arXiv:2403.19417v1 [cs.CV] 28 Mar 2024

OakInk2 : A Dataset of Bimanual Hands-Object Manipulation in Complex Task Completion

Xinyu Zhan[1*] Lixin Yang[1*] Yifei Zhao[1] Kangrui Mao[1] Hanlin Xu[1]
Zenan Lin[2,‡] Kailin Li[1] Cewu Lu[1†]
[1]Shanghai Jiao Tong University, [2]South China University of Technology

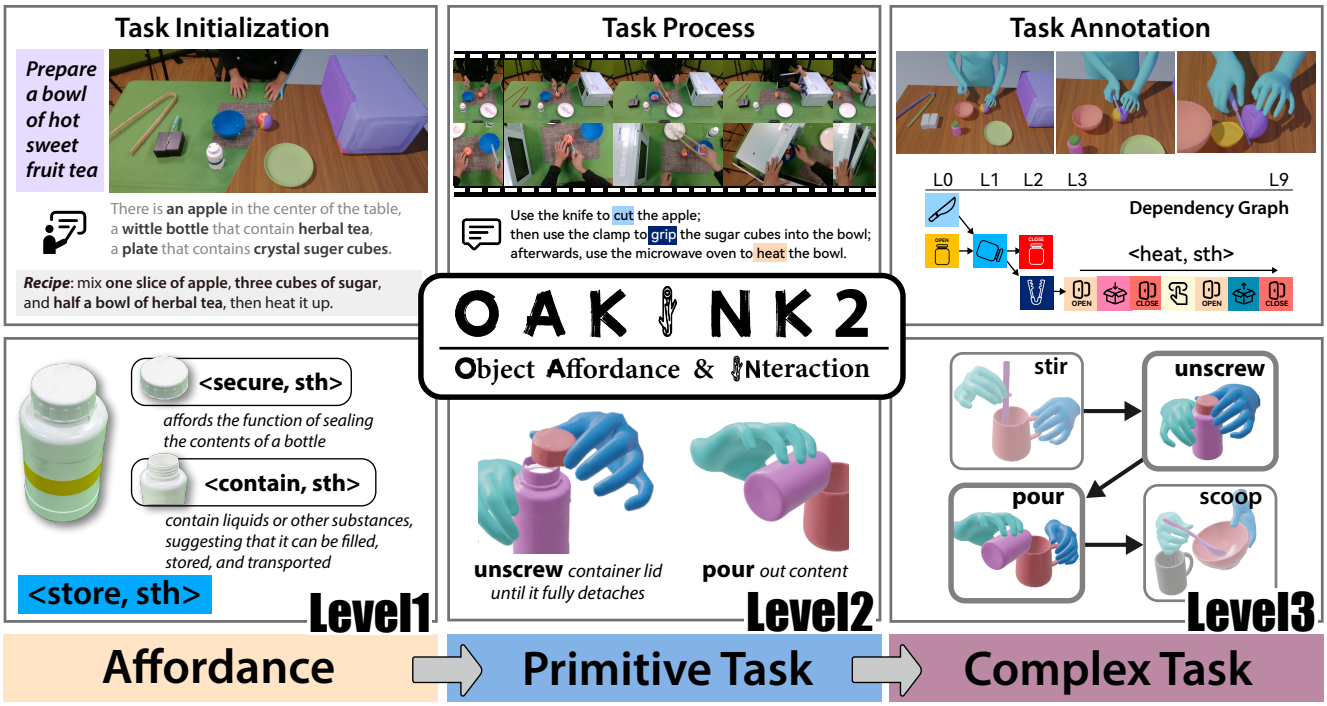


Figure 1. **An overview of the data and content of our proposed OakInk2 dataset.** OakInk2 dataset focuses on bimanual object manipulation tasks for complex daily activities. **1) The top row shows the data collection process**, including the task setup (top-left panel), human demonstration (top-center), and annotation (top-right). **2) The second row shows the three levels of abstraction constructed by OakInk2 for complex tasks**, including the Affordance, Primitive Task, and Complex Task. OakInk2 dataset provides allocentric and egocentric videos of human manipulation process, as well as the corresponding 3D-pose annotation and task specification.

Abstract

We present **OakInk2**, *a dataset of bimanual object manipulation tasks for complex daily activities. In pursuit of constructing the complex tasks into a structured representation,* OakInk2 *introduces three level of abstraction to organize the manipulation tasks:* ***Affordance***, ***Primitive Task***, *and* ***Complex Task***. OakInk2 *features on an object-centric perspective for decoding the complex tasks, treating them as a sequence of object affordance fulfillment. The first level, Affordance, outlines the functionalities that objects in the scene can afford, the second level, Primitive Task, describes the minimal interaction units that humans interact with the object to achieve its affordance, and the third level, Complex Task, illustrates how Primitive Tasks are composed and interdependent.* OakInk2 *dataset provides multi-view image streams and precise pose annotations for the human body, hands and various interacting objects. This extensive collection supports applications such as interaction reconstruction and motion synthe-*

*The first two authors contributed equally.
‡This work is done when Lin is an intern at SJTU.
†Cewu Lu is the corresponding author. He is the member of Qing Yuan Research Institute and MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, China.

sis. Based on the 3-level abstraction of OAKINK2*, we explore a task-oriented framework for Complex Task Completion (CTC). CTC aims to generate a sequence of bimanual manipulation to achieve task objectives. Within the CTC framework, we employ Large Language Models (LLMs) to decompose the complex task objectives into sequences of Primitive Tasks and have developed a Motion Fulfillment Model that generates bimanual hand motion for each Primitive Task.* OAKINK2 *datasets and models are available at* https://oakink.net/v2.

1. Introduction

Learning how humans achieve specific task objectives through diverse object manipulation behaviors has been a long-standing challenge. Recent data-driven approaches have made significant progress on this topic, including hand-object pose estimation [1, 7, 14, 22, 24–26, 39, 66], interaction synthesis [12, 17, 31, 57, 62, 71], and action imitation [53, 54]. However, the gap still exists for current methods to achieve a human-level understanding on object manipulation for complex task completion. In particular, humans possess a remarkable capacity to interact with specific objects in an appropriate sequence to achieve desired outcomes [33]. This inspires us to focus on the decomposition of hands-object interaction in complex manipulation tasks into sequential units.

Tracing prior research, the advancement in hand-object interaction understanding is inseparable from the emergence of a series of hand-object interaction datasets [3, 8, 12, 15, 21, 24, 30, 34, 40, 47, 54, 56, 67, 72] to support data-driven methods. A noteworthy example among these datasets is OakInk [67]. OakInk analyzed object affordances (*i.e.* functional properties of objects/object-parts [18]) and collected *human-centric* grasping interaction driven by intents to utilize these affordances. The term: *Oak* is for object affordance knowledge, and *Ink* for interaction knowledge. Nevertheless, the previous OakInk has two major limitations: 1) it lacks human demonstrations that cover the process of fulfilling those affordances, and 2) it lacks complex manipulation tasks that involve multiple object affordances.

In this paper, we present OAKINK2, extending the data and methodology of the previous OakInk. In order to manage the inherent complexity in complex manipulation tasks, OAKINK2 adopts an *object-centric* perspective and constructs three levels of abstraction upon manipulation tasks:

1) ***Affordance***: object/object-part level functionalities that enable manipulation. For example, a bottle cap affords securing and unsecuring of the content in the bottle.

2) ***Primitive Task (Primitive)***: a "minimal" sequence of hand-object interaction that fulfills a given object's affordance. For instance, to fulfill the affordance: securing, one needs to either `screw` or `press` the cap onto the bottle's opening to form a seal that prevents leaking.

3) ***Complex Task***: sequential combination of *Primitives* to address the long-horizon and multi-goals manipulation tasks. Tasks are characterized as "complex" for their goal requires more than one object affordance. *Complex Tasks* also detail the **dependencies** among the *Primitives* and dictate the **order** in which they are executed. To illustrate, to pour the fluid from a sealed bottle, one must first `unscrew` the cap and then `pour` out the liquid.

In this way, OAKINK2 delineates *Complex Tasks* as directed acyclic graphs, hereafter referred to as ***Primitive* Dependency Graphs (PDG)**. Within these graphs, each node represents a *Primitive*, serving to fulfill a specific affordance. The directed edges illustrate the sequence in which *Primitives* must be executed to achieve task completion.

Build upon the above methodology, OAKINK2 introduces a large-scale dataset for bimanual object manipulation. It encompasses human demonstrations for complex task completion, with multi-view image streams and paired pose annotations for human body, hands and objects. OAKINK2 contains 627 sequences of real-world bimanual manipulation sequences, where 264 of these sequences are for *Complex Tasks*. These sequences contain 4.01M frames from four different views (one egocentric and three allocentric views). The dataset includes four manipulation scenarios, 75 objects and 9 invited subjects in total.

The versatile and task-driven nature of OAKINK2 enables a wide range of applications. In this paper, we focus on the task and motion planning for Complex Task Completion (CTC). CTC involves two notable components: **1)** text-based *Complex Task* decomposition using *Primitives* and **2)** task-aware motion generation to fulfill each *Primitive*. For the first component, we design a task interpreter with Large Language Model (LLM) that can generate the PDG and program the execution order of these *Primitives*, based on textual descriptions of the *Complex Tasks*. For the latter component, we propose a generalist Task-aware Motion Fulfillment model (TaMF) to generate the hand motion at *Primitive* level, based on the task-related object trajectory.

In summary, our contributions are as follows:

- We build an object-centric, three-level abstraction to structure and understand complex manipulation tasks, *i.e.* *Affordance*, *Primitive* to fulfill affordance, and *Complex Task* with *Primitive* dependencies.
- We introduce OAKINK2, a large-scale real-world dataset for bimanual object manipulation with human demonstrations for both *Primitives* and *Complex Tasks*.
- We propose a task-oriented framework, CTC, for complex task and motion planning. CTC consists of a LLM-based task interpreter for *Complex Task* decomposition and a diffusion-based motion generator for *Primitive* fulfillment.

2. Related Works

Hand-Object Interaction Datasets. The recent research community has witnessed the emergence of numerous datasets on hand-object interactions. Earlier datasets [3, 12, 24] focused on static hand-object interactions with limited diversity. More recent datasets [8, 15, 21, 34, 41, 56, 72] captured dynamic hand-object interactions, covering bimanual interactions [15, 34] and interactions with articulated bodies [15, 72]. We pay particular attention to interaction datasets related to object affordances. [12] expressed affordances in grasp type labels. [3, 15, 56] collected intention labels for interactions. [30, 67] studied object affordance-based hand-object interaction and collected object segmentations and affordance labels. [41] studied hand-object interactions in tool-action-object pairs. Our proposed OAKINK2 captures both human demonstrations for minimal interaction fulfilling object affordance as *Primitive*, and demonstrations for *Complex Task* where these affordances are fulfilled in specific order constrained by their dependencies.

Decomposition of Manipulation Tasks. Decomposing complex manipulation tasks into multiple building blocks across different hierarchies represents a widely adopted paradigm in the research community. [10] utilize the symbolic interface of task planners to construct an abstract state space, facilitating the reuse of hierarchical skills. [27, 63] decompose task specifications into hierarchical neural programs, which feature bottom-level programs as callable subroutines interacting with the environment. [9] chain multiple dexterous policies for achieving long-horizon task goals. [2] adopt a language-based methodology for decomposing action hierarchies. In our work, we introduce an object(affordance)-centric, three-level abstraction framework within OAKINK2 for the decomposition of complex manipulation tasks into *Primitives*.

Motion Synthesis. Motion synthesis involves obtaining credible and realistic human action sequences. There are plenty of works to generate human motions [51, 52, 59], even interactions [17, 36, 37, 57, 58, 62] based on different probabilistic model backbones like cVAE or denoising diffusion. In particular, [36, 37, 58] synthesize human motion based on the object motion, delegating the latter part to preceding models serving as inputs. Inspired by these works, we propose a new task within OAKINK2: Task-aware Motion Fulfillment This task requires the model to synthesize hand motion trajectories based on given textual task descriptions and object motions.

Foundation Models in Manipulation Tasks. Recent days we have seen a significant increase in the application of foundation models in completing manipulation tasks. There are significant efforts for end-to-end foundation models [4, 5, 11] that outputs control signals from visual and textual inputs. Existing works [6, 28, 55] also leverage the in-context learning and zero-shot generalization abilities of Large Language Models (LLMs) for action selection from an array of choices to realize an autoregressive achievement of planning. Demonstration of LLM-based program generation for task completion in [29, 37, 55] inspires us to explore the ability of LLMs to reason code for discerning interdependencies between object affordances in complex tasks, along with the sequence in which they are implemented. Our OAKINK2 introduce the decomposition of *Complex Tasks* into interdependent affordance-based *Primitives*, accompanied by their diverse image-textual descriptions. Based on this, we show an application of OAKINK2 in Complex Task Completion utilizing existing power of foundation models.

3. Construction of OAKINK2

We first introduce how the three-level of abstractions are acquired in Sec. 3.1, then provide the details for data collection and annotation in Sec. 3.2.

3.1. Complex Task Acquisition

Task Initialization. Given a collected repository of objects, we first construct four manipulation scenarios. Each scenario has its unique characteristic and corresponds to a set of complex manipulation tasks. These scenarios are: 1) kitchen table; 2) study room table; 3) demo chem lab; 4) bathroom table. Then, we invite four annotators (👥) to propose *Complex Tasks* in these scenarios and select object cluster that required for these tasks (Fig. 2's 1st column).

3.1.1 Object Affordance Analysis

After the task targets are determined, we proceed to analyze the objects' affordances in given scenarios. The expression of affordance adheres to the definitions in the previous OakInk [67]: each affordance contains a specific object part segmentation (*e.g.* a bottle cap) and a descriptive phrase tuple (*e.g.* `<secure, sth>`), which elucidates the function of that part. We provide examples of these affordances in Fig. 2's 2nd column.

3.1.2 Primitive Task Design

In the second stage, We design *Primitives* as the **minimal** interactions that fulfill those object affordance. Here "minimal" indicates the task are required to fully complete the functionality of a certain affordance without any redundant interaction process. Each *Primitive* contains a starting condition, a terminal condition, and the in-between hand-object interaction process. For example, considering an affordance associated with a knife blade meant to `<cut, sth>`, a corresponding *Primitive*, `cut`, requires the subject to move

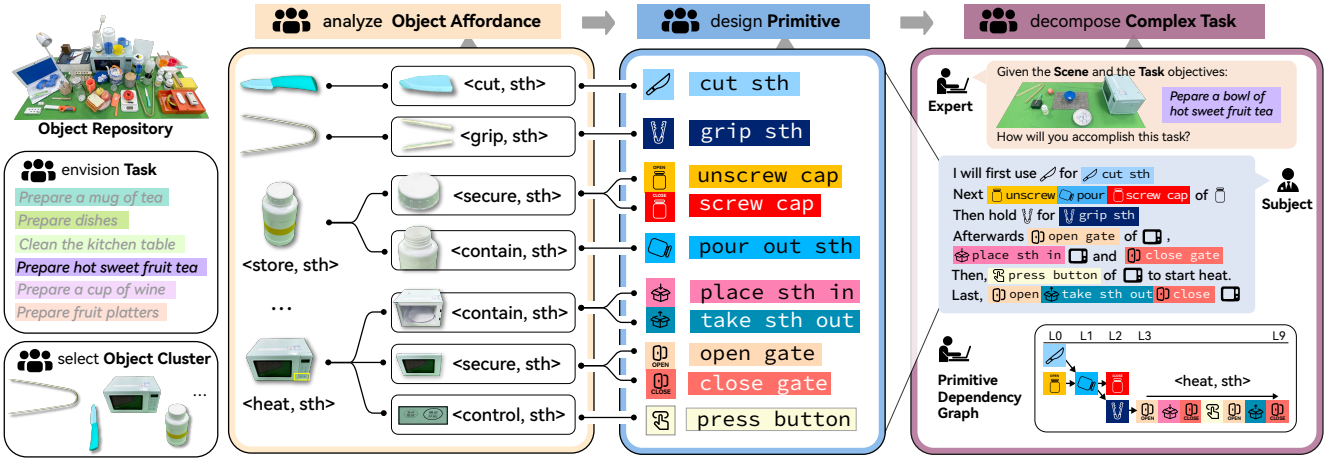


Figure 2. **Illustration of the complex task acquisition process.** This figure use a *Complex Task*: 'Prepare a bowl of hot sweet fruit tea.' to demonstrate the process. Initially, the annotators (👥) analyze the affordances of four essential objects (a gripper, a knife, a tea bottle, and a microwave oven) and design corresponding *Primitive*. For instance, to prepare fruit slices, the *Primitive*: `cut` associated with the knife blade is required. Following this, an expert (🧑‍💻) arranges the scene for the *Complex Task*, and then the subject (🧑), utilizing the designed *Primitive*, plans the execution path of the *Complex Task*. Later, these execution paths are structured into a *Primitive* Dependency Graphs.

the blade to completely pass through the object to be cut so that the separated parts could be detached. In this stage, we collect all available object affordances and their associated *Primitives*, leading to a *Primitive* tasks pool (Fig. 2's 3rd column).

3.1.3 Complex Task Decomposition

In the third stage, we proceed to decompose the previous proposed *Complex Task* – characterized by its long-horizon and multi-goal manipulation targets – into a series of short-term and single-goal *Primitives*. In emphasizing the ordering of *Primitive* completion is important for the *Complex Task* completion, our approach also delineates the **dependencies** between *Primitives*. Therefore, each *Complex Task* contains a series of *Primitives*, along with a *Primitive* Dependency Graph (PDG), which maps out the hierarchical execution order of these *Primitives*. *Primitives* at level 0 (L0) are independent, requiring no prior *Primitives* to be completed, while the final level include those *Primitives* that bring the *Complex Task* to completion.

We deploy a dedicated protocol to acquire the decomposition and dependencies. As shown in Fig. 2's 4th column, initially, an expert (🧑‍💻) instantiates the scene and target with specific description. Subsequently, a subject (🧑) is instructed to describe the order of the completion using the available *Primitive* in the pool. Then, the expert records and organizes this sequence into the PDG, concluding the *Complex Task* acquisition process.

3.2. Data Collection and Annotation

After the acquisition of the three-level of abstractions, the subjects are required to complete the *Primitive* and *Complex Task* respectively in a data capture platform (Fig. 3).

Figure 3. **Capture platform.** 12 MoCap cameras are circled in blue and 4 RGB cameras in red.

3.2.1 Capture Setup

The data capture platform contains two major components: the multi-camera system for recording the manipulation process and the optical MoCap system for pose tracking. The MoCap system uses 12 Optitrack Prime 13W infrared cameras to track the surface markers affixed to the subject's upper body, left and right hand, and interacting objects. The multi-camera system consists of 4 commodity RGB cameras, 3 of which are from allocentric views and 1 is from the egocentric view. We synchronize all sensors at 30 fps and calibrate the transformation between these two systems.

3.2.2 Data Annotation

Object Pose. Poses of rigid bodies are directly solved via the MoCap system. For the poses of articulated bodies, the base parts of articulated bodies are handled similarly to rigid bodies, while the articulated parts are divided into two categories. If the part is large enough to attach enough mark-

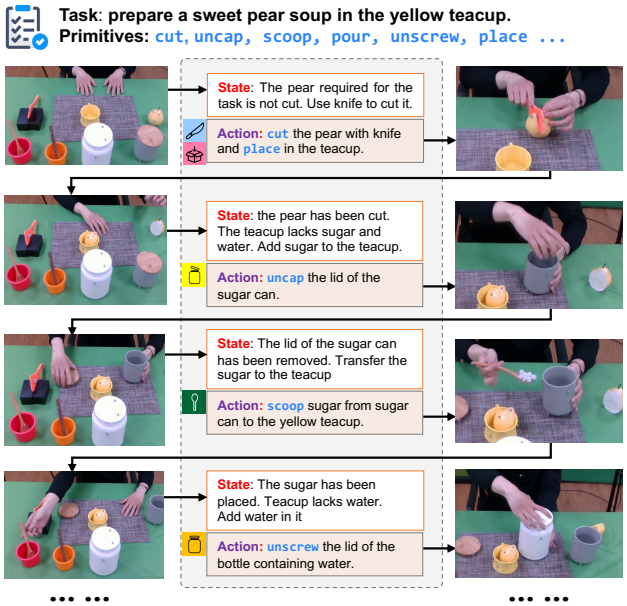


Figure 4. **Commentary of the task execution.** The left column shows the current state of the scene. The center column shows the narrative dialog retrieved from experts. The right column shows the upcoming *Primitive* task to be executed.

ers without blocking the interaction then it will be handled like rigid bodies. Otherwise, only one marker is attached to that part. The marker's position is calibrated in the object's canonical coordinate frame. Later, given the articulation type (*e.g.* revolution or prismatic), the parameter of the articulation joint is determined by minimizing the squared difference between the observed marker position and the recovered marker position in the object's canonical frame.

Human Pose and Surface. The annotation of human pose and surface relies on SMPL-X [50] body mesh. To actually acquire human pose and surface, we employ a two-stage fitting approach in align with the MoSH++ [44]. In the first stage, we use the captured markers when the subject in T-pose to fit the subject's SMPL-X shape parameter $\bar{\beta}$ and each marker's location $P_{\mathcal{M}}^{(c)}$ in SMPL-X canonical space. From stage one's optimization result, we can determine the correspondence $\mathcal{C}(\cdot)$ from the subject's surface markers to the vertices of the SMPL-X model. In the second stage, we fit the per-frame subject poses parameter θ throughout the task completion process. This fitting is grounded in the previously acquired shape $\bar{\beta}$ and marker correspondence $\mathcal{C}(\cdot)$. With pose and shape parameters obtained, the subject's body mesh is reconstructed using the SMPL-X model. Other body representations like MANO are derived from this result. Refer to Sup. Mat for details.

Commentary of Task Execution. After the manipulation process is completed, we send the video recording to experts for analysis, requesting them to furnish detailed commentary on the task execution process. At each *Primitive* step, experts are asked to provide comments on the current task state and the forthcoming action. Specifically, given the execution of the previous *Primitive*, experts are asked to 1) summarize the tasks yet to be completed to achieve the manipulation goals, considering both the current scene and the upcoming *Primitive* slated for execution; and 2) offer descriptions of the next action using the available *Primitives* in the pool. This process is illustrated in Fig. 4. The narrative text provided by experts are subsequently refined using GPT-4 [48] to serve as commentary. OakInk2 features on these commentaries as they encapsulate the expert's chain-of-thought when observing the manipulation process. These commentaries serve not only to interpret user behaviors but also to inform the generation of user actions.

4. The OakInk2 Dataset

4.1. Data and Annotation List

OakInk2 provide RGB videos that record the manipulation processes. These videos are collected from multi-view (1 egocentric and 3 allocentric) setup, synchronized at 30 fps, with resolution 848×480. The annotations contains two parts: **1) 3D motion**, including pose and shape for the human upper-body, hands, and objects (with articulation parameters) during the interaction process; and the **2) task specification**, including object affordances, *Primitives* that correspond to these affordances, *Complex Tasks* with task goals, initial conditions, PDGs, expert commentary, and subject's completion sequence. Evaluations of the 3D annotation qualities are provided in Sup. Mat. Annotation on 3D hand keypoints undergo cross-dataset validation with a reconstruction model, while the 3D poses associated with grasping actions are examined for the physical property integrity.

4.2. Dataset Statistics

OakInk2 sets up four scenarios of hand-object interaction with a total number of 38 long-horizon complex manipulation goals, which instantiates to 150 *Complex Tasks*. OakInk2 contains in total 75 objects and 39 affordance. These affordances map to 60 types of *Primitives*. OakInk2 contains 627 sequences of bimanual dexterous hand-object interaction in total. 363 of these are for *Primitives* and 264 are for *Complex Tasks*. In total, OakInk2 contains 4.01M image frames. We compare OakInk2 to multiple existing hand-object interaction datasets in Tab. 4. Here we highlight several notable features of OakInk2: 1) it provides interaction grounded in object affordance (*vs.* HO3D, DexYCB); 2) it features long-horizon manipulation goals (*vs.* ARCTIC, HOI4D, GRAB); 3) it includes 3D pose and shape annotation for both hands and objects (*vs.* EGO4D, AssemblyHands); and 4) it offers task decomposition using *Primitives*, which is not available in any datasets in Tab. 4.

Dataset	image mod.	resolution	#frame	#views	#subj	#obj	3D gnd.	real / syn.	label method	hand pose	obj pose	afford. inter.	dynamic inter.	long-horizon	task decomp.
EGO4D [19]	✓	∼	∼	1	931	–	✗	–	–	✗	✗	✗	✗	✓	✓
HO3D [21]	✓	640 × 480	78K	1-5	10	10	✓	real	auto	✓	✓	✗	✓	✗	✗
GRAB [56]	✗	–	1.62M	–	10	51	✓	real	mocap	✓	✓	✓	✓	✗	✗
H2O [34]	✓	1280 × 720	571K	5	4	8	✓	real	auto	✓	✓	✓	✓	✗	✗
HOI4D [40]	✓	1280 × 800	3M	1	9	1000	✓	real	crowd	✓	✓	✓	✓	✗	✗
ARCTIC [15]	✓	2800 × 2000	2.1M	9	10	11	✓	real	mocap	✓	✓	✓	✓	✗	✗
AssemblyHands [47]	✓	1920 × 1080	3.03M	12	34	–	✓	real	semi-auto	✓	✗	✓	✓	✓	✓
Ego-Exo4D [20]	✓	∼	∼	5-6	839	–	✓	real	semi-auto	✓	✗	✗	✓	✓	✓
OakInk-Image [67]	✓	848 × 480	230K	4	12	100	✓	real	crowd	✓	✓	✓	✓	✗	✗
OakInk2	✓	848 × 480	**4.01M**	4	9	75	✓	real	mocap	✓	✓	✓	✓	✓	✓

Table 1. **A cross-comparison among various public datasets.** (Refer to Sup. Mat for the full table.)

5. Selected Applications

5.1. Hand Mesh Reconstruction

The Hand Mesh Reconstruction (HMR) task is to estimate the 3D hand pose during the interaction process from the captured images. We benchmark HMR task under both single-view settings and multi-view settings. In single-view settings, the image input only contains one view, egocentric or allocentric. In multi-view settings, the image input will contain multiple views, together with the camera calibration parameters. For both settings we partition the corresponded task-specified subsets at the sequence level, maintaining the proportion of samples in train/val/test sets at approximately 70%, 5%, and 25%.

We evaluate mean per joint position error (MPJPE), mean per vertex position error (MPVPE) in world space, wrist(root)-relative (RR) systems and systems after Procrustes analysis (PA). We also evaluate area under curve (AUC) of correct keypoints percentage within range $0 - 20\,\mathrm{mm}$ in root-relative systems. We show HMR benchmark results under both settings in Tab. 2.

Setting	Methods	PA-MPJPE	PA-MPVPE	RR-MPJPE (AUC)	RR -MPVPE	MPJPE	MPVPE
Mono	METRO [38]	6.90	6.47	17.56 (0.410)	16.44	–	–
	RLE [35] + HandTailer [42]	5.46	6.86	13.08 (0.441)	14.03	–	–
Multi	KP-based Fit [68]	9.20	8.83	15.63 (0.349)	15.38	19.30	19.11
	POEM [68]	6.18	6.61	12.12 (0.581)	12.15	9.17	9.52

Table 2. Single- and multi-view HMR **evaluation results** in *mm*.

5.2. Task-aware Motion Fulfillment (TaMF)

To achieve task objectives in interaction scenarios, we introduce a novel task: Task-aware Motion Fulfillment (TaMF). It targets at the generation of hand motion sequences that can fulfill given object trajectories conditioned on textual task descriptions.

Task Formulation. Given a textual description of the *Primitive* task: $\texttt{text}_{\mathrm{PT}}$, we assume the involved objects geometries $\boldsymbol{\mathcal{V}}_o = \{\mathbf{V}_{o,m}\}$ and their motion trajectories $\boldsymbol{\mathcal{T}}_o = \{\mathbf{T}_{o,m}^{(i)}\}$ during the interaction process are known. We use subscript h to represent human hands, o to represent the object, m to index different object instances (and different parts of the same instance) and superscript (i) to index different timestamps. The task is to generate a corresponding hands motion trajectory $\boldsymbol{\mathcal{P}}_h = \{\mathbf{P}_h^{(0:L)}\}$ conditioned on the textual description $\texttt{text}_{\mathrm{PT}}$, object geometries $\boldsymbol{\mathcal{V}}_o$, and motion trajectories $\boldsymbol{\mathcal{T}}_o$.

Evaluation Metrics. We evaluate contact ratio (CR) and solid intersection volume (SIV) to measure the physical plausibility of the generated motion. On sequence-level, we evaluate motion smoothness with Power Spectrum KL divergence of joints (PSKL-J) as in human motion generation, and evaluate FID to measure distances between the ground-truth motions and the generated motions. We also conduct a perceptual study to evaluate the level of realism for the generated motion. Detailed definition of these metrics can be found in Sup. Mat.

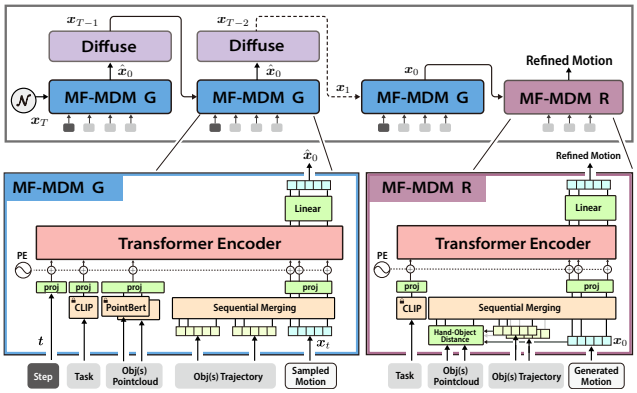


Figure 5. **Architecture of MF-MDM.** First sample random noises $\boldsymbol{x}_T$; then at each step iterating from T to 1, MF-MDM G predicts the cleaned sample $\hat{\boldsymbol{x}}_0$ and then diffuse it back to $\boldsymbol{x}_{t-1}$. After the generated sample $\boldsymbol{x}_0$ is acquired, it is refined by MF-MDM R for better interaction details.

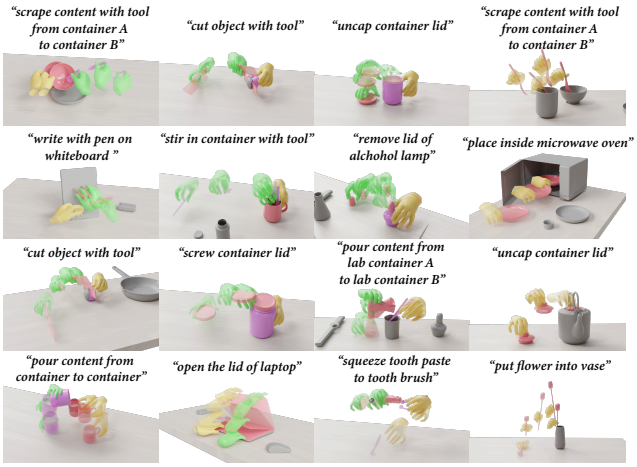


Figure 6. **Qualitative Visualization** of the generated hand motion in TaMF model.

Physical lausibility		Motion Smoothness	
CR↑	SIV (cm^3)↓	PSKL-J (**g.t., p.**) ↓	(**p., g.t.**) ↓
0.90	4.17	0.0446	0.0460

FID	Perceptual Score	
	Dataset	Generated
1.369	4.66 ± 0.48	3.64 ± 0.85

Table 3. **Evaluations** of generated hand motion in TaMF model. PSKL-J is evaluated between the training data (**g.t.**) and the generated hand motion trajectory (**p.**); both directions are included as PSKL-J is an asymmetric metric.

Model and Results. We enhance a diffusion-based motion generation model: MDM [59], tailoring it to the nuanced requirements of task-aware hand motion synthesis. The model architecture is visualized in Fig. 5. Our proposed model, named as MF-MDM, consists of two components: **1) MF-MDM G**, which generates human motion trajectory conditioned on textual descriptions of tasks and object motion trajectories; and **2) MF-MDM R**, which refines generated hand motion based on spatial hand-object relationships. The sampling process is modeled as a reversed diffusion process of gradually cleaning noised samples. The key difference for MF-MDM is to incorporate multi-object related probabilistic conditions into existing transformer encoder. To achieve this, we employ an extra layer, Sequential Merging, to aggregate spatial relationships in the interaction scene at each frame. The object motion trajectories and the previously diffused hand motion trajectory are projected to the same dimension and aggregated. For the refine model MF-MDM R, we append hand-object distances as an extra spatial information for Sequential Merging layer. The aggregated embedding sequence is combined with other tokens before being fed into the main transformer encoder: the noising step token, the text embedding of the task description from the CLIP text encoder, and the aggregated object geometry embeddings from the PointBert encoder. We also provide the quantitative evaluations in Tab. 3 and qualitative visualization in Fig. 6.

5.3. Complex Task Completion (CTC)

OakInk2 brings in a new application – breaking *Complex Task* goals into paths of *Primitive* motions. The Complex Task Completion (CTC) is to generate hands motion trajectories based on a textual description of the scene and the task objectives. Considering the challenge of direct translation from complex task and scene text to end-to-end motion generation, which involves a transition across multiple modalities, there is currently no adequate framework to address this problem. Therefore, we decompose CTC into three stages, tackling each one sequentially.

The process initially begins with text-based **1) *Primitive* planning**. The recent breakthroughs in foundation models [48, 69], such as Large Language Models (LLMs), allow us to utilize them as the task planner, as these models already have the capability to plan the *Primitive* execution path, while only requiring proper guidance and context. The output of this stage is a task planning script that includes the execution order for each *Primitive*. Subsequently, the problem is reformulated into generating the hand and object motion trajectories for each *Primitive*, based on the target task and scene state, thus modeling $P(\mathcal{P}_h, \mathcal{T}_o | \mathbf{text}_{\mathrm{PT}})$. We again break this down into two subtasks: **2) object trajectory retrieval**, *i.e.* $P(\mathcal{T}_o | \mathbf{text}_{\mathrm{PT}})$ and **3) hand motion generation** *i.e.* $P(\mathcal{P}_h | \mathcal{T}_o, \mathbf{text}_{\mathrm{PT}})$. The former is solved by re-targeting[1] object motion from expert's demonstration to meet the newly generated random scene. The latter is our pre-defined Task-aware Motion Fulfillment model (TaMF, Sec. 5.2).

① Primitive Planning by LLMs. In this stage, we leverage the off-the-shelf GPT-4 [48] to generate program that decompose the *Complex Task* as a sequence of *Primitive*. We first embed the scene description $\mathbf{text}_{\mathrm{scene}}$, the complex task description $\mathbf{text}_{\mathrm{goal}}$ and each object's description $\{\mathbf{text}_{\mathrm{obj}}\}$ into the prompt based on manually designed templates. GPT-4 will respond to the prompt using the **program**. As shown in Fig. 8's code block, this program instantiates the *Primitive* Dependency Graph (PDG) using a sequence of code snippets, where each node of the PDG (*Primitive*) is implemented as a `execute([primitive], ...)` function, and the edge of the PDG is implemented as function's calling order. Then we use a dependency checker built upon the PDG information in OakInk2 to test whether the generated program completes the *Complex Task* without violation of constraints. If a successful program is obtained, we move to the next stage.

[1] re-target refers to the process of adjusting pre-existing motion trajectories to align with new initial and target poses of objects, ensuring compatibility with the current scene

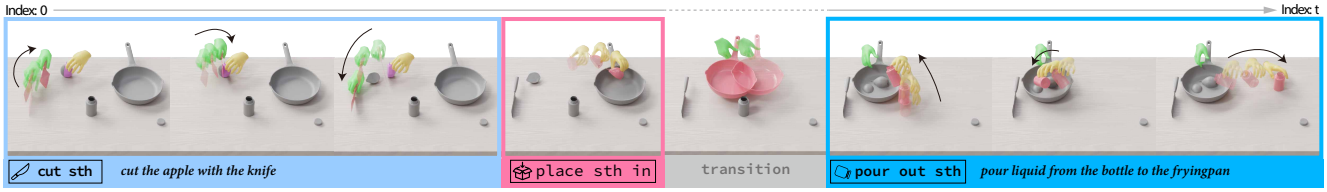


Figure 7. **Visualization of Motion Generation Outcome in Complex Task Completion.**

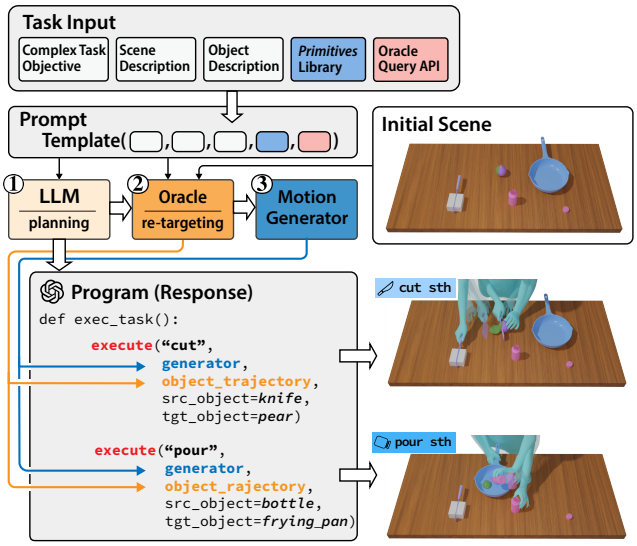


Figure 8. **The diagram of Complex Task Completion.** The task input populates a predefined template to generate the prompt for planning. The ① LLM (GPT-4) responds with code of the program's execution path, delineating the DAG for *Primitive* dependency. Within the code response block, the `orange` snippets marks the ② Oracle to re-target object trajectories; the `blue` snippets indicate ③ motion generators for *Primitives*.

At this moment, the `execute()` function in Fig. 8's code snippets remain incomplete, lacking two pivotal components: the `object_trajectory` and the hand motion `generator`. We will address these components in the following two stages.

② Object Trajectories Retrieval from Oracle. Accomplishing a *Primitive* task necessitates the object's motion trajectories within that context. In this stage, we leverage an Oracle to retrieve object motion trajectories based on a certain scene and *Primitive*. The term "Oracle" denotes a dual-function capability: 1) pursuant to a given *Primitive*, it fetches the object motion trajectories within the OakInk2 dataset, and 2) it re-targets these expert-derived trajectories based on the initial, functional and post poses of the objects, thereby conforming to new scene requirements and generating the desired `object_trajectory`.

③ Hand Motion Generation with TaMF. Once the object trajectories are obtained, the final stage is to generate hand motion trajectories for each *Primitive*. To this end, we utilize our previously designed Task-aware Motion Fulfillment model (TaMF, Sec. 5.2) as a generalist `generator` (indicating that a singular TaMF model accommodates all *Primitives*). After populating all `execute()` functions with the determined object trajectories and generator, the program is executed in sequel and all the *Primitive* trajectories are connected by interpolation. This interpolation ensures smooth transitions by linking the final state of a preceding trajectory with the initial state of the subsequent one.

We show an example of the generated motions for *Complex Task* in Fig. 7. Details of test scene generation, prompts and templates, evaluations of primitive planning, success/-failure cases are referred to Sup. Mat.

6. Future Works

OakInk2 is a dataset packing a variety of hand-object interactions for human completion of long-horizon and multi-goal complex manipulation tasks. OakInk2 incorporates *Primitive* demonstrations, characterized as minimal interactions that fulfill object affordance, and *Complex Tasks* demonstrations, which also include their decomposition into interdependent *Primitives*.

First, we expect OakInk2 to support large-scale language-manipulation pre-training, improving the performance of multi-modal (*e.g.* vision-language-action [69]) models for Complex Task Completion. In the longer term, we expect OakInk2 can potentially support learning frameworks capable of end-to-end text-to-manipulation generation.

Second, OakInk2 can empower various embodied manipulation tasks by re-targeting the collected demonstrations of *Primitives* to different embodiments, such as heterogeneous hands and platforms as [23, 53, 54, 61, 64] implied. The interaction scenarios constructed in OakInk2 can also be transferred and integrated into existing simulation environments [45, 60] to support embodied learning on object manipulation.

Acknowledgments. This work was supported by the National Key Research and Development Project of China (No. 2022ZD0160102), National Key Research and Development Project of China (No. 2021ZD0110704), Shanghai Artificial Intelligence Laboratory, XPLORER PRIZE grants, and 2023 Shanghai Pujiang X Program Project (No. 23511103104).

References

[1] Ahmed Tawfik Aboukhadra, Jameel Malik, Ahmed Elhayek, Nadia Robertini, and Didier Stricker. THOR-Net: End-to-end graformer-based realistic two hands and object reconstruction with self-supervision. In *Winter Conference on Applications of Computer Vision (WACV)*, 2023. 2

[2] Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quon Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. RT-H: Action hierarchies using language. *arXiv preprint arXiv:2403.01823*, 2024. 3

[3] Samarth Brahmbhatt, Chengcheng Tang, Christopher D Twigg, Charles C Kemp, and James Hays. ContactPose: A dataset of grasps with object contact and hand pose. In *European Conference on Computer Vision (ECCV)*, 2020. 2, 3, 15

[4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alex Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023. 3

[5] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics transformer for real-world control at scale. *Robotics: Science and Systems (RSS)*, 2023. 3

[6] 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 (CoRL)*, 2023. 3

[7] Zhe Cao, Ilija Radosavovic, Angjoo Kanazawa, and Jitendra Malik. Reconstructing hand-object interactions in the wild. In *International Conference on Computer Vision (ICCV)*, 2021. 2

[8] Yu-Wei Chao, Wei Yang, Yu Xiang, Pavlo Molchanov, Ankur Handa, Jonathan Tremblay, Yashraj S. Narang, Karl Van Wyk, Umar Iqbal, Stan Birchfield, Jan Kautz, and Dieter Fox. DexYCB: A benchmark for capturing hand grasping of objects. In *Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 3, 15

[9] Yuanpei Chen, Chen Wang, Li Fei-Fei, and C Karen Liu. Sequential dexterity: Chaining dexterous policies for long-horizon manipulation. *arXiv preprint arXiv:2309.00987*, 2023. 3

[10] Shuo Cheng and Danfei Xu. LEAGUE: Guided skill learning and abstraction for long-horizon manipulation. *IEEE Robotics and Automation Letters*, 2023. 3

[11] Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mo-

han Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Mart'in-Mart'in, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, and Zipeng Lin. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023. 3

[12] Enric Corona, Albert Pumarola, Guillem Alenya, Francesc Moreno-Noguer, and Grégory Rogez. GanHand: Predicting human grasp affordances in multi-object scenes. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 3, 15

[13] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. *International Journal of Computer Vision*, 2022. 15

[14] Bardia Doosti, Shujon Naha, Majid Mirbagheri, and David Crandall. HOPE-Net: A graph-based model for hand-object pose estimation. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[15] Zicong Fan, Omid Taheri, Dimitrios Tzionas, Muhammed Kocabas, Manuel Kaufmann, Michael J Black, and Otmar Hilliges. ARCTIC: A dataset for dexterous bimanual hand-object manipulation. In *Computer Vision and Pattern Recognition (CVPR)*, 2023. 2, 3, 6, 13, 15

[16] Guillermo Garcia-Hernando, Shanxin Yuan, Seungryul Baek, and Tae-Kyun Kim. First-person hand action benchmark with rgb-d videos and 3D hand pose annotations. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 15

[17] Anindita Ghosh, Rishabh Dabral, Vladislav Golyanik, Christian Theobalt, and Philipp Slusallek. IMoS: Intent-driven full-body motion synthesis for human-object interactions. In *Computer Graphics Forum*, 2023. 2, 3

[18] James J Gibson. *The ecological approach to visual perception: classic edition*. Psychology Press, 2014. 2

[19] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Computer Vision and Pattern Recognition (CVPR)*, 2022. 6, 15

[20] Kristen Grauman, Andrew Westbury, Lorenzo Torresani, Kris Kitani, Jitendra Malik, Triantafyllos Afouras, Kumar Ashutosh, Vijay Baiyya, Siddhant Bansal, Bikram Boote, et al. Ego-exo4d: Understanding skilled human activity from first-and third-person perspectives. *arXiv preprint arXiv:2311.18259*, 2023. 6, 15

[21] Shreyas Hampali, Mahdi Rad, Markus Oberweger, and Vincent Lepetit. Honnotate: A method for 3D annotation of hand and object poses. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 3, 6, 15

[22] Shreyas Hampali, Sayan Deb Sarkar, Mahdi Rad, and Vincent Lepetit. Keypoint transformer: Solving joint identification in challenging hands and object interactions for accurate 3D pose estimation. In *Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[23] Ankur Handa, Karl Van Wyk, Wei Yang, Jacky Liang, Yu-Wei Chao, Qian Wan, Stan Birchfield, Nathan Ratliff, and Dieter Fox. DexPilot: Vision-based teleoperation of dexterous robotic hand-arm system. In *International Conference on Robotics and Automation (ICRA)*, pages 9164–9170. IEEE, 2020. 8

[24] Yana Hasson, Gul Varol, Dimitrios Tzionas, Igor Kalevatykh, Michael J Black, Ivan Laptev, and Cordelia Schmid. Learning joint reconstruction of hands and manipulated objects. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 3, 15

[25] Yana Hasson, Bugra Tekin, Federica Bogo, Ivan Laptev, Marc Pollefeys, and Cordelia Schmid. Leveraging photometric consistency over time for sparsely supervised hand-object reconstruction. In *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[26] Yana Hasson, Gül Varol, Ivan Laptev, and Cordelia Schmid. Towards unconstrained joint hand-object reconstruction from rgb videos. In *International Conference on 3D Vision (3DV)*, 2021. 2

[27] De-An Huang, Suraj Nair, Danfei Xu, Yuke Zhu, Animesh Garg, Li Fei-Fei, Silvio Savarese, and Juan Carlos Niebles. Neural task graphs: Generalizing to unseen tasks from a single video demonstration. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[28] 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 (ICML)*. PMLR, 2022. 3

[29] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. VoxPoser: Composable 3D value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023. 3

[30] Juntao Jian, Xiuping Liu, Manyi Li, Ruizhen Hu, and Jian Liu. AffordPose: A large-scale dataset of hand-object interactions with affordance-driven hand pose. *arXiv preprint arXiv:2309.08942*, 2023. 2, 3, 15

[31] Hanwen Jiang, Shaowei Liu, Jiashun Wang, and Xiaolong Wang. Hand-object contact consistency reasoning for human grasps generation. In *International Conference on Computer Vision (ICCV)*, 2021. 2

[32] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 14

[33] Christopher A Kurby and Jeffrey M Zacks. Segmentation in the perception and memory of events. *Trends in cognitive sciences*, 2008. 2

[34] Taein Kwon, Bugra Tekin, Jan Stuhmer, Federica Bogo, and Marc Pollefeys. H2O: Two hands manipulating objects for first person interaction recognition. In *International Conference on Computer Vision (ICCV)*, 2021. 2, 3, 6, 15

[35] Jiefeng Li, Siyuan Bian, Ailing Zeng, Can Wang, Bo Pang, Wentao Liu, and Cewu Lu. Human pose regression with residual log-likelihood estimation. In *International Conference on Computer Vision (ICCV)*, 2021. 6

[36] Jiaman Li, Jiajun Wu, and C Karen Liu. Object motion guided human motion synthesis. *ACM Transactions on Graphics (TOG)*, 2023. 3

[37] Kailin Li, Lixin Yang, Zenan Lin, Jian Xu, Xinyu Zhan, Yifei Zhao, Pengxiang Zhu, Wenxiong Kang, Kejian Wu, and Cewu Lu. FAVOR: Full-body ar-driven virtual object rearrangement guided by instruction text. In *AAAI Conference on Artificial Intelligence*, 2024. 3, 17, 18

[38] Kevin Lin, Lijuan Wang, and Zicheng Liu. End-to-end human pose and mesh reconstruction with transformers. In *Computer Vision and Pattern Recognition (CVPR)*, 2021. 6, 16

[39] Shaowei Liu, Hanwen Jiang, Jiarui Xu, Sifei Liu, and Xiaolong Wang. Semi-supervised 3D hand-object poses estimation with interactions in time. In *Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[40] Yunze Liu, Yun Liu, Che Jiang, Kangbo Lyu, Weikang Wan, Hao Shen, Boqiang Liang, Zhoujie Fu, He Wang, and Li Yi. HOI4D: A 4d egocentric dataset for category-level human-object interaction. In *Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 6, 15

[41] Yun Liu, Haolin Yang, Xu Si, Ling Liu, Zipeng Li, Yuxiang Zhang, Yebin Liu, and Li Yi. TACO: Benchmarking generalizable bimanual tool-action-object understanding. *arXiv preprint arXiv:2401.08399*, 2024. 3, 15

[42] Jun Lv, Wenqiang Xu, Lixin Yang, Sucheng Qian, Chongzhao Mao, and Cewu Lu. HandTailor: Towards high-precision monocular 3D hand recovery. In *British Machine Vision Conference (BMVC)*, 2021. 6

[43] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022. 13

[44] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. AMASS: Archive of motion capture as surface shapes. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 5, 13, 14

[45] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021. 8

[46] Andrew T Miller and Peter K Allen. Graspit!: A versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122, 2004. 15

[47] Takehiko Ohkawa, Kun He, Fadime Sener, Tomas Hodan, Luan Tran, and Cem Keskin. AssemblyHands: Towards egocentric activity understanding via 3D hand pose estimation. In *Computer Vision and Pattern Recognition (CVPR)*, 2023. 2, 6, 15

[48] OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 5, 7, 17

[49] OptiTrack. Motive: Optical motion capture software. https://optitrack.com/software/motive/. 13

[50] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed AA Osman, Dimitrios Tzionas, and Michael J Black. Expressive Body Capture: 3D hands, face, and body from a single image. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 5, 13

[51] Mathis Petrovich, Michael J Black, and Gül Varol. Action-conditioned 3D human motion synthesis with transformer VAE. In *International Conference on Computer Vision (ICCV)*, 2021. 3

[52] Mathis Petrovich, Michael J Black, and Gül Varol. TEMOS: Generating diverse human motions from textual descriptions. In *European Conference on Computer Vision (ECCV)*, 2022. 3

[53] Yuzhe Qin, Hao Su, and Xiaolong Wang. From one hand to multiple hands: Imitation learning for dexterous manipulation from single-camera teleoperation. *IEEE Robotics and Automation Letters*, 2022. 2, 8

[54] Yuzhe Qin, Yueh-Hua Wu, Shaowei Liu, Hanwen Jiang, Ruihan Yang, Yang Fu, and Xiaolong Wang. DexMV: Imitation learning for dexterous manipulation from human videos. In *European Conference on Computer Vision (ECCV)*, 2022. 2, 8

[55] 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 *International Conference on Robotics and Automation (ICRA)*, 2023. 3

[56] Omid Taheri, Nima Ghorbani, Michael J Black, and Dimitrios Tzionas. GRAB: A dataset of whole-body human grasping of objects. In *European Conference on Computer Vision (ECCV)*, 2020. 2, 3, 6, 13, 15

[57] Omid Taheri, Vasileios Choutas, Michael J Black, and Dimitrios Tzionas. GOAL: Generating 4d whole-body motion

for hand-object grasping. In *Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3, 18

[58] Omid Taheri, Yi Zhou, Dimitrios Tzionas, Yang Zhou, Duygu Ceylan, Soren Pirk, and Michael J Black. Grip: Generating interaction poses using spatial cues and latent consistency. In *International Conference on 3D Vision (3DV)*, 2024. 3

[59] Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or, and Amit H Bermano. Human motion diffusion model. *International Conference on Learning Representations (ICLR)*, 2023. 3, 7

[60] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012. 8

[61] Weikang Wan, Haoran Geng, Yun Liu, Zikang Shan, Yaodong Yang, Li Yi, and He Wang. UniDexGrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning. *arXiv preprint arXiv:2304.00464*, 2023. 8

[62] Yan Wu, Jiahao Wang, Yan Zhang, Siwei Zhang, Otmar Hilliges, Fisher Yu, and Siyu Tang. SAGA: Stochastic whole-body grasping with contact. In *European Conference on Computer Vision (ECCV)*, 2022. 2, 3, 17

[63] Danfei Xu, Suraj Nair, Yuke Zhu, Julian Gao, Animesh Garg, Li Fei-Fei, and Silvio Savarese. Neural task programming: Learning to generalize across hierarchical tasks. In *International Conference on Robotics and Automation (ICRA)*, 2018. 3

[64] Yinzhen Xu, Weikang Wan, Jialiang Zhang, Haoran Liu, Zikang Shan, Hao Shen, Ruicheng Wang, Haoran Geng, Yijia Weng, Jiayi Chen, et al. Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy. In *Computer Vision and Pattern Recognition (CVPR)*, 2023. 8

[65] Lixin Yang, Xinyu Zhan, Kailin Li, Wenqiang Xu, Jiefeng Li, and Cewu Lu. CPF: Learning a contact potential field to model the hand-object interaction. In *Computer Vision and Pattern Recognition (CVPR)*, 2021. 14

[66] Lixin Yang, Kailin Li, Xinyu Zhan, Jun Lv, Wenqiang Xu, Jiefeng Li, and Cewu Lu. ArtiBoost: Boosting articulated 3D hand-object pose estimation via online exploration and synthesis. In *Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[67] Lixin Yang, Kailin Li, Xinyu Zhan, Fei Wu, Anran Xu, Liu Liu, and Cewu Lu. OakInk: A large-scale knowledge repository for understanding hand-object interaction. In *Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3, 6, 15, 16

[68] Lixin Yang, Jian Xu, Licheng Zhong, Xinyu Zhan, Zhicheng Wang, Kejian Wu, and Cewu Lu. POEM: Reconstructing hand in a point embedded multi-view stereo. In *Computer Vision and Pattern Recognition (CVPR)*, 2023. 6

[69] Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3D-VLA: A 3D vision-language-action generative world model. *arXiv preprint arXiv:2403.09631*, 2024. 7, 8

[70] Hao Zheng, Regina Lee, and Yuqian Lu. HA-ViD: A human assembly video dataset for comprehensive assembly knowledge understanding. *arXiv preprint arXiv:2307.05721*, 2023. 15

[71] Juntian Zheng, Qingyuan Zheng, Lixing Fang, Yun Liu, and Li Yi. CAMS: Canonicalized manipulation spaces for category-level functional hand-object manipulation synthesis. In *Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[72] Zehao Zhu, Jiashun Wang, Yuzhe Qin, Deqing Sun, Varun Jampani, and Xiaolong Wang. ContactArt: Learning 3D interaction priors for category-level articulated object and hand poses estimation. *arXiv preprint arXiv:2305.01618*, 2023. 2, 3, 15

[73] Christian Zimmermann, Duygu Ceylan, Jimei Yang, Bryan Russell, Max Argus, and Thomas Brox. FreiHAND: A dataset for markerless capture of hand pose and shape from single rgb images. In *International Conference on Computer Vision (ICCV)*, 2019. 16

Supplementary Materials

Table of Contents

A Annotation Details
- **A.1** Platform Calibration & Synchronization
- **A.2** Data Cleaning
- **A.3** Human Pose and Surface

B Dataset Meta Information
- **B.1** Task-specific Subsets

C Dataset Evaluation
- **C.1** Cross-Dataset Validation
- **C.2** Physical Property Assessment

D Tasks and Benchmarks
- **D.1** Task-aware Motion Fulfillment

E Application: Complex Task Completion

F Dataset Inspection
- **F.1** Task List
- **F.2** Visualization

A. Annotation Details

A.1. Platform Calibration & Synchronization

The MoCap system is calibrated via a specialized wand provided by the vendor. The cameras in the multi-camera system are calibrated using ArUco cubes. These cameras are attached with reflective markers to be tracked by the MoCap system. These calibration tools are shown in Fig. 9. The two systems are time synchronized with software synchronization tools bundled in the ROS2 [43].

A.2. Data Cleaning

In this section, we present a brief description of the process used to clean the reflective marker positions captured by the MoCap system, in preparation for subsequent object pose and human pose computations. Inherent limitations of the MoCap system inevitably lead to errors in the reflective marker positions obtained: in extreme cases of occlusion, the system may fail to detect and record some markers; ghost markers may be included due to unwanted environmental reflections; when two or more markers come into close proximity, the system may incorrectly assign their labels or falsely identify them as a single marker. These limitations lead to the introducing of a manual mechanism for cleaning and post-processing captured data.

The data cleaning procedure is composed of two components: 1) the MoCap post-processing software; 2) a multi--view interactive editor. We invite three professional annotators for data cleaning. The annotators first sequentially check the location of the captured reflective markers in the MoCap post-processing software [49]. They then proceed to eliminate ghost points, split overlapping markers, correct mislabeled markers, and fill short gaps in the marker trajectories. The annotators, following the order from articulated parts to rigid bodies, human bodies, and both hands, systematically clean the results of the collected markers. Subsequently, the sequences are exported to the multi-view interactive editor. In the editor, annotators verify the cleaned MoCap results and recover the marker positions in extreme occlusion cases through triangulation-based annotations from 2D point locations in multiple views. The results are combined to get the cleaned captured reflective marker positions in the capture volume.

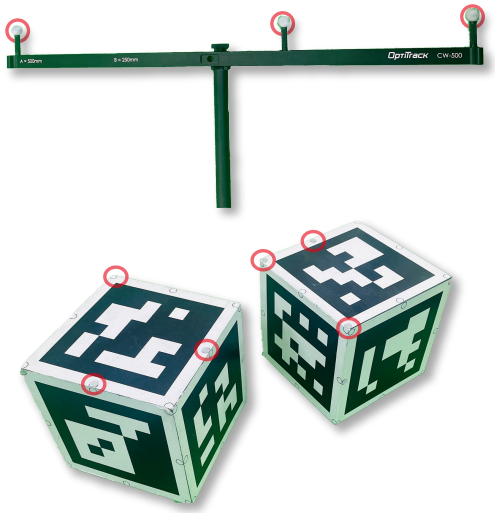


Figure 9. **Illustration of Platform Calibration Tools.** The top is the vender-provided calibration wand for the MoCap system. The bottom are the ArUco cubes attached with reflective markers (circled in red). The ArUco patterns are for unifying the camera in the multi-camera system, while the surface-attached reflective markers are used for unifying the cameras with the MoCap system.

A.3. Human Pose and Surface

We employ a two-stage fitting approach inspired by the application of the MoSH++ algorithm in [15, 44, 56] for the SMPL-X [50] annotations. The first stage registers the subject's SMPL-X *shape* parameters and establishes correspondence mapping from the markerset to the surface of SMPL-X model. The second stage registers SMPL-X *pose* parameters for each frame in the sequence. The two-stage fitting pipeline is implemented on PyTorch for its automatic differentiation support and common gradient descent based algorithms are used to solve for both stages.

The first stage. Let $\bar{\beta}$ be *shape* parameters. Let $P_{\mathcal{M}}^{(c)} \in \mathbb{R}^{N_{\mathcal{M}} \times 3}$ be surface marker positions lying in SMPL-X canonical space, where $N_{\mathcal{M}}$ is the number of markers in the target markerset. Let $\theta = \{\theta_i\}$ be SMPL-X *pose* parameters for each frame i when the subject is in T-pose. The first stage could be formulated as an optimization process to minimize the distance between the observed markers

and the reconstructed markers derived from surface marker positions lying in SMPL-X canonical space, as shown in Eq. (1).

$$\min_{\theta,\beta,P_{\mathcal{M}}^{(c)}} E = \lambda_1 E_{\text{recon}}\big(\theta,\beta,P_{\mathcal{M}}^{(c)}\big) + \lambda_2 E_{\text{prior(b)}}\big(\theta\big) + \lambda_3 E_{\text{plau(h)}}\big(\theta\big) + \lambda_4 E_{\text{reg}}\big(\theta,\beta,P_{\mathcal{M}}^{(c)}\big) \tag{1}$$

The main cost is E_{recon}, which is the distance between the observed markers $P_{\mathcal{M}}$ and the reconstructed markers $\hat{P}_{\mathcal{M}}$ derived from surface marker positions. Let $\boldsymbol{V}^{(c)}$ be the surface vertices in the canonical space of SMPL-X, $\boldsymbol{V}$ be the reconstructed surface vertices. The markerset correspondence function $\mathcal{C}(\cdot)$ uses markerset position $P_{\mathcal{M}}^{(c)}$ and surface vertices $\boldsymbol{V}^{(c)}$ in canonical space to recover the markerset positions $\hat{P}_{\mathcal{M}}$ from the current reconstructed surface vertices $\boldsymbol{V}$. It first projects the markers in canonical space $P_{\mathcal{M}}^{(c)}$ into local frames formed by the surface vertices to get the vertex index $I_{\mathcal{M}}$ and coefficients in local frames $C_{\mathcal{M}}$. Then it uses the index to recover frames on posed vertices $\boldsymbol{V}$ and the coefficients in local frames to recover marker positions on the posed SMPL-X bodies. E_{recon} can then be expressed as in Eq. (2).

$$\boldsymbol{V}^{(c)} = \text{SMPL-X}\big(\mathbf{0},\bar{\beta}\big) \qquad \boldsymbol{V} = \text{SMPL-X}\big(\theta,\bar{\beta}\big)$$
$$E_{\text{recon}} = \big\|P_{\mathcal{M}} - \hat{P}_{\mathcal{M}}\big\|^2 = \Big\|P_{\mathcal{M}} - \mathcal{C}\big(\boldsymbol{V},P_{\mathcal{M}}^{(c)};\boldsymbol{V}^{(c)}\big)\Big\|^2 \tag{2}$$

$E_{\text{prior(b)}}\big(\theta\big)$, $E_{\text{plau(h)}}\big(\theta\big)$, and $E_{\text{reg}}\big(\theta,\beta,P_{\mathcal{M}}^{(c)}\big)$ are auxiliary cost terms. $E_{\text{prior(b)}}\big(\theta\big)$ is an auxiliary term that minimizes negative log-likelihood of human body poses computed by prior from pre-existing datasets following the practice in [44]. $E_{\text{plau(h)}}\big(\theta\big)$ is an implementation of anatomy loss in [65] on the SMPLX model, designed to prevent distortion in the pose of the human body during the fitting process, enhancing its physical plausibility. $E_{\text{reg}}\big(\theta,\beta,P_{\mathcal{M}}^{(c)}\big)$ is an auxiliary term that regularize the optimization variables.

The second stage. In the second stage, we fit the subject's pose $\theta = \{\theta_t\}$ throughout the interaction process based on the shape $\bar{\beta}$ and marker correspondence $\mathcal{C}(\cdot)$ obtained in the first stage.

For each frame t in the sequence, we optimize the subject's SMPL-X *pose* parameter θ_t to minimize a combination cost composed of observed marker reconstruction error $E_{\text{recon}}\big(\theta_t\big)$, body pose prior $E_{\text{prior(b)}}\big(\theta_t\big)$, hand anatomy abnormality $E_{\text{plau(h)}}\big(\theta_t\big)$, hand-object intersection $E_{\text{plau(ho)}}\big(\theta_t\big)$ and other auxiliary regularization costs.

$$\min_{\theta_t} E = \lambda_1 E_{\text{recon}}\big(\theta_t\big) + \lambda_2 E_{\text{prior(b)}}\big(\theta_t\big) + \lambda_3 E_{\text{plau(h)}}\big(\theta_t\big) + \lambda_4 E_{\text{plau(ho)}}\big(\theta_t\big) + \lambda_5 E_{\text{reg}}\big(\theta_t\big) \tag{3}$$

$E_{\text{plau(h)}}\big(\theta_t\big)$, $E_{\text{prior(b)}}\big(\theta_t\big)$, and $E_{\text{plau(h)}}\big(\theta_t\big)$ are the same cost terms as in the first stage. $E_{\text{plau(ho)}}$ penalizes the penetration and intersection between the interacting hands and objects by sampling internal points inside hand meshes and computing the sum of their signed distance function values of the objects. $E_{\text{reg}}\big(\theta_t\big)$ not only includes regularization terms for the optimization variables but also contains velocity regularization terms to keep the smoothness of the annotated trajectories.

Optimization We implement the two-stage fitting pipeline on PyTorch for its automatic differentiation support. We adopt Adam [32] as the optimizer to solve for both stages, as it is widely applied and suitable for non-convex cost terms introduced in both stages. We propose an early stopping mechanism for better running speed of the second stage: if there is no significant reduction in the fitting cost over a number of consecutive frames that exceeds a specific threshold, the optimization process will be terminated.

B. Dataset Meta Information

B.1. Task-specific Subsets

Since OakInk2 is intended for various types of tasks, we create multiple subsets with different strategies for sample selection and data organization tailored to each specific task. To obtain these subsets, we apply a few heuristics to determine whether each sample meets the requirements of the task it needs to support. For instance, the visibility of hands or objects in image samples is essential for supporting related vision tasks. We verify the individual and combined segmentation masks for hands and objects in the images. If the proportion of the combined segmentation mask to its individual counterpart exceeds a certain threshold, we consider the instance as *visible* in the current frame. We regard the object to interact as *grasped* if it is close enough to hands (minimal distance $\leq 5\,\text{mm}$) and *lifted* (height displacement to the initial state $\geq 5\,\text{mm}$).

We show the features and the construction methods for task-specific subsets in the following list.

OakInk2-H-SV Subset for hand reconstruction from single-view images. We select views that the subjects' hands are *visible* to form this subset. This subset supports task single-view Hand Mesh Recovery.

OakInk2-H-MV Subset for hand reconstruction from multi-view images. We select combined views from different cameras as a single sample in this subset if the subjects' hands are *visible* in a majority of camera views. This subset supports task multi-view Hand Mesh Recovery.

OakInk2-HO Subset for hand-object pose estimation or reconstruction from images. We select views that the subjects' hands are *visible* and the object is *grasped* to form this subset.

OakInk2-Grasp Subset for grasps on the objects. We

Dataset	image mod.	resolution	#frame	#views	#subj	#obj	3D gnd.	real / syn.	label method	hand pose	obj pose	afford. inter.	dynamic inter.	long-horizon	task decomp.
EPIC-KITCHEN-100 [13]	✓	∼	20M†	1	37	–	✗	–	–	✗	✗	✗	✗	✓	✓
Ego4D [19]	✓	∼‡	∼†	1	931	–	✗	–	–	✗	✗	✗	✗	✓	✓
HA-ViD [70]	✓	1280 × 720	1.5M	3	30	40	✗	–	–	✗	✗	✗	✗	✓	✓
FPHAB [16]	✓	1920 × 1080	105K	1	6	4	✓	real	mocap	✓	✓	✓	✓	✗	✗
ObMan [24]	✓	256 × 256	154K	1	20	3K	✓	syn	simulate	✓	✓	✗	✗	✗	✗
YCBAfford [12]	✓	–	133K	1	1	21	✓	syn	manual	✓	✗	✗	✗	✗	✗
HO3D [21]	✓	640 × 480	78K	1-5	10	10	✓	real	auto	✓	✓	✗	✓	✗	✗
ContactPose [3]	✓	960 × 540	2.99M	3	50	25	✓	real	auto	✓	✓	✓	✗	✗	✗
GRAB [56]	✗	–	1.62M	–	10	51	✓	real	mocap	✓	✓	✓	✓	✗	✗
DexYCB [8]	✓	640 × 480	582K	8	10	20	✓	real	crowd	✓	✓	✗	✓	✗	✗
H2O [34]	✓	1280 × 720	571K	5	4	8	✓	real	auto	✓	✓	✓	✓	✗	✗
HOI4D [40]	✓	1280 × 800	3M	1	9	1000	✓	real	crowd	✓	✓	✓	✓	✗	✗
ARCTIC [15]	✓	2800 × 2000	2.1M	9	10	11	✓	real	mocap	✓	✓	✓	✓	✗	✗
ContactArt [72]	✓	–	332K	–	–	80	✓	real	transfer	✓	✓	✗	✓	✗	✗
AssemblyHands [47]	✓	1920 × 1080	3.03M	12	34	–	✓	real	semi-auto	✓	✗	✓	✓	✓	✓
AffordPose [30]	✗	–	–	–	–	641	✓	syn	manual	✓	✓	✓	✗	✗	✗
TACO [41]	✓	4096 × 3000‡	5.2M	13	14	196	✓	real	auto	✓	✓	✓	✓	✗	✗
Ego-Exo4D [20]	✓	∼‡	∼†	5-6	839	–	✓	real	semi-auto	✓	✗	✗	✓	✓	✓
OakInk-Image [67]	✓	848 × 480	230K	4	12	100	✓	real	crowd	✓	✓	✓	✓	✗	✗
OakInk-Shape [67]	✗	–	–	–	–	1700	✓	real	transfer	✓	✓	✓	✗	✗	✗
OakInk2	✓	848 × 480	**4.01M**	4	9	75	✓	real	mocap	✓	✓	✓	✓	✓	✓

Table 4. **A cross-comparison among various public datasets.** ∼: The value is either not provided on the paper or measured in a different unit. †: Datasets measure in record time rather than number of captured frames. In particular, EPIC-KITCHEN-100 contains more than 100 hours of video, Ego4D 3670 hours, and Ego-Exo4D 1422 hours. They are larger in scale than any other dataset listed in the table. ‡: Dataset has a mixed resolution.
Legend:
image mod. : Image Modality. ✓ means real image captures; ✓ means synthetic (rendered) images; ✗ means no image modality provided.
3D gnd. : 3D grounding. ✓ means the dataset contains 3D grounding annotations; ✗ means the dataset is 2D only.
real / syn. : Interaction is real / synthetic. Here syn indicates the interactions come from certain grasp/interaction synthesizer.
label method : Label Method of 3D grounding information. For synthetic interactions, "simulate" indicates interactions are retrieved from physical-based grasp simulators, *e.g.* GraspIt! [46]; "manual" indicates interactions are labeled with human labor. For real interactions, "mocap" indicates the interactions are captured by MoCap systems; "crowd" indicates the interactions are derived from crowd-source keypoint annotations; "auto" indicates the interactions are retrieved from automatic annotation pipelines; "semi-auto" indicates a hybrid of "crowd" and "auto" methods.
afford. inter. : Affordace-based Interaction. ✓ means the interactions captured are affordance-aware and explicitly labeled; ✓ means the interactions are afforance-aware but grouped in coarse-grained labels like intentions; ✗ means the interactions are not organized by object afforances.
dynamic inter. : Dynamic Interaction. ✓ means the dataset captures dynamic sequence of hand-object interactions; ✗ means the dataset captures static grasps that do not change during the interaction process.
long-horizon : Long-horizon Tasks. As in the main text, ✓ means the dataset contains captured interactions that involved more than one object afforances; ✗ vice versa.
task decomp. : Task Decomposition. As in the main text, ✓ means the dataset contains annotations that decomposition a complex task into multiple segments; ✗ vice versa.

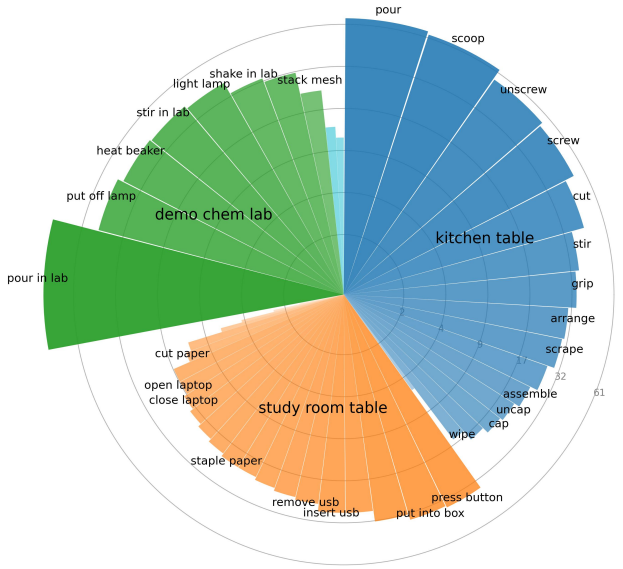
select frames that the object is *grasped* to form this subset.

OakInk2-Motion-Approach&Retreat Subset for the interaction process that the subjects approach and grab the object for future tasks. We select frames from the sequence in one *Primitive Task* that cover the process of *approach* and *grasp* the object are collected to form this subset. This subset provides auxiliary information in Task-aware Motion Fulfillment and Object Trajectories Retrieval from Oracle Queries in Complex Task Completion.

OakInk2-Motion-Task Subset for the interaction process that the subjects complete a task and fulfill one object affordance. We select frames from the sequence in one *Primitive Task* that cover the process from the *grasp* of the object to the *completion* of the task are collected to form this subset. This subset supports Task-aware Motion-Fulfillment.

C. Dataset Evaluation

Annotation on 3D hand keypoints undergo cross-dataset validation with a reconstruction model, while the 3D poses



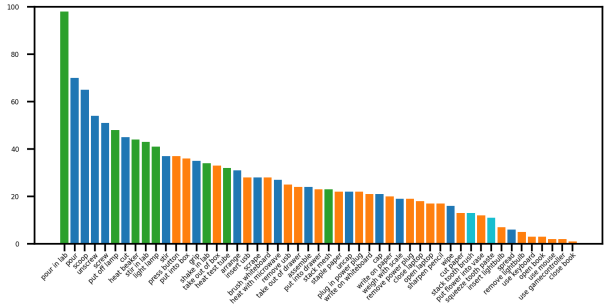


Figure 10. **Distribution of *Primitive Task* demonstrations.** The sub-figure above displays the proportion of *Primitive Task* demonstrations across various scenarios within the entire OAKINK2 dataset, with frequently occurring *Primitive Tasks* highlighted. The sub-figure below presents a list of *Primitive Tasks* recorded in OAKINK2, along with the illustration of their corresponding quantity distribution. A list of all recorded *Primitive Tasks* and *Complex Tasks* can be found at Tab. 7 and Tab. 8.

associated with grasping actions are examined for their physical property integrity.

C.1. Cross-Dataset Validation

We perform cross-dataset validation to verify the consistency of 3D hand keypoint annotations in OAKINK2 with pre-existing datasets. We train a single-view hand mesh recovery model [38] separately on three different training schemes: FreiHAND [73] only, OAKINK2-H-SV only, and a mixture of these two sets. We evaluate MPJPE and MPVPE after Procrustes analysis on OakInk-image (SP2) [67], and the results shown in Tab. 5 indicates a consistent improvement on these metrics, verifying that OAKINK2 complements existing datasets and boosts existing models.

Train	Test	PA-MPJPE (*mm*) ↓	PA-MPVPE (*mm*) ↓
1) FreiHAND	OakInk-image (SP2)	12.07	11.96
2) OAKINK2-H-SV	OakInk-image (SP2)	12.60	11.04
1) & 2) mixture	OakInk-image (SP2)	**10.94**	**9.67**

Table 5. **Cross dataset validation** for OAKINK2.

C.2. Physical Property Assessment

To evaluate the quality of the 3D poses associated with grasping actions in OAKINK2, we inspect several physical-based metrics that assess the feasibility and stability of captured hand-object interactions. We restrict the samples to be evaluated based on certain rules (the objects in interaction need to be grasped and lifted), ensuring that these physics-based quality metrics accurately reflect the quality of the dataset during the interaction process. We compare OAKINK2-Grasp (-G.) with two subsets of OakInk: OakInk-Core and OakInk-Shape (Tab. 6). We observe that, despite the use of the mocap system as the primary annotation method for easily scaling up the capture process, OAKINK2 still achieved annotation quality on par with OakInk built upon the hybrid of manual and mocap annotation. More qualitative visualizations of OAKINK2 are provided in Fig. 14.

Metrics	OAKINK2-G.	OakInk-Core	OakInk-Shape
Penet. Depth. cm↓	0.25	0.18	0.11
Solid Intsec. Vol. cm^3↓	0.61	1.03	0.62
Sim. Disp. Mean cm↓	1.83	0.98	0.94
Sim. Disp. Std cm↓	1.16	1.74	1.62

Table 6. **Quality assessment** of OAKINK2.

D. Tasks and Benchmarks

D.1. Task-aware Motion Fulfillment

Train-Val-Test Split Following the same practice as HMR, we partition the subsets at the sequence level, maintaining the proportion of samples in train/val/test sets at approximately 70%, 5%, and 25%, in alignment with OakInk.

Evaluation Metric Details

CR, Contact Ratio. This metric measures the ratio of the frames within the motion trajectories where the hand-object contact (minimum distance) is within a $5\,\mathrm{mm}$ threshold.

SIV, Solid Intersection Volume. This metric measures how much space intersection occurs during estimation. We voxelize the object mesh into 100^3 voxels, and calculate the sum of the voxel volume inside the hand surface.

PSKL-J, Power Spectrum KL divergence of Joints. This metric reflects the smoothness of the generated

motion. It measures the acceleration distribution variance between **predicted** and **g.t.** joint sequences, reporting results in both directions. We reference our implementation on [37, 62]. The notable difference is that we use hand joints for measurement, resulting in a distinct range of metric values compared to full-body joints.

FID, Fréchet Inception Distance score. This metric evaluates the realism of the generated motion trajectory. We develop a motion feature extractor based on the transformer encoder architecture. The embedding is obtained by appending a trailing token to hand motion trajectories. The embedding we use for each motion is of 64 dimensions. The encoder is trained by the classifying motion trajectories into their corresponding categories. We apply the encoder to both ground-truth trajectories and generated trajectories and compute Fréchet Inception Distance between them for motion realism evaluation.

E. Application: Complex Task Completion

Test Scene Generation. To evaluate the ability of the oracle-facilitated three-stage method described in the main text to accomplish complex tasks, we derive a set of test environments by perturbing the object positions within the complex scenarios contained in OAKINK2 dataset without altering the task objectives $\mathbf{text}_{\mathrm{goal}}$ and the descriptions of the objects' states $\{\mathbf{text}_{\mathrm{obj}}\}$.

We utilize ground truth annotations to generate variations in the test scenes. We treat specific object sets as clusters and place them in randomized locations. Objects within the same cluster share a unified offset to ensure collective randomization. This is crucial when groups of objects must maintain coherence in their movements. The process of randomization comprises four distinct *wander* steps. This helps prevent obstruction caused by other objects when an object ventures in a random direction. Hence, each object gains an enhanced opportunity to navigate around other structures within its environment. Each object's final location results from the cumulative effect of these four *wander* steps. Within each *wander* step, a maximum of eight iterations are employed for collision prevention between objects by reducing the step length to half.

Prompt Generation. We implement Primitive Planning by tweaking the Large Language Model – GPT-4 [48] in this study – so that it can generate Python code based on narrative prompts describing the current scenario and task objectives. The language model's role is to interpret this description, identify key objects involved in the task, determine the appropriate object affordance and trajectory, and generate suitable instances of *Primitives* execution organized into a feasible sequence.

Our approach to overcoming these challenges involves the design of a prompt template that incorporates both scene and task descriptions as referenced earlier. This template not only explicates the underlying code framework but also provides a sample of scenario-independent code. We further prompt the Language Learning Model (LLM) to produce a code implementation as a response, as opposed to providing an explanatory narrative of coding procedures.

Concerning the underlying code framework, to ensure robust and coherent code generation, we propose an Entity-Component-System (ECS) architecture. This structure encourages a decoupling of components, here referred to as data or state, from the system, representing the *Primitives* in our context. This approach endows us with the capability of generating uniformly styled code implementations, where the layout involves instantiating object entities, loading the affordance as a component, and submitting the *Primitives* to the execution system.

Evaluations of Primitive Planning. We employ a checker based on the *Primitive* Dependency Graph provided along with the *Complex Tasks* to be planned to benchmark the success rate of the program that is supposed to complete the task target. We analyze the checker results and observe an overall success rate of 36% in the generation of Planning codes. Concerning the number of *Primitives* incorporated within the *Complex Tasks*, we observed differing success rates. Specifically, in those *Complex Tasks* incorporating equal to or less than three *Primitives*, a success rate of 44% was obtained Conversely, in the *Complex Tasks* category incorporating between three and five *Primitives*, the success rate dropped to 20%. Notably, no success was recorded in *Complex Tasks* incorporating more than five *Primitives*. The results demonstrate that in the current setting, the Large Language Model (LLM) is adequate to handle relatively simpler *Complex Tasks*. However, in contexts of highly complex *Complex Tasks*, the LLM struggles to accurately comprehend the relationships and dependencies between objects' affordances and the corresponding *Primitives*.

Demo Planning Result. We provide a review of the results of a completed *Complex Task* within one of the constructed test scenes. The python programs generated are listed as Listing 2. This code joins all the relevant objects and their associated affordances, proceeding to execute the *Primitives* in the precise required order. We have also included an example of a failed case, presented as Listing 3, which highlights a failure in the execution of *Primitive* Planning. This failure is characterized by a superfluous *Primitive* that fulfills an unnecessary object affordance that blocks the execution path.

Alternative Motion Generation for Complex Task Completion. In addition to the TaMF-based motion generation approach presented in the main text, we explore an alternative strategy that leverages keyframe generation and motion

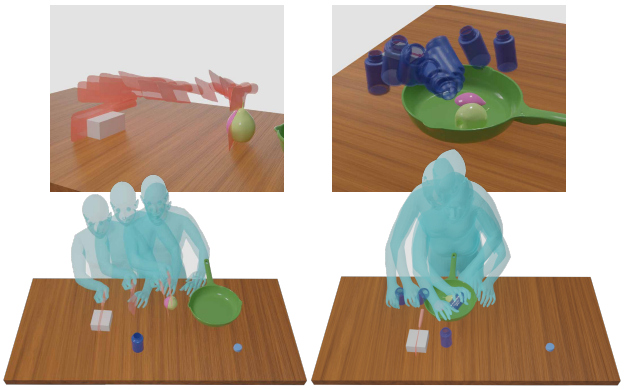


Figure 11. **Oracle Trajectories and Motion Generation** This figure illustrates the successful *Complex Task* completion of two Primitive Tasks. The top pair of images depict the oracle trajectories, while the bottom pair represents the sequential motion generated.

in-betweening as motion generator for Complex Task Completion. We adopt GNet and MNet in GOAL [57] and INet in FAVOR [37]. These models follow the pattern of first generating hand-object interactions in key frames, and then generating intermediary interaction trajectories within these frames. The generation contains three stages. In the first stage, GNet generates static grasps based on the object's initial and terminal poses. Subsequently, MNet generates motion trajectories to reach the object and retreat from the object. In the final stage, INet is fed with alternating object poses from the object motion trajectory to generate the in-between motion during the interaction process. The object oracle trajectories result in a sequence of human body movements depicted in Fig. 11. The left-hand images of Fig. 11 illustrate the sequential actions of approaching and utilizing a knife to cut a pear, representing the affordance *cut* associated with the knife under the *Primitive Task* category. The right-hand images illustrate the sequence of lifting a bottle and pouring its contents into a pan, indicative of the *Primitive Task* affordance, *pour*, as related to the bottle.

F. Dataset Inspection

F.1. Task List

Table 7. Collected *Affordances* and Designed *Primitive Tasks*.

Scenario	Affordance	Affordance Instantiation	*Primitive Task*
kitchen table	<be rearranged, _>		*rearrange*
	<store securely, sth>		
	<contain, sth>		
		<flow in, sth>	*pour*
		<pour, sth>	*pour*
		<shake, sth>	*shake*
	<secure, sth>		
		<screw into, sth>	*screw*
		<unscrew from, sth>	*unscrew*
		<cap onto, sth>	*cap*
		<uncap from, sth>	*uncap*
	<grip, sth>		*grip*
	<scoop, sth>		*scoop*
	<scrape, sth>		*scrape*
	<cut, sth>		*cut*
	<stir, sth>		*stir*
	<spread, sth>		*spread*
	<assemble into, sth>		*assemble*
	<wipe, sth>		*wipe*
	<heat with microwave, sth>		
	<contain, sth>		
		<place inside, sth>	*place inside*
		<take outside, sth>	*take outside*
	<secure, sth>		
		<shut, sth>	*close gate*
		<open, sth>	*open gate*
	<control, sth>		
		<be pressed, _>	*press button*
		<trigger, sth>	*trigger lever*
	<weigh, sth>		
	<support, sth>		*place onto*
study room table	<be rearranged, _>		*rearrange*
	<store securely, sth>		
	<contain, sth>		
		<place inside, sth>	
		<take outside, sth>	
	<secure, sth>		
		<cover, sth>	*put on lid*
		<uncover, sth>	*remove lid*
		<shut, sth>	*pull out drawer*
		<open, sth>	*push in drawer*
	<illuminate, sth>		
	<connect to, sth>		
		<connect to, power socket>	*plug in power plug*
		<deconnect from, power socket>	*remove power plug*
		<connect to, usb>	*insert usb*
		<deconnect from, usb>	*remove usb*
		<connect to, lightbulb socket>	*insert lightbulb*
		<deconnect from, lightbulb socket>	*remove lightbulb*
	<shear, paper>		
	<secure, sth>		
		<cap, pen tip>	*cap the pen*
		<uncap, pen tip>	*remove the pen cap*
	<write/draw, sth>		*write on paper*
			write on whiteboard
	<brush, whiteboard>		*brush whiteboard*
	<be sharpen by, sth>		*sharpen pencil*
	<sharpen, pencil>		*sharpen pencil*
	<staple together, paper>		*staple paper together*
	<be written/drawn by, pen/pencil>		*write on paper*
			write on whiteboard
	<be sheared by, scissors>		*shear paper*
	<be stapled together by, stapler>		*staple paper together*
	<be turn, _>		*close book*
			open book
	<display, sth>		
	<protect, sth>		
		<open, laptop lid>	*open laptop lid*
		<close, laptop lid>	*close laptop lid*
	<control, sth>		*use keyboard*
			use mouse

Table 7. Collected *Affordances* and Designed *Primitive Tasks*.

Scenario	Affordance	Affordance Instantiation	*Primitive Task*
			use gamecontroller
	<cultivate, flowers>		*put flower into vase*
	<be cultivated in, sth>		*put flower into vase*
demo chem lab	<be rearranged, _>		*rearrange*
	<store securely, experiment substances>		
	<contain, experiment substances>		
		<flow in, experiment substances>	*pour in lab*
		<pour, experiment substances>	*pour in lab*
		<shake, experiment substances>	*shake lab container*
	<secure, experiment substances>		
		<screw into, lab container>	*screw*
		<unscrew from, lab container>	*unscrew*
		<cap onto, lab container>	*cap*
		<uncap from, lab container>	*uncap*
	<contain, experiment substances>		
		<flow in, experiment substances>	*pour in lab*
		<pour, experiment substances>	*pour in lab*
		<shake, experiment substances>	*shake lab container*
	<be heated by, alcohol lamp>		*heat beaker/flask*
			heat test tube
	<stir, experiment substances>		*stir experiment substances*
	<be ignited, _>		*ignite alcohol lamp*
	<heat, lab container>		*heat beaker/flask*
			heat test tube
	<put off, alcohol lamp>		*put off alcohol lamp*
	<ignite, alcohol lamp>		*ignite alcohol lamp*
	<clamp, test tube>		*hold test tube*
	<conduct heat to, lab container>		*place asbestos mesh*
	<support, lab container>		*place asbestos mesh*
bathroom table	<be rearranged, _>		*rearrange*
	<contain, sth>		
		<squeeze out, sth>	*squeeze tooth paste*
	<secure, sth>		
		<shut, sth>	*flip open tooth paste cap*
		<open, sth>	*flip close tooth paste cap*

Table 7. Collected *Affordances* and Designed *Primitive Tasks*. The first column records the manipulation scenarios. The second column lists collected affordances of object instances and parts. The affordances of object parts are indented below their parent instance-level affordance. The third column lists the instantiations of object affordances. These instantiations are bound to certain object attributes, *e.g.* <screw, sth> is bound to actual screws on the bottle's opening and cap. The fourth column lists the designed *Primitives* corresponding to the affordances. Some *Primitives* are set to gray for these *Primitives* are difficult to demonstrate and capture in individual. Demonstrations of these tasks are embedded within *Complex Task* demonstrations.

Scenario	*Complex Task*
kitchen table	heat with microwave oven; weigh with scale; scoop and pour; scoop and grip; scoop and wipe; scoop and scrape; pour and stir; grip and pour; pour and arrange; weigh with scale and pour; pour and scrape; grip and arrange; weigh with scale and grip; grip and wipe; pour and grip; clean the kitchen table; prepare a cup of hot sweet drink; prepare a bowl of hot soup with salt; prepare a cup of hot sweet fruit tea; prepare a chilled apple platter; prepare a savory fruit salad; prepare a baked sweet donut with sauce; prepare a baked sweet donut with apple slices and jam; prepare a savory baked sweet donut; prepare a cheese-baked sweet donut with tomato sauce; prepare savory baked apple slices with cheese; prepare a chilled fruit platter; make a baked sandwich with a filling of donut and salt; make a sandwich with a filling of tomato sauce and sugar; make a sandwich with a filling of apple slices and donut, adding tomato sauce, mustard sauce, salt, and sugar; make a baked sandwich with a filling of cheese and donut, adding tomato sauce; scoop, unscrew, pour, and screw; grip and scoop; scoop and scoop; scoop and arrange; weigh with scale and scoop; cut and scoop; cut and pour; grip and stir; grip and scrape; cut and grip; grip and assemble; stir and arrange; stir and scrape; scrape and arrange; weigh with scale and assemble; unscrew and pour; pour and screw; uncap and scrape; scrape and cap; uncap, scrape, and cap; scrape and assemble; assemble and arrange; unscrew and heat with microwave; pour and heat with microwave; heat with microwave and pour; heat with microwave and stir; heat with microwave and assemble; cut and heat with microwave; uncap, pour, and cap; prepare a cup of chilled green tea; prepare a cup of apple green tea; prepare a cup of mixed flavor fruit juice; prepare a cup of savory fruit juice milk tea; prepare a cup of pear milk tea with fruit jam; prepare a cup of savory honey fruit juice milk tea; prepare a cup of savory strawberry orange juice mixed with milk; uncap and scoop; prepare a cup of wine; prepare a cup of milk tea; prepare a cup of chilled fruit tea; prepare a cup of honey coffee; prepare a cup of chilled juice milk with jam; prepare a cup of chilled sweet milk tea;

Scenario	*Complex Task*
study room table	put into box; take out of box; put into drawer; take out of drawer; ready the laptop on the desktop for work; tidy up the desktop with the laptop after work; ready the laptop on the desktop for entertainment; illuminate the desktop; sharpen the pencil and write; tidy up the desktop with the laptop after entertainment; tidy up the desktop after paper-cutting; write and bind the paper; design and cut out rectangle shape on the paper; press button and open laptop; press button and close laptop; press button and put into box; press button and take out of box; press button and remove power plug; insert usb and plug in power plug; tidy up the desktop after writing; plug in power plug and press button; design and cut out flower shape on the paper; design and draw on the paper; ready the laptop and the lamp on the desktop for work; design, write and bind the paper; ready the desktop for drawing; take out of drawer, insert usb, and open laptop; put into drawer and put into box; put into drawer, put into box, and close laptop; remove usb, close laptop, and put into drawer; tidy up the desktop after drawing; cut and bind paper; ready the laptop and the lamp on the desktop for entertainment; design, draw and cut out flower shape on the paper; design, draw and bind the paper; tidy up the desktop after binding paper;
demo chem lab	transfer and heat liquid in beaker; transfer and heat liquid in conical flask; heat liquid in beaker and transfer liquid; heat liquid in conical flask and transfer liquid; transfer and heat liquid in beaker and transfer liquid out; heat liquid in test tube and transfer liquid; prepare solution through heating; mix liquid; pour in lab and shake in lab; pour in lab and pour in lab; pour in lab and heat test tube; pour in lab and light lamp; stir in lab and pour in lab; stir in lab and heat beaker; shake in lab and pour in lab; shake in lab and heat test tube; shake in lab and heat beaker; heat beaker and put off lamp; heat test tube and put off lamp; light lamp and put off lamp; put off lamp and pour in lab; put off lamp and stir in lab; put off lamp and shake in lab; heat beaker and stir in lab; stack mesh and heat beaker; light lamp and heat beaker; light lamp and heat test tube; pour in lab, shake in lab, and heat test tube; stir in lab, pour in lab, and shake in lab; light lamp, heat beaker, and put off lamp; light lamp, heat test tube, and put off lamp; stack mesh, light lamp, and heat beaker; light lamp, heat beaker, and stir in lab; light lamp, heat test tube, and shake in lab; pour in lab, pour in lab, and pour in lab; pour in lab, shake in lab, and pour in lab; stir in lab, stack mesh, and heat beaker; shake in lab, pour in lab, and heat test tube; shake in lab, heat test tube, and pour in lab; heat beaker, stir in lab, and pour in lab; heat beaker, put off lamp, and pour in lab; heat beaker, put off lamp, and stir in lab; heat test tube, put off lamp, and shake in lab; pour in lab, stir in lab, and pour in lab; pour in lab, pour in lab, pour in lab, pour in lab, and pour in lab; stir and transfer liquid; heat liquid in beaker; heat liquid in test tube; put off lamp, pour in lab, and shake in lab; put off lamp, stir in lab, and pour in lab; prepare solution in beaker;
bathroom table	squeeze tooth paste tube to tooth brush; squeeze tooth paste and stack tooth brush; prepare for teeth brushing.

Table 8. Recorded *Complex Tasks*. We list the names of the recorded *Complex Tasks* here.

F.2. Visualization

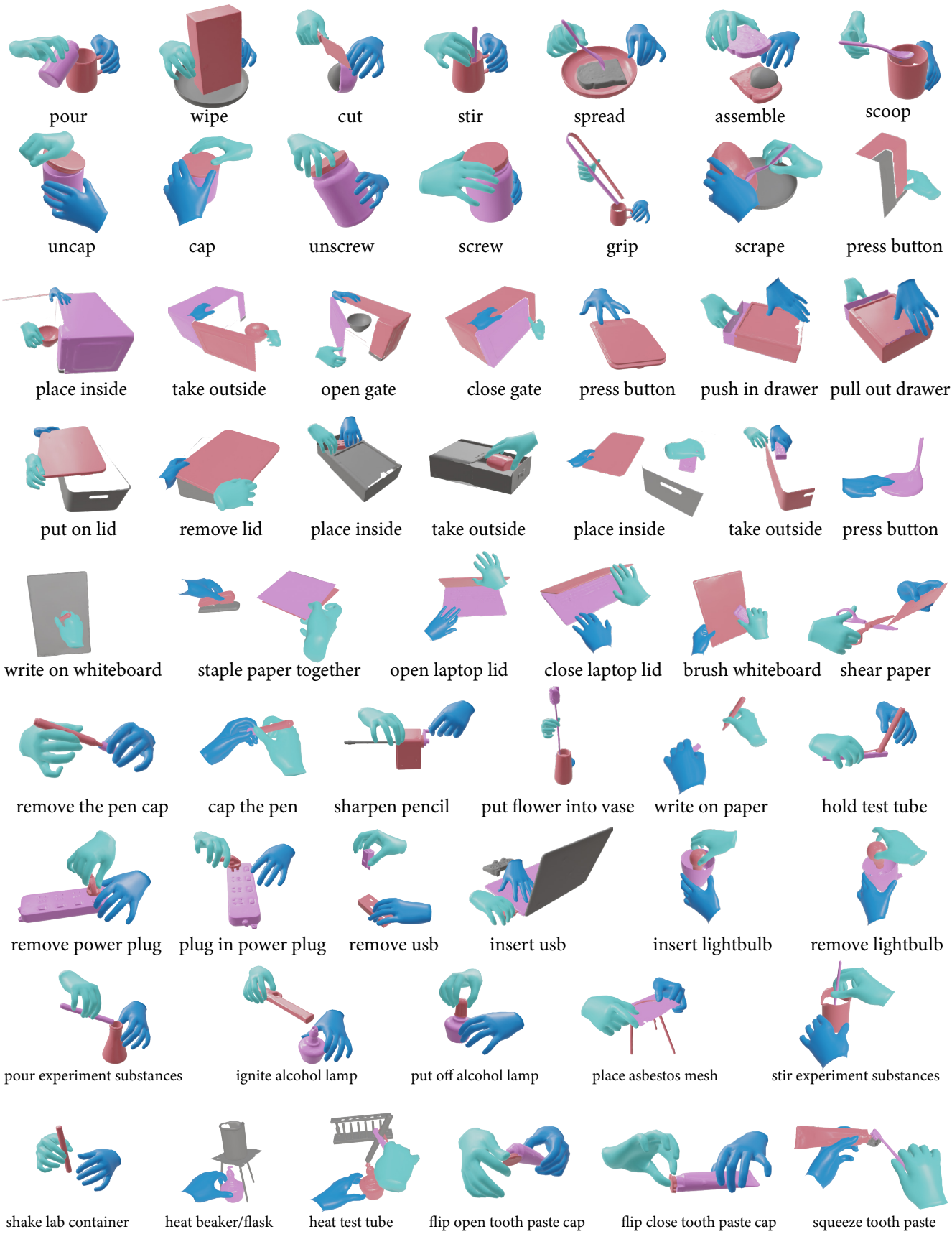


Figure 12. ***Primitives*** **visualization.**

Figure 13. **Object visualization.**

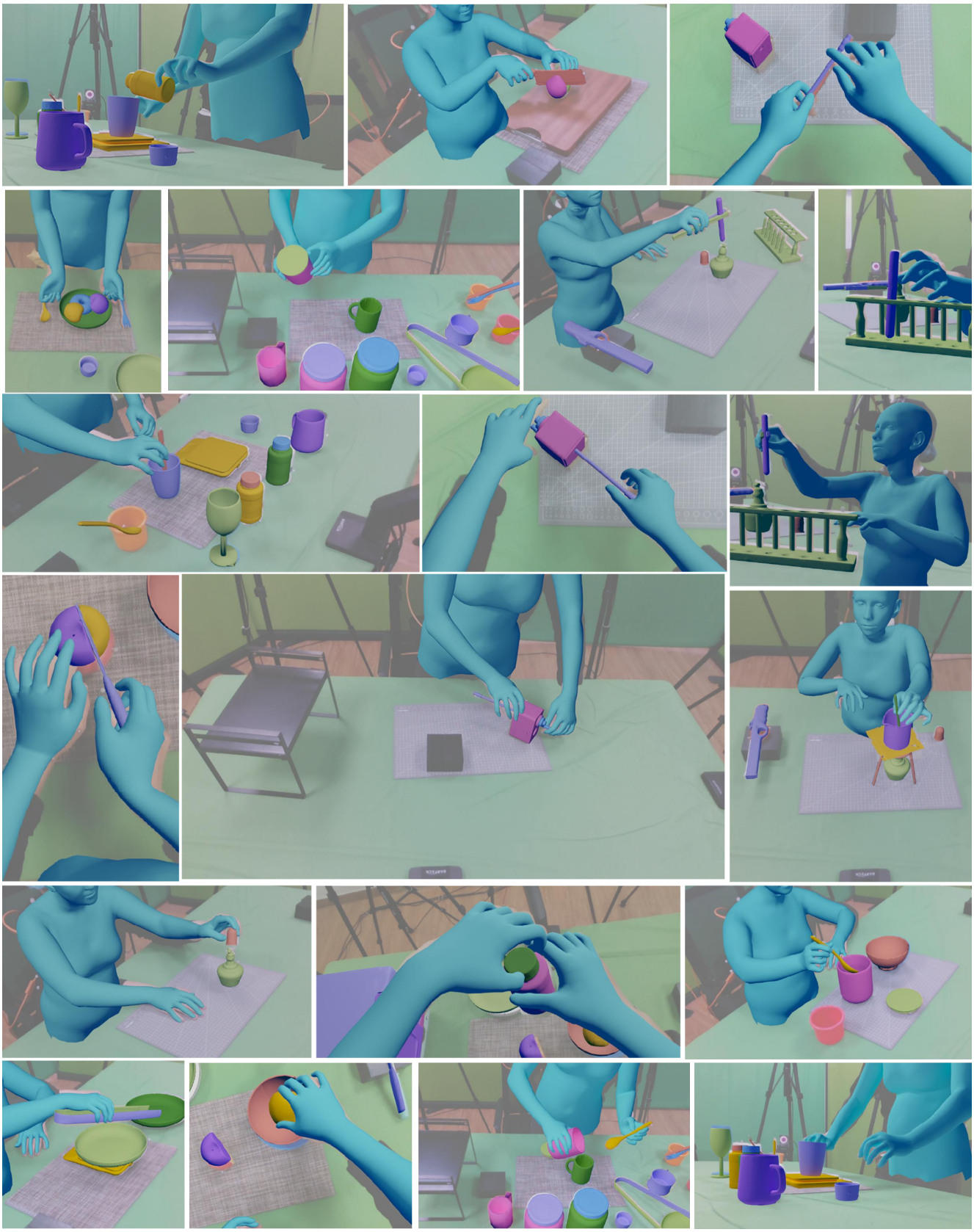


Figure 14. **Dataset visualization.** Human bodies and objects within the scene are rendered onto the captured raw images for visualization.

```
You are a python programming expert and you are asked to finish a certain bimanual robotics task.

Scene Description:
{scene_desc}

Task Description:
{task_desc}

The Structure of the code is a ECS architecture defined as
ECS File

The entities are defined as
Entity File

The components are defined as
Component File

The systems are defined as
System File

The task is to finish the methods called "exec_task" in this class {read_from_file(scene_path)}

You need to query the raw 3d object from the scene which contains object name as keys in scene description and you can use them to query different type of information from the scene. Build them into Objects and instantiate PrimitiveTasks to finish the job.
Leave the objects not mentioned in the task description as they are.
Avoid using any methods with underscore prefix. Explicitly specify the keyword arguments instead of using **kwargs.
For example:

an_object = Object('object_name',
    geometry=self.query_geometry_info('object_name'),
    ...
    )

a_primitive_task = PrimitiveTask(an_object.affordance.get_primitive_task_info('primitive_task_name'))

a_primitive_task.execute(src_object=an_object, tgt_object=another_object, trajectory=oracle.generate(a_primitive_task)
    )

Explanation of the code is unnecessary. Putting everything in method implementation would be admired. Respond with exec_task(self) itself.
```

Listing 1. Prompt Template. **Entity** marks object instances present in the scene. **Component** accommodates information of objects' initial status and affordances. **System** accommodates the interface of motion generators for *Primitives*. The example code incorporated in the prompt demonstrates the interface of **oracle queries** for object motion trajectories.

```
# Task
# The task is to cut the pear into pieces which to be put in the pan and add some season.

# Scene
# In a kitchen, on a table, these objects are placed: [knife, pear, bottle, pan]
def exec_task(self):
    # making an object for knife
    knife = Object('knife',
        geometry=self.query_geometry_info('knife'),
        state=self.query_state_info('knife'),
        affordance=self.query_affordance_info('knife')
    )

    # making an object for pear
    pear = Object('pear',
        geometry=self.query_geometry_info('pear'),
        state=self.query_state_info('pear'),
        affordance=self.query_affordance_info('pear')
    )

    # making an object for pan
    pan = Object('pan',
        geometry=self.query_geometry_info('pan'),
        state=self.query_state_info('pan'),
        affordance=self.query_affordance_info('pan')
    )

    # making an object for bottle
    bottle = Object('bottle',
        geometry=self.query_geometry_info('bottle'),
        state=self.query_state_info('bottle'),
        affordance=self.query_affordance_info('bottle')
    )

    # creating primitive task for cutting
    cut_pear = PrimitiveTask(knife.affordance.get_primitive_task_info('cut'))
    cut_pear.execute(src_object=knife, tgt_object=pear, trajectory=oracle.generate(cut_pear))

    # creating primitive task for placing pear inside the pan
    put_task = PrimitiveTask(pear.affordance.get_primitive_task_info('put_inside'))
    put_task.execute(src_object=pear, tgt_object=pan)

    # create primitive task for adding season
    add_season = PrimitiveTask(bottle.affordance.get_primitive_task_info('pour'))
    add_season.execute(src_object=bottle, tgt_object=pan, trajectory=oracle.generate(add_season))
```

Listing 2. Primitive Task Planning Success Example.

```
# Task
# The task is to light the alcohol lamp to heat the compound in the beaker. The lamp is capped at the beginning. Put the beaker on the support with mesh when heating

# Scene
# In a chemistry laboratory, on a table, these objects are placed: [lighter, alcohol_lamp, beaker, heat_support, mesh]
def exec_task(self):
    # Uncap
    alcohol_lamp = Object('alcohol_lamp',
        geometry=self.query_geometry_info('alcohol_lamp'),
        affordance=self.query_affordance_info('alcohol_lamp'),
        state=self.query_state_info('alcohol_lamp')
    )
    uncap_alcohol_lamp_task = PrimitiveTask(alcohol_lamp.affordance.get_primitive_task_info('uncap'))
    uncap_alcohol_lamp_task.execute(src_object=alcohol_lamp, trajectory=oracle.generate(uncap_alcohol_lamp_task))

    # Light
    lighter = Object('lighter',
        geometry=self.query_geometry_info('lighter'),
        affordance=self.query_affordance_info('lighter'),
        state=self.query_state_info('lighter')
    )
    light_alcohol_lamp_task = PrimitiveTask(lighter.affordance.get_primitive_task_info('light_lamp'))
    light_alcohol_lamp_task.execute(src_object=lighter, tgt_object=alcohol_lamp, trajectory=oracle.generate(light_alcohol_lamp_task))

    # Arrange (put on the mesh support)
    beaker = Object('beaker',
         geometry=self.query_geometry_info('beaker'),
         affordance=self.query_affordance_info('beaker'),
         state=self.query_state_info('beaker')
    )
    heat_support = Object('heat_support',
        geometry=self.query_geometry_info('heat_support'),
        affordance=self.query_affordance_info('heat_support'),
        state=self.query_state_info('heat_support')
    )
    mesh = Object('mesh',
         geometry=self.query_geometry_info('mesh'),
         affordance=self.query_affordance_info('mesh'),
         state=self.query_state_info('mesh')
    )
    put_beaker_on_mesh_task = PrimitiveTask(beaker.affordance.get_primitive_task_info('arrange'))
    put_beaker_on_mesh_task.execute(src_object=beaker, tgt_object=heat_support, trajectory=oracle.generate(put_beaker_on_mesh_task))

    # Stack mesh to heat
    stack_mesh_task = PrimitiveTask(mesh.affordance.get_primitive_task_info('stack_mesh'))
    stack_mesh_task.execute(src_object=mesh, tgt_object=beaker, trajectory=oracle.generate(stack_mesh_task))

    # Heat
    heat_beaker = PrimitiveTask(alcohol_lamp.affordance.get_primitive_task_info('heat_beaker'))
    heat_beaker.execute(src_object=alcohol_lamp, tgt_object=beaker, trajectory=oracle.generate(heat_beaker))
```

Listing 3. Primitive Task Planning Fail Example. One violation of dependency occurs (line 41): the extra `place onto` primitive erroneously positions the beaker upon the support prior to the asbestos mesh's placement on the support. This blocks the correct execution path that requires placing the asbestos mesh before heating the beaker.