A Implementation Details

A.1 Human Reconstruction From Videos

Method. For the 3D human reconstruction, we start by tracking the person in the video and getting an initial estimate of their 3D body pose using 4D Humans [54]. This body reconstruction cannot capture the hand pose details (i.e., the hands are flat). Therefore, for each detection of the person in the video, we detect the two hands using ViTPose [55], and for each hand, we apply HaMeR [56] to get an estimate of the 3D hand pose. However, the hands reconstructed by HaMeR can be inconsistent with the arms from the body reconstruction (e.g., different wrist orientation and location). To address this, we apply an optimization refinement to make the body and the hands consistent in each frame, and encourage that the holistic body and hands motion is smooth over time. This optimization is similar to SLAHMR [57], with the difference that besides the body pose and location of the SMPL+H model [58], we also optimize the hand poses. We initialize the procedure using the 3D body pose estimate from 4D Humans and the 3D hand poses from HaMeR. Moreover, we use the 2D projection of the 3D hands predicted by HaMeR to constrain the projection of the 3D hand keypoints of the holistic model using a reprojection loss. Finally, we can jointly optimize all the parameters (body location, body pose, hand poses) over the duration of the video, as described in SLAHMR [57].

Inference Requirements. The model of human reconstruction we use is large and needs to be run on a computer with sufficiently good computation speed. Here we provide details about the runtime performance of the human reconstruction model. We use a desktop that comes with a GPU RTX3090 that has the size of the memory 24 GB. For a 10 seconds video with fps 30, it processes 10 minutes.

A.2 Prompts of Using GPT4V

In order to use GPT4V in OKAMI, we need GPT4V's output to be in a typed format so that the rest of the programs can parse the result. Moreover, in order for the prompts to be general across a diverse set of tasks, our prompt does not leak any task information to the model. Here we describe the three different prompts in OKAMI for using GPT4V.

Identify Task-relevant Objects. OKAMI uses the following prompt to invoke GPT4V so that it can identify the task-relevant objects from a provided human video:

Prompt: You need to analyze what the human is doing in the images, then tell me: 1. All the objects in front scene (mostly on the table). You should ignore the background objects. 2. The objects of interest. They should be a subset of your answer to the first question. They are likely the objects manipulated by human or near human. Note that there are irrelevant objects in the scene, such as objects that does not move at all. You should ignore the irelevant objects.

Your output format is:

```
The human is xxx.
All objects are xxx.
The objects of interest are:
```
json
{
    "objects": ["OBJECT1", "OBJECT2", ...],
}
```
```

Ensure the response can be parsed by Python 'json.loads', e.g.: no trailing commas, no single quotes, etc. You should output the names of objects of interest in a list ["OBJECT1", "OBJECT2", ...] that can be easily parsed by Python. The name is a string, e.g., "apple", "pen", "keyboard", etc.

Identify Target Objects. OKAMI uses the following prompt to identify the target object of each step in the reference plan:

Prompt: The following images shows a manipulation motion, where the human is manipulating an object.

Your task is to determine which object is being manipulated in the images below. You need to choose from the following objects: {a list of task-relevant objects}.

Tips: the manipulated object is the object that the human is interacting with, such as picking up, moving, or pressing, and it is in contact with the human's {the major moving arm in this step} hand.

Your output format is:

~~~
```json
{
    "manipulate_object_name": "MANIPULATE_OBJECT_NAME",
}
```
~~~

Ensure the response can be parsed by Python 'json.loads', e.g.: no trailing commas, no single quotes, etc.

Identify Reference Objects. Here is the prompt that asks GPT4V to identify the reference object of each step in the reference plan:

Prompt: The following images shows a manipulation motion, where the human is manipulating the object {manipulate_object_name}.

Please identify the reference object in the image below, which could be an object on which to place {manipulate_object_name}, or an object that {manipulate_object_name} is interacting with. Note that there may not necessarily have an reference object, as sometimes human may just playing with the object itself, like throwing it, or spinning it around. You need to first identify whether there is a reference object. If so, you need to output the reference object's name chosen from the following objects: {a list of task-relevant objects}.

Your output format is:

~~~
```json
{
    "reference_object_name": "REFERENCE_OBJECT_NAME" or "None",
}
```
~~~

Ensure the response can be parsed by Python 'json.loads', e.g.: no trailing commas, no single quotes, etc.

A.3 Details on Factorized Process for Retargeting

Body Motion Retarget. To retarget body motions from the SMPL-H representation to the humanoid, we extract the shoulder, elbow, and wrist poses from the SMPL-H models. We then use inverse kinematics to solve the body joints on the humanoid, ensuring they produce similar shoulder and elbow orientations and similar wrist poses. The inverse kinematics is implemented using an open-sourced library Pink [59]. The IK weights we use for shoulder orientation, elbow orientation, wrist orientation, and wrist position are 0.04, 0.04, 0.1, and 1.0, respectively.

Hand Pose Mapping. As we describe in the method section, we first retarget the hands from SMPL-H models to the humanoid's dexterous hands using a hybrid implementation of inverse kine-

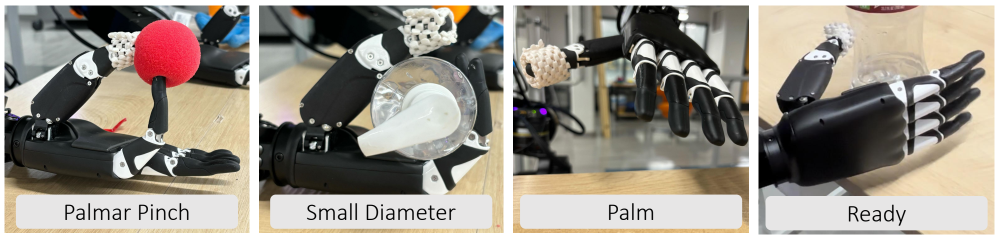

Figure 5: Visualization of the predefined hand poses. Palmar pinch, small diameter, and palm can be used as grasping poses, and palm and ready can be used as free-motion poses.

matics and angle mapping. In practice, the details of finger motions generate jerky motions of dexterous hands that impede the performance of manipulation. In order to have the humanoid interact with objects robustly, we filter the noisy motion by mapping the hand poses into a fixed set of hand poses [60], where a diverse set of hand poses are defined for dexterous hands. In this project, we implement four hand poses, shown in Figure 5. At each step of the reference plan, we map the estimated finger angles to one of the four hand poses we implement: two grasping poses selected from a grasp taxonomy [60], a stretched hand pose for non-prehensile interaction or articulated object interaction, and a preparatory grasping pose. The first three hand poses are used when the hand is interacting with objects. For free-motion reaching, we typically choose one of the last two hand poses depending on which hand pose the next step of the plan requires.

Given that the result of the hand reconstruction model is noisy, it is hard to directly decide if there is contact between the hand and the objects based on their geometric features. To reliably detect the contact relations between the hand and the object, we use an off-the-shelf model, Hand Object Detector [61], to determine whether the hand is in contact with an object. The classified result is then used to map the hand to either a grasping pose or a free-motion pose.

B Additional Experimental Details

B.1 Success Conditions

We describe the success conditions we use to evaluate if a task rollout is successful or not.

- `Sprinkle-salt`: The salt bottle reaches a position where the salt is poured out into the bowl.
- `Plush-toy-in-basket`: The plush toy is put inside the container, with more than 50% of the toy inside the container.
- `Close-the-laptop`: The display is lowered towards the base until the two parts meet at the hinge (aka the laptop is closed).
- `Close-the-drawer`: The drawer is pushed back to the containing region, either it's a drawer or a layer of a cabinet.
- `Place-snacks-on-plate`: The snack is placed on top of the plate, with more than 50% of the snack package on the plate.

B.2 Implementation of Baseline

We implement the baseline ORION [4] with minimal modifications to apply it to our humanoid setting. First, we estimate the palm trajectory from SMPL-H trajectories by using the center point of the reconstructed fingers as the palm position at each time step. Next, we warp the palm trajectory based on the test-time objects' locations. Finally, we use inverse kinematics to solve for the robot's body joints, with the warped trajectory serving as the target palm position.

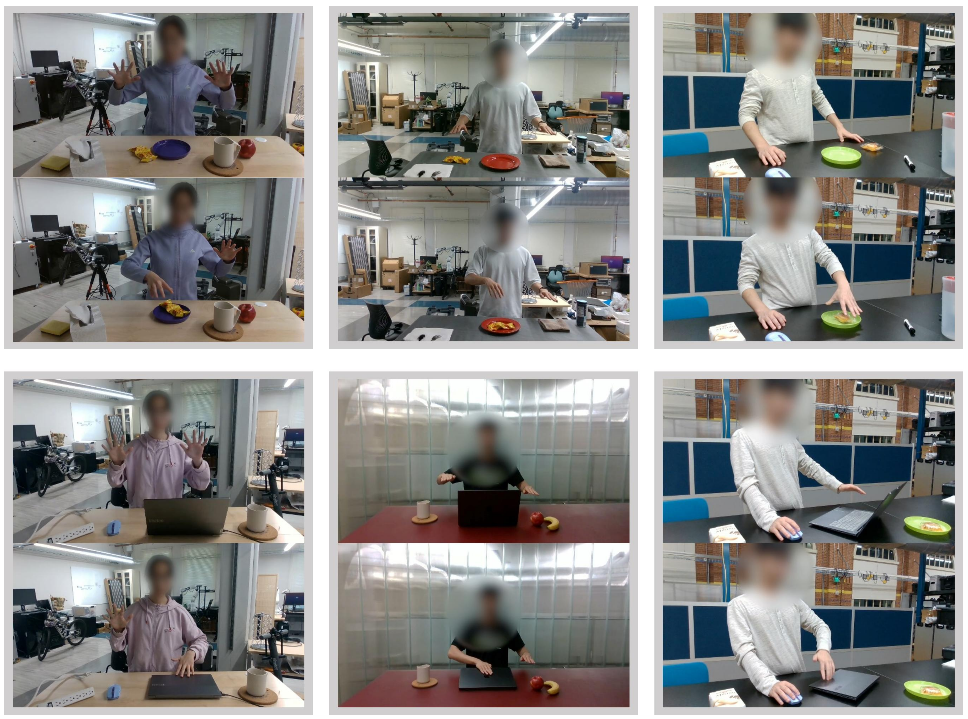
Figure 6: The initial and end frames of videos performed by different human demonstrators. The first row is `Place-snacks-on-plate` task, and the second row is `Close-the-laptop` task.

B.3 Details on Different Demonstrators

Figure 6 shows the videos of three different human demonstrators performing `Place-snacks-on-plate` and `Close-the-laptop` tasks. We calculate the success rates of imitating different videos, and the results are shown in Figure 4(b).