Human-Object Interaction from Human-Level Instructions

Supplementary Material

Contents

A. Details of High-Level Planner

A.1. Pseudo-code for Object Positions Calculation

In Algorithm 1, we provide the pseudo-code for the algorithm that calculates object positions in the scene graph, as described in Sec. 4.1.

ALGORITHM 1: Object Positions Calculation

Input : Scene graph $G(V, E)$ with vertices V and edges E, initial positions L, static nodes (V_S), the nodes to be moved (V_M).
Output : Updated positions L for all nodes.
Procedure `ComputePositions(`G, L`)`**:**
  Let the set of processed nodes $V_P \leftarrow V_S$
  while V_M *is not empty* **do**
    Let $V' \subseteq V_M$ be the nodes whose predecessors' positions are known
    foreach $v \in V'$ **do**
      $L(v) \leftarrow$ `Update(`v, G, L`)`
      $V_P \leftarrow V_P \cup \{v\}$
      $V_M \leftarrow V_M \setminus \{v\}$
    end
  end
  return L
return
Procedure `Update(`v, G, L`)`**:**
  Initialize an empty list of positions P
  foreach *predecessor* u *of* v *in* G **do**
    $O \leftarrow$ `ComputeOffset(`(u, v), $L(u)$`)`
    Append $L(u) + O$ to P
  end
  return Average(P)
return
Procedure `ComputeOffset(`(u, v), $L(u)$`)`**:**
  if *edge* (u, v) *indicates "on"* **then**
    Set O_{height} to align with the top surface of u
    Sample $O_{\text{horizontal}}$ within the horizontal bounds of u
    return $[O_{\text{horizontal}}, O_{\text{height}}]$
  end
  else if *edge* (u, v) *indicates "adjacent"* **then**
    Use the direction and distance provided by the LLMs to compute O
    return O
  end
return

A.2. Prompts for High-Level Planner

We provide the prompts used for generating the scene map and execution plan in Fig. S1. An example of the input and the corresponding output of the LLM planner is shown in Fig. S2. We also present the prompt used by the baseline (as described in Sec. 7.1) in Fig. S3.

B. Details of Low-Level Motion Generator

B.1. Diffusion Model Architecture

Our system includes four diffusion models: CoarseNet, RefineNet, FingerNet, and the navigation module, all of which adopt a similar transformer-based architecture. Here, we illustrate the architecture of RefineNet in Fig. S4. The model is provided with a motion sequence $\boldsymbol{x}_n$, the condition $\boldsymbol{c}_r$, and the noise step n, and predicts the clean motion $\hat{\boldsymbol{x}}_0$. For the other networks, we simply substitute the condition with the corresponding one.

B.2. Implementation Details of Grasp Pose Generation

As introduced in Sec. 5.2.2, we use a state-of-the-art grasp pose generation method [12] to generate the grasp pose.

Scene Map Generation Prompt: I will provide a scene description outlining the current layout of a room, along with instructions on how it should be rearranged. Your task is to create a detailed target layout based on these descriptions. When describing the target layout, please:
1. Identify the objects involved in the rearrangement, including those that need to be moved and those necessary to define the spatial relationships.
2. Specify the precise spatial relationships between objects in the target layout.
[start of rules]
1. When describing spatial relationships, use only the following relationships:
a. on: Signifies that object A is on top of object B.
b. adjacent: Indicates that object A is near object B. You may also describe the direction between them, using terms like east, west, south, and north. And you can also specify the distance between them.
c. facing: Denotes that object A is facing object B. When the direction is not important or can be arbitrary, say "with no specific facing direction".
2. List all relevant position and orientation relationships for all objects involved.
3. Avoid describing spatial relationships between objects that aren't involved in this rearrangement process.
[end of rules]

Execution Plan Generation Prompt: Determine the most natural order to move each object to achieve the target layout. Give a reason for your choice.
[start of rules]
1. You can move one object at a time.
2. You shouldn't move an object more than once.
3. You cannot move an object if there is currently another object on top of it.
[end of rules]

Figure S1. Prompts used by the LLM planner to generate the scene map and execution plan.

Input:
Scene description: The room contains static objects: [table], and movable objects: [chair, monitor].
The table has space only on its south side for placing items, as the other directions are obstructed.
Below is the information about the objects, with all center-of-mass values representing the height of the object when it is placed on the ground:
- The height of the table is 0.8 meters.
- The center of mass of the chair is at 0.5 meters.
- The center of mass of the monitor is at 0.3 meters.
The following are the positions of the objects in the initial layout:
- The table is at position (2.0, 1.0, 0.5).
The dimensions of the objects:
- The table has an XY bounding box of 1.2 meters in the X direction and 1.0 meter in the Y direction.
For orientation:
- North corresponds to the XY direction (1,0),
- South corresponds to (-1,0),
- East corresponds to (0,-1),
- West corresponds to (0,1).

Instruction: I want to set up my workspace.

Output:
Scene Map:
Monitor: {(2.0, 1.0, 0.8), south},
Chair: {(0.4, 1.0, 0.5), north}.

Execution Plan:
1. Lift the monitor, move the monitor, put down the monitor.
2. Lift the chair, move the chair, put down the chair.

Figure S2. Example of the input and output for the LLM planner.

Although originally designed for robotic hand grasping, this method relies solely on kinematic chain information, making it fully compatible with the SMPL-X [7] hand model. Starting with an object mesh and an initial hand pose, the grasp generation process is formulated as an optimization problem that minimizes a differentiable force closure energy term along with several additional terms that encourage the fingers to conform closely to the object surface while maintaining a natural hand shape. For motions involving both hands, such as carrying a box, we omit the force closure term.

```
Scene Map Generation Prompt: I will provide a scene description outlining the current layout of a room,
along with instructions on how it should be rearranged. Your task is to create a detailed target layout
based on these descriptions. When describing the target layout, please:
1. Identify the objects that need to be moved.
2. Specify the precise 3D position and orientation for those objects in the target layout.
```

Figure S3. Prompts used by the baseline, as described in Section 7.1.

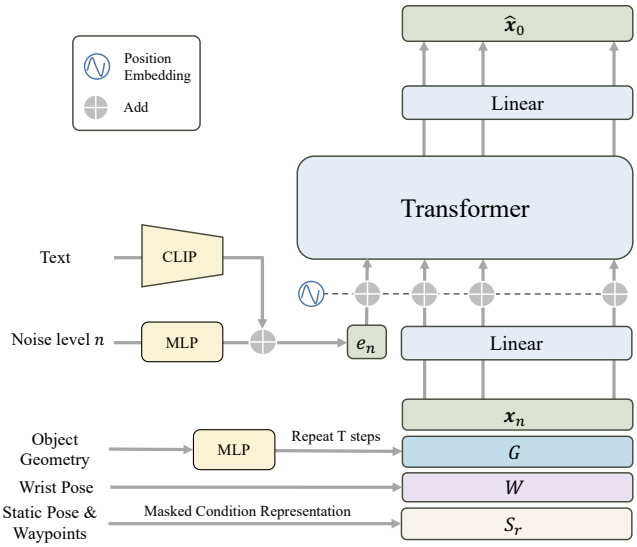


Figure S4. Architecture of RefineNet. The model is conditioned on $\boldsymbol{c}_r = \{\boldsymbol{W}, \boldsymbol{S}_r, \boldsymbol{G}, \boldsymbol{T}\}$, where $\boldsymbol{W}$ represents the wrist-object relative pose, $\boldsymbol{S}_r$ is the masked motion representation containing the static pose and waypoints, $\boldsymbol{G}$ denotes the object geometry, and $\boldsymbol{T}$ is the textual input.

B.3. Post-processing in RefineNet

As introduced in Sec. 5.2.3, we apply post-processing to RefineNet's output to better align it with the input condition. First, we replace the object poses in the pre-contact and post-contact phases with the corresponding static poses. To address the abrupt transitions at the phase boundaries, we identify the difference between the predicted object pose and the static object pose at the boundary frame denoted as $\Delta \boldsymbol{d} \in \mathbb{R}^6$ (suppose this is the boundary between pre-contact phase and contact phase), which encapsulates both the positional and rotational differences in axis-angle format. To smooth these transitions, we employ an interpolation strategy to the object motion in the contact phase. The interpolation function is defined by $\hat{\boldsymbol{O}}_t = \boldsymbol{O}_t + \alpha_t \Delta \boldsymbol{d}$, where α_t is a time-dependent scaling factor that decreases from 1 to 0 across the transition period, $\boldsymbol{O}_t$ denotes the predicted object pose.

Next, we use the rectified object motion and the optimized wrist pose $\hat{\boldsymbol{w}} = (\hat{\boldsymbol{R}}, \hat{\boldsymbol{T}})$ to compute the wrist trajectories during the contact phase. The rotation is computed as $\boldsymbol{R}_o \hat{\boldsymbol{R}}$, and the position is computed as $\boldsymbol{R}_o \hat{\boldsymbol{T}} + \boldsymbol{T}_o$, where $\boldsymbol{R}_o$ and $\boldsymbol{T}_o$ represent the object's rotation and position, respectively. The computed wrist trajectories may slightly deviate from the predicted wrist trajectories in the contact phase, leading to unsmooth transitions near the phase boundaries. We apply the same interpolation strategy to the wrist poses in the pre-contact and post-contact phases to produce smooth wrist motions. Finally, we employ an Inverse Kinematics (IK) algorithm to solve for the updated human pose constrained by the wrist poses.

B.4. Implementation Details of FingerNet

FingerNet generates finger motions for the pre-contact and post-contact phases to create natural transitions between rest and grasp poses. It is trained on the GRAB dataset [10]. Since FingerNet focuses solely on the hand approaching and releasing motions, we only use the corresponding portions of the dataset. Specifically, we extract one second of motion before the grasp begins and after it ends as training data, discarding the rest.

We adopt a mirroring strategy similar to [14] to handle both hands simultaneously. During training, left-hand data are mirrored and combined with the right-hand data so that the model only needs to predict motions for the right hand. At inference time, we mirror the left-hand input, predict the corresponding motions using the model, and then mirror the output back to obtain the left-hand motions.

B.5. Smooth Transition between Navigation and Interaction

The navigation and interaction modules alternate to generate a long-horizon interaction sequence. Specifically, when the human is far from the object, the navigation module is activated to guide the human towards it. Once the human is within a threshold distance (set at 1 meter), the system switches to the interaction module to begin object manipulation. After the interaction is completed and the object is put down, control reverts to the navigation module to move towards the next object.

Since both modules use the initial human pose as an input condition, we ensure a smooth transition by feeding the last pose from the previous module into the next. We apply the same smoothing technique described in Sec. B.3 at the switching boundary to further enhance continuity.

C. Implementation Details of Physics Tracker

We use IsaacGym [6] as the physics engine to perform simulation. The simulated character is skeleton-driven where each joint is modeled as a motor controlled through a PD controller. The control policy works at 30Hz, which is the same with generated kinematics motions, while the simulation runs at 120Hz. Our character has 62 body links and 49 controllable joints. This results in a state space $\mathbf{s}_t \in \mathbb{R}^{(62+|\mathcal{O}|)\times 13}$ including the position, orientation and linear and angular velocities of the character's body links and the manipulating objects $\mathcal{O}$ depending on the given tracking motions. To track the target motion, additional observation $\mathbf{o}_t \in \mathbb{R}^{(62+|\mathcal{O}|)\times 7}$ including the target position and orientation of each body link and object is also introduced as the input to the control policy. The action space is $\mathbf{a}_t \in \mathbb{R}^{49\times 3}$ given each controllable joint has 3 degrees of freedom. All our control policies are optimized using PPO [9] as the backbone reinforcement learning algorithm with network parameters updated by Adam optimizers [3]. The hyperparameters used for policy training are listed in Tab. S2.

C.1. Reward Design

Since fingers take 30 out of 49 controllable joints and have a higher requirement in control precision for accurate object manipulation, we define a reward function containing three terms to evaluate the full-body pose and finger pose separately:

$$r = 0.8r_{\text{body}} + 0.2r_{\text{hand}} + 0.05r_{\text{energy}}, \tag{S1}$$

where r_{body} evaluates the full-body poses excluding fingers, r_{hand} evaluates the hand poses related to the target manipulating object, and r_{energy} is an additional energy-related term penalizing end effectors' abrupt movement. Note that in our implementation, at any time step, at most one object is activated as the manipulation target, while multiple objects may exist in the scene and would be manipulated one by one.

For full-body pose evaluation, we consider the orientation and position accuracy of the character's body links and the target object:

$$\begin{aligned} r_{\text{body}} =& 0.5\exp\left(-15\sum_{b\in\mathcal{B}} w_{q,b}||\mathbf{e}_{q,b}||^2\right) \\ &+ 0.5\exp\left(-15\sum_{b\in\mathcal{B}} w_{p,b}||\mathbf{e}_{p,b}||^2\right), \end{aligned} \tag{S2}$$

where $\mathcal{B}$ is the set of all body links of the character and the activated target object at the current time step, $\mathbf{e}_{q,b}$ is the orientation error between each body link b of the simulated character and that in the tracking motion, $\mathbf{e}_{p,b}$ is the position error, and $w_{q,b}$ and $w_{p,b}$ are the associated weights. We measure $\mathbf{e}_{q,b}$ using the radian of the angle needed to rotate the link b to the target orientation, while $\mathbf{e}_{p,b}$ is measured by the Euclidean distance from the current position of link b to the target position. All weight values are listed in Tab. S1. For position tracking, we focus only on the position of the root link and end effectors to ensure that the trajectory of end effectors (foot and hand excluding fingers) matches the tracking target, while relying on orientation tracking to ensure the naturalness of full-body poses.

For hand pose evaluation, to ensure that the object would be manipulated by hands correctly, we consider the fingers' relative position to the target manipulating object when the hand is close to the object, or their relative position to the

Body Part	Value
root (pelvis)	$w_q = 1, w_p = 1$
lower abdomen	$w_q = 0.2$
upper abdomen	$w_q = 0.2$
chest	$w_q = 0.2$
neck	$w_q = 0.2$
head	$w_q = 0.2$
clavicles	$w_q = 0.1$
upper arms	$w_q = 0.2$
lower arms	$w_q = 0.2$
wrists	$w_q = 0.3, w_p = 0.3$
thighs	$w_q = 0.5$
calfs	$w_q = 0.3$
feet	$w_q = 0.2, w_p = 0.1$
target object	$w_q = 1, w_p = 1$

Table S1. Unnormalized weights of each body link of the character and the target object when computing the reward regarding body pose errors (cf. r_{body} in Equation S2). All weights unlisted are zero. The weights are normalized such that $\sum_b w_{p,b} = 1$ and $\sum_b w_{q,b} = 1$ before computing r_{body}.

Parameter	Value
policy network learning rate	5×10^{-6}
critic network learning rate	1×10^{-4}
reward discount factor (γ)	0.95
GAE discount factor (λ)	0.95
surrogate clip range (ϵ)	0.2
number of PPO workers (simulation environment instances)	2048
PPO replay buffer size	2048×8
PPO batch size	256
PPO optimization epochs	5

Table S2. Hyperparameters of policy training for physics tracker.

wrist when the hand is far away from the target object:

$$r_{\text{hand}} = \exp\left(-\frac{5}{|\mathcal{F}|}\sum_{f\in\mathcal{F}}\alpha||\mathbf{e}_{f,o}|| + (1-\alpha)||\mathbf{e}_{f,w}||\right), \tag{S3}$$

where $\mathcal{F}$ is the set of fingers, $\mathbf{e}_{f,o}$ is the position error between each finger f of the simulated character and the tracking target position in the object's local system, $\mathbf{e}_{f,w}$ is the position error in the wrist's local system, and α is an interpolation coefficient depending on the distance between the hand and object in the generated tracking motions. We define $\alpha = 1$ when the hand is equal to or less than 0.25m from the target object and $\alpha = 0$ if the hand is at least 1m far away from the object in the tracking motion.

r_{energy} takes into account the character's end effectors to prevent jittering or abrupt movement with larger acceleration:

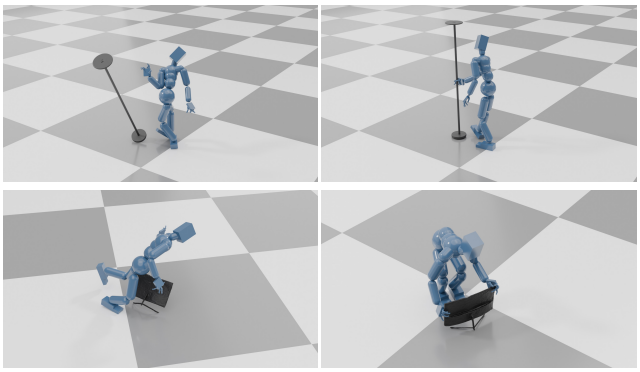
$$r_{\text{energy}} = \exp\left(-\frac{1}{900}\sum_{e\in\mathcal{E}}||\mathbf{a}_e||^2\right), \tag{S4}$$

where $\mathcal{E}$ is the set of the character's key end effectors (feet and hands excluding fingers), and $\mathbf{a}_e$ is the linear acceleration of the end effector e.

In contrast to recent works [1, 5, 13] on object manipulation, our reward function does not rely on the contact states between the hands (or fingers) and objects. This is because the contact data derived from the kinematic motions generated by diffusion models can be unreliable due to artifacts (cf. Fig. 7). Moreover, this approach aligns with the common scenario in which detailed contact information is typically absent in motions obtained from motion capture and video demonstrations [8].

C.2. Importance Sampling Strategy

Given that the generated kinematic motions can span several minutes and involve interactions with multiple objects, it would be highly inefficient to perform motion tracking from the very beginning of the sequence. Since locomotion without object manipulation and the phase following object grasping are relatively easier to learn, we adopt an importance sampling strategy that focuses on pre-grasp states to facilitate policy training. Specifically, we partition the parallel simulated physics environments into $|\mathcal{O}|$ batches, with each batch assigned to a specific manipulating object. In each batch, simulated characters are initialized in a random pre-grasp pose while interacting with the assigned object. This approach increases the likelihood that the policy learns the transition from the pre-grasp to the grasping phase, and subsequently, the locomotion that follows a successful grasp. Moreover, this strategy ensures that the policy consistently learns to interact with multiple objects simultaneously, thereby avoiding potential local minima during policy exploration.



(a) PhysHOI (b) Ours

Figure S5. PhysHOI [13] fails to track kinematic motion, while our policy successfully drives the character to interact with diverse objects.

D. Inference Speed

Achieving real-time performance remains an open challenge for both diffusion-based and optimization-based methods. In our low-level motion generator, producing a 4-second motion sequence takes about four seconds for each diffusion model in stages 1, 3, and 4, while the optimization process in stage 2 requires roughly three minutes. Additionally, training the physics tracker on a 4-second interaction segment takes approximately 2 hours. All experiments were run on a single NVIDIA RTX 4000 GPU.

E. Additional Experiemental Analysis

E.1. Implementation Details of Baseline

Since no prior work addresses the task of generating full-body motion, finger motion, and object motion from language instructions, we adapt prior works CHOIS [4] and GRIP [11] as fair baselines to compare against our system. CHOIS generates full-body motion and object motion from text and initial states, which also serves as our Stage 1. The output of CHOIS will serve as input to GRIP. GRIP consists of an arm denoising model and a two-stage finger motion generation model. It takes full-body motion (without finger motion) and object motion as input to denoise arm motion. It then proposes a hand sensor to extract hand-object spatial features and predict initial finger poses. In the second stage of finger motion generation, GRIP leverages these predicted finger poses to compute proximity features, which serve as input to refine the finger motion.

E.2. Results of Navigation Module

We report the evaluation scores for the navigation module in Tab. S3, using the metrics introduced in Sec. 7.2. The module is evaluated on the test split of the HumanML3D dataset [2]. It demonstrates that the module closely adheres to the

	T_{xy}	H_{feet}	FS	MPJPE
Ours	4.00	1.48	0.58	10.87

Table S3. Navigation synthesis on the HumanML3D dataset [2]. The joint error is relatively small, demonstrating that the model aligns well with the input waypoints.

input waypoints, with a relatively small joint position error when compared to the ground truth data, while maintaining reasonable foot sliding and foot height.

E.3. Results of Physics Tracker

We compare our physics tracker against PhysHOI [13], which trains reinforcement learning policies to control a humanoid character playing basketball. For our task, we have PhysHOI track the kinematic motion generated by our low-level motion generator. As shown in Fig. S5, PhysHOI fails to grasp objects, whereas our method successfully completes the task.

E.4. Human Perception Study Details

We show the interface of the human perceptual study for the high-level planner in Figure S6 and the low-level motion generator in Figure S7.

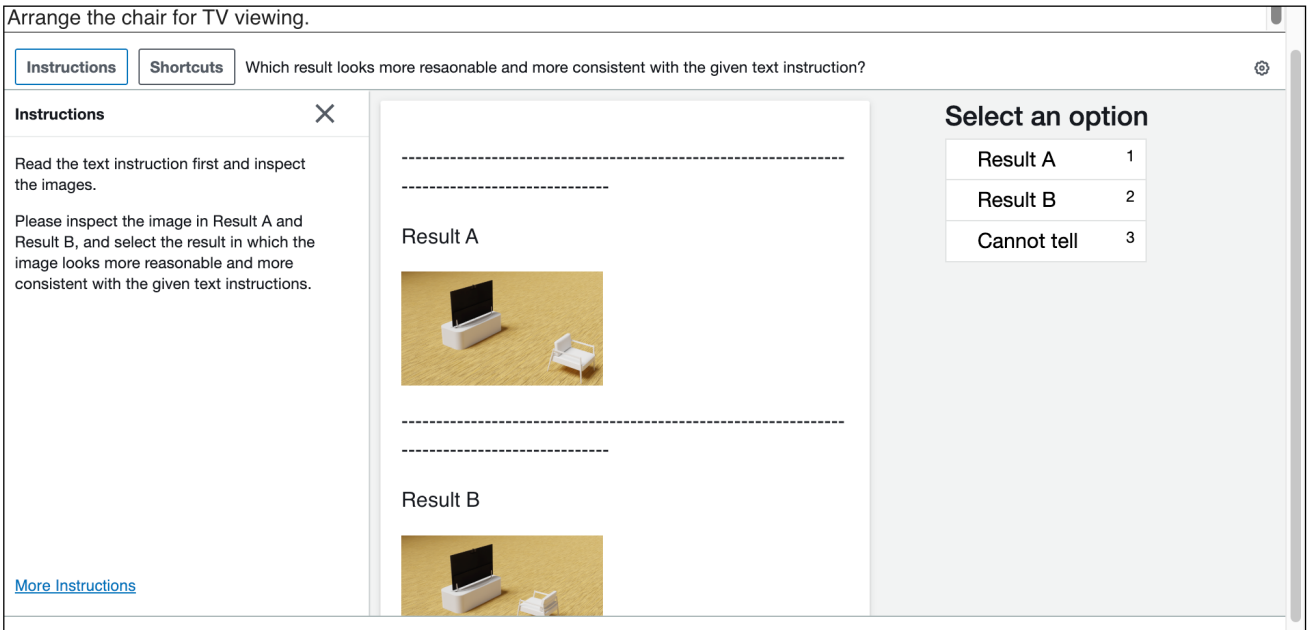


Figure S6. Interface of human perceptual study for the high-level planner.

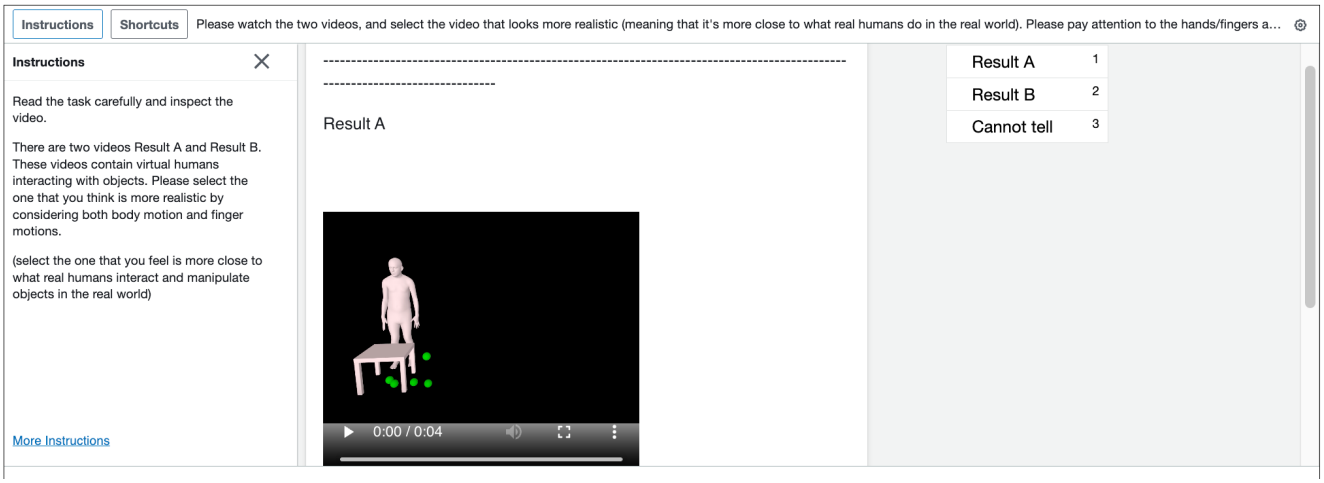


Figure S7. Interface of human perceptual study for the low-level motion generator.

References

[1] Samarth Brahmbhatt, Ankur Handa, James Hays, and Dieter Fox. Contactgrasp: Functional multi-finger grasp synthesis from contact. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2386–2393. IEEE, 2019. 5

[2] Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. Generating diverse and natural 3d human motions from text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5152–5161, 2022. 5, 6

[3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 4

[4] Jiaman Li, Alexander Clegg, Roozbeh Mottaghi, Jiajun Wu, Xavier Puig, and C Karen Liu. Controllable human-object interaction synthesis. *arXiv preprint arXiv:2312.03913*, 2023. 5

[5] Zhengyi Luo, Jinkun Cao, Sammy Christen, Alexander Winkler, Kris Kitani, and Weipeng Xu. Grasping diverse objects with simulated humanoids. *arXiv preprint arXiv:2407.11385*, 2024. 5

[6] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021. 4

[7] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3d hands, face, and body from a single image. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10975–10985, 2019. 2

[8] Nancy S Pollard and Victor Brian Zordan. Physically based grasping control from example. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 311–318, 2005. 5

[9] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. 4

[10] 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. 3

[11] Omid Taheri, Yi Zhou, Dimitrios Tzionas, Yang Zhou, Duygu Ceylan, Soren Pirk, and Michael J Black. Grip: Generating interaction poses using latent consistency and spatial cues. *arXiv preprint arXiv:2308.11617*, 2023. 5

[12] Ruicheng Wang, Jialiang Zhang, Jiayi Chen, Yinzhen Xu, Puhao Li, Tengyu Liu, and He Wang. Dexgraspnet: A large-scale robotic dexterous grasp dataset for general objects based on simulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11359–11366. IEEE, 2023. 1

[13] Yinhuai Wang, Jing Lin, Ailing Zeng, Zhengyi Luo, Jian Zhang, and Lei Zhang. Physhoi: Physics-based imitation of dynamic human-object interaction. *arXiv preprint arXiv:2312.04393*, 2023. 5, 6

[14] He Zhang, Yuting Ye, Takaaki Shiratori, and Taku Komura. Manipnet: neural manipulation synthesis with a hand-object spatial representation. *ACM Transactions on Graphics (ToG)*, 40(4):1–14, 2021. 3