A. Data Curation Prompts

A.1. LLM-i - Reformulation Suitability Screening

To operationalize stage (b) in our pipeline (Figure 2), we employ *GPT-5 [34]* as a binary judge that decides whether a raw problem can be *reformulated into a geometric construction task*. The model returns a single token (`true`/`false`) without explanations, enabling high-throughput triage before manual verification. Figure 6 shows the exact prompt template used in our experiments.

LLM-i: Reformulation Suitability Judge

Role. You are a *Geometric Construction Feasibility Judge*. Given a math problem (text + optional diagram), your task is to **only decide** whether the problem is **suitable to be reformulated as a geometric construction problem**. **Output only one word:** `true` **or** `false`. Do not output any explanation, punctuation, or spaces.

Criteria

All of the following must hold (otherwise output `false`):

- **Geometric objects are explicit:** the statement involves planar geometric objects/relations (points, lines, angles, circles, triangles/polygons, parallels/perpendiculars, angle bisectors, midpoints, centers, tangents, intersections, etc.).
- **Convertible into a construction goal:** there is a clear object/position/figure/relation to be *constructed*, not merely a proof or numeric calculation.
- **Reasonable determinacy:** under common assumptions (not to scale; free choice of references), the target has a clearly attainable solution (unique or finitely many), not severely underdetermined.

Output `false` if the problem is purely algebraic/calculus/probability/statistics/equation solving or analytic calculations with no geometric construction goal.

Output requirements (must be strictly followed)

- Only output: `true` or `false`.
- Do not output any other characters, spaces, or line breaks.

Inputs

- Problem text: {`PROBLEM_TEXT`}
- Problem diagram: {`OPTIONAL_DIAGRAM`}

Figure 6. Prompt template used by **GPT-5 [34]** in pipeline stage (b) to screen whether a raw problem can be reformulated as a geometric construction task. The template enforces a single-token decision (`true`/`false`) with explicit positive criteria and disqualifiers, enabling high-throughput automatic triage prior to human verification.

A.2. LLM-ii - Geometric Problem Generation

At stage (d) of the GGBench pipeline (Figure 2), **LLM-ii** rewrites pre-screened geometry questions into formal *geometric construction problems*. The model transforms general geometry descriptions into construction-oriented formulations that specify given elements, target relations, and construction goals—without revealing solutions or GeoGebra code. It also assigns metadata including difficulty level, construction type, and core geometric skills. Figure 7 shows the prompt template used in this stage (abridged with ellipses for brevity).

LLM-ii (Stage (d)): Geometric Construction Problem Generation

Task: Given a set of geometric information, *rewrite it as a construction problem*. The problem must state *what to construct* and must not include the solution.

Include the following fields:

1. Problem Difficulty: choose `Easy`, `Medium`, or `Hard`.

2. Problem Type: `Straightedge-and-compass`, `Analytic construction`, or `Geometric transformation construction`.

3. Core Skills: list key geometric principles (e.g., similarity, rotation, tangency, circumcenter, centroid, ...).

4. Modality: specify `Multimodal` (with figure) or `Text-based`.

5. Initial GeoGebra Information: define only the *given* objects (points, lines, segments, ...) as the starting state.

Output Format Example (abridged):

Problem Title (Medium)

Constructing a Triangle with a 30° Angle

Problem Type: Straightedge-and-compass construction

Problem Description:

Given a line segment AB, construct $\triangle ABC$ such that $\angle ACB = 30°$. Begin by specifying the required givens and the target object(s) to be constructed without revealing the solution steps.

Core Skills: Equilateral Triangle, Circumcenter, Inscribed Angle Theorem, Circle Construction

Modality: Multimodal

Initial GeoGebra Information (example)

Only the initial objects to seed the construction (no solution code).

GeoGebra Code

```
```geogebra
ShowAxes(false)
ShowGrid(false)
A = Point({2, 3})
B = Point({6, 3})
segAB = Segment(A, B)
```

Figure 7. Prompt template for **LLM-ii** at stage (d), aligned with the LLM-iii box in layout and syntax.
```

A.3. LLM-iii - Answer Generation

Based on the adapted problems produced in stage (d), **LLM-iii** generates the final *geometric construction answer* at stage (e) (Figure 2). The model is instructed to output stepwise reasoning aligned with executable GeoGebra code; per-step snapshots are required for visual traceability. Figure 8 shows the prompt template (abridged with ellipses for brevity).

LLM-iii (Stage (e)): Geometric Construction *Answer* Generation

Task: Given a geometric construction problem, generate a **step-by-step construction answer** including: (1) stepwise construction from primitives (points, lines, circles) to the final figure; (2) *valid GeoGebra code for each step*; (3) *reasoning for each step* (e.g., perpendicular bisector, symmetry, tangency, ...); (4) per-step snapshots showing figure evolution; (5) clear labels/annotations for all objects. **GeoGebra code requirements** (important): No comments and no blank lines; strictly follow GeoGebra syntax; define objects before use; only official commands; simple, non-self-intersecting polygons; consistent, descriptive names (`PointA`, `LineAB`, ...). **Constraints & notes** (abridged): Verify commands exist and are valid; initialize points/lines/angles before dependent objects; avoid CW/CCW mistakes; ensure angle/ratio sources are stated; check intersections and angles; use consistent notation; angles displayed should be integers when required; ... **Checklist**: (i) no undefined objects; (ii) no spelling/syntax errors; (iii) strict GeoGebra compliance. If errors exist, revise. **Output structure**: Numbered steps; each step provides *Method*, *Principle*, and a GeoGebra code block. *Example (abridged)*: **Step 1: Construct auxiliary circles to locate the circumcenter**
Method: draw c_1 centered at A with radius AB, and c_2 centered at B with radius BA.
Principle: intersections of c_1 and c_2 are equidistant from A and B. **GeoGebra Code**

```
geogebra
  ShowAxes(false)
  ShowGrid(false)
  A = Point({2, 3})
  B = Point({6, 3})
  c1 = Circle(A, B)
  c2 = Circle(B, A)
```

Continue stepwise until completion. Provide a final, single, self-contained GeoGebra code block that runs without errors.

Figure 8. Prompt template for **LLM-iii** at stage (e) in Figure 2, used to generate the complete *geometric construction answer* (stepwise reasoning aligned with executable GeoGebra). Ellipses (...) indicate omitted boilerplate for brevity.

A.4. LLM-iv Prompt for Final Automatic Screening

At stage (f) of the GGBench pipeline (Figure 2), **LLM-iv** acts as the final *automatic construction judge*. It reviews the generated construction tasks and corresponding solutions produced by earlier stages, verifying whether each sample meets all structural, syntactic, and geometric requirements before inclusion in the final dataset. The model outputs a binary score (`1` = pass, `0` = fail) together with diagnostic reasoning in JSON format. Figure 9 presents the abridged prompt template used for this purpose.

LLM-iv (Stage (f)): Construction Compliance Verification Prompt

Task: Given a geometric construction task, verify whether a model-generated submission satisfies all structural, syntactic, and field-level requirements. The input includes the construction problem, initial diagram code, and a stepwise solution with corresponding GeoGebra commands.
Evaluation requirements: (1) *Information extraction*: extract title, difficulty, problem type, description, core skills, modality, initial code block, and construction steps; (2) *Problem reasonableness*: target must be clear; constraints must be sufficient; construction must be geometrically feasible; (3) *Structural and field compliance*: allowed types/modalities only; English-only text; a single initial GeoGebra code block (no comments or blank lines); no undefined or malformed objects; (4) *Stepwise consistency*: each step must explain "what + why"; commands must match text; logical progression must be followed.
Scoring rubric: (1) reasonable construction and full compliance. (2) any mandatory rule violated.
Output structure: Provide a brief analysis, followed by a strict binary score:
[Scoring Rationale]: explain format, compliance, and syntax checks (e.g., English-only, valid commands, no undefined objects, no blank lines, consistent use of ShowAxes, ZoomIn, etc.)
[Score]: X point(s)
[JSON]:

```
{"answer_score": [[X]]}
```

End with: "In summary, this submission deserves X point(s)."

Figure 9. Prompt template for **LLM-iv** at stage (f) in Figure 2, used to verify structural integrity, field compliance, and command-level correctness in GeoGebra-based construction scripts. The prompt outputs a strict 0/1 score based on format and execution validity.

A.5. Human Checking & Filtering Standards

At stage (b) of the GGBench pipeline (Figure 2), human annotators perform the first-level screening to ensure that all retained samples are geometrically meaningful, visually grounded, and executable when applicable. This step complements the automated filtering conducted by LLM-i and forms a high-quality foundation for downstream prompt adaptation (stage (d)). The manual filtering process follows three evaluation dimensions as outlined below.

(1) Geometric relevance. Each problem must explicitly present geometric semantics or spatial relationships. Accepted problems typically contain characteristic geometric terminology such as "point," "line," "angle," "circle," "triangle," "parallel," "perpendicular," "radius," or "area." Problems without explicit geometric keywords may still qualify if their content clearly implies geometric reasoning, such as construction, angle relations, or area derivation. In contrast, problems focused purely on algebraic manipulation, logical deduction, or other non-spatial reasoning are excluded.

(2) Image–text correspondence. Each problem must include an accompanying geometric diagram or schematic that accurately reflects the textual conditions and spatial configuration. The image should support reasoning or serve as visual evidence for the described construction. Samples are removed if their images lack geometric characteristics, contradict the textual description, or contain unrelated visual content such as natural photographs, tables, or decorative graphics.

(3) Code executability and consistency. For problems containing executable code (e.g., GeoGebra scripts, or LaTeX drawing commands), annotators manually verify that all code runs correctly in standard environments. The generated output must align with the geometric meaning and textual description. Any instance with syntax errors, runtime failures, or inconsistencies between the produced figure and the problem statement is excluded from the dataset.

This threefold manual screening protocol ensures that all retained items exhibit clear geometric validity, accurate multimodal alignment, and functional code integrity—providing a robust and verifiable foundation for subsequent automated evaluation and dataset finalization.

A.6. Expert Validation Standards

At stage (f) of the GGBench pipeline (Figure 2), expert reviewers conduct the final validation to ensure the scientific soundness, structural completeness, and executable reliability of all retained samples. This stage serves as the last human-in-the-loop checkpoint before dataset finalization, guaranteeing that every item meets research-grade quality requirements. The expert validation process is guided by the following five criteria.

(1) Logical rigor of reasoning structure. Experts examine whether each problem demonstrates a complete and coherent reasoning chain that conforms to mathematical logic and geometric principles. Samples with missing premises, invalid inferential steps, or conclusions that cannot be justified by the given conditions are considered invalid.

(2) Consistency and correctness of drawing procedures. Problems involving geometric constructions, function plotting, or visual rendering are manually reviewed to ensure that all drawing instructions and steps follow established geometric conventions. The generated figures must correspond precisely to the textual descriptions. Samples are rejected if inconsistencies occur between text and graphics, if proportions are distorted, or if essential annotations—such as angles, lengths, or intersection points—are missing.

(3) Accuracy and reproducibility of results. Experts verify that each final outcome, including computed values, geometric figures, or rendered visualizations, is both correct and reproducible. This involves cross-checking the stated results with mathematical reasoning and the actual rendering outputs. Any instance of computational error, logical inconsistency, or semantic mismatch with the problem intent is disqualified.

(4) Clarity and formal correctness of presentation. Each problem is reviewed for linguistic clarity, symbolic correctness, and graphical readability. The use of mathematical symbols and notations must follow formal conventions, and the visual layout must support interpretability in research contexts. Problems containing ambiguous expressions, inconsistent symbols, or irregular formatting are revised or excluded.

(5) Global coherence and research alignment. Finally, experts assess whether the overall logic, visual presentation, and code execution results are semantically aligned. Only problems that demonstrate complete coherence across content, structure, and visualization—and that fall within the intended scope of multimodal geometric reasoning—are retained in the finalized GGBench dataset.

This expert validation phase ensures that the final dataset satisfies high standards of mathematical soundness, visual consistency, and executable reliability, enabling robust benchmarking for future research in multimodal geometric reasoning and construction.

B. Evaluation

B.1. Experimental Setups

To ensure deterministic outputs, we fix the temperature at 0.0 during inferencet that eliminates stochasticity and enforces consistent step-by-step reasoning. All code generation adheres strictly to the GeoGebra command syntax to ensure executability and structural correctness. Inference is parallelized for efficiency. All models are evaluated under the same settings with prompts and pipelines.

B.2. VLM-T Scoring Prompt

Figure 10 illustrates the evaluation prompt used to assess VLM-T performance, one of the core dimensions in GGBench. This component focuses on evaluating the textual reasoning in solving geometric construction problems.

LLM-text-score: Evaluation Prompt for VLM-T

Role: You are an evaluator of the "text steps" for geometric multimodal constructions.
Input:
- Problem description
- Reference answer: standard construction steps
- Model's answer: model's construction steps

Evaluation Criteria:
- **Completeness**: step order is clear; no skipped logic; dependency relations are explicit
- **Accuracy**: key geometric operations are correctly described
- **Geometric Consistency**: operations comply with geometric constraints
- **Reference Equivalence**: alternative but equivalent strategies are acceptable

Scoring Rubric (1–5):
- 5: Complete, rigorous, and equivalent to the reference answer
- 4: Minor omissions or differences that do not affect reproducibility
- 3: Most key points covered but with notable omissions or errors
- 2: Numerous logical flaws or incoherent reasoning
- 1: Invalid, unstructured, or irreproducible construction

Output Requirements: Return a single Arabic numeral (1–5) only. No text, punctuation, or justification.

Figure 10. Prompt used to compute VLM-T scores. The evaluator assesses textual construction steps on logical, geometric, and structural grounds, returning a scalar score between 1 and 5.

The evaluation process compares a model's generated construction steps against expert-written reference steps, emphasizing logical completeness, geometric correctness, and stepwise coherence. A rubric-based scoring from 1 to 5 is adopted, where higher scores indicate precise and faithful reasoning. To maintain scoring consistency, the evaluator is instructed to ignore stylistic variations and accept logically equivalent strategies that achieve the same geometric objectives. The final score serves as a proxy for the model's ability to translate visual reasoning tasks into accurate symbolic instructions, and directly contributes to the VLM-T metric reported in Table 4.

B.3. VLM-I-Res Scoring Prompt

Figure 11 shows the evaluation template used to obtain VLM-I-Res scores in GGBench.

LLM-image-score: Evaluation Prompt for VLM-I-Res

Task: You are a geometric image consistency evaluator.
Input:
- Reference answer image
- Model's final image

Evaluation Criteria:
- Element completeness: whether the key geometric elements required by the problem are included (points, line segments, circles/auxiliary circles, points of tangency, tangents, annotations/labels, etc.).
- Topology & constraints: whether relative positions and geometric relationships (perpendicular, parallel, tangency, concyclicity/collinearity, angle/proportional relationships, etc.) are correct.
- Visual tolerance: do not penalize non-critical style differences such as line width, color, fonts; do not penalize translation/rotation/uniform scaling/mirroring (similarity transforms) unless explicitly forbidden by the problem.
- Overall judgment: prioritize the correctness of geometric topology and constraints; LPIPS is only a reference for perceptual similarity—if there is a conflict, geometric consistency takes precedence.

Scoring Rubric (1–5):
- 5: Elements, topology, and constraints are highly consistent
- 4: Basically consistent, only slight positional/style deviations
- 3: Mostly consistent; elements are complete but there are several minor errors/deviations
- 2: Only partially consistent; key elements are missing or constraint errors are obvious
- 1: Inconsistent with the problem intent or missing key elements (such as circles/auxiliary circles/points of tangency/tangents), or the image is unusable

Boundary Handling:
- Missing/corrupted/unreadable image $\rightarrow$ 1
- Text only with no image $\rightarrow$ 1

Output Requirements: Output only a single Arabic numeral (1–5) as the final result, with no other text, punctuation, or spaces.

Figure 11. Prompt used to compute the VLM–I-Res score. The VLM judge compares the model's final rendered diagram against the reference image and assigns a 1–5 rating based on geometric consistency and constraint satisfaction.

This prompt guides GPT-4o to act as an impartial ge-

ometric evaluator that judges the consistency between a model's *final rendered diagram* and the *reference solution*. The design emphasizes geometric reasoning over superficial appearance, ensuring that perceptual similarity does not override structural correctness.

VLM-judge: Prompt for Step Accuracy Evaluation (Step)

Task: Given a geometric construction problem and a rendered long image showing the step-by-step solution, judge whether each step image strictly matches its corresponding textual instruction in terms of naming, geometric structure, and positioning.

Scoring Rules:

- **5 points**: The image and text match perfectly, including naming, topological relationships, and quantity, with no extraneous or missing elements.
- **4 points**: The image and text are mostly consistent, with only very slight visual deviations that do not affect understanding (e.g., a point label is slightly offset but named correctly).
- **3 points**: The image meets the main structural requirements but has more than one minor error (e.g., confusing names, incorrect position of auxiliary points).
- **2 points**: The image and text are difficult to correspond one-to-one; there are critical naming errors, connection errors, or missing elements.
- **1 point**: The overall structure of the image is incorrect, naming is chaotic, it is completely disconnected from the text, or the image is irrelevant/completely incorrect.

Mandatory Clauses:

- If naming errors or omissions make it impossible to uniquely identify the target point/line, the score must not exceed **1 point**.
- If the text requires connecting a specific pair of points, but the endpoints in the image cannot be reasonably matched (wrong position **and** wrong name), score **1 point**.
- If a name is slightly misplaced but still identifiable in context, a score of **2 points** may be given.

Figure 12. Prompt used for Step Accuracy (Step) evaluation by the judge model. This criterion assesses alignment between visual steps and symbolic instructions.

B.4. VLM-I-Mid Scoring Prompt

To evaluate the consistency and correctness of intermediate construction steps, we employ two complementary visual-judging criteria: **Step Accuracy (Step)** and **Process Consistency (Consist.)**. Each is independently scored by a frozen VLM (GPT-4o) using fixed prompts. The final $\texttt{VLM\{I}_{\texttt{Mid}}$ score is computed as the mean of the two ratings and rescaled to $[0, 100]$.

Prompt for Step Accuracy Figure 12 presents the evaluation prompt used for measuring Step Accuracy. This metric quantifies the degree of alignment between each textual instruction and its corresponding geometric subfigure in the rendered construction sequence. The prompt guides the judge model to assess whether visual elements—including object naming, positional structure, and geometric relationships—faithfully match the symbolic description at each reasoning step. It ensures that the geometric process remains verifiable and syntactically grounded at the step level.

VLM-judge: Prompt for Process Consistency Evaluation (Consist.)

Task: Evaluate whether each step reasonably builds upon the previous step's figure in a cumulative manner, avoiding skipped steps, missing procedures, or logical gaps.

Scoring Rules:

- **5 points**: Completely based on the previous step, structure is fully inherited, and the new construction is correct and complete.
- **4 points**: Mostly coherent, with only minor issues in inheritance clarity or naming deviations.
- **3 points**: The structural relationship between steps is roughly preserved, but there are missing key construction marks, logical skips, or incorrect step order.
- **2 points**: Most constructions are not inherited, or construction continues on an incorrect structure; the logical chain is unclear.
- **1 point**: Steps are severely incoherent, as if they are independent drawings, with no evolutionary relationship visible, or the figures in each step are completely unrelated, forming a broken chain.

Mandatory Clauses:

- If the long image contains only one single figure, the score for this dimension must not exceed **1 point**.
- If intermediate figures are missing or replaced by error messages, the score must be **1 point**.

Figure 13. Prompt used for Process Consistency (Consist.) evaluation by the judge model. This criterion checks visual and logical coherence across sequential construction steps.

Prompt for Process Consistency Figure 13 illustrates the prompt designed for evaluating Process Consistency. This indicator measures whether each step in a geometric construction logically inherits and extends the previous figure, maintaining coherent spatial and structural progression. The prompt enforces sequential reasoning validity, penalizing disjointed or skipped steps, reflecting the model's capacity for stable geometric evolution.

B.5. Prompt for Model Inference

To ensure fair and reproducible evaluation across heterogeneous model families, we design two complementary prompt templates corresponding to the two evaluation tracks in GGBench: (1) **LLM/LRM code–based prompting** for models that generate explicit *GeoGebra programs*, and (2) **UMM image–based prompting** for models that directly produce diagrams or visual construction steps. Both prompts are carefully standardized to (i) unify task semantics, (ii) minimize ambiguity in geometric intent, and (iii) encourage interpretable, stepwise reasoning.

Model Inference Prompt for LLMs/LRMs

Task:
According to the given drawing topic and image, create a construction using GeoGebra language. If the problem requires a ruler-and-compass construction, retain all auxiliary traces. Output step-by-step drawing instructions, geometric principles, and executable GeoGebra code.

Code Generation Rules:

1. Follow GeoGebra syntax strictly (capitalize commands; no underscores).
2. Use `Point({x, y})` for points and `Vector((x, y))` or `Vector(Point1, Point2)` for vectors.
3. Retain all auxiliary elements in the final figure.
4. Use a single coordinate system; no comments or blank lines.
5. Only static drawings are allowed (no animations).

Output Example:
Problem Title: Construct the incircle of triangle ABC.
Each step includes **Construction**, **Principle**, and **GeoGebra Code** sections.
GeoGebra Code (abridged):

```
ShowAxes(false)
ShowGrid(false)
A = Point({1, 5})
B = Point({0, 1})
C = Point({7, 2})
...
bisA = AngleBisector(C, A, B)
bisB = AngleBisector(A, B, C)
I = Intersect(bisA, bisB)
Circle(I, A)
...
ZoomIn(0, 0, 8, 6)
```

Figure 14. Prompt used for LLMs/LRMs.

Figure 14 shows the template used to elicit explicit *text–to–code* reasoning. Each instance provides a natural–language construction topic and, when available, a reference diagram. Strict code–generation rules are enforced to guarantee syntactic validity and deterministic rendering—e.g., capitalized command names, no underscores, static drawings only, and a single coordinate frame. For ruler–and–compass problems, auxiliary traces must be preserved to make the reasoning chain fully inspectable. An abridged output example is provided in the prompt box of Fig. 14. This design encourages models to *reason, formalize, and construct* in a single pipeline, directly linking linguistic reasoning to executable geometry.

Model Inference Prompt for UMMs

Task:
According to the given drawing topic, create a drawing that meets the requirements. If the problem requires the use of a ruler and compass construction, please retain the traces of ruler and compass construction. Finally, only output the drawing steps.
Drawing Step Generation Rules:

1. Strictly follow the description of the drawing steps and clearly explain each step.
2. The output should only include detailed step descriptions, without any code.
3. The steps should describe the drawing process, ensuring that each description leads to the creation of the corresponding image.
4. The steps should be clear and concise, avoiding dynamic effects or code.

Output Example:
Problem Title: Given two circles of different radii c_1 (center O_1, radius r_1) and c_2 (center O_2, radius r_2), which do not intersect. Using only ruler and compass construction, draw the two external tangents to these two circles.
Drawing (abridged):

1. Construct the angle bisector of ($\angle$ BAC).
2. Construct the angle bisector of ($\angle$ ABC).

Principle: The incenter of a triangle (the center of the incircle) is the intersection of its three angle bisectors. According to the angle bisector theorem, any point on an angle bisector is equidistant from the two sides of the angle. Therefore, the intersection of the two angle bisectors will be equidistant from the three sides of the triangle.

Figure 15. Prompt used for UMMs (e.g., Janus, Bagel, Qwen-Image, Nano Banana, Seeddream).

Figure 15 illustrates the template for end-to-end multimodal systems that generate drawings directly from text. These models receive the same construction topic but are instructed to produce only natural–language descriptions of drawing steps without emitting code. The rules emphasize clarity and procedural completeness: each step must describe a concrete geometric operation that would yield the corresponding figure; dynamic or stylistic embellishments are prohibited. If the problem involves ruler-and-compass construction, all traces should be retained in the generated image. This prompt design isolates the model's visual *generative reasoning* ability—its capacity to transform textual geometric constraints into coherent visual sequences—without relying on symbolic code execution.

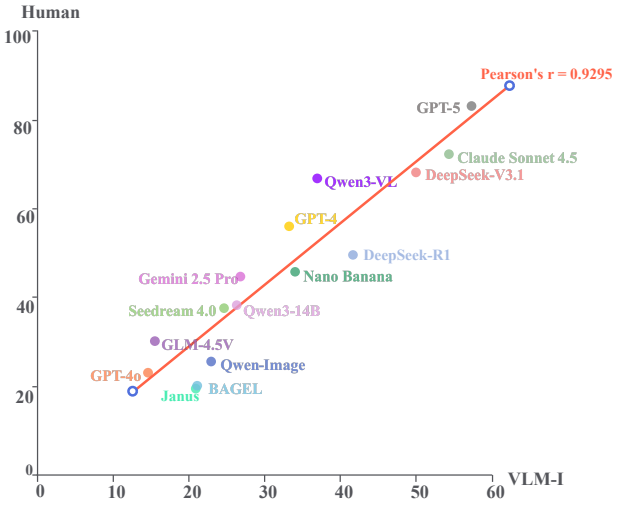

Figure 16. Correlation between VLM-I and human evaluation.

B.6. Human Evaluation

To verify the reliability and consistency of our automatic evaluation metric (VLM-I), we conducted a systematic human evaluation involving 3 domain experts with backgrounds in mathematics education and geometric modeling. For each model, 100 samples were randomly selected and independently scored to ensure representativeness and fairness.

Prior to the evaluation, all experts underwent standardized training to ensure familiarity with the assessment protocol, scoring rubric, and exemplar responses. A double-blind review mechanism was adopted to minimize subjective bias:

- Each sample was independently evaluated by two experts;
- If the score difference exceeded 1 point, a third expert acted as arbiter;
- The final score was the average of the three expert ratings.

Evaluation Criteria. Experts evaluated each model's generation by jointly considering its textual construction steps and the rendered figure. Scores were assigned based on three criteria: logical soundness of the geometric reasoning, accuracy of the drawing process, and overall consistency between text and image. A 1–5 scale was used:

- **5 – Perfect Match**: Both the textual steps and diagram are complete, logically sound, and closely match the reference. All geometric primitives (e.g., circles, auxiliary lines, intersections, tangents) are correctly included and rendered.
- **4 – Mostly Correct**: Minor deviations in position, style, or step detail, but the core reasoning and visual fidelity are intact.
- **3 – Partially Correct**: The key reasoning path is preserved, but notable omissions or inconsistencies exist (e.g., missing constructions, misaligned diagrams, or loose logical steps).
- **2 – Major Errors**: Either the text or diagram deviates significantly from the problem intent, omitting several key steps or introducing logical flaws.
- **1 – Invalid Result**: Severe inconsistency or mismatch between text and diagram, with critical geometric elements missing or misinterpreted.

Correlation Analysis. To assess alignment between human judgments and automated scores, we computed the Pearson correlation coefficient (r) between VLM-I scores and average human ratings. As shown in Figure 16, the correlation is exceptionally high ($r = 0.9295$), confirming that our automated metric closely reflects human-perceived quality and faithfully captures model performance trends.

C. Baselines

Track A: End-to-end UMMs These systems take natural-language specifications and output diagram images step-by-step. We include: *Qwen-Image* [56], *Seedream 4.0* [38], *Janus* [55], *BAGEL* [10], and *Nano Banana* [11]. Among them, *Qwen-Image* and *Seedream 4.0* produce only final images without stepwise explanations, whereas the other models generate multi-stage visual outputs for step evaluation.
Track B: Planning → Code → Render These models first produce a *textual plan* and then emit *executable GeoGebra code* to render into images: *GPT-4o* [19], *GLM-4.5V* [17], *Qwen3VL-235B-A22B* [62], *GPT-4* [1], *GPT-5* [34], *Claude Sonnet 4.5* [3], *Gemini-2.5-Pro* [9], *DeepSeek-R1* [24], *DeepSeek-V3.1* [24], and *Qwen3-14B* [62].

D. Case Study

D.1. Examples across Difficulty Levels

To illustrate the range of geometric reasoning tasks in GG-Bench, we present three representative examples drawn from the benchmark, each aligned with a specific difficulty level. Figures 17–19 visualize problems designed to probe distinct levels of spatial abstraction, procedural depth, and symbolic control.

Figure 17 corresponds to an entry-level task grounded in elementary circle geometry. The problem involves constructing chords that pass through given points while remaining parallel to two specified diameters. Despite its simplicity, solving the problem requires maintaining directional consistency, identifying appropriate auxiliary constructions, and accurately positioning endpoints within a bounded coordinate frame. It targets the model's ability to handle localized spatial constraints and execute basic primitives such as 'Segment', 'ParallelLine', and 'Point'.

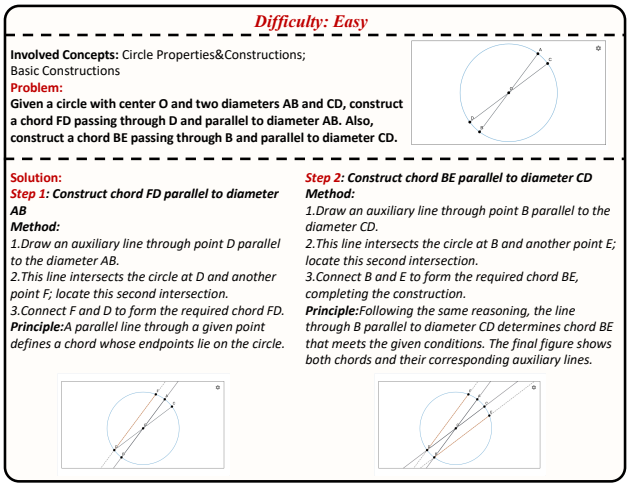

Figure 17. Example of an easy-level task.

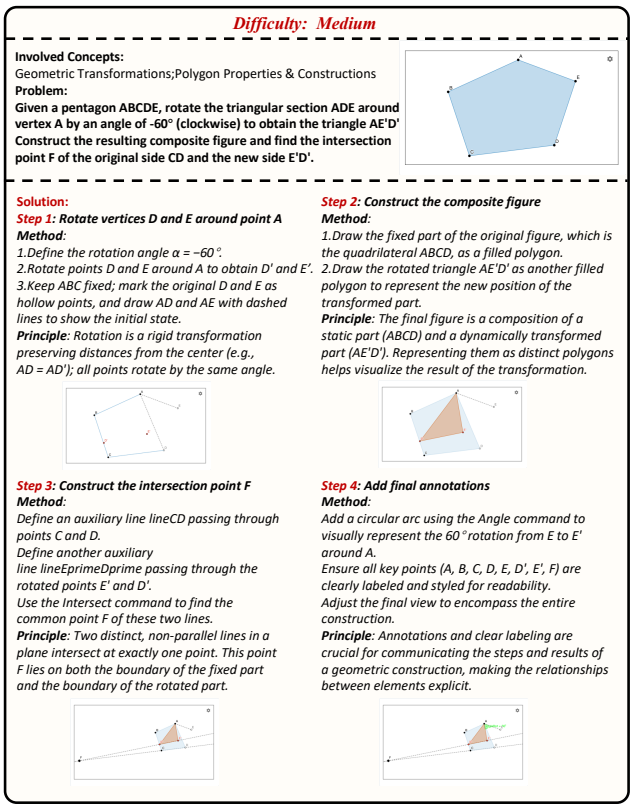

Figure 18. Example of a medium-level task.

Figure 18 illustrates a transformation-based task involving a regular pentagon and an inscribed triangle. The objective is to rotate the triangle about one vertex and identify the resulting intersections with the original figure. This setup tests the model's understanding of rigid motions, label consistency, and intersection logic. It requires chaining multiple dependencies while preserving geometric invariants such as distance and angle, and it penalizes any deviation from transformation semantics or object reuse policies.

Figure 19 represents a high-complexity construction spanning multiple geometric phases. Starting from a tangent circle, the problem unfolds into a sequence of regular polygon constructions—inscribing a square, then an octagon, followed by a 16-gon. The solution involves hierarchical decomposition, recursive angle bisection, and repeated application of symmetric placement rules. Success depends on long-horizon reasoning, internal consistency across stages, and robustness in symbolic command generation under rigid geometric constraints.

D.2. Additional Error Cases

This appendix presents representative qualitative error cases to illustrate typical reasoning failures across different model architectures. We divide these errors into four core categories, each reflecting a distinct type of breakdown in the construction process. All examples are selected from the GGBench benchmark and cover both Unified Multimodal Models (UMMs), which generate diagrams directly from text, and LLMs/LRMs, which follow a structured plan–code–render workflow. Figures 20–23 correspond to the four error types respectively.

1. Structural and Contextual Errors This type of failure refers to the disruption of consistency across the generation process, including input understanding, intermediate representation, and final visual output. It is primarily observed in End-to-end UMMs, which often struggle to maintain stable structural alignment over multiple steps. As shown in Figure 20, all four examples belong to this category. BAGEL starts with a reasonable diagonal construction of a kite but soon inserts irrelevant labels, unexplained formulas, and misaligned annotations. The resulting image lacks coherence, with its later elements no longer traceable to the geometric intent. Nano Banana displays duplicated vertex names and introduces disjointed lines that contradict the target rectangle EFGH, suggesting confusion in spatial tracking and relational preservation. Seedream 4.0 outputs a quadrilateral whose diagonals are not perpendicular as required, failing the core symmetry constraint. Qwen-Image presents a final image where key vertices are mislabeled and symmetries are broken, revealing a lack of global semantic control. These cases illustrate how UMMs often degrade structurally when tasked with complex multi-step visual reasoning.

2. Geometric Logic and Step-Level Errors This category includes errors that violate fundamental geometric principles or disrupt logical flow during construction. It spans both UMMs and LLMs/LRMs, as shown in Figure 21. Janus, the UMM in this figure, incorrectly substitutes a

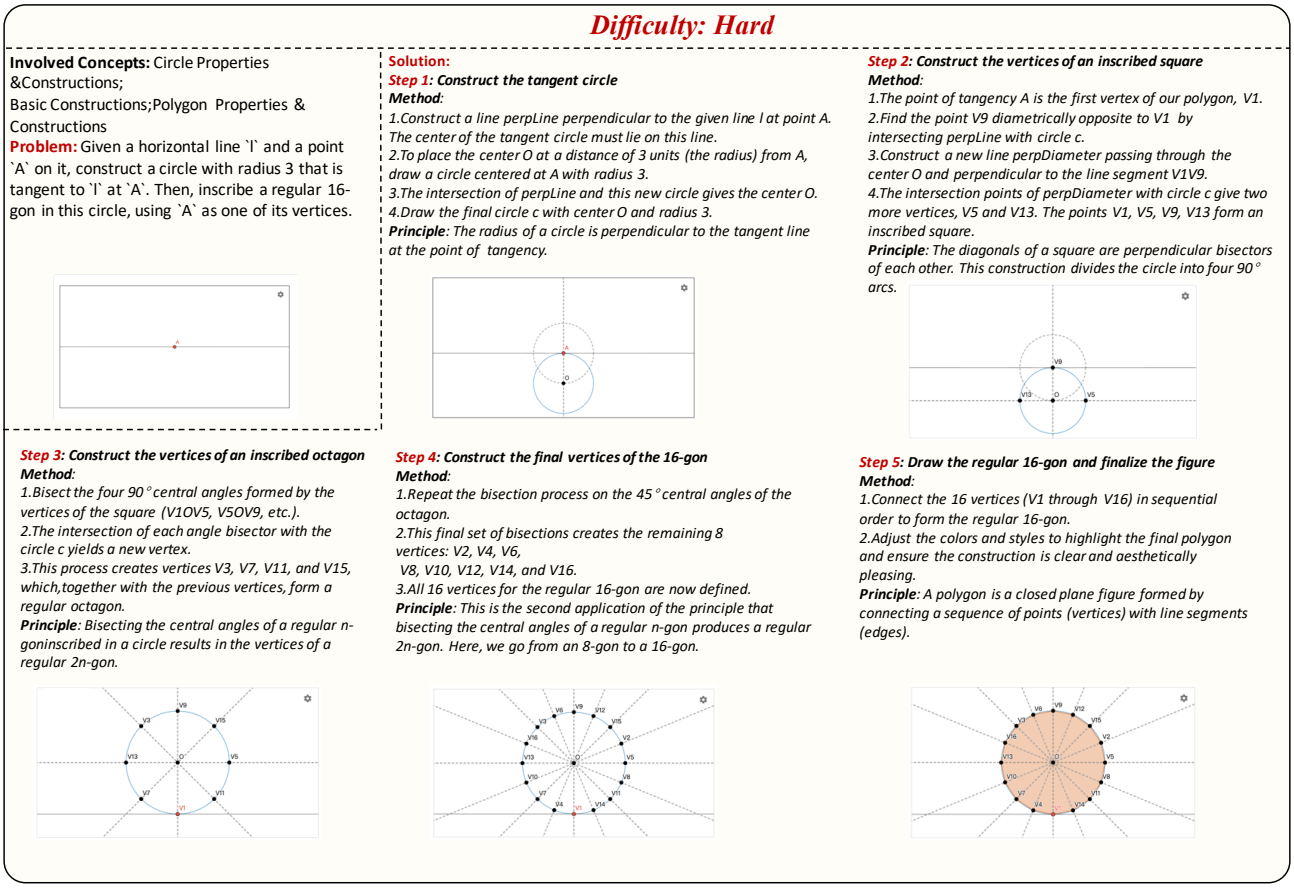

Figure 19. Example of a hard-level task.

circumcenter-based construction with an incenter-based approach. This leads to distorted geometry and an abstract diagram unfaithful to the problem constraints. Among the LLMs/LRMs, GPT-5 misapplies the Inscribed Angle Theorem: although the logic is cited in text, point C is wrongly placed on the minor arc, creating a 150° angle rather than the intended 30°. Qwen3-14B exhibits three errors—(1) it omits a critical intersection point required to anchor the desired angle, (2) it misstates the reasoning for placing the point, and (3) it fabricates symbolic justifications, invoking undefined elements like ‘circleB‘ or ‘Intersect(circle4, circleB)‘. These errors underscore the challenge of aligning formal reasoning with precise geometric operations.

3. Construction–Computation Confusion These errors occur when models mistake final computed values as direct construction inputs, violating causality between geometric reasoning and procedural execution. Figure 22 illustrates such cases. Qwen3-VL and GPT-4o both aim to create an inscribed angle of 40°, yet instead of deriving the necessary construction, they directly use 40° as a rotation command. This leads to the creation of central angles of 160°, effectively doubling the desired measure. BAGEL, on the other hand, avoids construction entirely—its output contains decorative expressions and numeric labels but lacks any geometric logic or execution path. These failures reveal an inability to differentiate between construction goals and input prerequisites.

4. Code Execution and Syntax Errors This category captures issues arising purely at the code generation and execution level, specific to the plan–code–render pipeline of LLMs/LRMs. Figure 23 shows representative examples. DeepSeek-R1 mismanages variable reuse, overwriting labels like ‘b‘ and ‘c‘, which breaks GeoGebra’s execution semantics. GPT-4 references undeclared points such as ‘B‘, violating dependency ordering and halting the rendering process. GLM-4.5V includes multiple syntax violations: it calls nonexistent points ‘R‘, ‘S‘, ‘T‘, misuses list indexing in ‘Intersect(...)‘, and improperly chains commands. Such code-level breakdowns illustrate persistent brittleness in symbolic translation pipelines and highlight the gap between textual plans and executable geometric instructions.

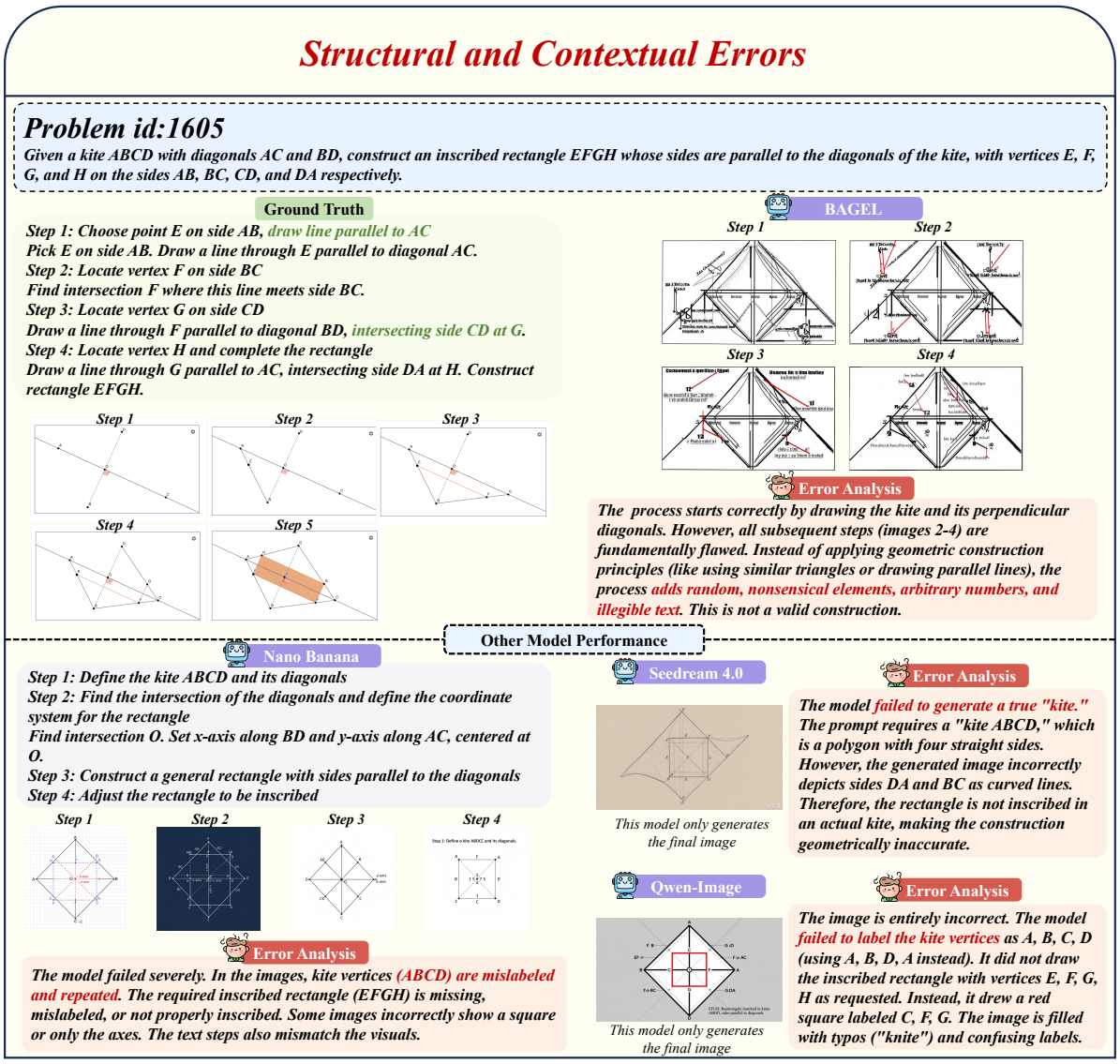

Figure 20. Structural failures in End-to-end UMMs.

E. Additional Experiments

E.1. Code Evaluation

Table 5 reports model performance on GGBench-Code across three categories of metrics: executability (pass@1), lexical and semantic similarity (BLEU, ROUGE-L, chrF, RUBY), and structural consistency (Edit Distance). These metrics offer fine-grained insight into how accurately models can produce syntactically valid and semantically faithful GeoGebra code beyond visual and textual reasoning.

Among all models, GPT-5 [34] demonstrates the strongest symbolic competence. It achieves a pass@1 score of 79.02, indicating its generated code is frequently executable and logically sound. Its performance across similarity metrics is also competitive, with a RUBY score of 39.92 and a chrF score of 62.44, suggesting both surface-level and deeper structural alignment with reference solutions. Claude-Sonnet-4.5 [3] and Qwen3VL [62] follow closely, consistently ranking high across all metrics, reflecting their ability to generate precise symbolic constructs within correct geometric contexts.

Text-only models like DeepSeek V3.1 [24] perform reliably as well, reaching a pass@1 of 70.09 and yielding strong BLEU and ROUGE scores. While their Edit Distance is marginally higher than top-performing multimodal models, the results still reflect structurally well-formed code with minor formatting deviations. By contrast, models such as GPT-4o [19] and GLM-4.5V [17] exhibit substantial deficiencies across all code dimensions, underscoring their difficulty in bridging visual context and symbolic execution.

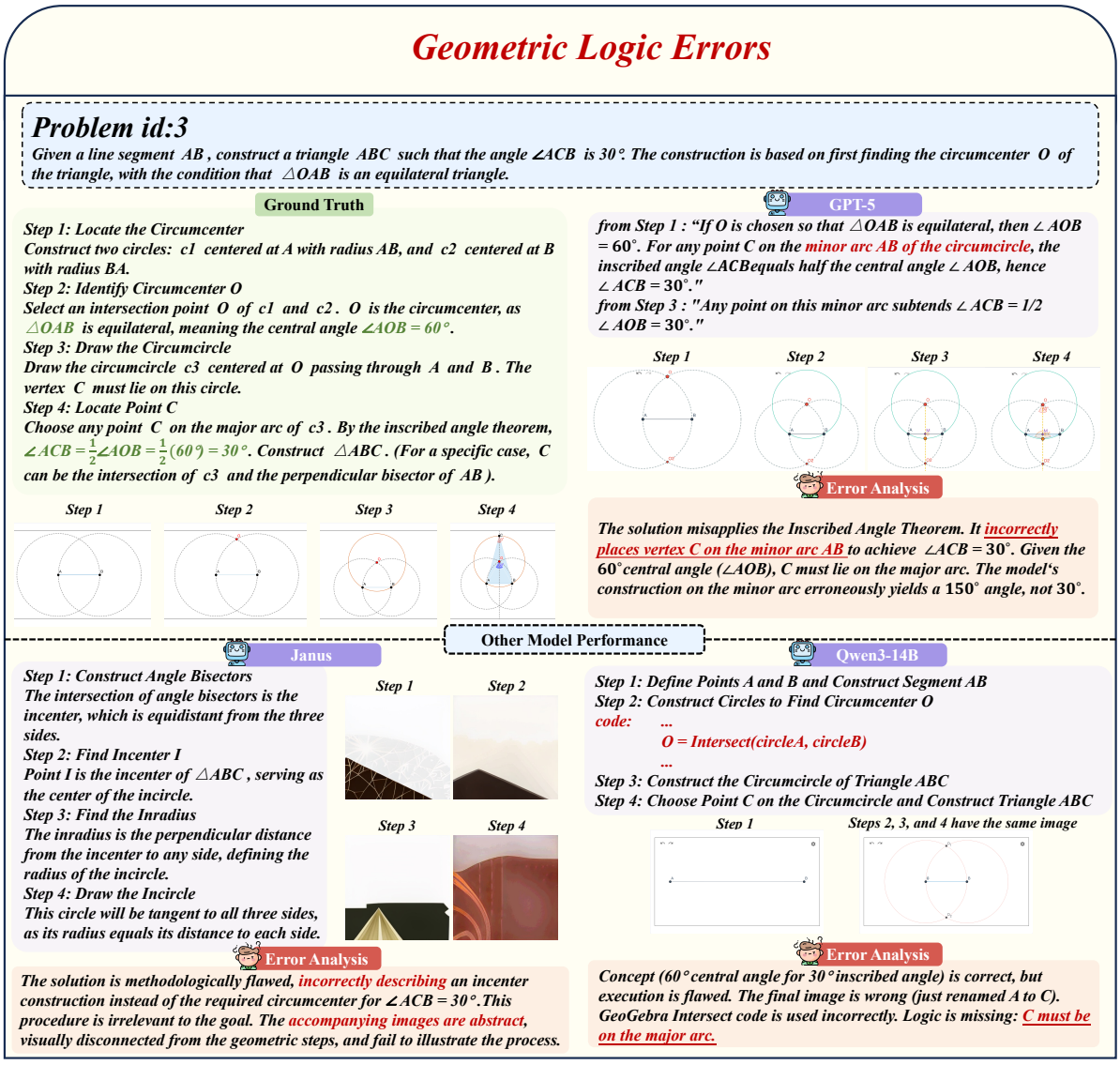

Figure 21. Logical and step-level errors across models.

Table 5. Evaluation results on GGBench-Code across execution, similarity, and structural metrics.

Model	pass@1 ↑	BLEU Mean ↑	RUBY Mean ↑	ROUGE-L Mean ↑	chrF Mean ↑	EditDist Mean ↓
GPT-4o [19]	7.87	8.33	12.85	27.41	29.17	78.92
GLM-4.5V [17]	14.25	10.28	17.53	30.48	38.32	79.61
Qwen3-14B [62]	34.30	14.04	23.85	38.07	44.36	76.89
Gemini 2.5 Pro [9]	32.67	24.21	40.56	50.50	65.87	71.14
DeepSeek-R1 [24]	54.36	18.46	33.68	42.69	60.11	73.64
GPT-4 [1]	50.53	17.31	28.54	44.62	51.25	70.34
Qwen3-VL [62]	69.31	23.24	38.61	50.73	63.51	70.24
DeepSeek V3.1 [24]	70.09	20.44	35.14	47.36	59.24	70.87
Claude Sonnet 4.5 [3]	75.34	22.04	36.77	49.68	60.28	70.81
GPT-5 [34]	79.02	18.87	39.92	44.08	62.44	74.13

UMMs are excluded from this evaluation due to their lack of code outputs.

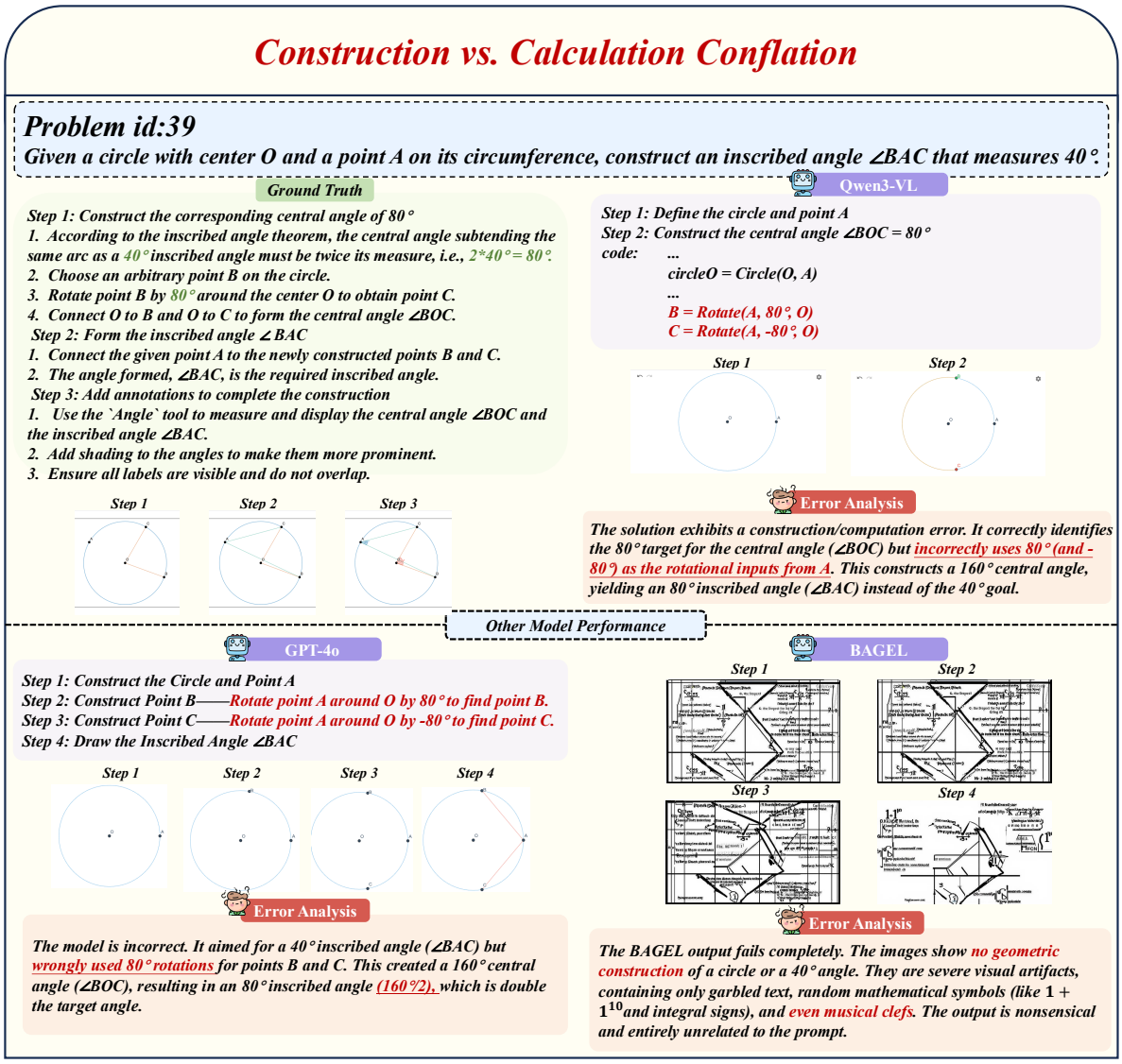

Figure 22. Confusion between construction and computation.

These findings corroborate the main results in Table 4, reaffirming that models with strong text reasoning capabilities tend to produce high-quality symbolic code. GGBench-Code thus serves as an effective diagnostic complement to our full benchmark, enabling precise evaluation of symbolic alignment and formalization in multimodal geometric tasks.

E.2. Performance by Question Type

Figure 24 presents the performance of all evaluated models across three geometric task types: analytic construction (AC), geometric transformation construction (GTC), and straightedge-and-compass construction (SCC), measured using the unified VLM-I metric. Clear distinctions emerge across categories. Models generally perform best on SCC tasks that emphasize rule-based geometric procedures, where GPT-5 [34] reaches 72.54, followed closely by Claude-Sonnet-4.5 [3] and DeepSeek V3.1 [24], both above 66. Performance on GTC tasks is moderately lower: Claude-Sonnet-4.5 records 66.26 and DeepSeek V3.1 achieves 53.53, reflecting the increased complexity of maintaining geometric invariants under rotation or reflection. Analytic construction tasks yield the lowest results, with GPT-5 scoring 51.90 and DeepSeek V3.1 reaching 46.07, likely due to the greater spatial flexibility and reduced structural constraints of such problems. UMMs such as Janus [55], BAGEL [10], and Seedream [38] remain consistently weaker due to the absence of symbolic reasoning, while text-only systems like DeepSeek V3.1 show competitive performance on structure-driven categories through accurate code synthesis. Overall, these findings indicate that

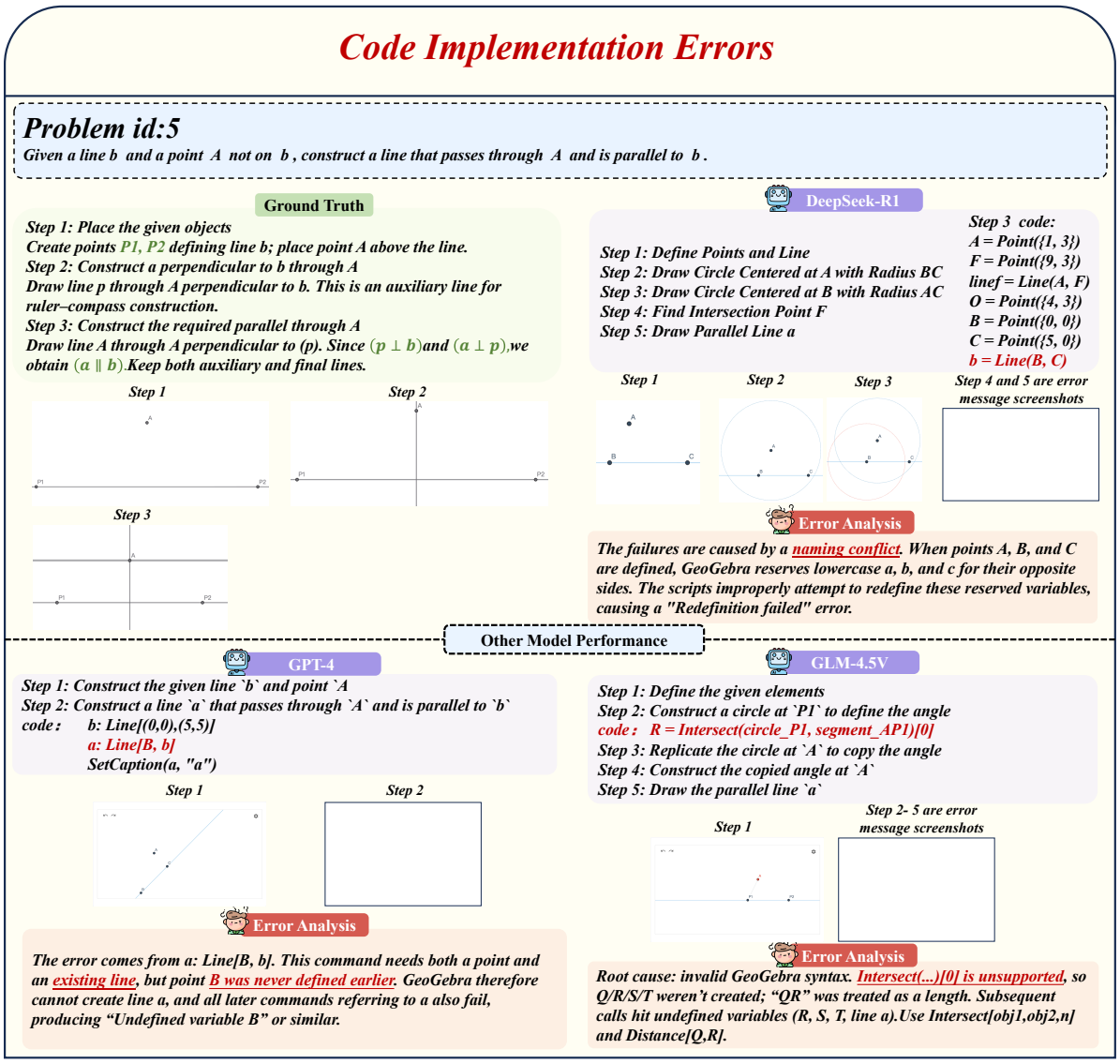

Figure 23. GeoGebra code execution errors.

model effectiveness is shaped jointly by modality alignment and the specific geometric reasoning paradigm required by the task.

E.3. Performance by Difficulty Level

Figure 25 presents the VLM-I scores across three difficulty levels—Easy, Medium, and Hard—providing a comprehensive view of how reasoning complexity impacts model performance. The majority of systems perform best on Easy problems and experience a steady decline as task difficulty increases, confirming the progressive design of GGBench. This pattern is especially pronounced for models like Qwen3-14B [62] and GPT-4 [1], which drop more than 20 points from Easy to Hard, suggesting that their performance is disproportionately sensitive to longer reasoning chains and higher geometric abstraction. In contrast, GPT-5 [34] maintains strong results across all difficulty tiers, scoring 72.70 on Easy and 66.77 on Hard, highlighting its robustness in both planning and execution. Claude-Sonnet-4.5 [3] follows a similar trend, while DeepSeek V3.1 [24] notably surpasses both on Easy problems with a peak of 73.57. UMMs like Nano Banana [9], Janus [55], and Seedream [38] show more stable but consistently lower performance across difficulties, indicating weaker adaptation to reasoning complexity despite visual fluency. These observations reinforce that difficulty scaling in GGBench effectively distinguishes between symbolic robustness and shallow pattern matching, making VLM-I a reliable diagnostic signal for geometric reasoning across complexity levels.

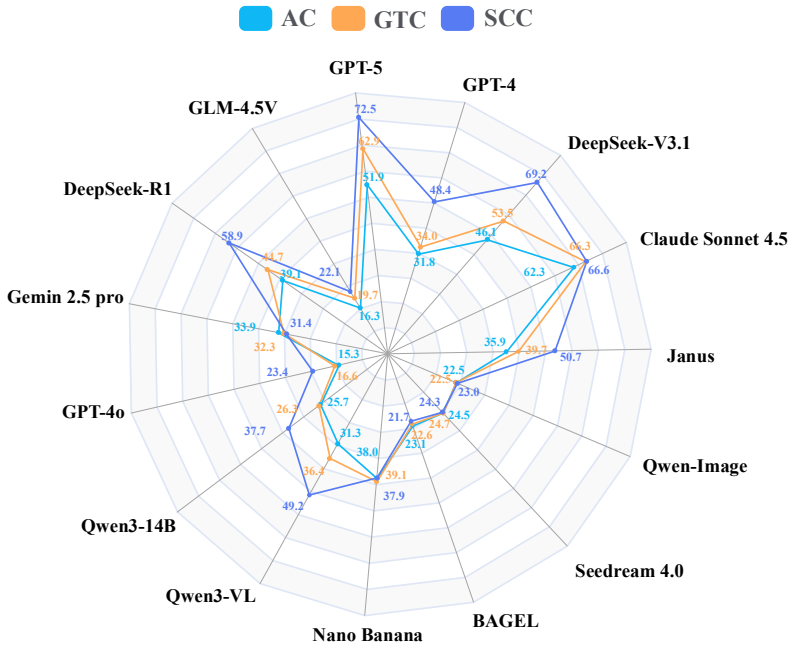

Figure 24. VLM-I scores across geometric task types: Analytic Construction (AC), Geometric Transformation Construction (GTC), and Straightedge-and-Compass Construction (SCC). Higher values indicate better performance.

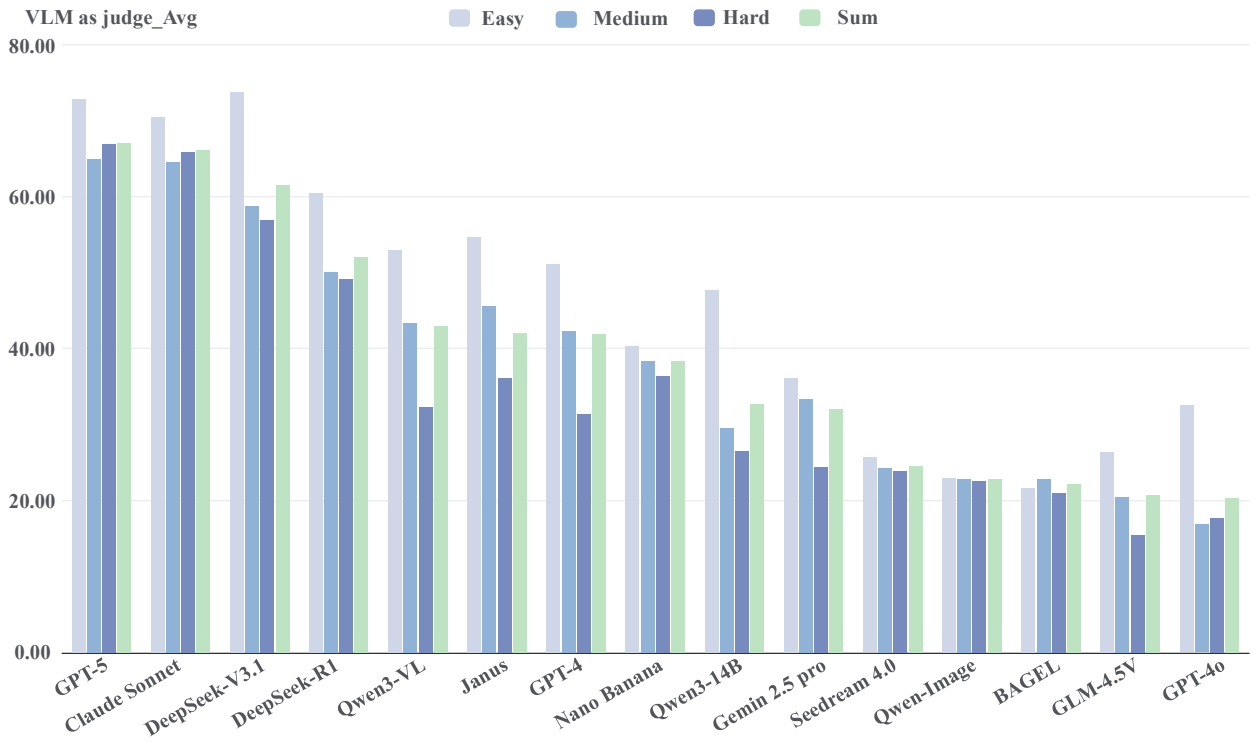

Figure 25. VLM-I performance across difficulty levels on GGBench. Bars represent *Easy*, *Medium*, *Hard*, and overall scores. VLM-I captures both intermediate reasoning quality and final visual correctness.