DORIC: DIGITAL-GENE-BASED OBJECT UNDERSTANDING FROM A SINGLE RGBD IMAGE VIA CHAIN-OF-THOUGHT REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Understanding 3D object knowledge and functionality in the physical world is a fundamental challenge for embodied AI. The recently proposed *Digital Gene* offers a promising explicit representation of objects via executable programs for object structure and function understanding. However, three major practical obstacles remain: 1) inaccurate gene grounding, 2) missing task-oriented reasoning for function utility, and 3) inflexible functional programs. To address these, we introduce **DORIC**, a framework that pairs MLLM's semantic and image reasoning with our proposed gene grounding modules for precise physical and numerical analyses. DORIC decomposes object-understanding requests into sub-tasks, each focusing on a single object and processed in a four-stage Chain-of-Thought (CoT) pipeline: I. Object and Structure Identification, II. Gene Selection and Grounding, III. Part-Level Program Selection, and IV. Object-Level Program Synthesis. By integrating multi-modal reasoning with specialized grounding modules, our contributions are four-fold: 1) a hybrid architecture for precise gene grounding involving MLLM and specialized modules, 2) a task-oriented four-stage CoT pipeline for function utility reasoning, 3) an in-context functional program synthesis paradigm, and 4) superior effectiveness and robustness demonstrated by extensive experiments.

1 INTRODUCTION

Understanding object structure and reasoning about functionality is fundamental yet remains a significant challenge for embodied agents due to the immense diversity of geometries and complex physical constraints. Recent advancements (Qi et al., 2024; Su et al., 2025; Team et al., 2023) have attempted to decode object functionality using implicit representations of structure and function. While these methods demonstrate a correlation between structure and functionality, implicit representations are difficult to translate into interpretable and analyzable reasoning processes that map structure to function. In contrast, the recent proposed Digital Gene (Sun & Lu, 2025), an explicit geometric representation that utilizes basic geometric shapes as fundamental units, encodes an object's hierarchical structure as an executable program. By defining functions on these primitives using structural parameters as variables, Digital Gene (Sun & Lu, 2025) provides a transparent and interpretable pathway from understanding physical structure to object functionality.

However, the practical application of Digital Gene (Sun & Lu, 2025) in the real world faces three significant challenges. **1. Inaccurate Gene Grounding:** Existing methods either struggle to precisely estimate the geometry and pose (Sun et al., 2025b), or neglect real-world scale (Su et al., 2025). **2. Missing Task-Oriented Reasoning for Function Utility:** Current methods (Su et al., 2025; Sun et al., 2025b) lack a logical reasoning chain to discern which specific functions provided by Digital Gene are required for object understanding tasks. **3. Inflexibility of Functional Programs:** Functional programs are typically manually defined (Sun & Lu, 2025; Sun et al., 2025a), lacking flexibility required to meet diverse object-understanding demands.

To address these challenges, we propose DORIC, a framework that leverages Digital Gene representations and CoT reasoning to understand objects from a single RGBD image. DORIC is built on the premise that while MLLMs excel at semantic reasoning and image processing, they are often deficient in spatial understanding and precise numerical analyses, which can be addressed by specialized small

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

models, enabling a hybrid architecture that pairs robust semantic reasoning with precise physical object understanding.

DORIC processes object-understanding request by firstly decomposing it into sub tasks each focusing on a single object, governed by a four-stage CoT process: **I. Object and Structure Identification**: MLLM is prompted to detect object and parts and invoke image segmentation tool for 2D masking. **II. Gene Selection and Grounding**: MLLM selects the most appropriate Digital Gene for each part. Parameter estimation modules then estimate the geometry, pose, and scale, followed by a parameter refinement step to better align the gene with the physical object. **III. Part-Level Program Selection**: MLLM selects task-relevant, predefined part-level functional programs (*e.g.*, handle grasp poses, door rotation axes). **IV. Object-Level Program Synthesis**: By inheriting part-level programs, MLLM synthesizes an object-level functional program tailored to the specific task (*e.g.*, grasping a handle to open a door by rotating it around its axis).

Extensive experimental results demonstrate the superiority of DORIC in geometry, pose, and interaction understanding. Our approach surpasses existing methods by 33% in geometric fidelity, and yields a 14.8% average improvement in interaction success across 21 categories. We further analyze the system's bottlenecks to quantify module-specific contributions and validate our framework's performance using different MLLM backbones for CoT reasoning.

In summary, our contributions are as follows: **1) Hybrid Architecture for Precise Digital Gene Grounding**: By decoupling MLLM semantic reasoning from specialized modules for precise numerical and spatial analysis, DORIC addresses the inherent deficiencies of MLLMs in spatial understanding while ensuring precise gene grounding and physically accurate object understanding. **2) Task-Oriented Chain-of-Thought Reasoning for Function Utility**: We establish a four-stage CoT pipeline that systematically maps visual perception to Digital Gene representation, ensuring a transparent and task-oriented reasoning way from physical structure to object functionality. **3) In-Context Functional Program Synthesis**: We curate part-level functional programs for 263 templates and leverage the in-context learning capabilities of MLLM to synthesize task-oriented, object-level programs via inheritance. This enables highly adaptable and executable object function reasoning tailored to specific tasks. **4) Effective Grounding and Interaction Performance**: DORIC yields a 14.8% improvement in interaction success and a 33% increase in geometric fidelity over existing baselines via extensive experiments, demonstrating the effectiveness, versatility, and robustness of DORIC across diverse object-understanding tasks.

2 Related Works

2.1 Physical Object Understanding

Physical object understanding is a cornerstone of embodied AI, requiring the integration of spatial perception, semantic reasoning, and physical grounding, involving several key aspects. First, foundational part-level understanding and segmentation (Geng et al., 2023; Thomas et al., 2024; Qian et al., 2022; Mo et al., 2019) provides the structural basis for complex objects. To ensure the object can be physically interacted with, 6-DoF pose estimation (Liu et al., 2023b; Wang et al., 2019; Wen et al., 2024; Geng et al., 2023) is required for precise localization. In robotic interaction planning (Ding et al., 2023; Driess et al., 2023; Huang et al., 2022; Brohan et al., 2023; Singh et al., 2023; Zhao et al., 2023), high-level reasoning is used to decompose complex tasks into executable steps. Finally, embodied manipulation strategies (Mo et al., 2021; Ning et al., 2023; Cui et al., 2025; Geng et al., 2023; Huang et al., 2025; Xiang et al., 2020) focus on how agents interact with objects.

2.2 Multi-modal Large Language Model

MLLM regarding object understanding can be categorized into general-purpose reasoning and domain-specific applications. General-purpose MLLMs prioritize broad reasoning and multi-modal fluency. Within this category, closed-source models (Hurst et al., 2024; Achiam et al., 2023; Team et al., 2023) excel at semantic and image reasoning. For example, GPT-5.6-solar excels at high-level task planning, while Gemini-3-Pro-Image (Nano-Banana Pro) enables precise image manipulation (Team et al., 2023). Open-source models (Touvron et al., 2023; Liu et al., 2023a; Zheng et al., 2023; Yang et al., 2025; Wang et al., 2025b) provide a versatile foundation that can be adapted through Supervised Fine-Tuning (SFT) to unlock specialized capabilities. Besides general-purpose ones, domain-specific

MLLMs focus on specialized tasks, including point cloud interpretation (Guo et al., 2023; Xu et al., 2024), affordance learning (Li et al., 2024), and perception-action alignment (Huang et al., 2024; Driess et al., 2023; Huang et al., 2025).

While MLLMs excel at categorical identification and broad spatial reasoning, they often lack the precise numerical estimation of geometry and pose necessary for fine-grained manipulation. This limitation has led to the emergence of hybrid approaches that decouple semantic reasoning from physical understanding and precise numerical analyses.

3 PRELIMINARY

To provide a clear understanding of our work, we first define our representational objectives. Our framework utilizes Digital Gene (Sun & Lu, 2025), which serve as an explicit and programmatic representation of physical object concepts.

First, we formulate how Digital Gene represent object parts. For a specific part P of an object, Digital Gene employs an executable Python class called *Concept Template*, denoted as $\mathcal{P}(\Theta_i, T_i)$, such that

$$P = \mathcal{P}(\Theta, T) \tag{1}$$

where $\Theta = \{\theta^{(k)}\}_{k=1}^{m}$ captures the geometric attributes of the part (*e.g.*, the handle radius or curvature). The pose T is defined by a position $p \in \mathbb{R}^3$ and an rotation $R(\alpha, \beta, \gamma)$ corresponding to the roll-yaw-pitch axes of an euler angle. The part-level function programs can be defined with the parameters, written as

$$f_{\mathcal{P}}(\Theta, T) \tag{2}$$

including object poses, geometric shapes, and affordance, *etc.* Please refer to Appendix.C for detailed examples of part-level programs.

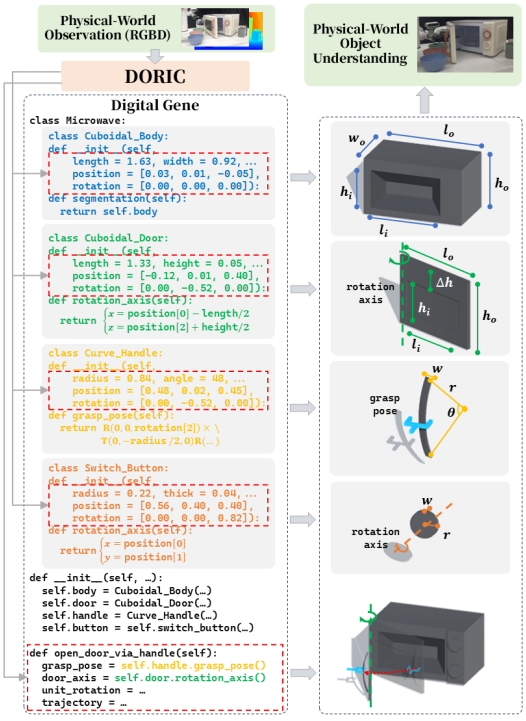


Figure 1: Illustration of Digital Gene. Digital Gene explicitly represent object structure and functionalities in executable programs. Gene *Microwave* comprises four concept templates: *Cuboidal_Body*, *Cuboidal_Door*, *Curve_Handle*, and *Switch_Button*. The parameters and object-level function are synthesized by DORIC, enabling Digital Gene to facilitate a robust and generalizable physical understanding of objects.

Building on the component-level descriptions, Digital Genes utilize programmatic inheritance to construct object-level templates from part-level *Concept Templates*. An object is formally represented as

$$O = \mathcal{O}(\{\Theta_i, T_i\}_{i=1}^{M}) \tag{3}$$

where M denotes the total number of components. Object-level functional programs $f_{\mathcal{O}}$ are synthesized by inheriting and composing part-level functions and their associated parameters. For example, the complex functionality of "opening a door" is synthesized by integrating the part-level "handle grasp pose" with the "door rotation axis", enabling the agent to execute a coordinated "grasp handle and rotate along door axis" sequence.

As illustrated in Figure 1, we aim to **precisely estimate parameters** Θ_i **and** T_i to ground real-world objects using Digital Gene. By leveraging manually pre-defined part-level functions $\{f_{\mathcal{P}}\}$, our framework enables MLLMs to **synthesize task-specific object-level programs** $f_{\mathcal{O}}$, establishing Digital Gene as a robust and versatile representation for embodied object understanding.

4 METHOD

As illustrated in Figure 2, DORIC utilizes a CoT pipeline designed for object understanding. The CoT process involves four stages: I. Object and Structure Identification, II. Gene Selection and Grounding,

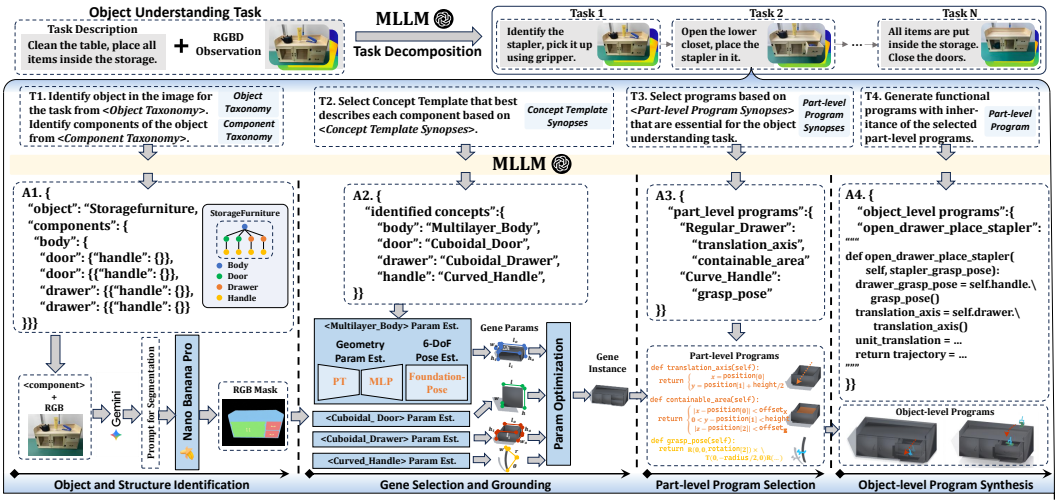


Figure 2: Illustration of pipeline. DORIC decomposes object understanding requests into sub-tasks, each focusing on a single object. It then employs a four-stage CoT to understand object structure and functionalities. Specialized downstream modules are invoked for numerical analysis and geometric refinement, facilitating precise Digital Gene representation and physical executable functions.

III. Part-level Program Selection, and IV. Object-level Program Synthesis. DORIC involves specific downstream modules in stage I and II for spatial understanding and precise numerical analyses. By decoupling semantic reasoning from numerical estimation, this architecture ensures that the resulting Digital Gene is both physically accurate and executable for physical-world object understanding.

We describe each stage of the CoT in Section 4.1. The Digital Gene grounding modules include geometry parameter estimation (Section 4.2), 6-DoF parameter estimation (Section 4.3), and parameter refinement (Section 4.4).

4.1 DORIC Chain-of-Thought

Given a single RGBD observation, DORIC employs GPT-5.6-solar to decompose the multi-object request into individual single-object-centric sub-tasks. For each sub-task, a four-stage CoT involves the following process:

Object and Structure Identification. GPT-5.6-solar reasons about object structure by utilizing the *object and component taxonomies*. Given the visual input and task description, it identifies the target object and its constituent parts to construct a hierarchical object tree. GPT-5.6-solar then formulates a targeted prompt for Nano-Banana Pro (Team et al., 2023) to generate a segmentation mask. By assigning distinct color identifiers to each component within the prompt, the framework facilitates the precise isolation and extraction of individual parts for downstream processing.

Gene Selection and Grounding. Each component maps to multiple concept templates representing distinct structural configurations in Digital Gene. GPT-5.6-solar selects the optimal template for each component instance by evaluating a candidate list of natural-language synopses that detail the representable geometry of each template.

For each component, a partial point cloud is extracted from its mask and depth data, normalized to a unit sphere of radius 0.5m, and processed by a geometric parameter estimation module to determine the geometric parameters Θ. Then, the rendered mesh is utilized as the input of FoundationPose (Wen et al., 2024) to estimate its 6-DoF pose T. Finally, the parameter refinement minimizes the Chamfer distance between the Digital Gene mesh and the observed point cloud, ensuring the gene optimally represents the physical object.

Part-level Program Selection. GPT-5.6-solar identifies the relevant part-level functional programs required for the task. It evaluates natural language synopses of the functionalities of the selected concept templates, and selects the essential programs $\{f_{\mathcal{P}}\}$ for the object understanding task.

Object-level Program Synthesis. GPT-5.6-solar acts as an in-context reasoner that inherits the selected part-level programs $\{f_{\mathcal{P}}\}$ to synthesize a task-oriented and executable object-level program $f_{\mathcal{O}}$. This program is instantiated with the parameters Θ_i and T_i, aligning the representation with the real-world object to facilitate comprehensive understanding.

This CoT approach allows GPT-5.6-solar to bridge object structure and function, utilizing Digital Gene as an explicit interface between physical space and programmatic logic. By delegating spatial and numerical tasks to specialized modules, the architecture allows GPT-5.6-solar to focus on its strengths in semantic reasoning. This decoupling of reasoning and perception ensures higher accuracy and broader generalizability across diverse object-understanding tasks. **Please refer to the Appendix.J for prompt examples.**

4.2 Geometric Parameter Estimation

A specialized geometric parameter estimation module is trained for each of the 263 concept templates. The architecture utilizes a PointNet++ (Qi et al., 2017) encoder to process partial point clouds and a three-layer MLP ($128 \times 512 \times 256 \times \dim(\Theta)$) activated with ReLU layers as a decoder to estimate the geometric parameters Θ.

Input. Component masks and depth maps are converted into partial point clouds, which is sampled to 2048 points and normalized to a unit sphere of radius 0.5m. The recorded scale factor is applied to the results of the geometric parameter estimation module, ensuring the final output reflects the object's accurate physical scale and geometry.

Training Settings. Our training set is derived from ConceptFactory (Sun et al., 2024), comprising all 4380 objects across 39 object categories and 263 concept templates annotated as Digital Gene. For each object from ConceptFactory asset, we capture depth data from four randomized viewpoints within altitude $\in [30°, 60°]$ and azimuth $\in (-90°, 90°)$ to establish partial point clouds from the component meshes. After point cloud normalization, these point clouds serve as input, with the corresponding normalized geometric parameters and meshes as the training supervision.

We optimize the module using a loss comprising geometric parameter loss and Chamfer distance loss, written as

$$\mathcal{L} = \lambda \mathcal{L}_{\delta}(\Theta, \hat{\Theta}) + (1 - \lambda)\mathrm{CD}(S_P, S_{\hat{P}}) \tag{4}$$

where $\lambda = 0.9$ balances two metrics, S_P denotes the point cloud sampled on part component P. $\mathcal{L}_{\delta}$ refers to Huber loss (Huber, 1992) with its threshold at 0.5. This approach minimizes the error between the ground truth mesh and the mesh instantiated from predicted parameters, ensuring both numerical and geometric accuracy. **Please refer to Appendix.E for more training details.**

4.3 6-DoF Pose Estimation

We employ FoundationPose (Wen et al., 2024) to estimate the 6-DoF pose parameters T for each individual component. The model is utilized off-the-shelf without additional fine-tuning, providing a robust and accurate pose initialization for the subsequent joint parameter refinement stage.

Input. For each component, we instantiate a 3D mesh using the predicted geometric parameters Θ. This established mesh is then paired with the RGB observation from the object-understanding task to serve as the input for FoundationPose (Wen et al., 2024).

4.4 Parameter Refinement

We employ simulated annealing to refine the initial parameters $\{\Theta_{i,0}, T_{i,0}\}$ obtained from the geometric parameter estimation and 6-DoF pose estimation modules for component P_i. At each iteration t, an element-wise stochastic perturbation $\alpha \in [\alpha_{\min}, \alpha_{\max}]$ is applied to generate a candidate:

$$\theta_{t+1}^{(j)} \leftarrow \alpha^{(j)} \theta_t^{(j)}. \tag{5}$$

Let D_t denote the one-way Chamfer distance between the instantiated part and the observed partial point cloud:

$$D(S_{P_{t+1}}, S^{\text{partial}}_{P_{\text{target}}}) = \sum_{x \in S^{\text{partial}}_{P_{\text{target}}}} \min_{y \in S_{P_{t+1}}} \|x - y\|_2^2. \tag{6}$$

Given $\Delta D = D_{t+1} - D_t$, the candidate is accepted with probability $\min\{1, \exp(-\Delta D/\tau_t)\}$, where τ_t is the current temperature. We use exponential cooling, $\tau_{t+1} = \rho\tau_t$, allowing early exploration and increasingly local refinement. Optimization terminates when the best distance falls below a threshold or the maximum number of iterations is reached. **Detailed pseudo code is provided in Appendix D.1.**

In our implementation, we set $[\alpha_{\min}, \alpha_{\max}] = [0.95, 1.05]$, initial temperature $\tau_0 = 0.1D_0$, and cooling rate $\rho = 0.9997$. These settings provide broad exploration initially and stable local refinement at low temperatures. Each target partial point cloud is extracted from the RGB mask and depth image and sampled to 4,096 points. We use at most $N = 20{,}000$ iterations and terminate early when the best distance is below $\varepsilon = 10^{-3}$.

5 Experiment

To evaluate the object understanding capability of DORIC, we focus on two questions: **1)** How effectively can DORIC understand object geometry and poses (Section 5.1)? **2)** Does DORIC robustly and generally facilitate object interaction tasks (Section 5.2)? In addition, we conduct system bottleneck analysis (Section 5.3) and CoT backbone replacement (Section 5.4) to validate our implementation.

5.1 Geometry and Pose Understanding

This evaluation focuses on understanding geometry and pose of objects. By using part-level programs that queries object geometries and poses, DORIC can infer the precise component-level configurations of an object. Furthermore, this evaluation measures the fidelity of the underlying geometric parameter estimation and pose estimation modules for Digital Gene representation.

Evaluation Setup. Our evaluation dataset follows GAPartNet (Geng et al., 2023), including 785 objects across 14 object taxonomy that overlap with Digital Gene taxonomy. Each object is placed in SAPIEN (Xiang et al., 2020) simulator, and RGBD observations with resolution 800×800 are captured from four different viewpoints with altitude $\in [30°, 60°]$ and azimuth $\in (-90°, 90°)$. We synthesize task descriptions targeting part-level manipulation on the GAParts. Ground truth geometric and 6-DoF parameters are obtained via the ConceptFactory annotation tool (Sun et al., 2024). The 6-DoF poses are additionally refined from GAPartNet's oriented bounding box annotations of each component to ensure precise gene annotation.

Evaluation Metrics. Our quantitative evaluation metrics include the following.

1) Geometry: We report **relative error ratio** E_Θ for the estimated geometric parameters Θ. We follow GeneVLM (Su et al., 2025) to analyze the parameter value distribution within the evaluation dataset for concept, and define the relative error as the absolute residual between the predicted and ground-truth parameters, normalized by the parameter's total range across all test instances.

$$E_\Theta = \frac{1}{M} \sum_{i=1}^{M} \frac{1}{|\Theta_i|} \sum_{\theta \in \Theta_i} \frac{|\theta - \hat{\theta}|}{|\theta_{\max} - \theta_{\min}|} \tag{7}$$

This normalization ensures that the error metric remains scale-invariant and comparable across different types of geometric parameters.

2) Pose: We evaluate the 6-DoF pose T of each component using **mean translation error** E_T, **mean rotation error** E_R, **mean IoU** of the components' bounding boxes with the ground truth ones following GAPartNet (Geng et al., 2023).

3) Mesh Similarity: We render Digital Gene-based meshes and calculating their chamfer distance against the ground-truth meshes from GAPartNet (Geng et al., 2023) (**NOT** their annotated genes), providing a comprehensive evaluation on both geometry and pose fidelity.

4) F-scores: The F-score represents the harmonic mean of precision and recall at specific error thresholds. We present F-score curves across varying thresholds for E_Θ, E_T, E_R, and CD, illustrating the system's robustness in geometric and pose understanding from a holistic perspective.

Furthermore, we provide qualitative results to facilitate a direct visual assessment of the fidelity between the output Digital Gene representations and the target objects, including samples from GAPartNet (Geng et al., 2023) and real world.

Baselines. We evaluate our method against GeneVLM (Su et al., 2025), which utilizes a unified multi-modal backbone to parse Digital Gene parameters. Since GeneVLM (Su et al., 2025) only takes an RGB image as input, the absolute scale of the parsed object may differ from the real scale. To make it comparable, we record the real scale and apply it to the GeneVLM generated parameters and then evaluate its performance on each metrics.

Results. Quantitative results are reported in Table 1. Compared with GeneVLM, our method substantially improves geometry and pose understanding, reducing the comprehensive CD metric by 33%. Figure 4 shows that ours consistently outperforms GeneVLM in F-score. However, its rotational errors exhibit a bimodal distribution: geometric ambiguity may cause inverse pose estimates for highly symmetric parts, while structurally distinctive components enable accurate estimates.

As illustrated in Figure 3, our method captures geometry, poses, and fine-grained details more precisely. In the eyeglass example (Figure 3-[Upper]), its predicted folding angle at the temple tips more closely matches the ground truth. For the faucet, it accurately estimates the handle poses, whereas GeneVLM produces rotational errors.

Figure 3-(b,c) show performance on real-world objects. Images in (b) are collected from the internet, with depths estimated using VGGT (Wang et al., 2025a), while those in (c) are captured using an Intel RealSense D415 camera. In both settings, our results exhibit geometric fidelity and precise object alignment, validating robustness in real-world geometry and pose understanding. Our method better preserves fine-grained geometry (*e.g.*, table-pole radius) and estimates rotations (*e.g.*, box folder).

5.2 Object Interaction Understanding

We evaluate the performance of DORIC in object understanding by conducting a quantitative analysis on object interaction tasks. These tasks serve as a representative benchmark to measure our method's ability to translate structural reasoning into executable actions.

Evaluation Setup. Our evaluation objects are collected from PartNet-Mobility (Xiang et al., 2020), including 1,170 objects across 21 objects that overlap with Digital Gene. Camera configurations are the same as in Section 5.1. Interactions are simulated within the SAPIEN simulator. We employ a Franka Panda Flying Gripper with a pair of parallel fingers to validate the executability of our synthesized programs. Each object is interacted 5 times with 1-budget.

Evaluation Metrics. We measure the interaction success rate (SR) for each task. Following GAPartNet (Geng et al., 2023), a success interaction is defined as opening up the target part for 90% of the motion range within 1,000 time-steps.

Baselines. Where2Act (Mo et al., 2021), Implicit3D (Zhong et al., 2023), ManipLLM (Li et al., 2024), and A3VLM (Huang et al., 2025) are adopted as baselines. These representative methods for object interaction understanding include affordance learning, articulation learning, heuristic policy, and MLLM reasoning. They are re-trained on 21 object categories for optimal performance to quantify the improvements provided by our method.

Results. The results reported in Table 2 show that our method outperforms the state-of-the-art method (Huang et al., 2025) with an improvement of 14.8%. Our method is capable of generating object-level programs for object interaction understanding, based on a concrete and precise foundation

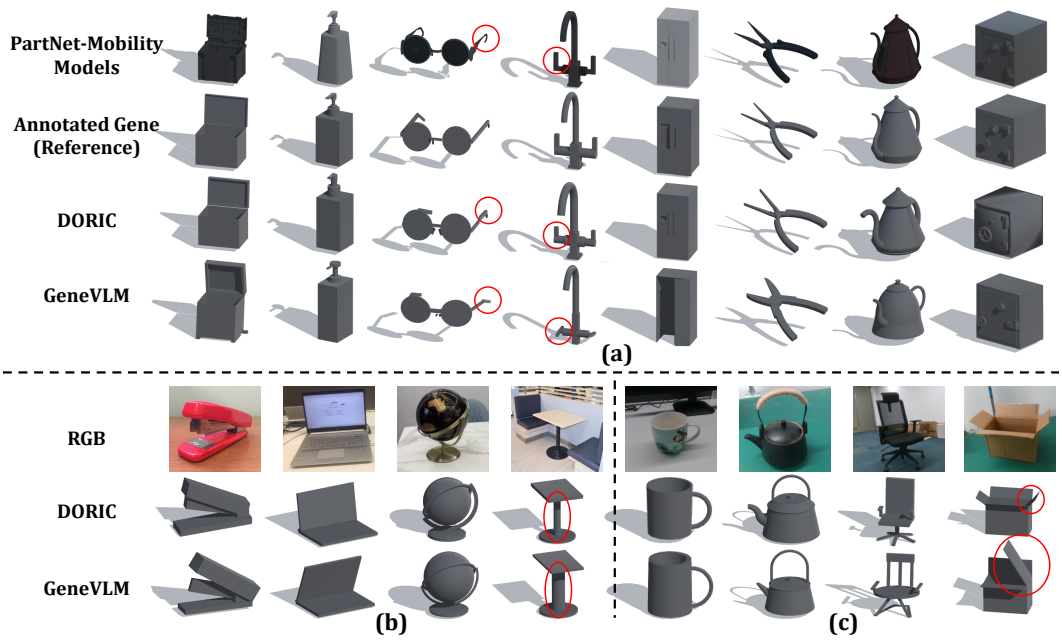


Figure 3: Digital Gene representation across simulated and real world. **(a):** Mesh comparison including original *PartNet-Mobility* models, *Annotated Genes* (by ConceptFactory annotation tool), and results from *DORIC* and *GeneVLM* based on estimated gene parameters. **(b):** Examples of images sourced from the internet. **(c):** Examples of physical objects captured in the real world.

Table 1: Results on object geometry (E_Θ) and pose (E_T, E_R) understanding. Chamfer distance (CD) comprehensively evaluates geometry and pose fidelity through mesh similarity.

Methods	$E_\Theta \downarrow$	$E_T \downarrow$	$E_R \downarrow$	mIoU↑	CD↓
GeneVLM	0.18	0.028	30.2	0.60	0.066
Ours	**0.13**	**0.025**	**23.5**	**0.63**	**0.044**

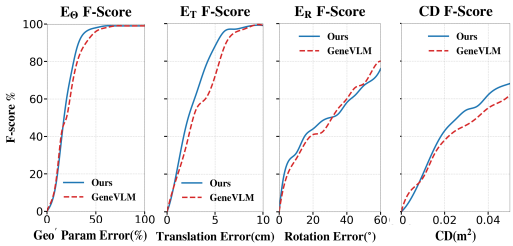


Figure 4: F-score curves of geometric parameter error, translation error, rotation error, and chamfer distance. Red curves refer to results of GeneVLM and blue ones refer to results of DORIC.

Table 2: Results on object interaction tasks using success rate (SR) as metric. All values represent the average success rate as a percentage.

Methods	Where2Act	Implicit3D	ManipLLM	A3VLM	Ours
SR	17.5	23.4	27.8	33.7	**38.7**

of geometry and pose understanding. **Please refer to Appendix.F.1 for detailed results, analysis, and real-world evaluation.**

5.3 System Bottleneck Analysis

Parameter Refinement. We assess the effectiveness of the parameter refinement module by removing it from the pipeline. The results shown in Table 3-Upper indicate that the refinement enhances the fidelity of object understanding, with a 13.3% improvement in geometry parameter estimation, a 15.3% improvement in CD regarding comprehensive geometry similarity, and a 9.3% improvement in object interaction understanding.

System Error Breakdown We provide a system error breakdown analysis by gradually replacing the output of each module with ground-truth values to identify system limitations. Specifically, the replacement of the "Object-Level Function" means that we directly provide precise grasp poses and motion according to part pose, following the heuristic policy in A3VLM (Huang et al., 2025), rather than using Digital Gene object-level programs to derive actions. The results are shown

Table 3: **[Upper]:** Results with and without the parameter refinement. **[Lower]:** System error breakdown analysis by gradually replacing outputs with ground-truth values.

Module	$E_\Theta \downarrow$	$E_T \downarrow$	$E_R \downarrow$	mIoU↑	CD↓	SR↑
w/ Param Refinement	0.13	0.025	23.5	0.63	0.044	38.7
w/o Param Refinement	0.15	0.028	24.2	0.60	0.052	35.4
RGB Segmentation	0.13	0.025	20.8	0.65	0.038	39.0
Geometric Parameters	-	0.022	15.2	0.72	0.035	42.5
6-DoF Pose Parameters	-	-	-	0.92	0.022	81.8
Object-Level Function	-	-	-	0.92	0.022	95.2

Table 4: Evaluation of DORIC using different MLLMs for CoT, including closed-source and open-source MLLMs. The open-source MLLMs are fine-tuned for evaluation.

Type	MLLM	$E_\Theta \downarrow$	$E_T \downarrow$	$E_R \downarrow$	mIoU↑	CD↓	SR↑
Closed-Source	GPT-5.6-solar	**0.13**	**0.025**	**23.5**	**0.63**	**0.044**	**38.7**
	Gemini-3.1	0.22	0.052	26.2	0.52	0.080	20.5
	GPT-5.6-luna	0.15	0.032	25.6	0.60	0.052	30.2
Open-Source	Qwen-3-VL-32B	0.30	0.036	22.8	0.58	0.152	26.2
	LLaMa-3.2-Vision	0.35	0.102	60.5	0.32	0.218	18.3

in Table 3-[Lower], where "-" denotes that relevant outputs are replaced by ground-truth values, zeroing the error and evaluation is not applicable. The results indicate that the major bottleneck is in pose understanding, where the replacement with ground-truth component poses leads to a 92.4% improvement in the comprehensive object interaction success rate.

For "RGB Segmentation", results show minimal improvement in object understanding. **We provide qualitative visualizations of the segmentation results using Nano-Banana Pro in Appendix.I. We also provide the visualization and analysis of failure cases. Please refer to Appendix.G.**

5.4 Ablation Studies: CoT Backbone

Since our framework relies on GPT-5.6-solar for CoT reasoning, we evaluate the system's performance and robustness by swapping the MLLM with other models, including GPT-5.6-luna, Gemini-3.1, Qwen-3-VL-32B (Yang et al., 2025), and LLaMa-3.2-11B-Vision (Touvron et al., 2023).

Setup. We fine-tune open-source MLLMs (Yang et al., 2025; Touvron et al., 2023) using LoRA (Hu et al., 2022) to support the four-stage CoT pipeline. The fine-tuning dataset comprises RGB observations paired with prompts and target outputs, utilizing the same object assets from ConceptFactory (Sun et al., 2024) used for training the geometry parameter estimation module. To ensure data consistency, prompts are generated via rule-based templates with dynamic placeholders, which are populated with specific component taxonomies, template synopses, and functional programs.

For the first three stages, target outputs (including structure trees and selections for concepts and part-level programs) are generated using scripts. For the final stage involving the synthesis of object-level programs $f_{\mathcal{O}}$, we leverage GPT-5.6-solar as an initialization and manually refine the resulting programs to ensure they correspond precisely to the specific interaction tasks. By focusing on object-interaction understanding, we ensure that the functional programs are physically grounded and consistently aligned with the kinematic requirements of each object category.

Results. The results shown in Table 4 demonstrate that the general object understanding and reasoning capabilities of closed-source MLLMs outperform open-source ones. Specifically, our implementation using GPT-5.6-solar achieves the highest success rate in object interaction and the lowest geometry and pose errors. We attribute the poor performance of open-source MLLMs in object interaction understanding to their failure to synthesize executable, physically aligned, and correct object-level programs.

6 Conclusion

We presented DORIC, a framework utilizing Digital Gene representations and CoT reasoning for robust RGBD object understanding from a single RGBD image. To overcome the practical limitations of using Digital Genes of existing methods, DORIC employs a hybrid architecture that decouples MLLM-based semantic reasoning from the precise numerical and spatial analysis of specialized modules. This design achieves high-fidelity parameter estimation, enables the flexible synthesis of object-level functional programs from pre-defined part-level programs, and allows for scalable incremental updates to new Digital Gene categories. Extensive experiments demonstrate that DORIC accurately parses Digital Gene parameters and effectively translates them into executable function programs, providing a practical way to use Digital Gene for physically precise object understanding.

AI USE STATEMENT

In this work, we used generative AI tools for 1) implementing DORIC's CoT backbone, and 2) annotating MLLM finetuning data, which is described in Appendix in detail. We have not used generative AI tools for generating research idea, methodology or experiment design, dataset preparation, result interpretation, or translation. And tasks involving mathematical claims or proofs are not applicable to this work. Additionally, we used generative AI tools for grammar checking and phrasing. We have reviewed all AI-assisted outputs and manuscript edits for correctness. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

DORIC utilizes MLLMs for executable object-level program synthesis, where MLLM hallucinations may cause physical execution to diverge from the intended task especially for object interaction. This misalignment may lead to potential risk of property damage or equipment failure.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, we report key hyper-parameters in our methods, data splits, method for processing and annotating training data, detailed evaluation settings, and computing resources. For clear description, We provide pseudo code for key algorithms of our method, and prompt examples for MLLMs. We also provide digital gene code examples for better understanding and reproducibility in our supplementary material. For the MLLMs involved in our experiments, experiments are conducted and the MLLMs are called during June-August 2026.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on robot learning*, pp. 287–318. PMLR, 2023.

Dmitry Chetverikov, Dmitry Svirko, Dmitry Stepanov, and Pavel Krsek. The trimmed iterative closest point algorithm. In *2002 International Conference on Pattern Recognition*, volume 3, pp. 545–548. IEEE, 2002.

Wenbo Cui, Chengyang Zhao, Songlin Wei, Jiazhao Zhang, Haoran Geng, Yaran Chen, Haoran Li, and He Wang. Gapartmanip: A large-scale part-centric dataset for material-agnostic articulated object manipulation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14791–14798. IEEE, 2025.

Yan Ding, Xiaohan Zhang, Chris Paxton, and Shiqi Zhang. Task and motion planning with large language models for object rearrangement. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2086–2092. IEEE, 2023.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: an embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 8469–8488, 2023.

Haoran Geng, Helin Xu, Chengyang Zhao, Chao Xu, Li Yi, Siyuan Huang, and He Wang. Gapartnet: Cross-category domain-generalizable object perception and manipulation via generalizable and actionable parts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7081–7091, 2023.

Ziyu Guo, Renrui Zhang, Xiangyang Zhu, Yiwen Tang, Xianzheng Ma, Jiaming Han, Kexin Chen, Peng Gao, Xianzhi Li, Hongsheng Li, et al. Point-bind & point-llm: Aligning point cloud

with multi-modality for 3d understanding, generation, and instruction following. *arXiv preprint arXiv:2309.00615*, 2023.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Siyuan Huang, Iaroslav Ponomarenko, Zhengkai Jiang, Xiaoqi Li, Xiaobin Hu, Peng Gao, Hongsheng Li, and Hao Dong. Manipvqa: Injecting robotic affordance and physically grounded information into multi-modal large language models. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7580–7587. IEEE, 2024.

Siyuan Huang, Haonan Chang, Yuhan Liu, Yimeng Zhu, Hao Dong, Abdeslam Boularias, Peng Gao, and Hongsheng Li. A3vlm: Actionable articulation-aware vision language model. In *Conference on Robot Learning*, pp. 1675–1690. PMLR, 2025.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pp. 9118–9147. PMLR, 2022.

Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*, pp. 492–518. Springer, 1992.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Yuheng Ji, Huajie Tan, Jiayu Shi, Xiaoshuai Hao, Yuan Zhang, Hengyuan Zhang, Pengwei Wang, Mengdi Zhao, Yao Mu, Pengju An, et al. Robobrain: A unified brain model for robotic manipulation from abstract to concrete. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1724–1734, 2025.

Xiaoqi Li, Mingxu Zhang, Yiran Geng, Haoran Geng, Yuxing Long, Yan Shen, Renrui Zhang, Jiaming Liu, and Hao Dong. Manipllm: Embodied multimodal large language model for object-centric robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18061–18070, 2024.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023a.

Xueyi Liu, Ji Zhang, Ruizhen Hu, Haibin Huang, He Wang, and Li Yi. Self-supervised category-level articulated object pose estimation with part-level se (3) equivariance. In *The Eleventh International Conference on Learning Representations*, 2023b.

Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 909–918, 2019.

Kaichun Mo, Leonidas J Guibas, Mustafa Mukadam, Abhinav Gupta, and Shubham Tulsiani. Where2act: From pixels to actions for articulated 3d objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6813–6823, 2021.

Chuanruo Ning, Ruihai Wu, Haoran Lu, Kaichun Mo, and Hao Dong. Where2explore: Few-shot affordance learning for unseen novel categories of articulated objects. *Advances in Neural Information Processing Systems*, 36:4585–4596, 2023.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

Zekun Qi, Runpei Dong, Shaochen Zhang, Haoran Geng, Chunrui Han, Zheng Ge, Li Yi, and Kaisheng Ma. Shapellm: Universal 3d object understanding for embodied interaction. In *European Conference on Computer Vision*, pp. 214–238. Springer, 2024.

Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *Advances in neural information processing systems*, 35:23192–23204, 2022.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11523–11530. IEEE, 2023.

Jiadi Su, Bizhe Bai, Jiude Wei, Zhou FANG, Mingyang Sun, Qichen He, Yi Yang, Tao Chen, Jianhua Sun, Cewu Lu, and Pengfei Liu. Genevlm: Automated parsing executable digital gene from a single image. https://openreview.net/pdf?id=JTVSWcM0xM, 2025.

Jianhua Sun and Cewu Lu. Digital gene: Learning about the physical world through analytic concepts. *arXiv preprint arXiv:2504.04170*, 2025.

Jianhua Sun, Yuxuan Li, Longfei Xu, Nange Wang, Jiude Wei, Yining Zhang, and Cewu Lu. Conceptfactory: Facilitate 3d object knowledge annotation with object conceptualization. *Advances in Neural Information Processing Systems*, 37:75454–75467, 2024.

Jianhua Sun, Yuxuan Li, Jiude Wei, Longfei Xu, Nange Wang, Yining Zhang, and Cewu Lu. Artipg: A toolbox for procedurally synthesizing large-scale and diverse articulated objects with rich annotations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6396–6405, 2025a.

Jianhua Sun, Yuxuan Li, Longfei Xu, Jiude Wei, Liang Chai, and Cewu Lu. Discovering conceptual knowledge with analytic ontology templates for articulated objects. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 14681–14689, 2025b.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Hugues Thomas, Yao-Hung Hubert Tsai, Timothy D Barfoot, and Jian Zhang. Kpconvx: Modernizing kernel point convolution with kernel attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5525–5535, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2642–2651, 2019.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 5294–5306, 2025a.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025b.

Xiaofei Wang, Mingliang Han, Tianyu Hao, Cegang Li, Yunbo Zhao, and Keke Tang. Advgrasp: Adversarial attacks on robotic grasping from a physical perspective. *arXiv preprint arXiv:2507.09857*, 2025c.

Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17868–17879, 2024.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11097–11107, 2020.

Runsen Xu, Xiaolong Wang, Tai Wang, Yilun Chen, Jiangmiao Pang, and Dahua Lin. Pointllm: Empowering large language models to understand point clouds. In *European Conference on Computer Vision*, pp. 131–147. Springer, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Xianchao Zeng, Xinyu Zhou, Youcheng Li, Jiayou Shi, Tianle Li, Liangming Chen, Lei Ren, and Yong-Lu Li. Diagnose, correct, and learn from manipulation failures via visual symbols. *arXiv preprint arXiv:2512.02787*, 2025.

Zirui Zhao, Wee Sun Lee, and David Hsu. Large language models as commonsense knowledge for large-scale task planning. *Advances in neural information processing systems*, 36:31967–31987, 2023.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

Chengliang Zhong, Yuhang Zheng, Yupeng Zheng, Hao Zhao, Li Yi, Xiaodong Mu, Ling Wang, Pengfei Li, Guyue Zhou, Chao Yang, et al. 3d implicit transporter for temporally consistent keypoint discovery. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3869–3880, 2023.

A APPENDIX

This is the appendix of submission **DORIC: Digital-Gene-based Object Understanding from a single RGBD Image via Chain-of-Thought Reasoning**. The appendix is organized as follows:

- Appendix.B: Discussion of the reasonability of using closed-source MLLMs.
- Appendix.C: Details and examples of the part-level programs we defined on Digital Gene.
- Appendix.D: Pseudo code of parameter refinement algorithm.
- Appendix.E: Description of training data and evaluation setups.
- Appendix.F: Object interaction task definitions and real-world experiments.
- Appendix.G-H: Failure cases, analysis, and discussions of generalizability.
- Appendix.I: Illustration and analysis of image segmentation by Nano-Banana Pro.
- Appendix.J: Detailed examples of prompts for the CoT in our method.
- Appendix.K-L: Limitation, future works, and computing resource report.

We provide real-world experiment videos and Digital Gene sample codes in **experiment_video** and **digital_gene_code_examples** folder of the supplementary material. Please refer to them for better understanding of our method.

B WHY USING ESTABLISHED CLOSED-SOURCE MLLMS?

While training a foundational backbone from scratch or fine-tuning one is a conventional choice for maximizing perceived academic contribution and reproducibility, the rapid evolution of closed-source MLLMs presents a new paradigm. Given that the MLLMs have reached a strong level of reasoning and commonsense understanding that is difficult to replicate at the individual research level, we propose that a strategic **tradeoff** exists between **utilizing closed-source MLLMs** and **developing a novel base model**. This approach is valid as long as the MLLM's capability scope is clearly delimited and does not overlap with the core contribution of the research.

Capability Scope and Core Contributions. When treating MLLMs as tools (like FoundationPose (Wen et al., 2024)) it is necessary to understand their capability scope and ensure their capability does not intrude upon the core contribution. As a representative of closed-source MLLMs, GPT *excels* in language-level task organization, image editing, and code generation. This allows us to utilize it for task decomposition, object part detection, and the in-context generation of functional programs. In contrast, it *cannot* independently perform precise physical-level numerical deduction and estimate digital gene parameters from single RGBD inputs. This functional gap ensures that our **core contribution**, "grounding genes", is achieved entirely **by our proposed modules**. Therefore, DORIC is **NOT a "GPT wrapper"**, as it cannot rely on GPT (or any other established MLLMs) for acquiring the physical-level knowledge (*e.g.*, structure, 6-DoF pose) that is essential for understanding object interaction. We ensure that MLLMs are restricted to high-level reasoning, while the solutions to the core physical problems constitute our primary academic contribution.

Engineering Tradeoffs. DORIC pipeline requires the 6-DoF poses of object parts via real-time inference from RGBD images. Among potential solutions (*e.g.*, FoundationPose (Wen et al., 2024), ICP (Chetverikov et al., 2002)), FoundationPose is the optimal choice for balancing performance and efficiency. Developing a custom algorithm for inference-time pose estimation would constitute an independent research topic exceeding the scope of this work. **Similarly, the decision to use closed-source MLLMs for problems within their capability scope is an engineering trade-off.** Problems such as task decomposition and code generation can be effectively addressed by closed-source MLLMs. Utilizing these models improves research efficiency and provides clear direction for early-stage investigations. At inference time, the choice between resource-intensive local deployments of open-source models and reliance on external costly closed-source MLLMs also represents a trade-off in computational resource management.

Furthermore, in an era of rapid foundational model iteration, adopting a modular architecture also means that **DORIC could benefit naturally as MLLM capabilities evolve**.

C PART-LEVEL PROGRAMS

In this section we demonstrate the part-level programs that we have already defined on the Digital Gene concept templates. We also discuss about the potential object-level programs that they can contribute to. These programs are categorized as follows:

Poses. A component's pose comprises its translation, rotation, and scale, as detailed in the implementation below. **Utility:** Relative poses between adjacent components can be computed from their respective poses. Components' bounding boxes can be derived from poses for collision detection and the formulation of heuristic interaction policies.

```
def get_poses(self):
   position = self.position
   rotation = self.rotation
   scale = np.max(self.vertices - np.mean(self.vertices, axis=0), axis=0)
   return {
      "position": position,
      "rotataion": rotation,
      "scale": scale
   }
```

Kinematics. The kinematic part-level programs specify motion constraints, such as the **translation axis** for prismatic components (*e.g.*, drawers and buttons) and the **rotation axis** for revolute parts (*e.g.*, doors, laptop lids, and bottle caps). The program for `lid_translation_axis` and `door_rotation_axis` are shown below. **Utility:** When integrated with affordance programs, these kinematic priors facilitate the formulation of precise object interaction policies, including the synthesis of robotic gripper motion trajectories.

```
def get_lid_translation_axis(self):
   rot_mat_x = get_rodrigues_matrix([1, 0, 0], self.rotation[0])
   rot_mat_y = get_rodrigues_matrix([0, 1, 0], self.rotation[1])
   rot_mat_z = get_rodrigues_matrix([0, 0, 1], self.rotation[2])
   R_global = np.matmul(rot_mat_z, np.matmul(rot_mat_y, rot_mat_x))

   translation_axis = np.matmul(R_global, self.local_translation_axis)
   return {
      "lid_translation_axis": translation_axis
   }

def get_door_rotation_axis(self):
   self.joint_origin = apply_transformation(local_pivot, position,
       rotation_rad)[0]

   rot_mat_x = get_rodrigues_matrix([1, 0, 0], rotation_rad[0])
   rot_mat_y = get_rodrigues_matrix([0, 1, 0], rotation_rad[1])
   rot_mat_z = get_rodrigues_matrix([0, 0, 1], rotation_rad[2])

   R_global = rot_mat_z
   R_global = np.matmul(rot_mat_x, R_global)
   R_global = np.matmul(rot_mat_y, R_global)

   rotation_axis = np.matmul(R_global, local_axis)

   return {
      "door_rotation_axis": rotation_axis
   }
```

Affordances. Affordances include **grasp poses** (*e.g.*, for handles and doors), **containment volumes** (*e.g.*, for closets and drawers), and **functional regions** (*e.g.*, for buttons and heaters). Here we use `get_handle_grasp_pose` program as a code example. **Utility:** These affordance knowledge can be integrated with kinematic constraints or semantic knowledge from other objects to facilitate complex task execution and manipulation planning.

```
def get_handle_grasp_pose(self):
    grasp_z = mounting_size[2] + grip_size[2] / 2
    local_grasp_center = np.array([[0, 0, grasp_z]])
    grasp_position = apply_transformation(
        local_grasp_center, position, rotation, rotation_order="ZXY"
    )

    rot_mat_x = get_rodrigues_matrix([1, 0, 0], rotation[0])
    rot_mat_y = get_rodrigues_matrix([0, 1, 0], rotation[1])
    rot_mat_z = get_rodrigues_matrix([0, 0, 1], rotation[2])
    R_global = rot_mat_z
    R_global = np.matmul(rot_mat_x, R_global)
    R_global = np.matmul(rot_mat_y, R_global)

    fix_mat = np.array([
        [0, 1, 0],
        [-1, 0, 0],
        [0, 0, 1]
    ])
    grasp_rotation_matrix = np.matmul(R_global, fix_mat)

    grasp_pose = np.matmul(grasp_rotation_matrix, grasp_position)

    return {
        "handle_grasp_pose": grasp_pose
    }
```

D Parameter Refinement Algorithm

D.1 Pseudo Code

In this section we provide the pseudo code to formulate the algorithm of parameter refinement, as shown in Algorithm.1.

D.2 Simulated Annealing vs. Gradient-Based Refinement

As described in Section 4.4, we employ simulated annealing to refine the NumPy-implemented Digital Gene parameters. For each component, we initialize the temperature as $\tau_0 = 0.1D_0$ and apply exponential cooling $\tau_{t+1} = \rho\tau_t$ with $\rho = 0.9997$. Improving candidates are always accepted, whereas worse candidates are accepted with probability $\exp(-\Delta D/\tau_t)$, enabling early exploration followed by local refinement.

We also evaluate gradient-based refinement by translating the templates into PyTorch and minimizing Chamfer distance using SGD. It achieves a CD of 0.052 and an SR of 38.0%, close to simulated annealing's 0.044 CD and 38.7% SR. However, gradient-based refinement requires $0.20\,\mathrm{s}$, which is $10\times$ slower than simulated annealing at $0.02\,\mathrm{s}$. Simulated annealing therefore provides a better accuracy-efficiency trade-off in our implementation.

E Training Details and Setups

E.1 Geometric Parameter Estimation

In Section 4.2, we introduced some information regarding the training of the geometric parameter estimation modules. Here, we provide more detailed information.

Training Data Distribution. Table 5 lists the training data volume for geometric parameter estimation. The first line of each object category shows the number of data prototype acquired from ConceptFactory (Sun et al., 2024), totaling 4,380 training data.

Algorithm 1 Digital Gene Parameter Refinement

Require: Initial parameters $\{\Theta_{i,0}, T_{i,0}\}_{i=1}^{M}$, target point clouds $\{S_{P_{i,\text{target}}}^{\text{partial}}\}_{i=1}^{M}$, maximum iterations N, threshold ε, cooling rate ρ
Ensure: Refined parameters $\{\Theta_i^*, T_i^*\}_{i=1}^{M}$

for each component $i \in [1, M]$ **do**
  $t \leftarrow 0$, $(\Theta_{i,t}, T_{i,t}) \leftarrow (\Theta_{i,0}, T_{i,0})$
  $P_t \leftarrow P(\Theta_{i,t}, T_{i,t})$
  $D_t \leftarrow D(S_{P_t}, S_{P_{i,\text{target}}}^{\text{partial}})$
  $\tau_t \leftarrow 0.1 D_t$, $(\Theta_i^*, T_i^*, D^*) \leftarrow (\Theta_{i,t}, T_{i,t}, D_t)$
  while $t < N$ and $D^* \geq \varepsilon$ **do**
    $(\Theta', T') \leftarrow (\Theta_{i,t}, T_{i,t})$
    for each $q^{(j)} \in \Theta' \cup T'$ **do**
      Sample $\alpha^{(j)} \sim \mathcal{U}(\alpha_{\min}, \alpha_{\max})$; $q^{(j)} \leftarrow \alpha^{(j)} q^{(j)}$
    end for
    $P' \leftarrow P(\Theta', T')$; $D' \leftarrow D(S_{P'}, S_{P_{i,\text{target}}}^{\text{partial}})$
    $\Delta D \leftarrow D' - D_t$; sample $u \sim \mathcal{U}(0, 1)$
    if $\Delta D \leq 0$ or $u < \exp(-\Delta D / \tau_t)$ **then**
      $(\Theta_{i,t+1}, T_{i,t+1}, D_{t+1}) \leftarrow (\Theta', T', D')$
    else
      $(\Theta_{i,t+1}, T_{i,t+1}, D_{t+1}) \leftarrow (\Theta_{i,t}, T_{i,t}, D_t)$
    end if
    if $D_{t+1} < D^*$ **then**
      $(\Theta_i^*, T_i^*, D^*) \leftarrow (\Theta_{i,t+1}, T_{i,t+1}, D_{t+1})$
    end if
    $\tau_{t+1} \leftarrow \rho \tau_t$; $t \leftarrow t + 1$
  end while
end for
return $\{\Theta_i^*, T_i^*\}_{i=1}^{M}$

Training Data Augmentation. The dataset comprises 4,380 objects across 263 distinct concept templates, each corresponding to a specific geometric parameter estimation module. As indicated in Table 5, certain categories have few data prototype, which can lead to suboptimal performance for their respective module. To address this, we implement two data augmentation techniques, aiming at preventing overfitting. **1) Stochastic Rotation:** We apply random rotations around the object's center, with roll-yaw-pitch values sampled uniformly from $[-30°, 30°]$. This process alters the orientation of the input point clouds without modifying the intrinsic geometric parameters of the concept templates. This not only expands the dataset volume but also enhances the model's robustness to varied viewpoints and partial point cloud observations. **2) Gaussian Noise:** We apply Gaussian noise with $\mu = 0, \sigma = 0.05$ onto the point cloud, introducing perturbations without damaging the point cloud overall shape. This process expands the dataset volume and improves the model's resilience to point cloud noise. The augmented dataset volume is shown in the second line of each object category in Table 5. An average of about 634 concept template instances are used for the training of each module.

Hyper-parameters. The geometric parameter estimation modules are trained with batch size $= 128$, epochs $= 300$, learning rate $= 10^{-4}$, Adam optimizer with weight decay $= 5 \times 10^{-4}$, gradient clip at 1.0. The loss balancing hyperparameter λ (Equation (4)) is determined using E_Θ (Equation (7)) as the evaluation metric through a grid search. We set $\lambda = 0.9$ as it consistently yields the best performance across the majority of cases.

E.2 Open-Source MLLM

Section 5.4 discussed the performance of using different MLLM for CoT, involving open-source MLLMs that requires fine-tuning.

Table 5: Statistics of training data volume for geometric parameter estimation modules. "Prototype" refers to data volume from ConceptFactory. "Augmented" refers to practical training data volume after augmentation. The abbreviations of the object categories are listed as follows: Bottle, Box, Bucket, Chair, Clip, Dishwasher, Dispenser, Display, Door, Doorhandle, Eyeglasses, Faucet, Fridge, Globe, Gluestick, Kettle, Kitchenpot, Knife, Laptop, Lighter, Microwave, Mug, Oven, Pen, Pliers, Refrigerator, Ruler, Safe, Scissors, Shampoo, Shaver, StorageFurniture, Switch, Table, Trashcan, USB, Washing-machine, and Window. "TTL" refers to total volume.

	Bot	Box	Bkt	Chr	Clp	Dsw	Dsp	Dpl	Dor	Drh	Egl	Fct	Fdr	Glb	Gsk	Ket	Ktp	Knf	Ltp	Lgt
Prototype	446	98	54	746	49	185	86	262	28	3	119	250	8	38	44	24	24	76	48	60
Augmented	4460	4900	5400	5968	4900	5550	3440	5240	5600	3000	3570	5000	4000	3800	4400	3600	3600	2280	4800	3600
	Mcw	Mug	Ovn	Pen	Plr	Rfg	Rlr	Saf	Scs	Smp	Shv	Stp	Stf	Swt	Tab	Tcn	Usb	Wsm	Win	TTL
Prototype	76	113	18	112	119	65	16	22	100	46	9	34	637	84	136	124	54	17	51	4380
Augmented	2280	3390	3600	3360	3570	3900	3200	4400	5000	4600	3600	3400	5096	4200	5440	4960	5400	5100	5100	166704

Data Preparation. The fine-tuning dataset consists of RGB images, prompts, and ground-truth answers for each of the four stages in the CoT pipeline. The fine-tuning dataset uses the same objects dataset used for training the geometric parameter estimation modules. **1) Image:** Each object is rendered in SAPIEN (Xiang et al., 2020), and 800×800 resolution images are captured from four different four viewpoints with altitude $\in [30°, 60°]$ and azimuth $\in [-90°, 90°]$. To enhance the realism and robustness of the data and to maintain the performance of the MLLMs for real-world images, we employ Nano-Banana Pro to manipulate the images, generating images featuring objects with diverse textures and geometries while ensuring that all components remain pixel-wisely aligned as much as possible, as shown in Figure 5. **2) Prompts:** The prompts are generated by scripts as demonstrated in Appendix.J, conditioned on the object metadata. **3) Ground-Truth Answers:** The answers are also structured and generated via scripts conditioned on object metadata, like the `Example Output` in the prompts in Appendix.J. Specifically, for the *Object-Level Program Synthesis* stage, we utilize GPT-5.6-solar to produce programs, which are then manually refined to ensure high-fidelity functional execution as ground-truth answers.

Hyper-parameters. We adopt LoRa (Hu et al., 2022) to fine-tuning the MLLMs (Yang et al., 2025; Touvron et al., 2023). We set lora rank $= 8$, learning rate $= 10^{-4}$, epochs $= 3$, batch size $= 1$, using a cosine scheduler with warm-up rate $= 0.1$, and BF16 precision for both training and inference.

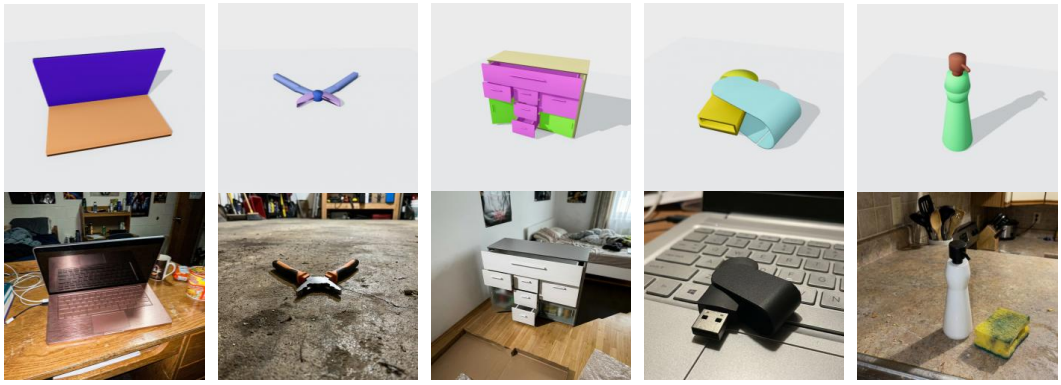


Figure 5: Visualization of MLLM fine-tuning image data. The top row displays raw mesh models from ConceptFactory rendered in SAPIEN. The bottom row illustrates the corresponding images manipulated by Nano-Banana Pro, conditioned on the source meshes to enhance realism and texture diversity while maintaining pixel-wise alignment as much as possible.

E.3 Data Split for Geometry and Pose Understanding

Here we provide the data split of evaluation for geometry and pose understanding (Section 5.1). As reported in Table 6, the evaluation includes 785 objects across 14 object taxonomy.

Table 6: Number of evaluation objects in each category for geometry and pose understanding.

Category	Box	Buc	Dis	Dor	Fdr	Pot	Ltp	Mic	Ovn	Saf	Stf	Tab	Tsc	Wsm	Total
#Eval objects	65	68	41	14	37	20	48	16	23	29	256	77	74	17	785

E.4 MLLM Versions and Inference Settings.

We accessed all closed-source models through their official APIs during June–August 2026. The main DORIC pipeline used `gpt-5.6-solar` for task decomposition, structural reasoning, Digital Gene selection, part-level program selection, and object-level program synthesis, while `gemini-3-pro-image` (Nano Banana Pro) was used for component segmentation. In the backbone replacement experiments, we additionally evaluated `gpt-5.6-luna` and `gemini-3.1-pro-preview`. These closed-source models were used without fine-tuning.

For the open-source backbones, we used `Qwen3-VL-32B-Instruct` and `Llama-3.2-11B-Vision-Instruct`. Both models were fine-tuned using the LoRA configuration described above and evaluated in BF16 precision. For each backbone, we used one response per query and fixed the prompts and decoding configuration across all evaluation samples. No test-time voting, self-consistency sampling, or best-of-N selection was applied. Because hosted API models may be updated by their providers, the reported closed-source results correspond specifically to the model versions served during the stated evaluation period. The exact prompts used for each stage are provided in Appendix J.

F Details of Object Interaction Understanding

F.1 Simulation Interaction Tasks

Section 5.2 presents an evaluation of object understanding tasks regarding interaction, involving 21 objects categories from PartNet-Mobility (Xiang et al., 2020) that has overlap with Digital Gene (Sun & Lu, 2025). Here we report the specific object interaction tasks in Table 7 for the 21 objects categories involved.

Table 7: List of specific tasks in object interaction.

Category	Tasks
Bottle	Open Lid
Box	Rotate Lid
Dispenser	Press Lid
Door	Open Door
Faucet	Turn on/off Switch
Fridge	Open Door (via Handle)
Kettle	Switch Handle
Kitchenpot	Lift Lid
Laptop	Rotate Screen
Lighter	Press Button, Rotate Lid
Microwave	Open Door (via Handle), Press Button
Oven	Open Door (via Handle), Press Button
Pliers	Open Pliers
Safe	Open Door (via Handle), Press Button
Scissors	Open Scissors
Storage	Open Door (via Handle), Open Drawer (via Handle)
Stapler	Rotate Lid, Press Lid
Table	Open Door (via Handle), Open Drawer (via Handle)
Trashcan	Open Lid
Usb	Rotate Cap, Open Cap
Washing'	Open Door, Open Lid

Detailed Interaction Results. Here we provide the detailed object interaction results for each of the 21 object categories in Table 8. Error bars are provided alongside the mean results (**AVG**), calculated over five independent runs.

Table 8: Evaluation results on object interaction tasks across different object categories with four baselines. All values represent the average success rate as a percentage.

Methods	Bot	Box	Dsp	Dor	Fct	Fdr	Ket	Lpt	Lgt	Mcw	Ovn	Pli	Pot	Saf	Scs	Stf	Stp	Tab	Tcn	Usb	Wsm	**AVG**
Where2Act	10.4	6.8	20.5	31.5	17.1	31.2	9.9	10.2	16.5	31.3	35.6	5.2	6.2	20.4	15.0	37.1	12.3	17.8	17.1	5.5	9.1	17.5±0.8
Implicit3D	16.3	14.5	30.6	45.8	18.2	30.5	10.5	20.0	25.2	38.3	38.2	8.5	11.5	28.9	12.5	45.2	38.0	2.0	15.8	26.5	15.3	23.4±1.4
ManipLLM	21.3	11.0	38.2	50.2	16.5	34.4	10.6	22.5	28.8	41.5	40.7	10.6	12.4	37.0	15.8	47.0	40.5	39.1	15.0	31.3	19.1	27.8±0.9
A3VLM	20.1	12.2	45.3	56.2	24.1	40.8	20.1	25.2	40.0	49.7	45.8	25.7	11.0	40.0	25.5	53.3	55.2	39.1	18.2	38.5	22.3	33.7±0.8
Ours	**24.2**	**15.9**	**48.2**	**58.4**	**28.0**	**46.7**	**24.3**	**27.1**	**41.2**	**52.2**	**51.5**	**40.0**	**15.5**	**50.6**	**38.1**	**58.6**	**55.5**	**51.6**	**22.9**	**40.0**	**22.7**	**38.7**±1.2

Table 9: Success rate of real-world object interaction. All values are in percentage.

Bottle	Box	Dispenser	Door	Kettle	Laptop	Microwave	Pot	Suitcase	Storage	Stapler	Trashcan	**AVG**
88.0	96.0	98.0	82.0	90.0	96.0	88.0	94.0	98.0	90.0	92.0	94.0	92.2

F.2 Real-World Object Interaction

Real-World Settings. Real-world experiments are conducted using a Flexiv Rizon robotic arm equipped with a parallel gripper, with RGBD data captured by an Intel RealSense D415 camera. All objects are positioned within the shared workspace to ensure they are simultaneously within the camera's field of view and the robot's reachable area.

Quantitative Results. The quantitative results of real-world object interaction are shown in Table 9.

Real-world Computational Overhead. Our framework maintains high computational efficiency, ensuring that inference overhead does not impede real-time robotic interaction. **Task Planning** incurs a *one-time initialization* cost of 0.8s. For each individual sub-task, the system executes a sequence of modular components, comprising **Segmentation** (1.8s), **Geometry Parameter Estimation** (0.4s), **Pose Estimation** (0.9s), **Parameter Optimization** (0.02s), and **Program Synthesis** (0.8s), resulting in a cumulative per-sub-task latency of 3.92s. When contrasted with a typical robotic execution cycle of over 20s, this total latency is negligible and supports real-time performance.

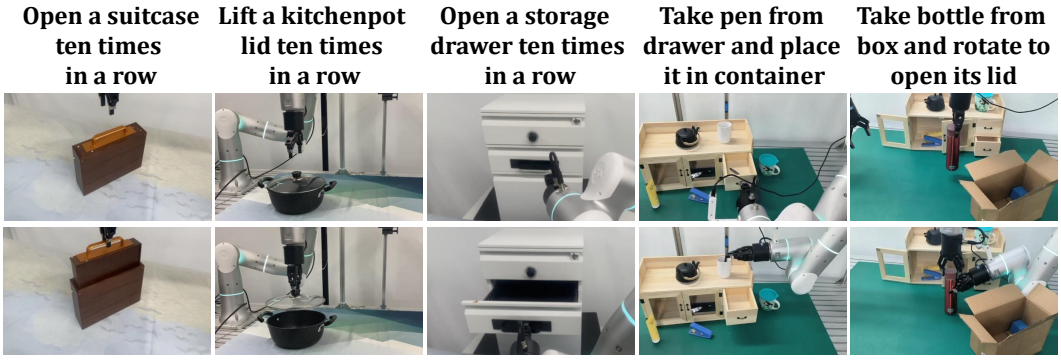


Figure 6: Visualization of five real-world object interaction experiments. Please refer to **experiment_video** folder of the supplementary material for the videos demonstrations.

Video Examples. The video examples (illustrated in Figure 6) of real-world object interaction experiments are provided in the **experiment_video** folder of the supplementary material. The videos involve three single-object interaction tasks: **opening a suitcase**, **lifting a kitchenpot lid**, and **opening a storage drawer**, each conducted over ten consecutive trials to demonstrate our system robustness. Additionally, we include two multi-object interaction tasks: **retrieving a pen from a storage drawer and placing it into a container**, and **take out a bottle from a box and rotate to open its lid**.

Real-World Robustness. We analyze and categorize real-world failure modes as shown in Figure 7-(a). Errors from real-world attributes (in warm color) constitute 41% of all failure cases. Notably, only 12% of tasks fail in the analysis, real-world attributes impact only 4.9% of total tasks, demonstrating our robustness in the real world.

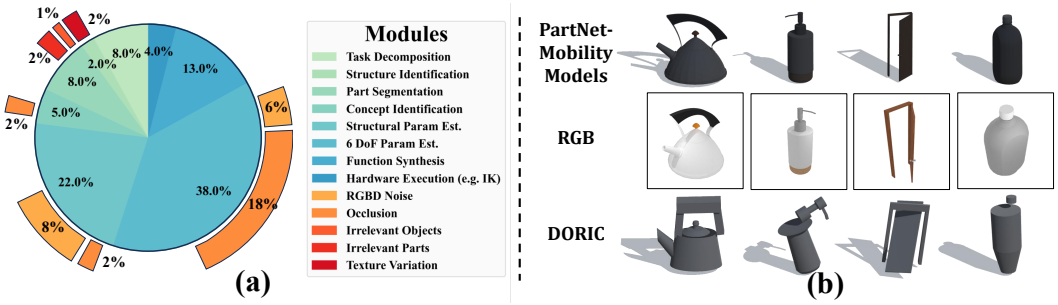


Figure 7: **(a)**: Failure mode analysis in real-world experiments. **(b)**: Illustration of failure cases for geometric and pose understanding. The objects are *kettle*, *dispenser*, *door*, and *bottle*.

G Failure Cases and Analysis

Figure 7-(b) illustrates four primary failure modes in geometric and pose understanding. These cases are categorized as follows:

Digital Gene Selection Failure. As demonstrated by the *kettle* example, DORIC selects a "C_Shaped_Handle" where a "Curved_Handle" would be more appropriate. Because the geometric structure of the selected template is fundamentally incompatible with the target object, the geometric understanding fails.

Pose Estimation Failure Triggered by Geometric Inaccuracy. In the *dispenser* case, error geometric parameters for the body and spray nozzle occur. Then, using an inaccurate mesh as a reference prevents FoundationPose (Wen et al., 2024) from successfully matching the poses to the observation.

Pose Understanding Failure Triggered by Perspective and Occlusion. As shown by the *door*, the randomized camera view in the reference RGB image makes the door board occluded and difficult to identify. Given such limited visual information, FoundationPose fails to identify sufficient pixel correspondences for reliable pose matching.

Pose Understanding Failure Triggered by Pose Ambiguity. In the *bottle* example, FoundationPose predicts an upside-down pose. By tracking the scoring process of FoundationPose, we found that the symmetrical nature of the object causes the incorrect candidate pose to receive a similar and even slightly higher score than the ground truth pose, leading to a catastrophic pose estimation error.

Cascaded Error. In Table 3, we provide an ablation study on the contribution of each module, which also means there exists **cascaded error propagation** within our framework. Specifically, the refinement stage is introduced to mitigate upstream errors, yielding a significant $+15.3\%$ performance gain. The errors introduced by segmentation (0.8%) and geometry parameter estimation (8.9%) are relatively insubstantial. The pose estimation stage accounts for the highest cascaded error, which is consistent with the most failure cases observed above. However, this error is hard to mitigate within our current pipeline, as FoundationPose (Wen et al., 2024) currently represents the optimal choice for balancing performance and efficiency for pose estimation from single RGBD.

H Generalization Capability in Object Interaction Understanding

The generalization capability of our approach in understanding object interactions arises from two sources: (1) the **high expressiveness of Digital Genes** in describing object structures, and (2) the **adaptability of functional programs** across diverse interaction tasks.

Highly Expressive Capability of Digital Genes. Digital Genes are capable of describing the structures of various objects, including novel and unseen ones, as the design and manufacturing of

man-made objects are fundamentally based on combinations of basic geometric primitives (*e.g.*, CAD). Specifically, we introduce 10 basic geometries (*Cuboid, Cylinder, Sphere, Cone, Triangular Prism, Ring, Torus, Rectangular Ring, Cuboid with Circular Hole, Cylinder with Box Hole*, please refer to **geometry_templates.py** in the supplementary **digital_gene_code_examples** for implementation) as the fundamental building blocks of Digital Genes. Each geometry encapsulates a category of shapes sharing common spatial properties, while its associated parameters instantiate specific instances. By combining these basic geometries, Digital Genes can represent an object's structural knowledge with exponential diversity. The complexity and variety covered by Digital Genes reach a real-world scale, demonstrating their high expressive capacity and adaptability to novel objects.

Adaptability of Functional Programs. The adaptability of functional programs stems from two factors: 1) **Correlation between functions and geometries:** The functionalities of objects are intrinsically linked to their geometric attributes, which generalize across parts with similar shapes. For example, the function of rotating a door can be adapted to rotating a box lid or a window. Since Digital Genes capture these geometric attributes and generalize to novel objects, they also capture and generalize the underlying functionalities. **2) Program Synthesis.** Our approach involves a program synthesis process that inherits from predefined part-level programs and generates novel programs for specific interaction tasks, further enhancing adaptability.

Beyond Conventionally Designed Objects. While the previous two properties enable digital gene allow Digital Genes to cover a wide range of interaction tasks and novel objects, there might be exceptions where the interaction task involves unconventionally designed objects. Understanding the physical properties of unconventional designs remains a significant challenge for embodied learning (Wang et al., 2025c). Even established frameworks (*e.g.*, RoboBrain (Ji et al., 2025), Cosmos) may fail on such objects (Zeng et al., 2025). Therefore, these cases represent a broader open challenge within the field rather than a specific limitation of our framework.

I Image Segmentation Effectiveness

Figure 8 presents qualitative examples of real-world image segmentation using Nano-Banana Pro. The specific prompts utilized to guide the segmentation process are detailed in Appendix.J. Each illustration includes the original image, the segmentation mask generated by Nano-Banana Pro, and the resulting segmented regions overlaid on the original input. These results demonstrate that Nano-Banana Pro excels at image understanding and component detection. Provided with appropriate prompts, Nano-Banana Pro generates masks with sharp boundaries and precise pixel-wise alignment across both coarse and fine-grained scales. The conclusion also aligns with the quantitative results of the system error breakdown in Section 5.3 that image segmentation by Nano-Banana Pro does not contribute to a bottleneck of our method.



Figure 8: Illustration of image segmentation using Nano-Banana Pro.

J PROMPT EXAMPLES

J.1 CHAIN-OF-THOUGHT PROMPTS

In this section, we present the CoT prompts utilized in our framework, using the task "open the drawer of the storagefurniture" as an example. Throughout the following prompt templates, content enclosed within "< >" brackets serves as placeholders that are replaceable with context-specific tasks, objects, components, or Digital Gene concept templates. All prompts are programmatically generated for in-context learning by scripts that combine intermediate MLLM outputs with a predefined corpus (*e.g.* Digital Gene concept template synopses). This automated integration ensures consistency throughout the multi-stage reasoning process across diverse object understanding tasks.

Object Identification. This prompt presents a Digital Gene supported object taxonomy to guide the MLLM in identifying object category.

```
Please identify the object category in the image based on the following
    candidate list:\n
[Box, Bucket, Door, Faucet, Refrigerator, Kettle, KitchenPot, Microwave,
    Safe, StorageFuriture, Switch, Table, Trashcan, Washingmachine,
    Window]\n\n
Output the result in strict JSON format with a single key "object"\n
Do not include markdown formatting or explanation.\n\n
Example Output:\n
{\n
   "object": "Faucet"\n
}\n
```

Structure Identification. Conditioned on the identified object taxonomy, this prompt provides component candidates for structure identification, allowing the MLLM to systematically identify the constituent parts and object structure topology from the observation, which is essential for Digital Gene grounding.

```
This object is a <StorageFurniture>. Please identify the structure of
    the <StorageFurniture> in a tree-format.\n
A type of component should exist multiple times in the output object
    structure tree.\n\n
Output the result in strict JSON format with a single key "components"\n
Do not include markdown formatting or explanation.\n\n
Example Output:\n
{\n
   "components": {\n
      "body":{\n
         "switch": {\n
            "handle": {},\n
            "handle": {}\n
         },\n
         "spout": {}\n
      }\n
   }\n
}\n
```

Digital Gene Selection. Conditioned on the identified object taxonomy, this prompt provides a set of concept template candidates, enabling the MLLM to select the most appropriate Digital Genes to represent the structural topology of each observed component.

```
This object is a <StorageFurniture>. All concepts and corresponding
    synopses for the <StorageFurniture> class are listed below:\n\n
{\n
   "Object": "StorageFurniture",\n
   "Components":\n
   [\n
```

```
      {\n
         "Component_Name": "Body",\n
         "Concept_Name":\n
         {\n
            "Storagefurniture_Regular_body": "Cabinet assembled from
                multiple cuboid boards(Cuboid): left/right sidepanels, a
                back panel, a base (optional top). Inside, shelves
                (cuboids) partition the interior along Y;side and back
                panels bound a rectangular cavity. Additional blocks can
                create dividers/drawer bays.",\n
            "Storagefurniture_Panel_body": "Single cuboid trim/face
                panel (Cuboid) covering the cabinet or drawer frontto
                unify the facade; thickness and offsets are
                adjustable."\n
         }\n
      },\n
      {\n
         "Component_Name": "Door",\n
         "Concept_Name":\n
         {\n
            "Regular_door": "Per leaf: a rectangular door slab (Cuboid)
                plus a small handle (Cuboid). The slab yaw-rotates about
                its side edge; the handle sits on the front face at an
                offset."\n
         }\n
      },\n
      {\n
         "Component_Name": "Drawer",\n
         "Concept_Name":\n
         {\n
            "Regular_drawer": "Drawer composed of left/right side
                plates, rear/front narrow plates, a bottom plate, and a
                front face plate (allCuboids). Supports one or two
                handles. The plates enclose the drawer cavity; the face
                plate sitsforemost and may protrude."\n
         }\n
      }\n
      <...> # Omit Long Prompts
   ],\n
}\n\n
Act as a structural engineer. Please analyze the image and identify
    which specific concepts (parts) are presentin this object instance.\n
lnclude parts that are structurally necessary even if they are partially
    occluded.\n
Select the most suitable Concept Name for each part.\n\n
Output the result in strict JSON format with a single key
    "identified_concepts" containing a dict mapping of components and
    concepts.\n
Do not include markdown formatting.\n\n
Example Output: \n
{\n
   "identified_concepts":\n
   {\n
      "body": "Regular_Body",\n
      "switch": "Round_Switch"\n
   }\n
}\n
```

Part-level Program Selection. The part-level functional program synopses of the selected concept templates are presented to the MLLM within the prompt. MLLM evaluates and selects the task-oriented programs, ensuring essential part-level functions are incorporated into the subsequent object-level program synthesis.

```
This object is a <StorageFurniture>. The user wants to perform the task:
    <"Open the door and the drawer of the storage".>\n
From previous steps, we identified the parts involved in this object:
    <['Regular_door", "Regular_drawer',
    "Storagefurniture_Regular_body"]>.\n
Below is the list of available Python functions defined within these
    active part classes:\n
{\n
   "Regular_door":\n
   [\n
      "get_door_kinematics"\n
   ],\n
   "Regular_drawer":\n
   [\n
      "get_drawer_kinematics"\n
   ],\n
   <...>    # Omit Long Prompts
}\n
Act as a robotics software engineer. Your goal is to identify which
    specific function should be called toretrieve the kinematic
    properties (such as axis, grasp pose) for the task.\n\n
Output the result in strict JSON format.\n
The keys should be the Concept Name, and the values should be the
    selected Function Name.\n
Do not include markdown formatting.\n
Example Output:\n
{\n
   "part_level_programs":\n
   {\n
      "Roller_Door": "get_door_kinematics",\n
      "Cuboid_Handle": "grasp_handle"\n
   }\n
}\n
```

Object-level Program Synthesis. Upon receiving the detailed part-level programs, the MLLM synthesizes a task-specific object-level program through inheritance. This enables our approach to combine localized part functions into a comprehensive, executable program for the entire object towards the object understanding task.

```
You are a robotic planning agent. We are preparing to execute a
    manipulation task.\n
Here is the resolved context from previous analysis steps:\n
{\n
   "object": <StorageFurniture>,\n
   "task": <open the door and the drawer of the storage>,\n
   "components":\n
   {\n
      "body":\n
      {\n
         "door":\n
         {\n
            "handle": {},\n
         },
         "button": {}\n
         "button": {}\n
         <...>
      }\n
   },\n
   "part_level_program_source":\n
   {\n
      "Regular_door":\n
      [\n
         "get_door_rotation_axis"\n
```

```
      ]\n
      "Curve_Handle":\n
      [\n
         "get_handle_grasp_pose"\n
      ]\n
   }\n
}\n\n
Below are the detailed available part level function programs:\n
<Digital Gene Python Codes with Functions>\n\n
**Goal**: Generate a single appropriate object-Level function that
    inheritate the part-level programs for the task "<open the door and
    the drawer of the storage>".\n
The object-level function must effectively utilize the kinematic
    capabilities described in the documentation. \n\n
Output the result in strict Python script.\n
Do not include markdown formatting.\n
```

J.2 Object Segmentation Prompts for Nano-Banana Pro

We employ Nano-Banana Pro (Team et al., 2023) as our image segmentation method, leveraging its general semantic understanding and precise image manipulation capabilities. Given that a component's structure, topology, and spatial orientation vary significantly across different objects and scenes, Nano-Banana Pro (Team et al., 2023) struggles to perform precise image manipulation without accurate guidance. Since providing such information via standardized scripts is impractical, we first utilize GPT-5.6-solar to interpret the image and extract structural and topological priors of the target component. By feeding these critical semantic information into Nano-Banana Pro (Team et al., 2023), we enable it to manipulate image with high-fidelity and generate masks for target components in specific colors.

Prompt for GPT-5.6-solar. We employ a three-step pipeline to guide GPT-5.6-solar in generating prompts for Nano-Banana Pro (Team et al., 2023) for image segmentation. **1.** it extracts visual appearance of the component in the image. **2.** it incorporates negative constraints and boundary refinements to prevent potential segmentation errors, *e.g.* internal gaps or aliasing artifacts. **3.** All information is incorporated into a structured prompt including the target identity, visual descriptions, spatial information, segmentation constraints, and mask specifications, *etc.* In addition, we provide some examples for few-shot. The "*generation_prompt*" is the prompt for Nano-Banana Pro.

```
# Role\n
You are an expert Visual Grounding and Generative Segmentation
    Assistant. Your goal is to translate structured user inputs into
    precise, low-level image generation prompts for a segmentation
    model.\n\n
# Input Context Explanation\n
1. **User Input**: A structured command in the format `<part name>`.\n
  - `part name`: The natural language name of the target object part
      (e.g., "kettle handle", "top drawer").\n
2. **Image Context**: The input image of the scene.\n
3. **Component Library**: A JSON list of object parts and their visual
    features.\n\n
# Task: Targeted Visual Description Generation\n
You must perform the following steps to generate the final JSON
    output:\n\n
## Step 1: Component Lookup & Extraction\n
- Search the **Component Library** for the specific `part_name` provided
    in the user input.\n
- Extract the `visual_desc` (visual description) associated with that
    part.\n
- **Constraint**: You do not need to classify the object. Trust the
    user's `part_name` and find its definition.\n\n
## Step 2: Negative Constraint Reasoning\n
- Analyze the nature of the target part to generate context-aware
    negative prompts.\n
```

```
- **Rules**:\n
   - If targeting a **Handle** (especially loops/rings): You MUST add
       constraints to "strictly exclude the gap/hole inside the handle"
       and "exclude the object body".\n
   - If targeting a **Drawer/Door**: You MUST add "exclude the cabinet
       frame", "exclude neighboring drawers".\n
   - If targeting a **Small Part** (knob/key): You MUST add "exclude the
       surface behind it".\n\n
## Step 3: Prompt Construction\n
Generate the segmentation prompt strictly following this format:\n
"Fine-grained segmentation for <part name>, [Extracted Visual
    Description], [Spatial Details from Image], [Context-Aware Negative
    Constraints], [Highlighting Instruction], [Technical Quality
    Keywords]"\n\n
1. **Target Identification**: Start explicitly with the user's inputs:
    `Fine-grained segmentation for <part name>`.\n
2. **Visual Description**: Append the `visual_desc` found in the
    library.\n
3. **Spatial Details**: Briefly describe the specific location in the
    image (e.g., "on the left side", "top-most").\n
4. **Context-Aware Negative Constraints**: Add the specific exclusions
    generated in Step 2. Be extremely verbose here (e.g., "strictly
    exclude the gap between handle and body", "do not merge with
    background wall").\n
5. **Highlighting Instruction**: ALWAYS use: "highlight only the
    [target] areas in solid bright magenta #FF00FF".\n
6. **Technical Quality Keywords**: Append: "geometric isolation, sharp
    edges, artifact-free mask, strict pixel preservation for
    surroundings."\n\n
# Component list (Dynamic Data) \n
<component_list_json>\n\n
# Output Format\n
1. **Return ONLY a valid JSON object.**\n
2. **DO NOT** include markdown code blocks.\n
3. The output must be directly parsable by a JSON parser.\n\n
# Few-Shot Examples\n\n
## Example 1 (Handle - Complex Negative Prompts)\n
**User Input**: "<kettle handle>"\n
**Component Library**: {\n
  "Kettle": {\n
    "components": {\n
      "body": {"visual_desc": "Main stainless steel container" },\n
      "handle": {"visual_desc": "Black plastic overhead arched grip, flat
          geometry" }\n
    }\n
  }\n
}\n
**Scene**: A silver kettle with a black handle on top.\n
**Output**:\n
{\n
  "reasoning": {\n
    "thought_process": "User targets 'handle'. Library desc is 'Black
        plastic overhead arched grip, flat geometry'. Since it is a
        U-handle, I must strictly exclude the empty space under the arch
        and the metal body it connects to."\n
  },\n
  "segmentation_tasks": [\n
    {\n
      "step_index": 1,\n
      "target_component": "kettle handle",\n
      "generation_prompt": "Fine-grained segmentation for kettle handle
          (Flat_U_Handle), Black plastic overhead arched grip, flat
          geometry, located on top of the silver kettle, strictly exclude
          the empty air gap under the handle, exclude the stainless steel
          kettle body, exclude the lid knob, do not merge handle with
```

```
        background wall, highlight only the black handle areas in solid
        bright magenta #FF00FF, geometric isolation, sharp edges,
        artifact-free mask, strict pixel preservation for
        surroundings."\n
   }\n
  ]\n
}\n\n
## Example 2 (Drawer - Boundary Constraints)\n
<...>       # Omit for Simplification. \n
# Current Mission\n
**User Input**: "{input_context}"\n
Generate the JSON response now.\n
```

Prompt for Nano-Banana Pro. Here we provide an example of a piece of prompt for Nano-Banana Pro generated by GPT-5.6-solar following the method mentioned above. We use "segment a box cover" as an example.

```
Fine-grained segmentation for box's cover: Solid single-piece lid
    covering the top, located on the upper section of the white textured
    box, strictly exclude the main box body below the horizontal seam,
    exclude the dark desk surface and background cables, highlight only
    the lid areas in solid bright magenta #FF00FF, geometric isolation,
    sharp edges, artifact-free mask, strict pixel preservation for
    surroundings.
```

K Limitations and Future Work

Digital Gene primarily can represent rigid and articulated objects. Therefore, our approach focuses on understanding tasks within these taxonomies. In the future, we aim to investigate the extension of Digital Gene to deformable objects and garments, broadening the generalizability of our method across a more diverse field of object taxonomies and functional properties.

L Computing Resources

The training procedures and evaluations are conducted on a computer with NVIDIA 4090 GPU and Intel(R) Xeon(R) Gold 6133CPU @ 2.25GHz.