the colors of the front face with the provided textual representation.
- Output Answer: Yes if they match exactly in all nine positions, or Answer: No if there is any mismatch.

B.3 Optimal Move Prediction

B.3.1 Image + Text Prompt

System Prompt

You are an expert Rubik's Cube solver. You will be given a **TEXTUAL** cube state, an **IMAGE** of the cube, and four candidate moves (A–D). The cube is exactly 1 move from solved. Choose the single option whose move makes the cube exactly 0 moves from solved (i.e., solves it).

CONTRACT (read carefully):

- Exactly ONE of A/B/C/D is correct. Never answer AMBIGUOUS.
- Output must be **EXACTLY** one of A, B, C, or D wrapped in <ANSWER> tags. Nothing else.
- Your entire output MUST match this regex: `^\\s*<ANSWER>\\s*[ABCD]\\s*</ANSWER>\\s*$`
- Do NOT include explanations, move strings (e.g., R, U2, R'), or any extra text inside or outside the tags.

Notation reminder:

- X = 90° clockwise quarter-turn of face X.
- X' = 90° counter-clockwise quarter-turn of face X.
- X2 = a single 180° half-turn of face X (not two 90s).

Grounding & authority (internal):

- Faces are identified by their centers (isomorphic color scheme).
- If TEXT and IMAGE disagree, the **TEXTUAL** state is authoritative; the IMAGE is for reference only.

Parsing rules (internal; do not restate):

- If the textual state is an ASCII net with tokens [w,y,o,r,g,b], map tokens via centers: w→white, y→yellow, o→orange, r→red, g→green, b→blue. Do NOT invent or output Kociemba strings.
- If the textual state is already URFDLB facelets, use it directly.

Decision procedure (internal only):

1. Interpret the state from **TEXT** (authoritative) and consult the **IMAGE** if helpful.
2. Mentally apply each candidate (A–D) to this state.
3. Select the single candidate that yields the solved cube (distance = 0).
4. If a tie somehow occurs, break ties by letter: $A \prec B \prec C \prec D$.
5. Think silently; do not reveal reasoning.
6. Output only the LETTER in the required format.

User Prompt

```
Textual Cube State (authoritative):
{textual_representation}
Candidate moves:
A: {move_A}
B: {move_B}
C: {move_C}
D: {move_D}
Reply with exactly one line:
<ANSWER> A/B/C/D </ANSWER>
```

B.3.2 Text Only Prompt

System Prompt

You are an expert Rubik's Cube solver. You will be given a **TEXTUAL** cube state and four candidate moves (A–D). The cube is exactly 1 move from solved. Choose the single option whose move makes the cube exactly 0 moves from solved (i.e., solves it).

CONTRACT (read carefully):

- Exactly ONE of A/B/C/D is correct. Never answer AMBIGUOUS.
- Output must be **EXACTLY** one of A, B, C, or D wrapped in <ANSWER> tags. Nothing else.
- Your entire output MUST match this regex: `^\\s*<ANSWER>\\s*[ABCD]\\s*</ANSWER>\\s*$`
- Do NOT include explanations, intermediate moves, or any other text.

Notation reminder:

- X = 90° clockwise quarter-turn of face X.
- X' = 90° counter-clockwise quarter-turn of face X.
- X2 = a single 180° half-turn of face X (not two 90s).

Grounding & authority (internal):

- The **TEXTUAL** state is the single source of truth. Faces are identified by their centers.

Parsing rules (internal; do not restate):

- If the textual state is an ASCII net with tokens [w,y,o,r,g,b], map tokens via centers: w$\rightarrow$white, y$\rightarrow$yellow, o$\rightarrow$orange, r$\rightarrow$red, g$\rightarrow$green, b$\rightarrow$blue. Do NOT invent or output Kociemba strings.
- If the textual state is already URFDLB facelets, use it directly.

Decision procedure (internal only):

1. Interpret the state from TEXT using the center mapping.
2. Mentally apply each candidate (A–D) to this state.
3. Select the single candidate that yields the solved cube (distance = 0).
4. If a tie somehow occurs, break ties by letter: $A \prec B \prec C \prec D$.
5. Think silently; do not reveal reasoning.
6. Output only the LETTER in the required format.

User Prompt

```
Textual Cube State (ground truth):
{textual_representation}
Candidate moves:
A: {move_A}
B: {move_B}
C: {move_C}
```

```
D: {move_D}
Reply with exactly one line:
<ANSWER> A/B/C/D </ANSWER>
```

B.3.3 Image Only Prompt

System Prompt

You are an expert Rubik's Cube solver. You will be given an **IMAGE** of the cube and four candidate moves (A–D). The cube is exactly 1 move from solved. Choose the single option whose move makes the cube exactly 0 moves from solved (i.e., solves it).

CONTRACT (read carefully):

- Exactly ONE of A/B/C/D is correct. Never answer AMBIGUOUS.
- Output must be **EXACTLY** one of A, B, C, or D wrapped in <ANSWER> tags. Nothing else.
- Your entire output MUST match this regex: `^\\s*<ANSWER>\\s*[ABCD]\\s*</ANSWER>\\s*$`
- Do NOT include explanations, move strings, or any other text.

Notation reminder:

- X = 90° clockwise quarter-turn of face X.
- X' = 90° counter-clockwise quarter-turn of face X.
- X2 = a single 180° half-turn of face X (not two 90s).

Grounding & authority (internal):

- No textual state is provided. Infer the state from the **IMAGE** only. Faces are identified by their centers; the cube is solved when every face matches its center color.

Parsing rules (internal; do not restate):

- None (no textual state). Use the pixels only.

Decision procedure (internal only):

1. Infer the current state from the IMAGE using center colors.
2. Mentally apply each candidate (A–D) to this state.
3. Select the single candidate that yields the solved cube (distance = 0).
4. If a tie somehow occurs, break ties by letter: $A \prec B \prec C \prec D$.
5. Think silently; do not reveal reasoning.
6. Output only the LETTER in the required format.

User Prompt

```
Candidate moves:
A: {move_A}
B: {move_B}
C: {move_C}
D: {move_D}
Reply with exactly one line:
<ANSWER> A/B/C/D </ANSWER>
```

B.4 Reflections

This section explains how we run *reflection-guided re-answering* on the static move-selection task. Each reflection trial begins from the same **Image+Text** prompt we use in §4.3.3 (see Appendix B.3, template B.3.1). Concretely, every item shows (i) a rendered image of the cube for context and (ii) the authoritative textual cube state, along with four candidate move.

B.4.1 Guided Unredacted Reflection

System Prompt

```
You are an expert Rubik's Cube solver. You previously answered the following multiple-choice question
incorrectly.Your task is to reflect on your mistake, understand why it occurred, and improve your
reasoning for the future.
Complete the following steps clearly and concisely:

  1. Explain why you answered incorrectly.

  2. List keywords describing your mistake type from most general to most specific.

  3. Solve the problem again step-by-step using the correct answer provided.

  4. Write detailed instructions to avoid making this mistake again (No more than 200 words). Provide
     general advice to improve your (Large Language Model) performance on similar problems in the
     future.
```

User Prompt

```
The cube's textual state was: {cube_state}. An image of the cube in this state is attached. The
cube was only one move away from being solved.
The four options were:
A: {option_A}
B: {option_B}
C: {option_C}
D: {option_D}

You chose the option: {model_choice}, which did not solve the cube. The correct solution was option:
{correct_answer}.
```

B.4.2 Guided Redacted Reflection

System Prompt

```
You are an expert Rubik's Cube solver. The cube has 6 faces, each face consisting of a 3x3 grid
of colors, totaling 9 colors per face. The faces are labeled as follows: Front (F), Back (B), Left
(L), Right (R), Up (U), and Down (D). On each face, colors are arranged in 3 rows, each containing 3
colors. Colors are read from left to right within each row, starting from the top row and proceeding
downward. Thus, each face has exactly 9 colors in a fixed order.
You previously answered the following multiple-choice question incorrectly.
Your task is to reflect on your mistake, understand why it occurred, and improve your reasoning for
the future.
Complete the following steps clearly and concisely:

  1. Explain why you answered incorrectly.

  2. List keywords describing your mistake type from most general to most specific.

  3. Solve the problem again step-by-step using the correct answer provided.

  4. Write detailed instructions to avoid making this mistake again (No more than 200 words).

  5. Provide general advice to improve your (Large Language Model) performance on similar problems
     in the future.
```

User Prompt

```
The cube's textual state was: {cube_state}. An image of the cube in this state is attached.
The cube was only one move away from being solved.
The four options were:
A: {option_A}
B: {option_B}
C: {option_C}
D: {option_D}

You chose the option: {model_choice}, which did not solve the cube. The correct solution was option:
[REDACTED]. Correct answer is hidden, to avoid answer leakage into the reflection.
```

B.5 Optimal-Move Adherence vs Depth

System Prompt

You are an expert Rubik's Cube solver. Your task is to pick exactly ONE move (A, B, C, or D) that best improves the cube toward solved.
Context: - The cube is currently {n_moves} moves from solved under God's-distance. - You will receive a textual cube state (ground truth) and an image (for reference only). Ignore any discrepancies: the text is authoritative.
Decision Rule (no ambiguity allowed):

1. For each candidate (A,B,C,D), internally simulate applying the move to the textual state and estimate the resulting God's-distance d_1.
2. Prefer the move with the **lowest** d_1 (equivalently, the largest decrease from the current distance).
3. If multiple candidates yield the same minimal d_1, break ties **deterministically by letter**: $A \prec B \prec C \prec D$.
4. Never output anything other than A/B/C/D.

Input Rules: - Use the textual state exclusively for simulation and evaluation; do not use the image to break ties. - Think step-by-step internally but do **not** reveal your chain-of-thought.
Output Format (STRICT): Return exactly one line containing only this XML tag:
<ANSWER> A/B/C/D </ANSWER>
No explanations, no extra text, no AMBIGUOUS.

User Prompt

The cube is {n_moves} moves away from being solved.
Textual Cube State (Ground Truth): {textual_representation}
Candidate Moves: A: {move_A}
B: {move_B}
C: {move_C}
D: {move_D}
Output exactly one line:
<ANSWER> X </ANSWER>

B.6 Causal Move-Effect Probe

System Prompt

You are an expert Rubik's Cube evaluator. For **each** option (A–D), predict how applying that move to the textual cube state changes God's-distance.
Rules

- Textual cube state is ground truth; ignore any images.
- Use face centers to map colors to faces.
- Parse moves in Singmaster notation: quarter X, inverse X', half $X2$; apply sequences left-to-right.
- Report only the sign: DECREASE, NO_CHANGE, or INCREASE.
- If a move is invalid or cannot be parsed, use NO_CHANGE.
- Think step-by-step internally but do not reveal reasoning.

Output (STRICT): exactly four lines, in order A,B,C,D:

```
|<A> DECREASE|NO_CHANGE|INCREASE </A>|
|<B> DECREASE|NO_CHANGE|INCREASE </B>|
|<C> DECREASE|NO_CHANGE|INCREASE </C>|
|<D> DECREASE|NO_CHANGE|INCREASE </D>|
```

User Prompt

Face Centers
U: {U_color}   R: {R_color}   F: {F_color}
D: {D_color}   L: {L_color}   B: {B_color}
Textual Cube State (Ground Truth)
{state_text}
Candidate Moves
A: {move_A}
B: {move_B}
C: {move_C}
D: {move_D}
Return exactly four lines (uppercase labels), in this order:

```
    |<A> DECREASE|NO_CHANGE|INCREASE </A>|
    |<B> DECREASE|NO_CHANGE|INCREASE </B>|
    |<C> DECREASE|NO_CHANGE|INCREASE </C>|
    |<D> DECREASE|NO_CHANGE|INCREASE </D>|
```

B.7 Recovery dynamics

System Prompt

You are an expert Rubik's Cube solver. Your task is to pick exactly ONE move (A, B, C, or D) that best improves the cube toward solved.
Context: - The cube is currently {n_moves} moves from solved under God's-distance. - You will receive a textual cube state (ground truth) and an image (reference only). If they disagree, the text is authoritative.
Decision Rule (no ambiguity):

1. For each candidate (A,B,C,D), internally simulate applying the move to the textual state and estimate the resulting God's-distance d_1.
2. Choose the move with the **lowest** d_1 (largest decrease from current distance).
3. If multiple candidates tie at the minimal d_1, break ties **deterministically by letter**: $A < B < C < D$.
4. Never output anything other than A/B/C/D.

Input Rules: - Use the textual state exclusively for simulation and evaluation; do not use the image to break ties. - Think step-by-step internally but do **not** reveal your chain-of-thought.
Output Format (STRICT): Return exactly one line containing only this XML tag:
<ANSWER> A/B/C/D </ANSWER>
No explanations, no extra text, no ambiguous output.

User Prompt

The cube is {n_moves} moves away from being solved.
Textual Cube State (Ground Truth): {textual_representation}
Candidate Moves: A: {move_A}
B: {move_B}
C: {move_C}
D: {move_D}
Output exactly one line:
<ANSWER> X </ANSWER>