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

Supplementary Materials

Table of Contents

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. 1. The two systems are time synchronized with software synchronization tools bundled in the ROS2 [18].

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.

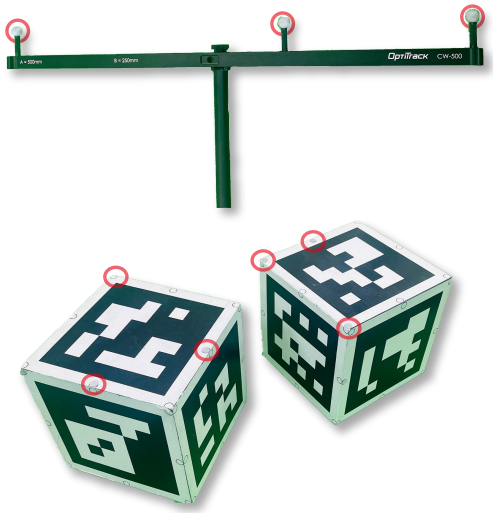

Figure 1. **Illustration of Platform Calibration Tools.** The top is the vender-provided calibration wand for the MoCap system. The bottom are the ArUco cubes attached with reflective markers (circled in red). The ArUco patterns are for unifying the camera in the multi-camera system, while the surface-attached reflective markers are used for unifying the cameras with the MoCap system.

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 [23]. 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.

A.3. Human Pose and Surface

We employ a two-stage fitting approach inspired by the application of the MoSH++ algorithm in [5, 19, 25] for the SMPL-X [24] 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).

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

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).

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

$E_{\text{prior(b)}}\big(\theta\big)$, $E_{\text{plau(h)}}\big(\theta\big)$, 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 [19]. $E_{\text{plau(h)}}\big(\theta\big)$ is an implementation of anatomy loss in [28] 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.

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

$E_{\text{plau(h)}}\big(\theta_t\big)$, $E_{\text{prior(b)}}\big(\theta_t\big)$, 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 [12] 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

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 [4]	✓	∼	20M†	1	37	–	✗	–	–	✗	✗	✗	✗	✓	✓
Ego4D [7]	✓	∼‡	∼†	1	931	–	✗	–	–	✗	✗	✗	✗	✓	✓
HA-ViD [30]	✓	1280 × 720	1.5M	3	30	40	✗	–	–	✗	✗	✗	✗	✓	✓
FPHAB [6]	✓	1920 × 1080	105K	1	6	4	✓	real	mocap	✓	✓	✓	✓	✗	✗
ObMan [10]	✓	256 × 256	154K	1	20	3K	✓	syn	simulate	✓	✓	✗	✗	✗	✗
YCBAfford [3]	✓	–	133K	1	1	21	✓	syn	manual	✓	✗	✗	✗	✗	✗
HO3D [9]	✓	640 × 480	78K	1-5	10	10	✓	real	auto	✓	✓	✗	✓	✗	✗
ContactPose [1]	✓	960 × 540	2.99M	3	50	25	✓	real	auto	✓	✓	✓	✗	✗	✗
GRAB [25]	✗	–	1.62M	–	10	51	✓	real	mocap	✓	✓	✓	✓	✗	✗
DexYCB [2]	✓	640 × 480	582K	8	10	20	✓	real	crowd	✓	✓	✗	✓	✗	✗
H2O [13]	✓	1280 × 720	571K	5	4	8	✓	real	auto	✓	✓	✓	✓	✗	✗
HOI4D [16]	✓	1280 × 800	3M	1	9	1000	✓	real	crowd	✓	✓	✓	✓	✗	✗
ARCTIC [5]	✓	2800 × 2000	2.1M	9	10	11	✓	real	mocap	✓	✓	✓	✓	✗	✗
ContactArt [31]	✓	–	332K	–	–	80	✓	real	transfer	✓	✓	✗	✓	✗	✗
AssemblyHands [21]	✓	1920 × 1080	3.03M	12	34	–	✓	real	semi-auto	✓	✗	✓	✓	✓	✓
AffordPose [11]	✗	–	–	–	–	641	✓	syn	manual	✓	✓	✓	✗	✗	✗
TACO [17]	✓	4096 × 3000‡	5.2M	13	14	196	✓	real	auto	✓	✓	✓	✓	✗	✗
Ego-Exo4D [8]	✓	∼‡	∼†	5-6	839	–	✓	real	semi-auto	✓	✗	✗	✓	✓	✓
OakInk-Image [29]	✓	848 × 480	230K	4	12	100	✓	real	crowd	✓	✓	✓	✓	✗	✗
OakInk-Shape [29]	✗	–	–	–	–	1700	✓	real	transfer	✓	✓	✓	✗	✗	✗
OAKINK2	✓	848 × 480	**4.01M**	4	9	75	✓	real	mocap	✓	✓	✓	✓	✓	✓

Table 1. **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! [20]; "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.

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 $\leq 5\,\mathrm{mm}$) and *lifted* (height displacement to the initial state $\geq 5\,\mathrm{mm}$).

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.

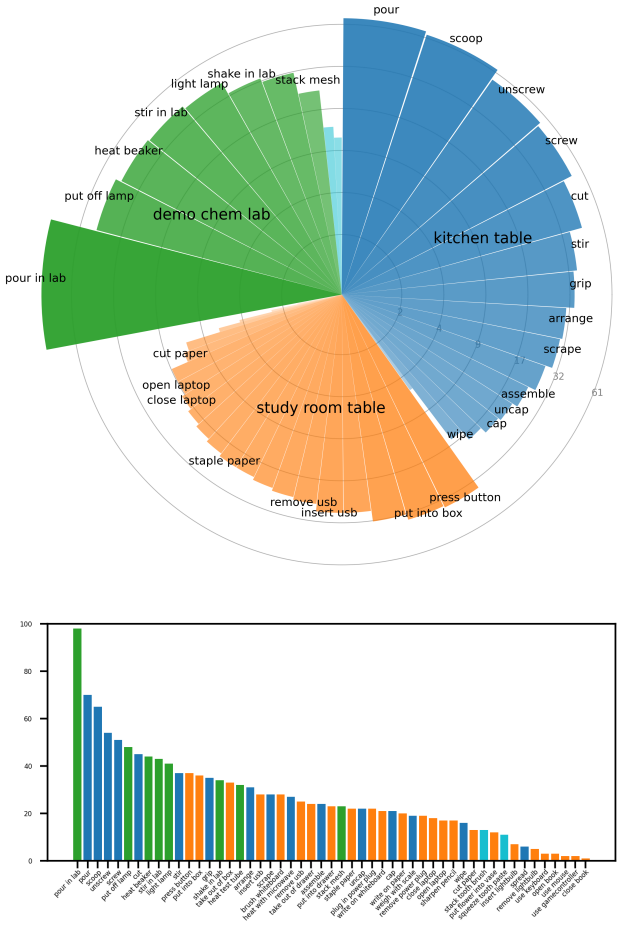

Figure 2. **Distribution of *Primitive Task* demonstrations.** The sub-figure above displays the proportion of *Primitive Task* demonstrations across various scenarios within the entire OAKINK2 dataset, with frequently occurring *Primitive Tasks* highlighted. The sub-figure below presents a list of *Primitive 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. 4 and Tab. 5.

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 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 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 [15] separately on three different training schemes: FreiHAND [32] only, OAKINK2-H-SV only, and a mixture of these two sets. We evaluate MPJPE and MPVPE after Procrustes analysis on OakInk-image (SP2) [29], and the results shown in Tab. 2 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 2. **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. 3). 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. 6.

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 3. **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 [14, 27]. 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 [22] 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 suc-

cess 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 in-betweening as motion generator for Complex Task Completion. We adopt GNet and MNet in GOAL [26] and INet in FAVOR [14]. 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. 3. The left-hand images of Fig. 3 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.

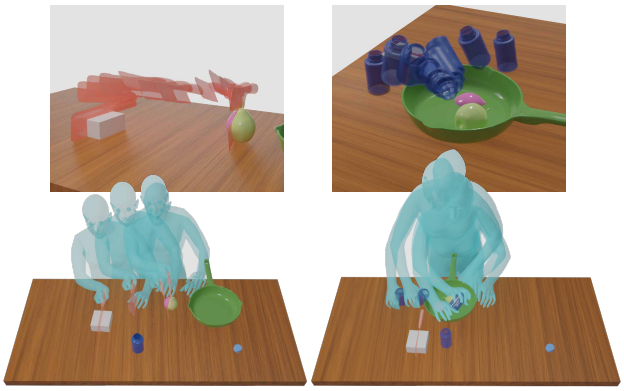

Figure 3. **Oracle Trajectories and Motion Generation** This figure illustrates the successful *Complex Task* completion of two Primitive Tasks. The top pair of images depict the oracle trajectories, while the bottom pair represents the sequential motion generated.

F. Dataset Inspection

F.1. Task List

Table 4. 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 4. 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 4. 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 5. Recorded *Complex Tasks*. We list the names of the recorded *Complex Tasks* here.

F.2. Visualization

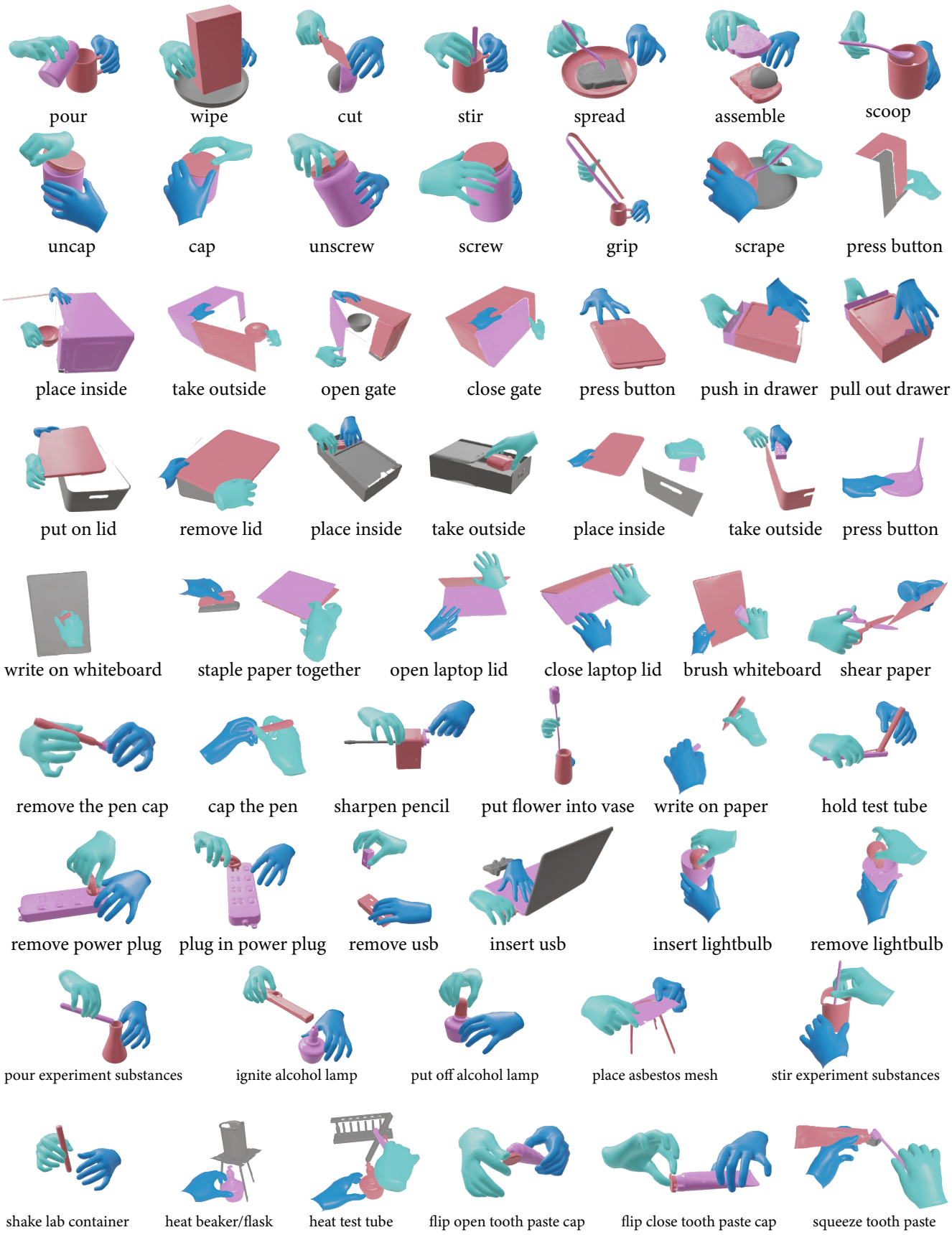

Figure 4. ***Primitives* visualization.**

Figure 5. **Object visualization.**

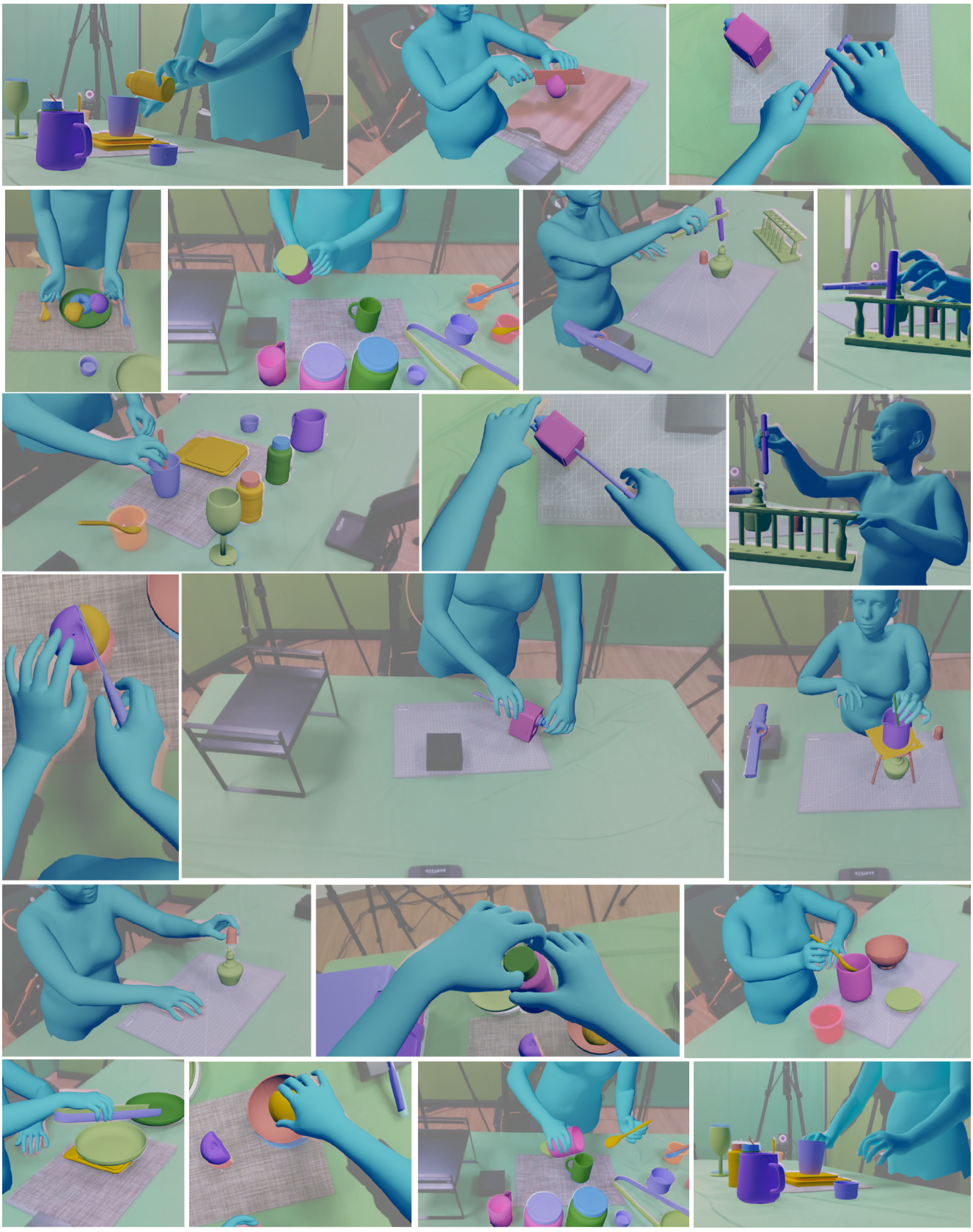

Figure 6. **Dataset visualization.** Human bodies and objects within the scene are rendered onto the captured raw images for visualization.

```
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.

```python
# 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.

References

[1] 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. 3

[2] 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. 3

[3] 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. 3

[4] 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. 3

[5] 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] 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. 3

[7] 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. 3

[8] 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. 3

[9] 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. 3

[10] 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. 3

[11] 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. 3

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[13] 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. 3

[14] 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. 5, 6

[15] 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. 4

[16] 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. 3

[17] 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

[18] 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. 1

[19] 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. 2

[20] Andrew T Miller and Peter K Allen. Graspit!: A versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122, 2004. 3

[21] 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. 3

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

[23] OptiTrack. Motive: Optical motion capture software. https://optitrack.com/software/motive/. 1

[24] 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. 2

[25] 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

[26] 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. 6

[27] 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. 5

[28] 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. 2

[29] 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. 3, 4

[30] 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. 3

[31] 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. 3

[32] 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. 4