3DrawAgent: Teaching LLM to Draw in 3D with Early Contrastive Experience

Supplementary Material

Overview

This supplementary material provides additional details, analyses, and results that further support the findings of our **3DrawAgent** framework. Specifically, it is organized as follows:

- **Section A** presents comprehensive **Implementation Details**, including renderer configurations, hyperparameters for the adopted LLMs (DeepSeek and Gemini), and the settings of all evaluation metrics.
- **Section B** offers an extended analysis of **Stroke Count Constraints** and the **Variance in CKE**, illustrating how our method adapts to complexity budgets and demonstrating the necessity of CLIP-guided contrastive selection over random selection.
- **Section C** describes the design and outcomes of our **User Study**, which assesses the perceptual quality of our generated 3D sketches compared to baseline approaches.
- **Section D** provides a candid discussion on the **Limitations and Failure Cases** of our approach, analyzing common geometric challenges and outlining potential avenues for future research.
- **Section E** provides the detailed **Prompts** and execution logs used throughout our framework, followed by an extended gallery of 3D generation results across a wide range of object categories.

A. Implementation Details

In this section, we present the detailed configurations used in our experiments, including the differentiable renderer setup, the CLIP-based evaluation metric, the Large Language Model (LLM) settings, and an analysis of the computational cost.

A.1. Renderer and Evaluation Settings

Differentiable Renderer. Our rendering pipeline is built upon `pydiffvg` [16]. To ensure consistency across all experiments, we employ a unified `BatchRenderer` with the following fixed hyperparameters:

- **Canvas & Projection:** We render all 3D sketches onto a 512×512 canvas with perspective projection. The camera focal length is set to 907.32 (derived from fov $\approx 60^\circ$ for a 512px width).
- **Curve Style:** The 3D Bézier curves are rasterized with a fixed stroke width of 2.0 pixels. The stroke color is set to dark gray (i.e., RGBA $= [0.1, 0.1, 0.1, 1.0]$) on a white background (i.e., RGBA $= [1.0, 1.0, 1.0, 1.0]$), composited via alpha blending.
- **Viewpoints:** We adopt a fixed set of *16 camera poses* uniformly distributed around the object to capture multi-view geometric structure.

CLIP-based Scoring (CLIP-S).

We utilize the pre-trained `ViT-B/32` model to measure text-3D sketch semantic alignment. Following Dream3DVG [17], for a given object category C (e.g., "car"), we construct a reference text prompt using a sketch-oriented template:

> "{C}, `minimal 2d line drawing, on a white background, black and white.`"

For each generated 3D sketch, we render 16 viewpoints and compute the cosine similarity between the embedding of text and each rendered view. The final CLIP-S score is the average similarity across all 16 views.

A.2. LLM Configurations

We employ fixed hyperparameters for both Foundation Models (`DeepSeek-V3.2-Exp` and `Gemini-2.5 Pro`) to ensure a controlled balance between exploratory diversity during experience accumulation and deterministic behavior at inference.

- **Exploration Phase (Training-Free CKE):** To promote diverse candidate sketches for contrastive critique, we set a sampling temperature of 0.7. The maximum output length is fixed at 32,768 tokens to support long chains of thought and large Bézier-curve lists. The GRPO contrastive group size is set to $K = 5$.
- **Inference Phase:** For final 3D sketch generation, we reduce the sampling temperature to 0.3 for more stable and deterministic outputs, while keeping the token limit at 32,768 to preserve full-structure curve descriptions.

A.3. Computational Cost and Efficiency

Unlike optimization-based methods (e.g., SDS) that require per-instance gradient updates, our framework is training-free and relies on API-based LLM inference. Below, we report the empirical computational cost measured using `DeepSeek-V3.2-Exp`.

Cost Comparison. Table 3 compares the inference latency and average monetary cost per sample of our method against state-of-the-art optimization-based baselines. While prior methods require 60 to 120 minutes of expensive GPU

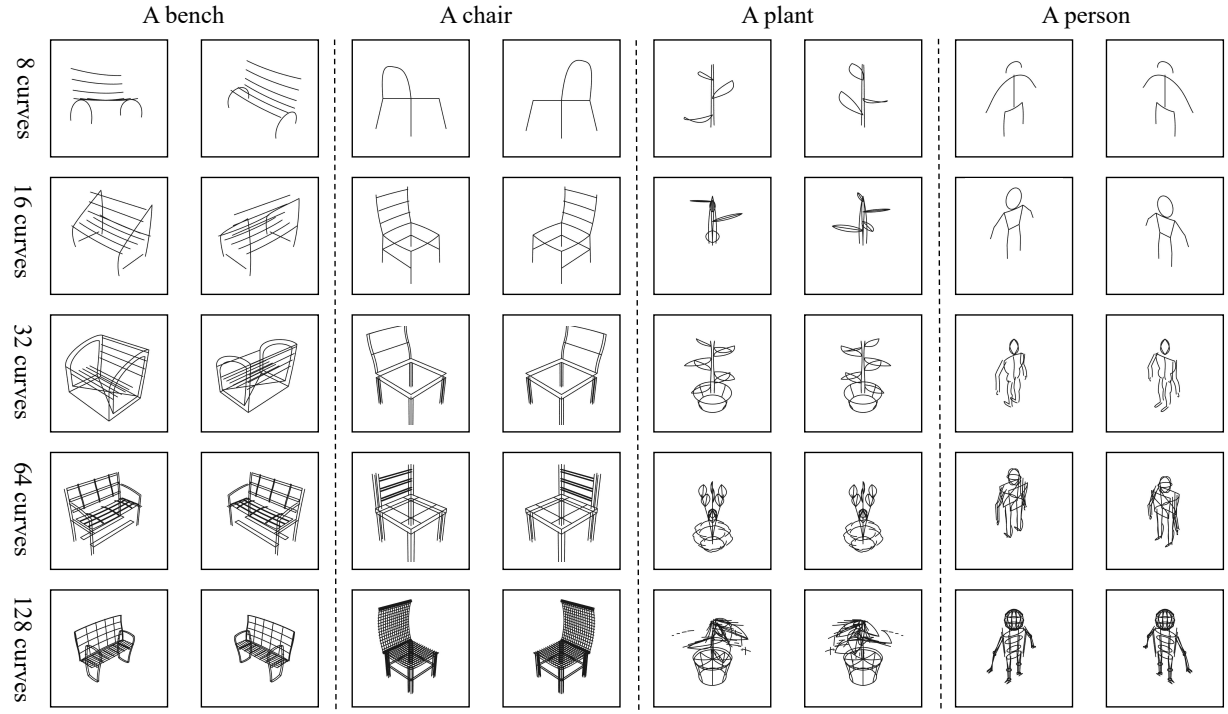

Figure 6. **Impact of Stroke Constraints on 3D Abstraction across Categories.** We evaluate the model's generation capability under varying Bézier curve budgets (rows from 8 to 128) across diverse categories: `Bench`, `Chair`, `Plant`, and `Person`. At minimal budgets (8 curves), the model performs high-level semantic abstraction, producing skeletal representations (e.g., a stick figure for the person or a simple stem for the plant). As the budget increases to 32–64 curves, structural details emerge, such as the pot geometry for the plant or parallel slats for the furniture. At 128 curves, the sketches evolve into dense wireframes. This demonstrates the model's versatility in adapting its planning strategy—from abstract symbolism to geometric fidelity—for both rigid and organic shapes.

computation per object, our training-free, API-based approach drastically reduces the generation time to approximately 2 minutes per sample, lowering the average cost to just \$0.09.

Table 3. Cost comparison with single objects.

Method	GPU / API	min / Sample	Avg. Cost (USD)
3Doodle	NVIDIA RTX3090	~ 120	0.80
Diff3DS	NVIDIA A10	~ 120	1.50
Dream3DVG	NVIDIA A100	~ 60	1.30
Ours	DeepSeek-V3.2	~ 2	0.09

Monetary Cost for CKE. To further detail our API consumption, we conducted a full CKE (Contrastive Knowledge Extraction) run on a dataset of 100 prompts over 3 epochs with a group size of $K = 5$. The total API cost was $\approx$ \$11 USD. The detailed pricing model and estimated consumption are listed in Table 4.

B. More Results and Analysis

In this section, we provide further analysis of the model's behavior and learning stability. Specifically, we evaluate its controllability over abstraction levels via stroke count constraints, and investigate the variance of our Contrastive Knowledge Extraction (CKE) compared to a random selection baseline.

Table 4. **Training-Free Cost Analysis (DeepSeek-V3.2-Exp).** Costs are estimated for a complete experience extraction run (100 prompts, 3 epochs, $K = 5$).

Metric	Unit Price (USD)	Total Volume	Est. Cost (USD)
Input (Cache Hit)	0.027 / 1M tokens	$\sim$150M tokens	4.1
Input (Cache Miss)	0.275 / 1M tokens	$\sim$10M tokens	2.8
Output	0.413 / 1M tokens	$\sim$10M tokens	4.1
Total	-	-	**11.0**

B.1. Abstract Level with Stroke Number Control

A key advantage of our language-driven framework, relative to pixel-space or fixed-representation generative methods, is its explicit controllability over the abstraction level via natural-language constraints. By specifying the desired number of curves (e.g., "draw a bench using exactly 16 curves"), the LLM is encouraged to allocate its limited ge-

ometric budget toward semantically important structures.

To evaluate this controllability, we prompt the model to sketch a `bench` under different stroke-count constraints: 8, 16, 32, 64, and 128 curves. The resulting sketches are visualized in Figure 6.

Abstraction vs. Detail. As shown in Figure 6, under a tight budget of 8 curves (i.e., the first row), the model demonstrates emergent reasoning by focusing on the most essential components. Curves are primarily allocated to outline the seat and the four legs, while finer details and textures are omitted. This behavior indicates that the LLM encodes an internal hierarchy of shape semantics, prioritizing structural integrity over decorative elements.

Progressive Refinement. With a higher curve budget of 16 and 32, the model gradually transitions from a *skeletal* to a more *descriptive* representation. Additional curves are allocated to the backrest and seat, capturing details such as the slats of a wooden bench. This smooth and coherent refinement demonstrates that the learned experience library $\mathcal{E}$ effectively guides the model in managing increased complexity while maintaining geometric consistency.

High-Density Generation. With 64 and 128 curves (i.e., the last two rows in Figure 6), the sketches form dense wireframes. Unlike standard mesh reconstruction methods that can struggle with topology, our approach preserves clean vector curves. However, beyond a certain point (e.g., 128 curves), perceptual improvement plateaus, and the model may introduce redundant or overlapping lines to exhaust the curve budget. This demonstrates the trade-off between efficiency and fidelity, suggesting that a medium budget (32–64 curves) typically provides the optimal balance for concept design tasks.

B.2. Variance in CKE and Random Selection

To further validate the effectiveness of our CLIP-guided Contrastive Knowledge Extraction (CKE), we compare it against a random selection baseline. In the random selection setting, instead of forming contrastive pairs based on multi-view CLIP similarity scores, we randomly sample generated sketches to form "relatively better" and "worse" pairs for the LLM to critique. The comparison of semantic alignment (CLIP-S_T) over multiple epochs is presented in Table 5.

Table 5. **Comparison of CKE against Random Selection.** We report the CLIP-S_T scores across different experience extraction epochs.

Setting	Component	Ep 0	Ep 1	Ep 2	Ep 3
Base	(w/o CKE)	0.5735	-	-	-
Random	(w/ CKE)	0.5735	0.6420	0.5595	0.6094
Ours	**(w/ CKE)**	0.5735	0.6461	**0.6643**	0.6428

Impact of Random Pairs. As shown in Table 5, random pair selection leads to highly unstable performance across CKE iterations, with a significant performance drop in Epoch 2 (0.5595, which is even lower than the Base model's 0.5735). We attribute this instability to the fact that randomly sampled pairs often lack a clear semantic ordering, producing noisy and sometimes contradictory preference signals. In contrast, our CLIP-guided selection provides reliable and consistent guidance, allowing the model to steadily accumulate beneficial spatial principles and achieve consistent gains.

CKE Variance and Over-reasoning. Table 5 also reveals a slight performance drop for our method in Epoch 3 (from 0.6643 down to 0.6428). As CKE progressively extracts and integrates experiences over multiple iterations, the newly extracted rules in later stages can sometimes become local, task-biased, or overly specific. When these overly specific constraints are integrated into the experience bank, they may reduce the LLM's drawing flexibility or cause "over-reasoning," which slightly degrades its generalization to novel prompts. This observation highlights the importance of maintaining a concise, abstract, and high-level experience library.

C. User Study

To better evaluate the perceptual quality of the generated 3D sketches, we conducted a user study comparing our **3DrawAgent** against two state-of-the-art baselines: **Diff3DS** [35] and **Dream3DVG** [17].

Participants and Dataset. We recruited 30 volunteers, primarily university students and researchers with backgrounds in computer science and design (ages 18–28), with a gender ratio of roughly 5:1 (male to female). The evaluation set comprised 40 randomly selected prompts spanning both rigid objects (e.g., furniture, vehicles) and organic shapes (e.g., animals, plants), consistent with the categories presented in the qualitative comparisons of the main paper.

Procedure and Criteria. For each prompt, participants were shown three anonymized 3D sketches, i.e., rendered as rotating videos to display full 360-degree structure, generated by Diff3DS, Dream3DVG, and our method. The presentation order was randomized to avoid bias. Participants were asked to select the best sketch based on two criteria:

- **Semantic Fidelity:** How accurately the sketch reflects the input text description.
- **Geometric Plausibility:** Whether the 3D structure is coherent, clean, and free of floating artifacts or fragmented curves.

Results. The results of the user study are summarized in Figure 7. Our method, **3DrawAgent**, achieved the highest preference rate with **46.66%** of votes. **Dream3DVG** followed with **36.67%**, while **Diff3DS** received **16.67%**. Participants noted that **Dream3DVG** often captures overall object volume well, benefiting from its 3DGS guidance,

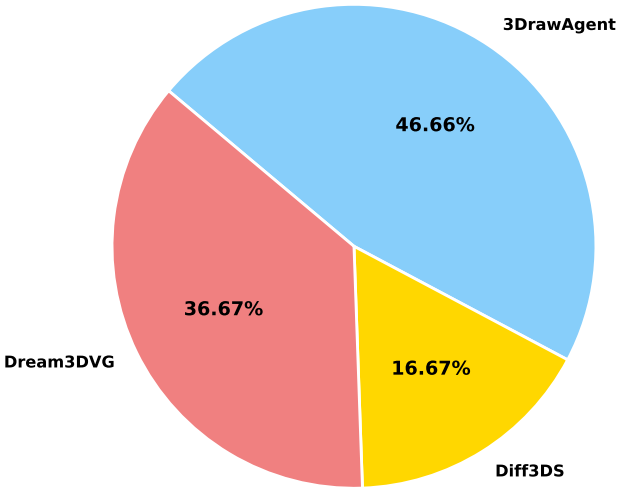

Figure 7. **User Study Results.** Percentage of user preference votes. **3DrawAgent** (46.66%) is the most preferred method, showing a clear advantage over **Dream3DVG** (36.67%) and **Diff3DS** (16.67%) in terms of combined semantic and geometric quality.

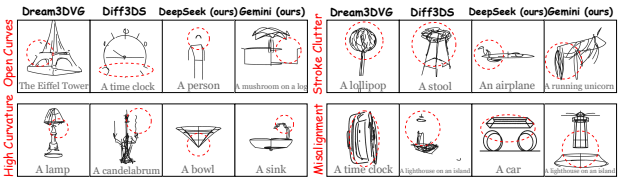

Figure 8. **Visual Examples of Common Failure Modes.** Despite our experience bank's guidance, 3DrawAgent encounters challenges with strict geometric connectivity and handling semantic ambiguity in complex structures. Key issues include (a) disconnected junctions where strokes should intersect, (b) floating components, and (c) visual clutter when managing ambiguous topological constraints.

but occasionally produces over-smoothed or noisy strokes. In contrast, **3DrawAgent** was consistently praised for its clean, *designer-like* vector curves and superior structural logic, particularly in handling complex topologies where explicit geometric reasoning is critical.

D. Limitations and Failure Cases

Despite 3DrawAgent's capacity to generate semantically accurate and abstract 3D sketches training-free, our method exhibits certain limitations common to parametric curve generation via text-guided optimization. Below, we dissect specific failure modes and propose directions for future refinement, referencing visual examples in Figure 8.

Strict Geometric Plausibility and Connectivity. A primary challenge lies in enforcing strict geometric constraints, such as perfect connectivity between adjacent curves that form semantic joints (e.g., where table legs meet the table top). While the extracted experience bank $\mathcal{E}$ provides high-level structural guidance (e.g., "legs should be placed vertically under the table top"), it lacks a dense vector point supervision to enforce localized endpoint intersection. As visualized in Figure 8 (a) and (b), strokes corresponding to different semantic components may fail to intersect precisely, resulting in slightly disconnected joints or "floating" elements. The current loss function, primarily a holistic CLIP-based similarity score, optimizes for overall semantic recognition rather than local topological exactness. Incorporating explicit intersection-promoting or endpoint-matching loss terms during optimization could mitigate this issue.

Semantic Ambiguity and Component Placement. As a text-driven agent relying on a frozen, generalized geometric model, 3DrawAgent sometimes struggles with ambiguous semantic placement of geometric components, particularly for non-canonical object structures. As shown in Figure 8 (a), while the semantics of a "stool" are captured, the spatial relationships between the individual curves defining the support legs are geometrically loose. In Figure 8 (c), when prompted to sketch a "stroller," the agent generates a plausible overall structure but creates complex, overlapping line-clusters rather than clean contours for detailed components (like wheels), leading to visual clutter.

Future Work Direction. These failure cases highlight that high-level linguistic reasoning alone is insufficient for precise geometric reasoning. Future research could focus on two avenues: (1) integrating learned geometric priors (e.g., pre-trained wireframe reconstruction models) into the generation pipeline to impose better local structural order, or (2) developing dense, multi-view reward functions that specifically penalize floating primitives or incomplete geometric loops.

E. Prompts and More Results

In this section, we provide the exact prompt specifications and execution logs used in our framework **3DrawAgent**. To ensure full reproducibility, we present the raw content of our System Prompt with illustrative generation logs that highlight the step-by-step execution and output behavior. Finally, we present an extended gallery of qualitative results to demonstrate the robust zero-shot generalization of our method.

E.1. System Prompt Specification

Figure 9 shows the *Input to the Agent (LLM)*. This prompt is a comprehensive instruction set designed to initialize the LLM as a 3D spatial planner. It serves several critical roles in our framework:

- **Role & Format Definition:** The "Role Instruction" and "Output Format Specification" constrain the LLM's out-

put to a Python list of 3D coordinates, ensuring deterministic parsing by the renderer without syntax errors.

- **Coordinate Grounding:** The "Coordinate System" defines physical bounds ($[-0.8, 0.8]$) and orientation (Right-handed, Z-up), providing a geometric prior that prevents out-of-view or distorted generations.
- **In-Context Learning:** The "Ground Truth Example" (e.g., `a Benz car`) provides a dense, high-quality reference. This allows the LLM to internalize the expected *density* and *topology* of curves before generating new targets (e.g., `A wardrobe`).

E.2. Generation Logs

Figure 10 shows the raw logs, which serve as a key data carrier in our framework, during 3D drawing using LLM. Beyond indicating success or failure, these logs capture the intermediate states of our training-free loop, functioning both as the *output of the exploration phase* and the *input to the reflection phase*.

- **Source (Agent–Environment Interaction):** The logs capture the LLM's interaction with the evaluator, i.e., the recorded trajectory containing the user prompt and the assistant's response. The `response` field stores the 3D curve coordinates generated by the Agent, while the `reward` field provides the feedback computed by the Environment (CLIP-based renderer).
- **Destination (Input to Judge):** These logs are fed into the Judge LLM (i.e., Experience/Knowledge Extractor), which contrasts high-scoring entries (e.g., `runid: 200`, Reward 0.65) with low-scoring ones (e.g., `runid: 0`, Reward 0.0) to perform causal reasoning.
- **Algorithmic Function:** The contrastive analysis allows the system to self-diagnose: failures in low-reward runs are attributed to *geometric sparsity* (i.e., short rollouts, simple curve lists), whereas successes arise from *dense spatial planning* (i.e., longer inference, utilization of Z-axis). These insights are distilled into textual *Experiences* to guide future 3D sketch generations.

E.3. Additional Qualitative Results

In addition to the analysis above, we present more 3D generation results across diverse categories to demonstrate the robustness of our method, as shown in Figures 11, 12, 13, and 14.

Role Instruction: You are a professional 3D artist specializing in generating procedural wireframe models using 3D Bézier curves.Your task is to create geometry for a given theme, adhering to extremely strict data format and scene constraints.

Output Format Specification: You must and only generate a Python list literal within a specific section of your final answer.

Unique Delimiter: You must wrap the raw Python list code between <curves> and </curves> tags. Outside of these tags, you may write explanations, but inside the tags, there must be no non-list content.

Data Constraints: The content wrapped within the tags must be a Python list [...].

Each element of this list is an independent curve, which itself must be a list.
Each curve's list must contain exactly 4 control points: [P0, P1, P2, P3].
Each control point P must be a list of exactly 3 floating-point numbers: [x, y, z].

Unlimited Quantity: The number of curves in the list is not limited. You can use any number of curves necessary to construct the model based on the complexity of the theme.

Edge-case Rules: Inside the <curves> and </curves> tags, it is absolutely forbidden to include:Any comments (# ...).\nAny variable assignments (e.g., generated_curves = ...).Any text or code that is not part of the list literal.

Coordinate System:

Origin: The object must be centered around the world coordinate system origin (0, 0, 0).
Scale: The entire object should fit within a bounding box from -0.8 to 0.8 on all axes.
Orientation: Use a right-handed coordinate system, where the +Z axis is up, the +X axis is to the right, and the +Y axis is forward.

Ground Truth Example:

This is a perfect example of 'A benz car'.
<curves>
[[[-0.269, -0.431, 0.033], [-0.255, -0.584, 0.014], [-0.277, -0.536, 0.308], [-0.27, -0.342, 0.233]],
...
[[-0.167, -0.289, 0.376], [-0.125, 0.084, 0.454], [-0.207, 0.113, 0.304], [-0.198, 0.228, 0.274]],
...
[[-0.207, -0.411, 0.321], [-0.224, 0.257, 0.319], [-0.256, 0.437, 0.273], [-0.215, -0.422, 0.307]]]
</curves>

Now, strictly adhering to all the rules above, generate curves for the following theme:A wardrobe.

Figure 9. **Full System Prompt.** Raw text input provided to the LLM, combining role definition, strict syntax constraints (code-only output), coordinate system rules, and a few-shot example (“A benz car”) to guide 3D sketch generation.

runid: 0
response:
[[[-0.3, -0.15, -0.4], [-0.1, -0.15, -0.4], [0.1, -0.15, -0.4], [0.3, -0.15, -0.4]],
...
[[[-0.3, 0.0, -0.4], [-0.25, 0.0, -0.45], [0.25, 0.0, -0.45], [0.3, 0.0, -0.4]]]
trajectories:
role: user,
content: You are a professional 3D artist specializing in generating procedural wireframe models using 3D Bézier curves... Now, strictly adhering to all the rules above, generate curves for the following theme:A wardrobe
role: assistant,
content:
[[[-0.3, -0.15, -0.4], [-0.1, -0.15, -0.4], [0.1, -0.15, -0.4], [0.3, -0.15, -0.4]],
...
[[[-0.3, 0.0, -0.4], [-0.25, 0.0, -0.45], [0.25, 0.0, -0.45], [0.3, 0.0, -0.4]]]
rollout_time: 191.22343802452087,
reward: 0.0
…
runid: 200
response:
[[[-0.3, -0.5, 0.0], [-0.2, -0.5, 0.0], [0.2, -0.5, 0.0], [0.3, -0.5, 0.0]],
...
[[0.15, -0.5, 0.65], [0.15, -0.4, 0.65], [0.15, 0.4, 0.65], [0.15, 0.5, 0.65]]]
trajectories:
role: user,
content: You are a professional 3D artist specializing in generating procedural wireframe models using 3D Bézier curves... Now, strictly adhering to all the rules above, generate curves for the following theme:A wardrobe
role: assistant,
content:
[[[-0.3, -0.5, 0.0], [-0.2, -0.5, 0.0], [0.2, -0.5, 0.0], [0.3, -0.5, 0.0]],
...
[[0.15, -0.5, 0.65], [0.15, -0.4, 0.65], [0.15, 0.4, 0.65], [0.15, 0.5, 0.65]]]
rollout_time: 1129.8924763202667,
reward: 0.65185546875

Figure 10. **Data Flow in Experience Extraction.** Raw logs collected during exploration, capturing the Agent's outputs (3D curves) and the Environment's feedback (rewards). These paired samples serve as the *input* to the "Contrastive Knowledge Extraction" module, where the "Judge LLM" derives spatial principles by contrasting high- and low-reward trajectories.

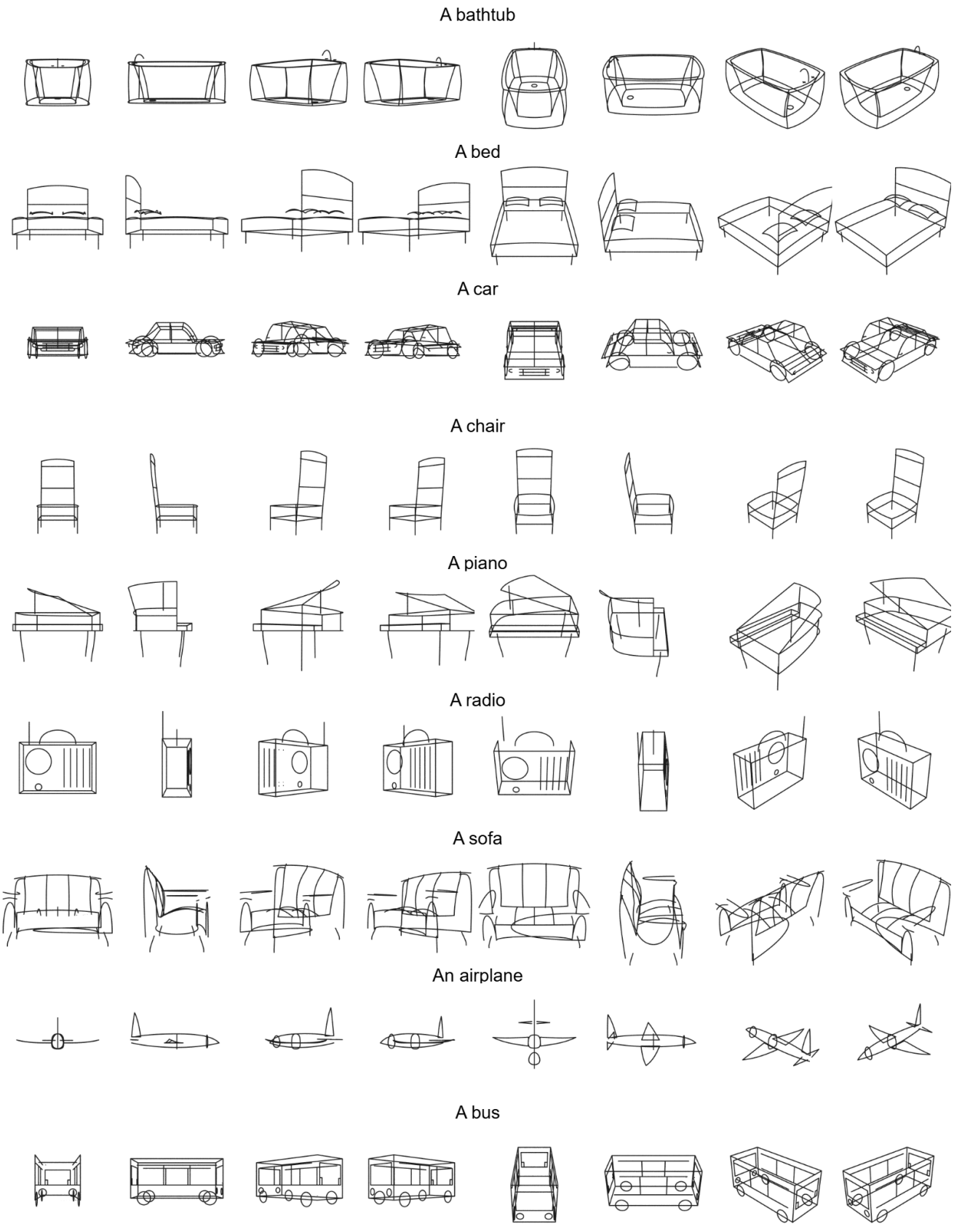

Figure 11. Additional Results: Text-to-3D Generation (Simple Prompts).

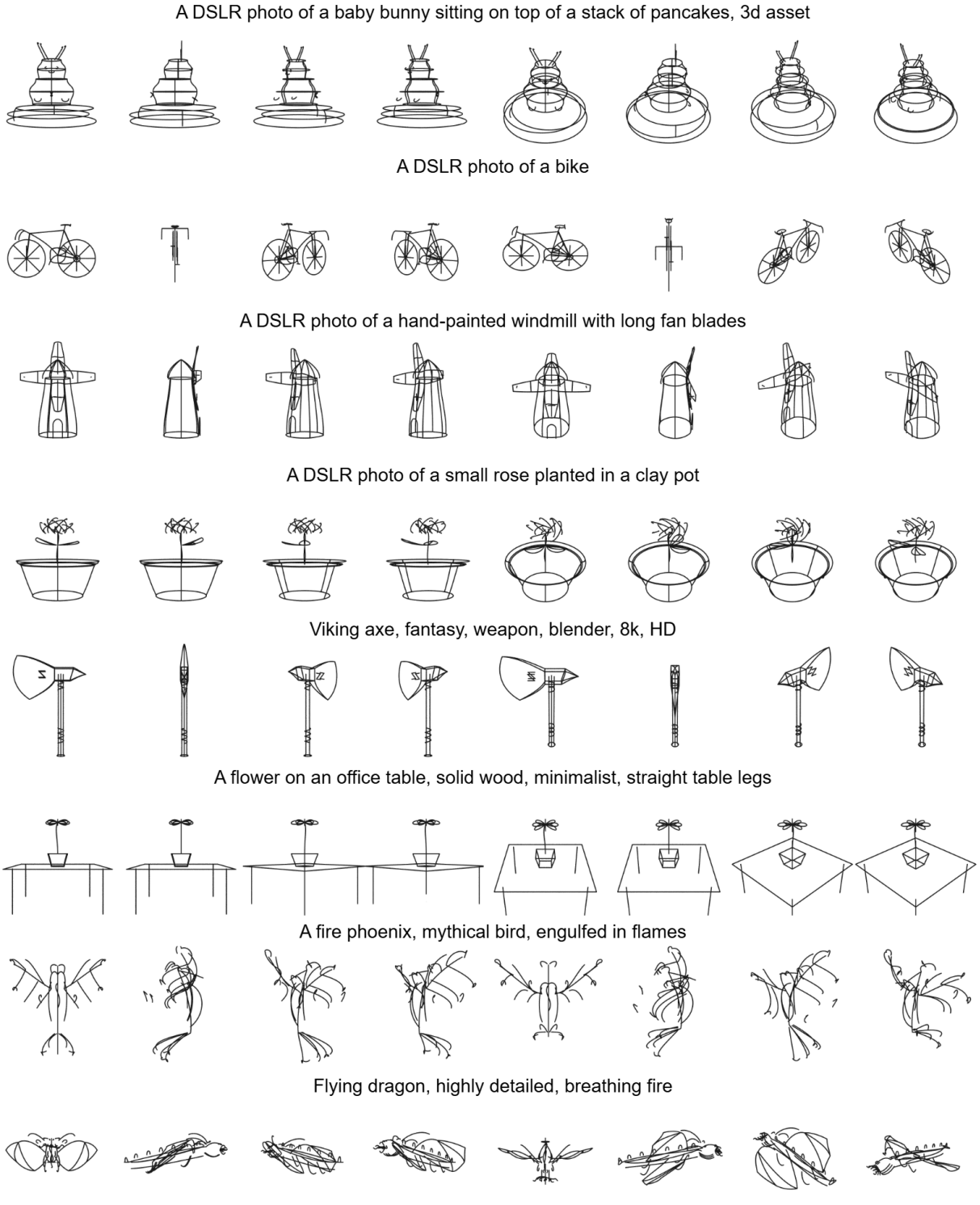

Figure 12. Additional Results: Text-to-3D Generation (Complex Prompts).

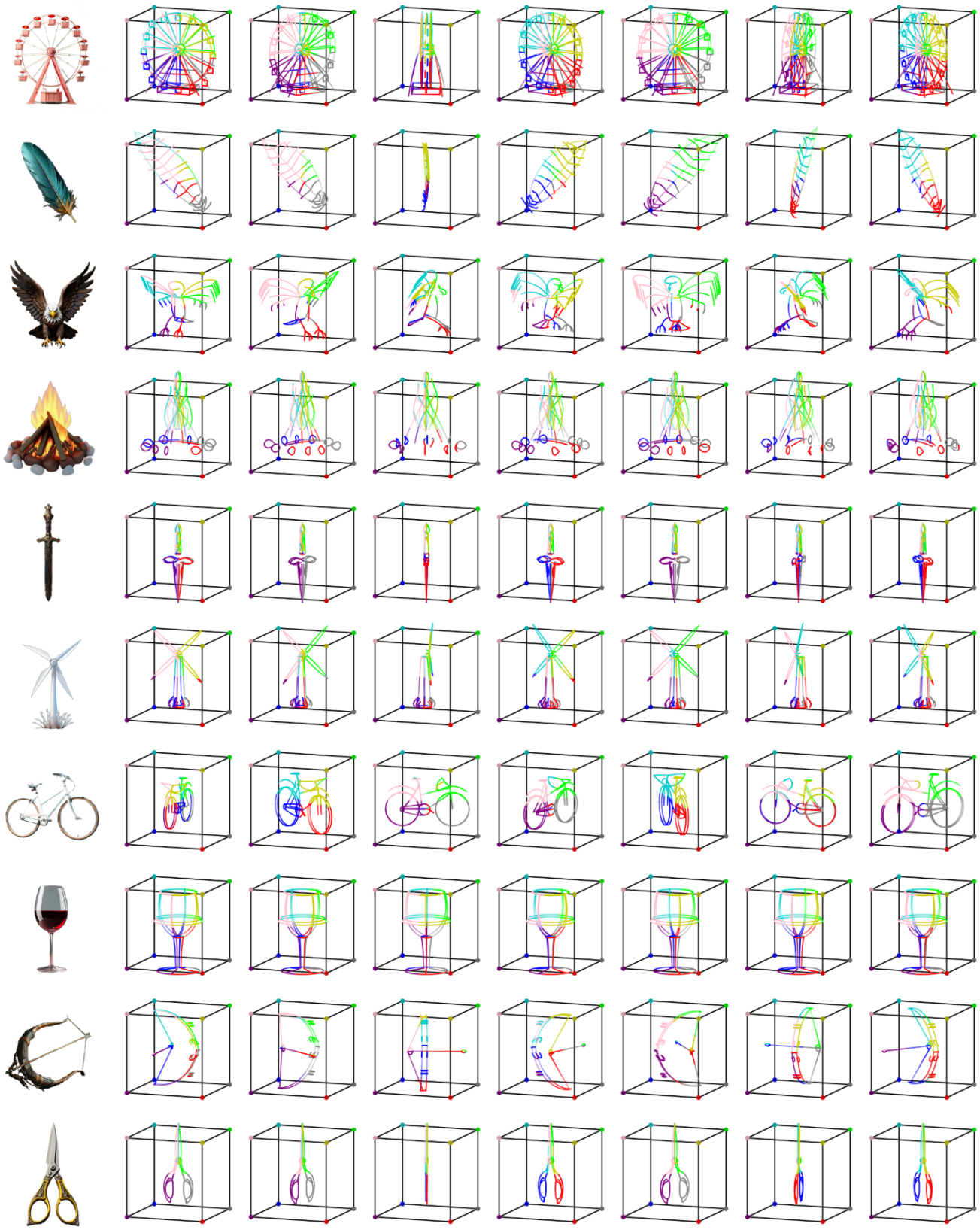

Figure 13. Additional Results: Image-to-3D Generation (Part I).

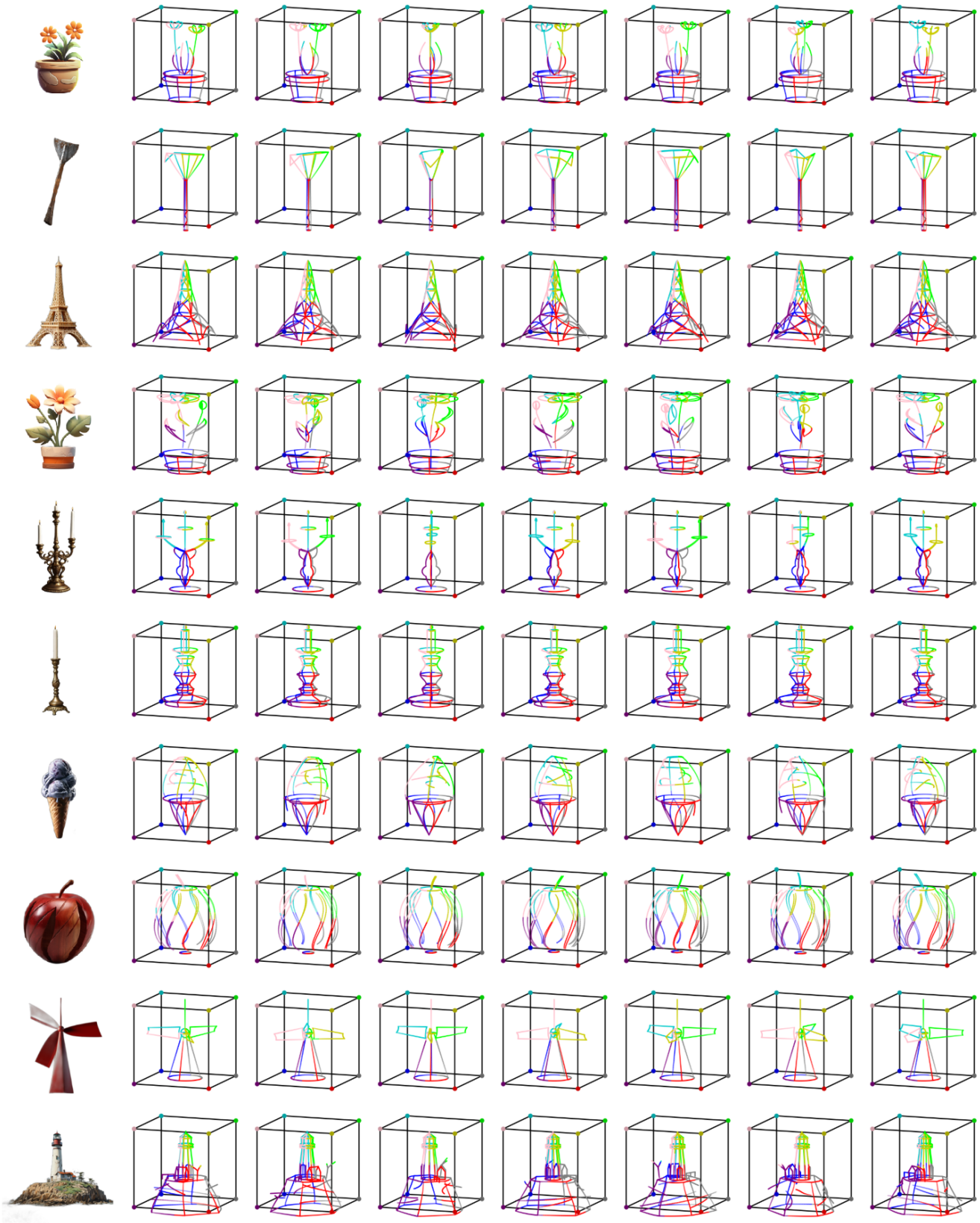

Figure 14. Additional Results: Image-to-3D Generation (Part II).