INTERASSEMBLY: ASSEMBLING LONG-HORIZON HUMAN-OBJECT INTERACTIONS VIA RETRIEVAL, OPTIMIZATION, AND GENERATION

Anonymous authors
Paper under double-blind review

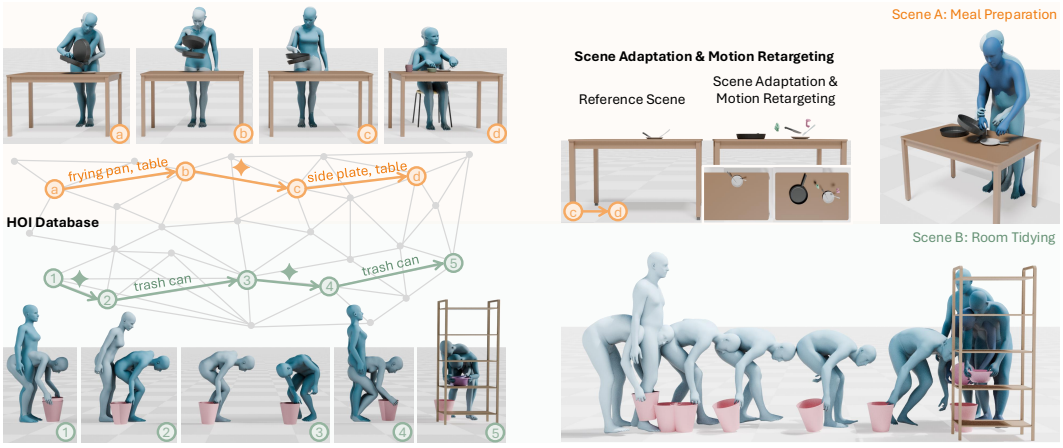


Figure 1: InterAssembly generates long-horizon human-object interaction sequences by traversing HOI databases. Each node represents a reference motion clip; edges connect clips sharing interactable objects. An LLM agent chains clips into a coherent sequence, adapts the reference scene to the target environment, and retargets motion endpoints (center, clips c→d) into object trajectories and motion keyframes, which a motion generation model turns into full-body motion. Scene A (orange) chains four clips for meal preparation; Scene B (green) chains five clips for room tidying.

ABSTRACT

Human-object interaction (HOI) datasets have advanced motion synthesis and embodied behavior modeling, but remain limited to short, isolated clips. This clip-level supervision cannot capture long-horizon activities that require temporally coherent manipulation of multiple objects and consistent scene evolution, essential for embodied intelligence. Collecting such data is prohibitively expensive, and no existing method generates long-horizon, multi-object HOI activities under today's data budgets. We introduce **InterAssembly**, an agentic system that assembles activity-oriented long-horizon HOI animations from an HOI database of short captured clips. Given only a 3D scene and this HOI database, an LLM agent iteratively proposes the next sub-action, retrieves candidate reference clips by structural scene similarity and selects among them by semantic fit to the proposed action, and adapts object placements to the target scene using collision-aware feedback. An interaction mesh retargeting stage then transfers the retrieved human motion to the new scene while preserving bone lengths, foot grounding, and temporal smoothness. The retargeted joint trajectories serve as sparse control signals for a controllable masked motion generation model, which synthesizes dense, smooth full-body motion consistent with the established interaction structure. InterAssembly is, to our knowledge, the first system to generate coherent, long-horizon, multi-object HOI activities, including tool use, object transfer, and multi-step tabletop cooking, at durations far beyond existing HOI generation methods, with low human-object penetration and consistent scene evolution throughout.

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

1 INTRODUCTION

Human–object interaction (HOI), such as cooking, cleaning, assembling tools, and organizing objects, involves sequential decisions, multi-object dependencies, and persistent state changes, and modeling it is critical for motion synthesis, character animation, and embodied-agent training. Large-scale HOI datasets have advanced learning-based motion generation (Li et al., 2023; Kim et al., 2024; Bhatnagar et al., 2022; Huang et al., 2022), but capture only isolated, single-interaction clips. This clip-level supervision cannot represent long-horizon activities requiring coherent multi-object manipulation and scene evolution, so current models (Hu et al., 2024a; Kobayashi et al., 2023) generate individual interactions rather than complete activities.

Collecting longer sequences does not resolve this: capture effort and annotation cost grow with sequence length, and pose errors and inter-penetrations accumulate, making collision-free, semantically correct long-horizon capture notoriously difficult. The same bottleneck limits text-conditioned HOI generators, which work well where large annotated datasets exist but lack long-horizon supervision at scale. We instead ask how far composition, retrieval, geometric adaptation, and generative refinement can stretch an HOI database of few-second clips into extended activities it does not contain. Recent LLM advances in scene understanding, planning, and spatial reasoning (Fu et al., 2025; Feng et al., 2023; Ahn et al., 2022) make this feasible, but doing so requires solving three interdependent problems: deciding what to do next, adapting the interaction to the target scene's geometry, and generating spatially plausible motion for the human and objects involved.

We address these three problems through InterAssembly, a unified pipeline with three components. The first is an LLM-driven agentic loop for high-level sequential planning: given a 3D scene and an HOI database of captured clips, the agent iteratively proposes the next action, retrieves the most compatible reference clip, and maps objects in the current scene to those in the retrieved interaction. The agent does not merely replay the clip: LLM-in-the-loop placement stages, guided by real-time scene descriptions and penetration feedback, adapt both the pre-action object configuration and the post-action resting state to the target scene's geometry.

The second component, an interaction mesh-based (Ho et al., 2010) retargeting method, bridges the retrieved motion and the scene layout. We couple the human skeleton and object point clouds into a unified volumetric interaction mesh, then solve for deformed human joint positions consistent with the adapted object trajectories via an As-Rigid-As-Possible (ARAP) (Sorkine & Alexa, 2007) optimization, producing per-joint trajectories that respect the new geometry.

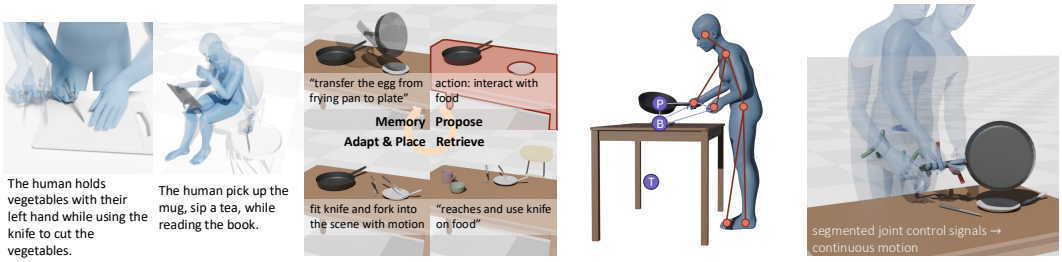
The third component turns these trajectories into natural motion. We use the retargeted joint positions as sparse control signals for MaskControl, a controllable masked generative motion model. MaskControl acts as a learned motion prior: it generates dense, smooth full-body motion that stays faithful to the contact and interaction structure established by retargeting.Together, these three components turn a bare 3D scene and an HOI database into extended multi-action sequences, flipping then eating a pancake, transferring objects across a shelf, cleaning a room, without any manual keyframing or scene-specific tuning.

To our knowledge, InterAssembly is the first system to address this composition problem (Sec. 2). Our main contributions are:

- **A modular, training-free pipeline** that assembles atomic motion clips into long-horizon, multi-object activities, built from decoupled planning, retrieval, retargeting, and synthesis stages that benefit directly from advances in each component.
- **An LLM-driven agentic framework** for sequential human-object interaction planning that jointly handles action proposal, database retrieval, object mapping, and iterative scene adaptation with collision-aware feedback.
- **An interaction mesh-based retargeting formulation** that couples human skeletal motion and object trajectories in a unified volumetric structure, supporting cross-scene transfer with bone-length adaptation.

2 RELATED WORK

LLM Agents for 3D Retrieval and Generation. Large language models have been gradually introduced to 3D generation tasks like object and scene generation. These methods leverage the



(a) HOI Database (b) LLM-Driven Sequential Planning and Scene Adaptation (c) Interaction Mesh Retargeting (d) Smooth Motion Synthesis via MaskControl

Figure 2: InterAssembly generates long-horizon HOI animations from (a) an HOI Database of reference clips, through three pipeline stages: (b) LLM-Driven Planning and Scene Adaptation proposes the next action, retrieves matching clips, and adapts placements to the target scene. (c) Interaction Mesh Retargeting deforms retrieved joints to fit the new scene, preserving bone lengths. (d) MaskControl synthesizes temporally coherent motion from retargeted joints as sparse control signals.

LLM's emergent reasoning and planning capabilities, primarily falling into two schools of thought. The first utilizes a procedural paradigm, employing LLMs as agentic programmers. These systems, such as (Lu et al., 2025b; Sun et al., 2023; Zhang et al., 2025a), do not generate geometry directly but instead write interpretable code to create editable assets. The second paradigm treats 3D as a new modality, using a tokenizer or an encoder to tokenize shapes so that the LLM can generate them autoregressively (Wang et al., 2024; Ye et al., 2025). At the frontier of this field, in complex scene generation, agentic frameworks like (Liu et al., 2025; Ling et al., 2025; Hu et al., 2024b) are converging on an LLM-as-orchestrator model, where a central LLM agent coordinates specialized sub-agents to plan layouts, ensure visual plausibility, and compose entire worlds.

Motion Retargeting and Generation. The synthesis of 3D human motion has evolved from VAE models (Petrovich et al., 2022) to a paradigm dominated by diffusion-based generative models (Tevet et al., 2023; Zhang et al., 2024a). This new generation of models primarily focuses on controllable synthesis from various modalities. Text-to-motion models aim for open-vocabulary and zero-shot capabilities by training on massive datasets (Liang et al., 2024; Fan et al., 2025), while audio-to-motion models specialize in generating beat-synchronized dance choreography (Siyao et al., 2022; Tseng et al., 2023; Huang et al., 2024). Alongside pure generation, a significant line of work addresses fine-grained controllability, using signals like spatial trajectories (Karunratanakul et al., 2023; Wan et al., 2024) or keypoints for motion retargeting (Chang et al., 2024).

Human-Object Interaction Datasets and Generation. Capturing and modeling human-object interactions is a long-standing challenge in vision and graphics. Early HOI datasets (Bhatnagar et al., 2022; Taheri et al., 2020; Li et al., 2023; Zhao et al., 2024; Huang et al., 2022; Jiang et al., 2022; Zhang et al., 2023) recorded 3D human motion with object trajectories and contact annotations, and later efforts expanded to richer object categories, more diverse interactions, and fine-grained bimanual manipulation (Kim et al., 2024; Jiang et al., 2024). HUMOTO (Lu et al., 2025a) and HIMO (Lv et al., 2024) capture sequences with multiple simultaneously manipulated objects, and HUMOTO further records complete multi-step tasks; our HOI database is built on HUMOTO. InterAssembly goes beyond any single captured sequence by composing sub-clips across source sequences and adapting them to target scenes never captured with that motion.

On the synthesis side, early work framed HOI generation as motion generation conditioned on the object (Li et al., 2023), with later methods moving toward joint human-object synthesis (Diller & Dai, 2024; Li et al., 2024). Diffusion-based approaches (Xu et al., 2023; Peng et al., 2023; Zhang et al., 2024b) produce temporally coherent interactions conditioned on language or object geometry, and recent systems target long-horizon or multi-object settings (Lv et al., 2024; Li et al., 2024; Zhang et al., 2025b; Deng et al., 2025), but each requires a trained per-segment or per-task conditional model, composes one object per step, or is limited to hands. As conditional models trained end-to-end on captured HOI data, they are constrained by the same data scarcity that makes long-horizon capture difficult. InterAssembly instead treats HOI synthesis as retrieval and adaptation over existing short clips.

3 OVERVIEW

Given an initial scene, either a target 3D environment or one derived from a seed clip, and an HOI database of short human-object interaction clips, InterAssembly produces a long-horizon animation in which a virtual human performs a temporally coherent sequence of manipulation actions consistent with the scene. The pipeline has three stages (Fig. 2), detailed in Secs. 4 to 6; Sec. C.1 gives the full algorithmic loop.

HOI Database. Each clip is annotated with object trajectories, per-joint human poses, semantic descriptions, and spatial relationship graphs, and is indexed by a structural descriptor (Sec. 4.2); semantic matching is performed by the LLM during selection (Sec. 4.1). All interaction content in the output originates from retrieved clips, and the generative model in Stage 3 acts as a motion prior that densifies and smooths the retargeted trajectories. We use HUMOTO (Lu et al., 2025a), though any database with these annotations is compatible.

Open-Ended Sequencing. InterAssembly requires no user-specified activity goal. Starting from the initial scene, the agent proposes and chains sub-actions that follow logically from one another until a fixed action budget is reached (7–12 sub-actions in our experiments, Sec. C.8).

Given the current scene state and action history, **Stage 1** (Sec. 4) uses an LLM agent to propose the next action, retrieve the best-matching reference clip, and adapt object placements to the target scene through iterative, collision-aware placement. **Stage 2** (Sec. 5) deforms the retrieved human motion to match the adapted object trajectories via an interaction-mesh ARAP optimization that preserves bone lengths, foot contact, and temporal smoothness. **Stage 3** (Sec. 6) feeds the retargeted joint trajectories as sparse control signals to MaskControl, a controllable masked generative motion model, which synthesizes dense, smooth full-body motion.

4 SCENE AND ACTION PLANNING VIA LLM AGENT

Our LLM agent governs InterAssembly's high-level loop: given the current scene state and action history, it decides what to do next, retrieves a matching reference clip, and adapts the scene and object trajectories so the retrieved motion is valid in the new environment. This loop executes the four substeps of pipeline Stage 1 shown in Fig. 3: **Propose**, **Retrieve**, **Select**, and **Adapt & Placement**.

4.1 SEQUENTIAL ACTION PROPOSAL

At the start of each iteration, the agent receives the full action history and scene setup as a natural-language memory string summarizing, for each past action, the objects involved, the motions, and the scene state before and after. Given this context and the current object list, an LLM proposes the next action as a natural-language description paired with an object list ranked by importance.

The proposal drives a two-stage retrieval pipeline. First, the proposed object list filters the database to clips involving all proposed objects, relaxing to the most important subset if too few candidates remain. Remaining candidates are re-ranked by structural similarity (Sec. 4.2), retaining the top-K; since structural similarity alone cannot distinguish candidates that share a layout but differ in action or outcome, we resolve this ambiguity at the Select step using each candidate's semantic description of its full motion, not just its start or end pose. The agent jointly reasons over the pro-

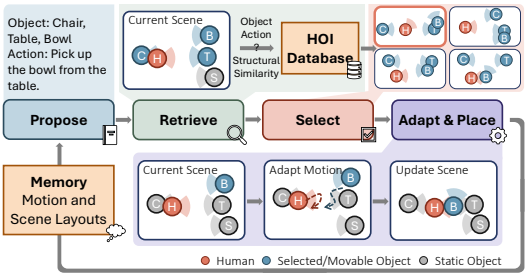


Figure 3: LLM-driven sequential planning and scene adaptation. Given the memory of executed actions, the LLM **proposes** the next sub-action with relevant objects, then **retrieves** candidate clips from the HOI Database by structural scene similarity and **selects** the best candidate by its semantic fit to the proposed action. The selected candidate is passed to **Adapt & Placement**, which adapts the human and object motions to the current scene geometry and updates object poses. A **Memory** module accumulates motion and scene layout history to inform subsequent sub-action proposals, enabling coherent long-horizon execution.

posed action text, these semantic descriptions, and structural scores to select the best match, then resolves an object mapping, prunes non-essential objects, and produces a rewritten action description grounded in the selected instances.

4.2 Structural Similarity and Reference Alignment

To rank candidates by spatial compatibility with the current scene, we compute a structural similarity score for each candidate. Given a query layout $Q = \{(\mathbf{p}_i^Q, \mathbf{R}_i^Q)\}_{i=1}^n$ and a candidate layout $C = \{(\mathbf{p}_i^C, \mathbf{R}_i^C)\}_{i=1}^n$, where $\mathbf{p}_i \in \mathbb{R}^3$ and $\mathbf{R}_i \in SO(3)$ denote the position and orientation of the i-th entity, the structural similarity is defined as $S_{\text{struct}} = \exp(-L_{\text{best}})$, where L_{best} is the minimum pairwise layout cost over all symmetry-equivalent configurations:

$$L = \frac{2}{n(n-1)} \sum_{i<j} \left(w_d \|\mathbf{v}_{ij}^Q - \mathbf{v}_{ij}^C\| + w_r d_R(\mathbf{R}_{ij}^Q, \mathbf{R}_{ij}^C) + w_\theta |\theta_{ij}^Q - \theta_{ij}^C| \right) \tag{1}$$

Here $\mathbf{R}_H \in SO(3)$ is the rotation of the anchor, i.e., the human pelvis frame at that timestep in the layout being scored, so $\mathbf{v}_{ij} = \mathbf{R}_H^\top(\mathbf{p}_i - \mathbf{p}_j)$ expresses the relative displacement between objects i and j in that local frame, making the score invariant to global scene orientation; $d_R(\mathbf{R}_{ij}^Q, \mathbf{R}_{ij}^C)$ is the geodesic distance between the relative rotations $\mathbf{R}_{ij} = \mathbf{R}_i^\top \mathbf{R}_j$; and θ_i, θ_j are the signed horizontal bearing angles of objects i and j as seen from the anchor (wrapped to $(-\pi, \pi]$), so $\theta_{ij} = \theta_i - \theta_j$ penalizes reflected configurations that the unsigned d_R term alone would miss. w_d, w_r, w_θ (Sec. B) are set empirically to balance these differing units (meters vs. radians) for ranking purposes rather than derived from a physical model. We minimize over the symmetry group Σ of the involved objects, $L_{\text{best}} = \min_{\sigma \in \Sigma} L(\sigma)$, so the score reflects the best semantically valid match rather than a single arbitrary orientation. The Select prompt (Sec. E.2) additionally computes this same layout cost anchored on one of the proposed objects rather than the human, as a secondary consistency check on the non-anchor objects; we refer to this application of the identical formulation as the alignment similarity score.

Using this symmetry-optimized configuration, we align the retrieved clip to the target scene: object rotations are corrected, the retrieved scene is rigidly aligned to the current scene by anchoring to the human, and any object required by the proposed action but not yet in the scene is loaded at its aligned position by a dedicated initial-placement LLM call that shares the pre-retargeting round's prompt structure and toolbox (Sec. 4.3). This coarse initialization keeps subsequent LLM-driven placement corrections local.

4.3 Iterative LLM-Driven Agentic Placement

Once the scene is initialized, the agent runs an iterative placement loop to make the retrieved motion spatially plausible for the target environment. Before the action begins, the LLM is allowed to adjust the starting poses of movable objects to better match the preconditions of the reference motion. We highlight two key designs in this stage.

Retargeting-aware adaptation. Rather than treating retargeting as a black box, the LLM observes each object's full executed trajectory, including where it landed, and declares one of three per-object *adjust modes*: **retarget** recomputes the full trajectory to the corrected pose; **inbetween** keeps the executed trajectory and appends a short interpolation bridge; and **both** recomputes the trajectory then applies a further settling correction. This visibility and declarative control let the agent handle geometric mismatches between scenes without manual intervention.

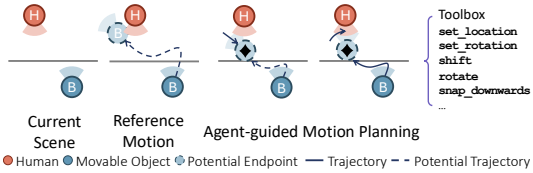


Figure 4: Agent-guided adaptation and placement. Given the current scene and a retrieved reference motion, an LLM agent uses a spatial toolbox to determine a valid endpoint for the movable object, then retargets the reference trajectory to that endpoint. An inbetween motion is further synthesized to smoothly transition the object from its current position to the start of the retargeted trajectory.

Collision-aware placement. The LLM also receives penetration reports covering the pre-action approach path and the action trajectory, and is instructed to resolve any reported collisions,

such as a bowl clipping through a table or a tool intersecting a shelf, within a fixed iteration budget (Sec. B). This coupling between the LLM's spatial reasoning and a geometry-level collision checker keeps the agent stage's output physically grounded, not merely semantically plausible, but non-penetration is not strictly enforced: if the budget is exhausted first, the loop proceeds with the best configuration reached so far, which may still contain residual collisions (Sec. D.6). We therefore call this mechanism collision-aware rather than collision-free.

4.4 Memory and Long-Horizon Continuity

After each action, the agent updates a persistent memory with a structured summary: the action, objects involved, motion descriptions, and start/end spatial relationship graphs. This memory is included verbatim in all subsequent proposal, selection, and placement calls, discouraging unnecessary repetition (repetition the task itself requires, e.g., repeatedly repositioning an unwieldy object, is not blocked), informing object mapping when duplicate object types are present, and explaining why objects are positioned as they are.

5 Motion Adaptation via Interaction Mesh

Directly generating human motion conditioned on a target scene remains intractable with current HOI datasets and methods, which lack the diversity and precision to synthesize plausible interactions across varying object geometries. Instead, we decouple the problem into two stages. The Interaction Mesh (Ho et al., 2010) first transfers the spatial semantics of a retrieved reference motion (contact relationships, relative proximity, limb configurations) into the target scene, providing a structured, scene-aware motion prior that is not directly usable as a final result but substantially constrains the subsequent generation step, reducing it from unconditioned synthesis to tractable refinement.

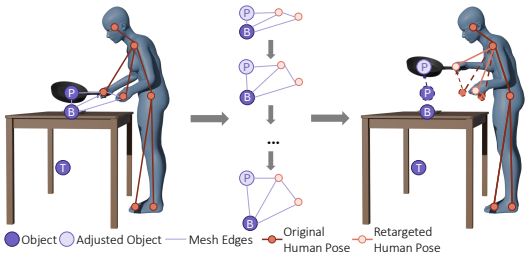


Figure 5: Interaction mesh retargeting. Human joints and object vertices from the retrieved clip (left) are jointly triangulated into a volumetric interaction mesh. As object positions are adapted to the target scene, ARAP deformation iteratively propagates these displacements through the mesh to retarget the human joint trajectories (center), yielding poses that respect the new object geometry (right).

This encodes spatial semantics through volumetric Laplacian deformation, but its linear formulation assumes small rotational deviations: scenes with significant rigid-body rotations (rotating machinery, complex multi-object interactions) introduce shearing artifacts and volume loss. We address this with an iterative, rotation-invariant adaptation scheme that preserves local rigidity and decouples object trajectories: per-vertex rotations are estimated via a weighted Kabsch step so an As-Rigid-As-Possible (ARAP) (Sorkine & Alexa, 2007) energy E_{ARAP} targets rotated differential coordinates rather than shrinking the mesh; a Gaussian-decay edge weighting decouples independent objects' kinematics while preserving local rigidity; and an anisotropic foot term E_{foot} lets feet slide horizontally while preventing floor interpenetration or floating. The full derivation of these three components is given in Sec. B.3.

The final motion minimizes a composite energy balancing spatial semantics, spatial plausibility, and target constraints:

$$E_{total} = w_L E_{ARAP} + w_B E_{bone} + w_A E_{accel} + w_{rig} E_{rigidity} + w_{traj} E_{traj} + w_{foot} E_{foot} \tag{2}$$

where E_{ARAP} enforces rotation-invariant spatial relationships, E_{bone} preserves skeletal lengths, E_{accel} ensures temporal smoothness, $E_{rigidity}$ maintains manipulated object shapes, and E_{traj}, E_{foot} guide objects and feet toward their target locations.

Table 1: **Quantitative comparison of HOI generation quality.** *Evaluated sequence characteristics*: whether the method is conditioned on a 3D scene, average sequence duration, and average number of sub-actions and manipulated objects per sequence (maximum over sequences in parentheses). *Object & contact*: object jerk and human-object penetration. *Human motion*: foot sliding and jerk. *Perceptual*: motion, interaction, and overall quality ratings (5-point Likert, mean±std) and mean diversity & complexity rank among the three generation methods (1=best) from our user study (Sec. 7.2). Gray rows are real-capture references and are not ranked; †HUMOTO is the HOI database from which InterAssembly retrieves reference clips and thus serves as an upper reference for transferred clips. (✓): partial scene conditioning (object waypoints only). Among generation methods, **bold** marks the best value, and $(n\times)$ gives the reduction relative to the best baseline. ↓/↑: lower/higher is better.

Dataset / Method	Evaluated Sequence Characteristics				Object & Contact		Human Motion		Perceptual			
	Scene aware	Duration (s)	#Sub-actions	#Obj-ects	Object Jerk ↓ (m/s^3)	H-O Pen. ↓ (cm)	Foot Sliding ↓ (cm)	Jerk ↓ (m/s^3)	Motion Rating ↑ (1–5)	Interaction Rating ↑ (1–5)	Overall Rating ↑ (1–5)	Div. & Cmplx. ↓ (rank)
Real-capture references												
BEHAVE (Bhatnagar et al., 2022)	–	–	–	–	10.40	0.0606	4.55	4.08	–	–	–	–
OMOMO (Li et al., 2023)	–	–	–	–	27.40	0.0602	2.13	15.10	–	–	–	–
IMHD (Zhao et al., 2024)	–	–	–	–	24.06	0.1172	1.47	1.14	–	–	–	–
HUMOTO† (Lu et al., 2025a)	–	–	–	–	1.13	0.0068	0.95	1.87	–	–	–	–
Generation methods												
HOI-Diff (Peng et al., 2023)	×	2.6	1.0	1.0	158.73	0.6065	15.87	**1.18**	2.47±1.22	1.37±0.53	1.71±0.61	2.91
CHOIS (Li et al., 2024)	(✓)	4.0	1.0	1.0	145.37	0.3446	**3.36**	5.61	3.90±1.17	2.84±1.30	3.31±1.16	2.09
InterAssembly (Ours)	✓	39.1	10.7 (max 12)	3.0 (max 7)	**6.13** (24×)	**0.0354** (10×)	6.35	4.69	**4.00**±1.02	**3.92**±1.02	**4.02**±0.99	**1.00**

6 Motion Synthesis via MaskControl

The joint trajectories from interaction mesh retargeting lack a motion prior due to their geometric-optimization origin. We therefore use them as sparse control signals for MaskControl (Pinyoanuntapong et al., 2025), which synthesizes smooth, natural full-body motion conditioned on these controls and LLM-generated action descriptions.

MaskControl operates on a fixed-length window, so for long-horizon sequences we adopt a sliding-window scheme: each window's control trajectory is recentered and reoriented by the accumulated root rotation from the previous window, and overlapping frames, transformed into the new coordinate frame, serve as hard joint constraints for boundary continuity. Since the pretrained MaskControl weights have no representation of scene geometry, we add an SDF-based object-avoidance loss on the hands and feet as inference-time guidance, penalizing interpenetration and encouraging plausible contact. Other joints receive no such guidance, so the pipeline can still miss an intended interaction whose contact falls outside these joint groups (Sec. D.6).

Since MaskControl uses a body-only skeleton, we recover hand poses post hoc: given fingertip targets from the interaction mesh, we optimize hand pose parameters jointly with the wrist, elbow, and shoulder joints via a fingertip position loss, restricted to interaction-relevant frames by the contact mask; other frames inherit the nearest optimized pose. This yields SMPL-H sequences with interaction-consistent hand poses.

7 Evaluation

We compare InterAssembly against HOI-Diff and CHOIS, the closest available generative HOI baselines, both run from their official public releases without modification (Sec. D.1).

All LLM calls throughout the pipeline use Gemini-3-flash-preview (Google DeepMind, 2025). We evaluate on 15 sequences selected from 60 generated candidates, spanning a total of 587 seconds (39.1 s on average) across diverse activities and scene configurations; Sec. D.3 details the selection process and compute cost. All other components run without any scene- or task-specific fine-tuning.

7.1 Quantitative Evaluation

We measure the evaluated sequences' characteristics, object and contact quality, human motion quality (metrics from HUMOTO (Lu et al., 2025a), Sec. D.4), and perceptual quality from our user study (Sec. 7.2), comparing against HOI-Diff (Peng et al., 2023) and CHOIS (Li et al., 2024) with real-capture datasets as unranked references (Tab. 1).

Sequence characteristics. Both baselines generate a single interaction with one object, lasting 4.0 s for CHOIS and 2.6 s on average for HOI-Diff. InterAssembly produces sequences averaging 39.1 s, roughly $10\times$ and $15\times$ longer, that chain 10.7 sub-actions over 3.0 objects (up to 7), and it is the only method fully conditioned on the 3D scene. Differences in evaluation settings across methods are discussed in Sec. D.2.

Object and contact quality. InterAssembly achieves the lowest object jerk ($6.13\,m/s^3$) and human-object penetration (0.0354 cm) among generation methods, $24\times$ and $10\times$ lower than the best baseline. Both values are also below BEHAVE, OMOMO, and IMHD; only HUMOTO, the database InterAssembly retrieves from, scores lower. We attribute this to the retrieval-and-retargeting design: InterAssembly inherits captured object trajectories through ARAP retargeting and preserves their contact structure in the new scene, whereas CHOIS and HOI-Diff synthesize object motion from scratch, yielding object jerk above $145\,m/s^3$.

Human motion quality. Foot sliding (6.35 cm) is less than half of HOI-Diff's 15.87 cm but higher than CHOIS and the capture datasets. Human jerk ($4.69\,m/s^3$) falls between the two baselines; HOI-Diff pairs the smoothest human motion ($1.18\,m/s^3$) with the least smooth object motion ($158.73\,m/s^3$). These gaps stem from the synthesis stage, where we use a controllable motion generation model trained without any interaction-aware data: the large-scale motion datasets such models are trained on do not pair body motion with object manipulation, so the model has no prior for foot contact and body dynamics during interaction. As interaction-aware motion data becomes available, a model trained on it can replace this stage directly.

Retrieval vs. Generation. Adaptation substantially transforms the retrieved clips (Sec. D.5). Retrieved clips often come from dissimilar scene layouts, and adaptation reshapes human joint positions by 5.3–5.9 cm on average (up to 10.3 cm in the most-adapted sequence) and object trajectories by 19.6 cm on average (up to 182.8 cm) to fit the target scene. Synthesized transitions lie $1.24\times$ farther from the 735-clip HOI database than the retrieved clips themselves, indicating that they contain motion absent from the database.

7.2 User Study and Qualitative Results

We conducted a perceptual study with 17 participants, comparing InterAssembly against HOI-Diff (Peng et al., 2023) and CHOIS (Li et al., 2024) on motion quality, interaction quality, and overall quality. For absolute quality ratings, each participant was shown a single video per method and rated it on a 5-point Likert scale. For pairwise comparisons, participants were shown two videos side by side and selected the preferred method. For diversity and complexity, participants were shown a set of four videos sampled from one method and ranked all three methods. The absolute ratings and rankings are reported in the *Perceptual* columns of Tab. 1.

InterAssembly achieves the highest scores in interaction quality and overall quality, substantially outperforming HOI-Diff on all axes and leading CHOIS on interaction plausibility despite comparable motion quality scores. In pairwise comparisons, InterAssembly wins or ties HOI-Diff in 82.4% of comparisons on motion quality and 100% on interaction quality, and wins or ties CHOIS in 62.7% and 66.7%, respectively. On overall quality, InterAssembly wins or ties HOI-Diff in 96.1% of pairwise comparisons and CHOIS in 60.8%. All participants unanimously ranked InterAssembly first in both motion diversity and behavioral complexity (mean rank 1.00, vs. 2.09 for CHOIS and 2.91 for HOI-Diff), reflecting the agentic planner's ability to compose extended, multi-action sequences beyond the reach of single-clip generation methods.

We present a breakdown of a long-horizon sequence in Fig. 6, showing how InterAssembly composes a semantically coherent multi-step interaction by chaining heterogeneous retrieved clips with consistent scene geometry and smooth transitions. Further qualitative results are provided in Fig. 8 and Fig. 7 (in the appendix), including single-object and multi-object interaction sequences. We analyze representative failure cases, limited scene awareness during motion synthesis and residual penetration in hard weave-through configurations, in Sec. D.6.

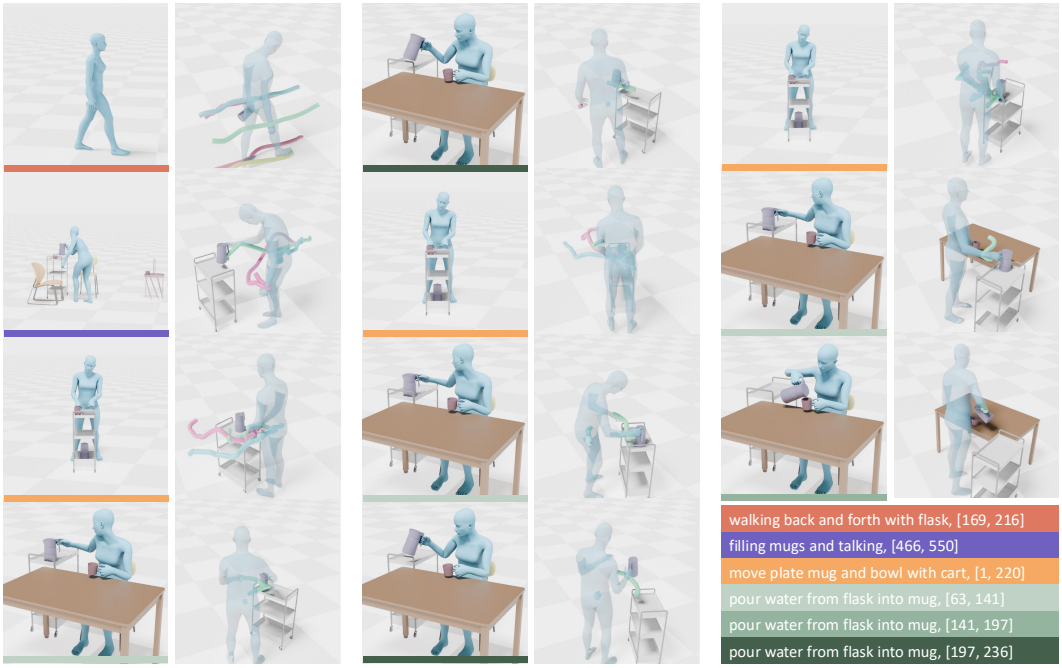


Figure 6: **Pipeline breakdown for an example long-horizon sequence**: the character walks carrying a vacuum flask, places it on a utility cart, pushes the cart while holding the flask, then pours water into a mug on a table. **Left**: The reference motion for each sub-action. The color bar identifies which retrieved clip each sub-action originates from, illustrating that the full sequence is assembled from multiple HOI database clips. **Right**: Progressive scene composition across sub-actions. Objects are introduced incrementally as each new action requires them, and the per-object control trajectory (derived from interaction mesh retargeting) is visualized alongside each stage. Together, the two views demonstrate how InterAssembly composes a semantically coherent long-horizon interaction by chaining heterogeneous retrieved clips with consistent scene geometry and smooth transitions.

8 Conclusions and Future Directions

We present InterAssembly, an agentic framework that assembles long-horizon human-object interaction animations from short captured clips, targeting coherent multi-object activities well beyond the few-second clips in current HOI datasets, a setting no prior HOI generation or retrieval system addresses (Sec. 2), without requiring long-horizon training data. By combining LLM-driven planning and scene adaptation, interaction mesh retargeting, and sliding-window masked motion generation, it produces extended, semantically coherent HOI sequences that no individual component could achieve alone. On the evaluated sequences, object motion and contact quality are competitive with several real-capture datasets, and a perceptual study shows consistent preference for our outputs in interaction plausibility and behavioral complexity. We view the modular design as InterAssembly's central contribution: since each stage feeds the next, gains in the HOI database, retargeting, or motion model propagate through the system, although we cannot yet isolate each stage's contribution via ablation (Sec. D.1).

Because the agent reasons in natural language at every stage, steering it toward a specified goal (e.g., "wash the dishes") is a direct extension, and the retrieval-based design complements emerging video and world models, which can learn from the 3D-grounded data InterAssembly produces and supply new priors to its HOI database (Sec. C.8). Remaining imperfections, including hand articulation, scene awareness during synthesis, residual penetration, and occasional LLM spatial-reasoning errors, mark where our pipeline and the fields it builds on can advance together; since each stage is independently upgradable, progress on either front carries into the full system. With broader evaluation in cluttered environments and finer dexterous control, we see a path toward populating 3D scenes at scale with rich, purposeful human activity, for animation, embodied agent simulation, and beyond.

AI DISCLOSURE

In this work, we used generative AI tools for generating synthetic datasets (as part of the proposed pipeline, Sec. 4), providing feedback on research methodology and experiments, implementing methods, translation, cleaning and reformatting data, and interpreting results. We have not used generative AI tools for developing theoretical models or conceptual frameworks, formulating mathematical claims, or supporting qualitative and thematic data analysis, and providing ingredients for or writing mathematical proofs and proposing or refining hypotheses are not applicable to this work. Additionally, we used generative AI tools for drafting parts of the paper and editing it to improve readability. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have made several efforts to ensure the reproducibility of InterAssembly. Algorithm 1 in Sec. C.1 summarizes the full pipeline, and Sec. B reports the hyperparameters for algorithms used in the pipeline (with interaction mesh retargeting full derivation in Sec. B.3). The agentic system is specified in Secs. C.2 to C.4 and C.6, which define the scene, object, human-state, and memory representations exchanged with the LLM, and in Sec. C.7, which documents every scene-manipulation tool available to the agent. The prompt structure of each LLM stage is summarized in Sec. C.5, and the full prompt templates are provided in Sec. E; all LLM calls use the publicly available Gemini-3-flash-preview (Sec. 7). For evaluation, Sec. D.1 describes how each baseline is run from its official public release, Sec. D.4 gives the formula and normalization for each reported metric, and Sec. D.3 details how the evaluated sequences are generated and selected, together with the compute cost and the limits of what that selection process lets us report.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022.

Bharat Lal Bhatnagar, Xianghui Xie, Ilya Petrov, Cristian Sminchisescu, Christian Theobalt, and Gerard Pons-Moll. Behave: Dataset and method for tracking human object interactions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2022.

Jake Bruce, Michael D. Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal M. P. Behbahani, Stephanie C. Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott E. Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4603–4623. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/bruce24a.html.

Di Chang, Yichun Shi, Quankai Gao, Jessica Fu, Hongyi Xu, Guoxian Song, Qing Yan, Yizhe Zhu, Xiao Yang, and Mohammad Soleymani. Magicpose: Realistic human poses and facial expressions retargeting with identity-aware diffusion, 2024.

Zekai Deng, Ye Shi, Kaiyang Ji, Lan Xu, Shaoli Huang, and Jingya Wang. Human-object interaction via automatically designed VLM-guided motion policy. *arXiv preprint arXiv:2503.18349*, 2025.

Christian Diller and Angela Dai. Cg-hoi: Contact-guided 3d human-object interaction generation. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2024.

Ke Fan, Shunlin Lu, Minyue Dai, Runyi Yu, Lixing Xiao, Zhiyang Dou, Junting Dong, Lizhuang Ma, and Jingbo Wang. Go to zero: Towards zero-shot motion generation with million-scale data, 2025. URL https://arxiv.org/abs/2507.07095.

Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. Layoutgpt: Compositional visual planning and generation with large language models. *arXiv preprint arXiv:2305.15393*, 2023.

Rao Fu, Jingyu Liu, Xilun Chen, Yixin Nie, and Wenhan Xiong. Scene-llm: Extending language model for 3d visual reasoning. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2195–2206, 2025. doi: 10.1109/WACV61041.2025.00220.

Google DeepMind. Gemini 3 flash preview. https://deepmind.google/models/gemini/, 2025.

Edmond S. L. Ho, Taku Komura, and Chiew-Lan Tai. Spatial relationship preserving character motion adaptation. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450302104. doi: 10.1145/1833349.1778770. URL https://doi.org/10.1145/1833349.1778770.

Zhiming Hu, Zheming Yin, Daniel Haeufle, Syn Schmitt, and Andreas Bulling. Hoimotion: Forecasting human motion during human-object interactions using egocentric 3d object bounding boxes. *IEEE Transactions on Visualization and Computer Graphics*, 2024a.

Ziniu Hu, Ahmet Iscen, Aashi Jain, Thomas Kipf, Yisong Yue, David A. Ross, Cordelia Schmid, and Alireza Fathi. Scenecraft: An llm agent for synthesizing 3d scene as blender code, 2024b. URL https://arxiv.org/abs/2403.01248.

Yinghao Huang, Omid Taheri, Michael J. Black, and Dimitrios Tzionas. InterCap: Joint markerless 3D tracking of humans and objects in interaction. In *German Conference on Pattern Recognition (GCPR)*, volume 13485 of *Lecture Notes in Computer Science*, pp. 281–299. Springer, 2022.

Zikai Huang, Xuemiao Xu, Cheng Xu, Huaidong Zhang, Chenxi Zheng, Jing Qin, and Shengfeng He. Beat-it: Beat-synchronized multi-condition 3d dance generation. In *European Conference on Computer Vision (ECCV)*, 2024.

Nan Jiang, Tengyu Liu, Zhexuan Cao, Jieming Cui, Zhiyuan Zhang, Yixin Chen, Heng Wang, Yixin Zhu, and Siyuan Huang. Full-body articulated human-object interaction. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9331–9342, 2022.

Nan Jiang, Zhiyuan Zhang, Hongjie Li, Xiaoxuan Ma, Zan Wang, Yixin Chen, Tengyu Liu, Yixin Zhu, and Siyuan Huang. Scaling up dynamic human-scene interaction modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1737–1747, 2024.

Korrawe Karunratanakul, Konpat Preechakul, Supasorn Suwajanakorn, and Siyu Tang. Guided motion diffusion for controllable human motion synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2151–2162, 2023.

Jeonghwan Kim, Jisoo Kim, Jeonghyeon Na, and Hanbyul Joo. Parahome: Parameterizing everyday home activities towards 3d generative modeling of human-object interactions. *ArXiv*, abs/2401.10232, 2024. URL https://api.semanticscholar.org/CorpusID:267035145.

Makito Kobayashi, Chen-Chieh Liao, Keito Inoue, Sentaro Yojima, and Masafumi Takahashi. Motion capture dataset for practical use of ai-based motion editing and stylization, 2023.

Jiaman Li, Jiajun Wu, and C. Karen Liu. Object motion guided human motion synthesis. *ACM Transactions on Graphics (TOG)*, 42:1 – 11, 2023.

Jiaman Li, Alexander Clegg, Roozbeh Mottaghi, Jiajun Wu, Xavier Puig, and C. Karen Liu. Controllable human-object interaction synthesis. In *ECCV*, 2024.

Han Liang, Jiacheng Bao, Ruichi Zhang, Sihan Ren, Yuecheng Xu, Sibei Yang, Xin Chen, Jingyi Yu, and Lan Xu. Omg: Towards open-vocabulary motion generation via mixture of controllers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 482–493, 2024.

Lu Ling, Chen-Hsuan Lin, Tsung-Yi Lin, Yifan Ding, Yu Zeng, Yichen Sheng, Yunhao Ge, Ming-Yu Liu, Aniket Bera, and Zhaoshuo Li. Scenethesis: A language and vision agentic framework for 3d scene generation, 2025. URL https://arxiv.org/abs/2505.02836.

Xinhang Liu, Chi-Keung Tang, and Yu-Wing Tai. Worldcraft: Photo-realistic 3d world creation and customization via llm agents, 2025. URL https://arxiv.org/abs/2502.15601.

Jiaxin Lu, Chun-Hao Paul Huang, Uttaran Bhattacharya, Qixing Huang, and Yi Zhou. Humoto: A 4d dataset of mocap human object interactions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10886–10897, October 2025a.

Sining Lu, Guan Chen, Nam Anh Dinh, Itai Lang, Ari Holtzman, and Rana Hanocka. Ll3m: Large language 3d modelers, 2025b. URL https://arxiv.org/abs/2508.08228.

Xintao Lv, Liang Xu, Yichao Yan, Xin Jin, Congsheng Xu, Shuwen Wu, Yifan Liu, Lincheng Li, Mengxiao Bi, Wenjun Zeng, and Xiaokang Yang. Himo: A new benchmark for full-body human interacting with multiple objects, 2024. URL https://arxiv.org/abs/2407.12371.

NVIDIA, :, Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, Daniel Dworakowski, Jiaojiao Fan, Michele Fenzi, Francesco Ferroni, Sanja Fidler, Dieter Fox, Songwei Ge, Yunhao Ge, Jinwei Gu, Siddharth Gururani, Ethan He, Jiahui Huang, Jacob Huffman, Pooya Jannaty, Jingyi Jin, Seung Wook Kim, Gergely Klár, Grace Lam, Shiyi Lan, Laura Leal-Taixe, Anqi Li, Zhaoshuo Li, Chen-Hsuan Lin, Tsung-Yi Lin, Huan Ling, Ming-Yu Liu, Xian Liu, Alice Luo, Qianli Ma, Hanzi Mao, Kaichun Mo, Arsalan Mousavian, Seungjun Nah, Sriharsha Niverty, David Page, Despoina Paschalidou, Zeeshan Patel, Lindsey Pavao, Morteza Ramezanali, Fitsum Reda, Xiaowei Ren, Vasanth Rao Naik Sabavat, Ed Schmerling, Stella Shi, Bartosz Stefaniak, Shitao Tang, Lyne Tchapmi, Przemek Tredak, Wei-Cheng Tseng, Jibin Varghese, Hao Wang, Haoxiang Wang, Heng Wang, Ting-Chun Wang, Fangyin Wei, Xinyue Wei, Jay Zhangjie Wu, Jiashu Xu, Wei Yang, Lin Yen-Chen, Xiaohui Zeng, Yu Zeng, Jing Zhang, Qinsheng Zhang, Yuxuan Zhang, Qingqing Zhao, and Artur Zolkowski. Cosmos world foundation model platform for physical ai, 2025. URL https://arxiv.org/abs/2501.03575.

Xiaogang Peng, Yiming Xie, Zizhao Wu, Varun Jampani, Deqing Sun, and Huaizu Jiang. Hoi-diff: Text-driven synthesis of 3d human-object interactions using diffusion models. *arXiv preprint arXiv:2312.06553*, 2023.

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

Ekkasit Pinyoanuntapong, Muhammad Saleem, Korrawe Karunratanakul, Pu Wang, Hongfei Xue, Chen Chen, Chuan Guo, Junli Cao, Jian Ren, and Sergey Tulyakov. Maskcontrol: Spatio-temporal control for masked motion synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9955–9965, 2025.

Li Siyao, Weijiang Yu, Tianpei Gu, Chunze Lin, Quan Wang, Chen Qian, Chen Change Loy, and Ziwei Liu. Bailando: 3d dance generation by actor-critic gpt with choreographic memory. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In Alexander G. Belyaev and Michael Garland (eds.), *Proceedings of the Fifth Eurographics Symposium on Geometry Processing, Barcelona, Spain, July 4-6, 2007*, volume 257 of *ACM International Conference Proceeding Series*, pp. 109–116. Eurographics Association, 2007. doi: 10.2312/SGP/SGP07/109-116. URL https://doi.org/10.2312/SGP/SGP07/109-116.

Chunyi Sun, Junlin Han, Weijian Deng, Xinlong Wang, Zishan Qin, and Stephen Gould. 3d-gpt: Procedural 3d modeling with large language models. *arXiv preprint arXiv:2310.12945*, 2023.

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. URL https://grab.is.tue.mpg.de.

Guy Tevet, Sigal Raab, Brian Gordon, Yoni Shafir, Daniel Cohen-or, and Amit Haim Bermano. Human motion diffusion model. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=SJ1kSyO2jwu.

Jonathan Tseng, Rodrigo Castellon, and Karen Liu. Edge: Editable dance generation from music. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 448–458, June 2023.

Weilin Wan, Zhiyang Dou, Taku Komura, Wenping Wang, Dinesh Jayaraman, and Lingjie Liu. Tlcontrol: Trajectory and language control for human motion synthesis. In *ECCV*, 2024.

Zhengyi Wang, Jonathan Lorraine, Yikai Wang, Hang Su, Jun Zhu, Sanja Fidler, and Xiaohui Zeng. Llama-mesh: Unifying 3d mesh generation with language models. *arXiv preprint arXiv:2411.09595*, 2024.

Sirui Xu, Zhengyuan Li, Yu-Xiong Wang, and Liang-Yan Gui. InterDiff: Generating 3d human-object interactions with physics-informed diffusion. In *ICCV*, 2023.

Junliang Ye, Zhengyi Wang, Ruowen Zhao, Shenghao Xie, and Jun Zhu. Shapellm-omni: A native multimodal llm for 3d generation and understanding. *arXiv preprint arXiv:2506.01853*, 2025.

Juze Zhang, Haimin Luo, Hongdi Yang, Xinru Xu, Qianyang Wu, Ye Shi, Jingyi Yu, Lan Xu, and Jingya Wang. Neuraldome: A neural modeling pipeline on multi-view human-object interactions. In *CVPR*, 2023.

Mingyuan Zhang, Zhongang Cai, Liang Pan, Fangzhou Hong, Xinying Guo, Lei Yang, and Ziwei Liu. Motiondiffuse: Text-driven human motion generation with diffusion model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(6):4115–4128, 2024a. doi: 10.1109/TPAMI.2024.3355414.

Shuyuan Zhang, Chenhan Jiang, Zuoou Li, and Jiankang Deng. Shapecraft: Llm agents for structured, textured and interactive 3d modeling, 2025a. URL https://arxiv.org/abs/2510.17603.

Yixuan Zhang, Hui Yang, Chuanchen Luo, Junran Peng, Yuxi Wang, and Zhaoxiang Zhang. Oodhoi: Text-driven 3d whole-body human-object interactions generation beyond training domains, 2024b. URL https://arxiv.org/abs/2411.18660.

Zhenhao Zhang, Ye Shi, Lingxiao Yang, Suting Ni, Qi Ye, and Jingya Wang. OpenHOI: Open-world hand-object interaction synthesis with multimodal large language model. In *NeurIPS*, 2025b.

Chengfeng Zhao, Juze Zhang, Jiashen Du, Ziwei Shan, Junye Wang, Jingyi Yu, Jingya Wang, and Lan Xu. I'm hoi: Inertia-aware monocular capture of 3d human-object interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 729–741, June 2024.

APPENDIX CONTENTS

A Additional Qualitative Results **15**

B Implementation Details **16**

B.1 Structural Similarity Weights 16

B.2 Collision Checker 16

B.3 Interaction Mesh Retargeting Details (Sec. 5) 16

B.4 Motion Synthesis Details (Sec. 6) 18

C Agentic System Details **18**

C.1 Full Pipeline Algorithm 18

C.2 Coordinate System and Scene Representation 18

C.3 Object Library Format 19

C.4 Human State Representation 20

C.5 LLM Prompt Structure 20

C.6 Action Memory Format 21

C.7 Scene Manipulation Tools Provided to the Agent 21

C.8 Discussion: LLM Backbone Sensitivity and Limitations 23

D Evaluation **24**

D.1 Evaluation Setup 24

D.2 Interpreting the Quantitative Comparison 25

D.3 Sequence Selection and Compute Cost 25

D.4 Evaluation Metrics 26

D.5 Retrieval-vs-Generation Similarity Analysis 27

D.6 Failure Case Analysis 28

E Full Prompt Templates **29**

E.1 Propose Prompt 29

E.2 Select Prompt 30

E.3 Iterative Placement Prompt 33



Figure 7: **Qualitative results of InterAssembly across five interaction scenarios.** Each row shows one synthesized sequence. The leftmost column provides an overview of the full sequence, and the remaining columns show uniformly sampled frames breaking down the progression of the interaction over time. From top to bottom: (1) meal preparation, where the human pours food onto a side plate and sorts out the food; (2) vase rearrangement across a multi-tier shelf, where the human moves the vase from the top tier to a utility cart and then places it on the second tier; (3) chair recovery and dining, where the human drops a chair, recovers it, sits down, and begins eating; (4) wash tub cleaning, where the human carries the tub to a tap, crouches to clean it, and walks away; and (5) repeated food preparation, where the human chops food on a cutting board and empties the scraps into a trash can. Across all scenarios, InterAssembly produces temporally coherent, multi-step HOI sequences with plausible contact and natural transitions between sub-actions.

A Additional Qualitative Results

We provide additional qualitative results of InterAssembly, complementing the pipeline breakdown in Fig. 6. Figure 7 shows five multi-step, multi-object interaction scenarios, and Fig. 8 shows extended single-object sequences with a clothes rack, a side table, and a shelf.

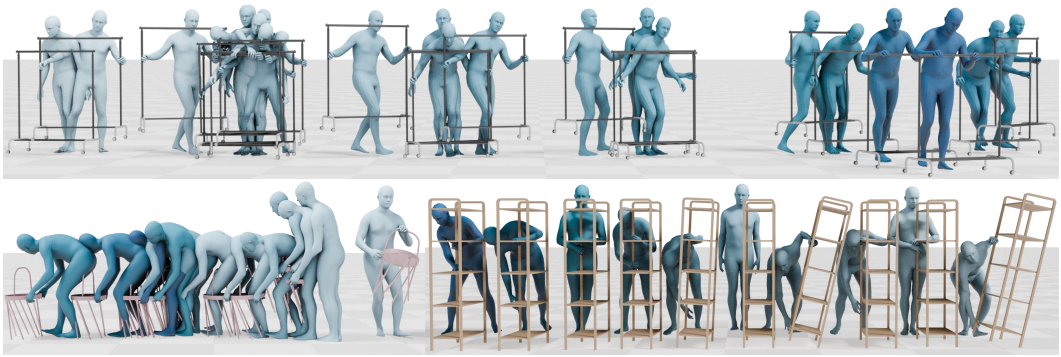


Figure 8: **Single-object HOI sequences generated by InterAssembly.** Human color progresses from light to dark over time. *Top:* Interaction with a clothes rack (sampled every 50 frames). The human repeatedly transports the rack leftward, pausing to lift it several times before reversing direction and pushing it forward. *Bottom-left:* Interaction with a side table (sampled every 200 frames). The human lifts the table off the ground, sets it down, then alternates between pushing and pulling it while periodically lifting it again. *Bottom-right:* Interaction with a shelf (sampled every 200 frames). The human begins by adjusting a screw at the base, then continually repositions and rebalances the shelf throughout the sequence. Horizontal span is scaled for visualization.

B Implementation Details

This section reports implementation details not already covered in Secs. 4 to 6.

B.1 Structural Similarity Weights (Sec. 4.2).

We set w_d=1.0, w_r=0.01, and w_θ=1.0 in the pairwise layout cost; the low rotation weight reflects that relative position dominates perceptual layout match for ranking purposes. Retrieval first keeps only sequences containing all proposed objects; when too few such sequences exist (fewer than 10), it relaxes to sequences containing the higher-ranked subset of the proposed objects (e.g., the first two of three), and a candidate missing a lower-ranked proposed object incurs an additional fixed penalty of 10.0 added to its pairwise cost before the $\exp(-\cdot)$ transform, rather than being excluded outright. After ranking by S_{struct}, the top K=20 candidates are passed to the Select step. The symmetry-equivalent search $\min_{\sigma\in\Sigma}$ samples continuous rotational symmetries at 5° resolution (72 samples) for non-anchor objects and 10° resolution (36 samples) for the anchor object; when human-orientation optimization is enabled, the candidate human yaw is searched at the same 5° resolution.

B.2 Collision Checker (Sec. 4.3).

The geometry-level collision checker underlying the penetration reports in the iterative placement stage performs full-mesh checks rather than bounding-box approximations: every object in the library is pre-processed offline into a signed distance field (SDF), and at query time each object's mesh vertices are tested against the SDFs of all other objects it may collide with. This evaluates both the pre-action approach path and the action trajectory at 5-frame intervals, flagging any inter-penetration between movable objects and static scene geometry in the report passed to the agent. These reports guide placement but do not gate it: each placement round runs for at most 10 LLM iterations (plus up to 5 for the settling pass, Sec. E), and if this budget is exhausted the loop proceeds with the best configuration reached so far.

B.3 Interaction Mesh Retargeting Details (Sec. 5)

This section gives the full derivation of the rotation-invariant, spatially-adaptive interaction mesh formulation summarized in Sec. 5.

Mesh Construction. At each frame i, the interaction mesh's vertex set $\{\mathbf{v}_i\}$ is the union of the human's skeletal joint positions and, for every object involved in the action, the 8 corners of its axis-aligned bounding box (Sec. C.3); we do not sample the object's full surface mesh. These vertices are connected into a single volumetric mesh via Delaunay tetrahedralization over their combined point set, taking every edge of every resulting tetrahedron as a mesh edge. This triangulation is recomputed independently at every frame from that frame's own (pre-optimization) point cloud, not fixed once from a single reference frame, so $\mathcal{N}(i)$ below is a per-frame 1-ring neighborhood.

Rotation-Invariant Laplacian Deformation. Standard volumetric Laplacian coordinates $\delta_i = \mathbf{v}_i - \sum_{j\in\mathcal{N}(i)} w_{ij}\mathbf{v}_j$ are translation-invariant but rotation-sensitive: large target rotations cause the linear solver to minimize energy by shrinking the mesh rather than rotating it. We adopt an As-Rigid-As-Possible (ARAP) (Sorkine & Alexa, 2007) strategy, estimating per-vertex rotation matrices $\mathbf{R}_i$ at each iteration k via the weighted Kabsch algorithm:

$$\min_{\mathbf{R}_i} \sum_{j\in\mathcal{N}(i)} w_{ij}\|(\mathbf{v}_i^k - \mathbf{v}_j^k) - \mathbf{R}_i(\mathbf{v}_i^0 - \mathbf{v}_j^0)\|^2 \tag{3}$$

The Laplacian energy then targets the rotated differential coordinates $\mathbf{R}_i\delta_i^0$, enforcing structural preservation consistent with the instantaneous orientation of each interaction cluster. This term is what Sec. 5 refers to as E_{ARAP}, with weight w_L=1.0.

Gaussian Weighting and Foot Constraints. To decouple the kinematics of independent objects while maintaining local rigidity, we replace the inverse-distance weighting with a Gaussian decay $w_{ij} = \exp(-(d_{ij}^0)^2/2\sigma^2)$, where $d_{ij}^0 = \|\mathbf{v}_i^0 - \mathbf{v}_j^0\|$ is the Euclidean distance between mesh vertices i and j in that frame's own pre-optimization configuration (computed once per frame, before the morph loop begins, and held fixed across its iterations for that frame), and σ=0.2 m. This introduces a soft locality cutoff, ensuring that each vertex is driven by its immediate geometric context and is unaffected by distant scene elements with conflicting trajectories. To adapt ground contacts without artifacts, we introduce anisotropic constraints penalizing deviations strictly along the vertical axis $\mathbf{n}_{up}$:

$$E_{foot} = \sum_{j\in\mathcal{F}} \|(\mathbf{v}_j \cdot \mathbf{n}_{up}) - h_j^{ref}\|^2 \tag{4}$$

where $\mathcal{F}$ is the set of foot joints and h_j^{ref} is joint j's target height along $\mathbf{n}_{up}$ in the target scene at that frame, with weight w_{foot}=2.0. Because only the vertical component is penalized, this term guides feet toward their target heights, preventing floor interpenetration or floating, while still allowing feet to slide horizontally to satisfy the other retargeting terms.

Bone, Acceleration, Rigidity, and Trajectory Terms. The remaining terms of E_{total} (Sec. 5) are, for each frame:

$$E_{bone} = \sum_{(a,b)\in\mathcal{B}} \big(\|\mathbf{v}_a - \mathbf{v}_b\| - \ell_{ab}\big)^2, \quad E_{accel} = \sum_i \|\mathbf{v}_i^{t+1} - 2\mathbf{v}_i^t + \mathbf{v}_i^{t-1}\|^2, \tag{5}$$

$$E_{rigidity} = \sum_{o\in\mathcal{O}} \sum_{(i,j)\in\text{pairs}(o)} \big(\|\mathbf{v}_i - \mathbf{v}_j\| - d_{ij}^0\big)^2, \quad E_{traj} = \sum_{j\in\mathcal{T}} \|\mathbf{v}_j - \mathbf{p}_j^{target}\|^2, \tag{6}$$

where $\mathcal{B}$ is the set of skeletal bone pairs with target length ℓ_{ab} taken from the retrieved clip's skeleton, $\mathcal{O}$ ranges over each movable object and $\text{pairs}(o)$ its 8 bounding-box vertices taken pairwise, each held at its original pairwise distance d_{ij}^0 (so $E_{rigidity}$ penalizes an object's shape changing relative to its own rest-pose bounding box, keeping it rigid independent of the human), and $\mathcal{T}$ ranges over every one of a movable object's 8 bounding-box vertices, each pulled toward its own adapted target position $\mathbf{p}_j^{target}$ from the placement stage (Sec. 4.3).

Weights and Iteration Budget. The above terms are weighted w_{rig}=2.0 and w_{traj}=0.1. The bone-length term anneals its weight w_B from 0.5 to 10.0, so early iterations only loosely constrain bone length and later iterations enforce it. The acceleration term w_A is set to 0.2 for the main action and relaxed to 0.1 during inbetween transitions, allowing larger inter-frame movement when bridging between poses. The composite energy is minimized for a fixed budget of 5 iterations for the main action and 10 iterations for inbetween transitions, rather than to a convergence tolerance.

B.4 Motion Synthesis Details (Sec. 6)

The sliding-window scheme uses windows of 196 frames with an overlap of 30 frames between consecutive windows; the overlapping frames serve as the hard joint constraints for boundary continuity described in Sec. 6. The SDF-based object-avoidance guidance on the hands and feet is applied with weight 0.5.

C Agentic System Details

The InterAssembly agent is purely language-based: the LLM receives no images, depth maps, or rendered views of the scene at any point. All scene information, including object identities, poses, sizes, spatial relations, and motion histories, is serialized into text and passed as part of the prompt context. The subsections below describe the coordinate conventions, object and human representations, and prompt structures that together define the shared vocabulary through which the agent perceives and manipulates the 3D scene.

C.1 Full Pipeline Algorithm

Algorithm 1 summarizes how the three stages described in Secs. 4 to 6 compose into the full iterative loop that turns a bare scene and an HOI database into a long-horizon animation. The loop is not goal-conditioned: it does not detect task completion, and PROPOSE receives no activity instruction. Termination is a fixed action-count budget N (7–12 in our evaluated sequences, Sec. C.8), not a discovered end state.

Algorithm 1 InterAssembly: Long-Horizon HOI Assembly

Require: initial scene S_0, HOI database $\mathcal{D}$, action budget N
Ensure: long-horizon animation: object trajectories and full-body motion segments $\{(T_{\text{obj}}, m)\}$

memory $M \leftarrow \emptyset$; $S \leftarrow S_0$; $k \leftarrow 0$
while $k < N$ **do**
  ▷ *Stage 1: LLM-Driven Planning & Scene Adaptation (Sec. 4)*
  $a \leftarrow \textsc{Propose}(S, M)$ ▷ next sub-action + relevant objects
  $\mathcal{C} \leftarrow \textsc{Retrieve}(a, \mathcal{D}, S)$ ▷ filter by objects, rank by structural similarity S_{struct}
  $c^* \leftarrow \textsc{Select}(\mathcal{C}, a, M)$ ▷ best candidate clip + object mapping
  $S', T_{\text{obj}} \leftarrow \textsc{Adapt\&Place}(c^*, S, M)$ ▷ iterative LLM placement with penetration feedback; invokes ARAPRETARGET below
  ▷ *Stage 2: Motion Adaptation via Interaction Mesh (Sec. 5)*
  $T_{\text{human}} \leftarrow \text{ARAPRETARGET}(c^*, T_{\text{obj}}, S')$ ▷ minimize E_{total} over the interaction mesh; run within ADAPT&PLACE
  ▷ *Stage 3: Motion Synthesis via MaskControl (Sec. 6)*
  $m \leftarrow \textsc{MaskControl}(T_{\text{human}}, a)$ ▷ sliding-window masked motion generation
  $m \leftarrow \textsc{HandRefine}(m, T_{\text{human}})$ ▷ fingertip-target hand optimization
  $M \leftarrow \textsc{UpdateMemory}(M, a, T_{\text{obj}}, m)$; $S \leftarrow S'$; $k \leftarrow k + 1$
end while
return assembled sequence $\{(T_{\text{obj}}, m)\}$

C.2 Coordinate System and Scene Representation

All scene data in InterAssembly is expressed in a right-handed, Z-up coordinate system. The axes are defined as follows: the X-axis points right (+X) and left (–X); the Y-axis points back (+Y) and front (–Y); and the Z-axis points up (+Z) and down (–Z). All positions are in meters; all rotations are Euler angles in XYZ order, given in degrees.

The human's facing direction is encoded solely by the Z-axis rotation of the pelvis. A rotation of 0° corresponds to the human facing the front (along $-Y$), and a rotation of 90° corresponds to facing right (along $+X$). This convention is shared across the LLM prompts, the object library, and the scene description module, so that all system components reason in the same spatial frame.

All animation, and evaluation in Sec. D.4, runs at 30 fps. Object poses are represented as $[x, y, z, r_x, r_y, r_z]$ six-tuples. When describing motion trajectories to the LLM, keyframes are sampled every 10 animation frames (approximately $\frac{1}{3}$ s at this frame rate), yielding a compact sequence that captures the salient structure of each action without exceeding the context budget.

Spatial relations. Scene structure is described using ten directed relation types, each grounded in the coordinate axes above. A relation $(A, \textit{rel}, B)$ is one-directional: if $A \to B$ is stated, $B \to A$ is not repeated.

- **in**: A is contained within the interior volume of B.
- **on**: A rests directly on top of and is supported by B.
- **under**: A is directly beneath B, in contact or near-contact.
- **above**: A is higher than B along the Z-axis but is not resting on it.
- **below**: A is lower than B along the Z-axis but is not directly under it.
- **in_front_of**: A has a smaller Y-coordinate than B (closer to $-Y$, the front).
- **behind**: A has a larger Y-coordinate than B (closer to $+Y$, the back).
- **left_of**: A has a smaller X-coordinate than B.
- **right_of**: A has a larger X-coordinate than B.
- **adjacent_to**: A and B are next to each other with minimal or no gap. When two objects are touching, both **adjacent_to** and the primary directional relation (e.g., **on** or **left_of**) must be stated.

The relations **in** and **on** take priority and must be used whenever they apply, as they are the most semantically informative for reasoning about object placement and interaction preconditions.

C.3 Object Library Format

Each object in the InterAssembly object library is stored as a JSON record with the following fields.

name. A unique identifier string, which may include a suffix encoding a canonical orientation variant (e.g., `basket_90` for a basket rotated 90° about Z).

description. A natural-language paragraph describing the object's appearance, material, and default orientation. The description explicitly states the direction of any prominent axis (e.g., the default handle direction of a basket), so the LLM can reason about reorientation without additional context.

object_size. A three-vector $[w, d, h]$ in meters giving the axis-aligned bounding box of the object in its canonical pose. This is used by the retargeting stage to sample object point clouds.

position_and_relationship. A dictionary mapping spatial relation keywords (e.g., `"in"`, `"on"`) to precise natural-language placement instructions, including metric offsets where necessary. These instructions are injected verbatim into the placement prompt so the LLM can satisfy contact constraints without computing coordinates manually.

rotation. A three-vector $[r_x, r_y, r_z]$ specifying the object's symmetry group along each axis. A scalar entry (e.g., 0) fixes that axis to a single value; a two-element list (e.g., $[0, 180]$) indicates that the object is symmetric under both listed rotations about that axis, so the structural similarity optimizer (Section 4.2) evaluates the layout cost at each element and selects the minimum.

A representative entry is shown below (fields abbreviated to those relevant to symmetry and orientation; a second, complete entry follows). The vase has no preferred orientation about Z due to its circular cross-section, so its Z-rotation is marked `"any"`. This continuous symmetry is not treated exactly: like the other continuous symmetries in Sec. B, it is approximated by sampling candidate orientations at 5° increments (72 samples) during structural-similarity optimization, rather than searched in closed form.

Object Library Entry — Example (vase)

```
"vase": {
    "name": "vase",
    "description": "The object is a vase with a vertically elongated, symmetrical form.
     Its central axis is parallel to the +z axis, with its flat bottom resting on the
     ground. The top of the neck forms a circular opening facing +z.",
    "object_size": [0.156, 0.156, 0.216],
    "rotation": [0, 0, "any"]
```

```
}
```

By contrast, an object such as a table would carry a discrete list, e.g., $[0, 0, [0, 180]]$, indicating symmetry only at $0°$ and $180°$ about Z.

A second entry, including the `position_and_relationship` field omitted above for brevity, illustrates an object whose placement instructions are keyed by facing direction relative to a nearby reference object:

Object Library Entry — Example (working chair, complete)

```
"working_chair": {
    "name": "working_chair",
    "description": "The chair consists of a flat, oval-shaped seat oriented parallel to
     the XY plane, supported by four slender legs. A rounded rectangular backrest is
     positioned above and behind the seat, with its main surface generally parallel to the
      YZ plane and facing forward. The four legs extend downwards from the seat, largely
     parallel to the -Z axis and extends outwardly.",
    "object_size": [0.45, 0.45, 0.766],
    "rotation": [0, 0, 0],
    "position_and_relationship": {
        "facing_front": "When you put the chair back of something and you want the chair
     to face that object, you need to make the chair face front. Make sure the rotation
     around z axis is zero degree to make it face front.",
        "facing_back": "When you put the chair in front of something and you want the
     chair to face that object, you need to make the chair face back. Call
     rotate_object_around_global_axis with angle 180 degree around axis z",
        "facing_left": "When you put the chair right of something and you want the chair
     to face that object, you need to make the chair face left. Call
     rotate_object_around_global_axis with angle -90 degree around axis z",
        "facing_right": "When you put the chair left of something and you want the chair
     to face that object, you need to make the chair face right. Call
     rotate_object_around_global_axis with angle 90 degree around axis z"
    }
}
```

C.4 Human State Representation

The LLM receives the human's state as a compact, two-part description. First, the pelvis is described by its world-space XY position (in meters) and its Z-axis rotation (in degrees), which jointly encode where the character stands and which direction they face. Second, the full skeletal pose is given as a list of world-space joint positions, one per MIXAMO bone (excluding end-effector joints), in the form $\{joint_name: [x, y, z]\}$.

This representation is intentionally minimal: the LLM does not receive raw rotation matrices or bone-local quaternions, as these are difficult to reason about in natural language. Instead, joint positions in world space convey proximity, reachability, and contact information in terms the model can interpret directly.

The downstream retargeting and motion synthesis stages consume a richer format. Trajectories passed to the interaction mesh and to MaskControl are per-frame sequences of joint positions (retargeting) or full six-DoF object poses paired with joint control signals (motion synthesis), as described in Sec. 5 and Sec. 6 of the main paper.

C.5 LLM Prompt Structure

Of the four Stage-1 substeps in Sec. 4 (**Propose**, **Retrieve**, **Select**, and **Adapt & Place**), only RETRIEVE involves no LLM call: it is the deterministic structural-similarity computation of Sec. 4.2, not a language-model prompt. The remaining three correspond to LLM calls and follow a consistent prompt structure, summarized in Tab. 2; full verbatim prompt templates for each are given in Sec. E. Note that **Adapt & Place** runs as two rounds per action – once before retargeting, to reach a low-penetration starting configuration, and once after, to reconcile the retargeted result with the intended end state (Sec. 4.3) – which share the same prompt structure and are therefore listed as a single *Iterative Placement* row below; the two rounds' prompts differ only in the specific object-pose and penetration-report values supplied, as shown in Sec. E. A separate initial-placement round, sharing

this same prompt structure and toolbox, places any object required by the proposed action that is not yet present in the scene (Sec. 4.2); we do not list it as a fourth row or reproduce its template separately in Sec. E since it differs from the pre-retargeting round only in operating on not-yet-placed rather than already-placed objects. The pre-retargeting round's own scope is therefore restricted to objects already in the scene by that point.

Table 2: Prompt structure across the agent's three LLM-driven stages. All stages receive the full memory string as context.

Stage	Key inputs	Expected output
Propose	Memory string; list of objects currently in scene; scene dimensions	Next action description; ordered list of relevant objects
Select	Memory string; top-K candidate descriptions and similarity scores; proposed objects and action	Selected candidate index; object mapping; rewritten action description
Iterative Placement	Memory string; current or retargeted object poses; reference or intended end-state poses; penetration reports (approach path and action trajectory)	Sequence of tool calls via `execute_operations`; per-object adjust mode (*retarget*, *inbetween*, or *both*) in the post-retargeting round

All structured outputs are returned as JSON objects. The prompts instruct the model to reason step by step before emitting the final JSON, and to prefer high-level semantic operations (e.g., `snap_downwards`) over raw coordinate setters where possible.

C.6 Action Memory Format

After each action completes, InterAssembly appends a structured summary to the persistent memory string. This summary contains the following fields.

Action. The rewritten natural-language description of the action that was executed, grounded in the specific object instances involved.

Objects involved. The list of scene-object names that participated in the action.

Human motion. A natural-language description of the overall human movement and how it related to each object, generated by an LLM from the pelvis trajectory and joint data.

Object motion. Per-object natural-language descriptions of how each object moved and interacted with the human, generated from the six-DoF trajectory keyframes.

Spatial relations. Two scene graphs, one at the start of the action and one at the end, each expressed as a list of directed relation triples $(\textit{object}_a, \textit{relation}, \textit{object}_b)$ drawn from the ten allowed relation types defined in Sec. C.2. These graphs record where every object was before and after the action, enabling the planner to track persistent state changes across the full sequence.

The memory string is constructed by concatenating all summaries in chronological order and is passed verbatim into the context of every subsequent Propose, Select, and Placement call. It serves three roles: preventing the agent from proposing actions that have already been completed unnecessarily, informing object mapping when multiple instances of the same object type are present, and providing the placement stage with the history needed to understand why objects are positioned as they are at the start of each new action.

C.7 Scene Manipulation Tools Provided to the Agent

The placement agent manipulates objects in a Blender scene by issuing calls to a library of tools via a batched `execute_operations` interface. Each call submits a JSON list of function calls that execute in order, with per-operation success or error messages returned to the agent. The agent is

instructed to prefer high-level semantic tools over coordinate-based setters. The tools are grouped below by function.

Absolute Setters

These are low-level escape hatches, used only when relative helpers are insufficient.

`set_object_location(object_name, location)` Sets the object's world-space position to the given `[x, y, z]` coordinates by directly assigning `obj.location`.

`set_object_rotation(object_name, rotation)` Sets the object's Euler rotation to `[rx, ry, rz]` in degrees, converting to radians before assigning to `obj.rotation_euler`.

`rotate_object_around_global_axis(object_name, angle, axis)` Rotates the object about a global axis (`x`, `y`, or `z`) by `angle` degrees, centered at the object's own origin. Implemented by decomposing `matrix_world` into translation and rotation-scale parts, applying a global rotation matrix, and recomposing.

Displacement

`shift_object(object_name, displacement)` Translates one or more objects by `[dx, dy, dz]` without changing their rotation, by adding the displacement directly to `obj.location`.

Axis-Based Alignment

These tools align objects relative to a reference without requiring explicit coordinates.

`align_objects_location(target_names, reference_name, axis)` Copies the location of a reference object to each target along the specified axes (any subset of `xyz`). Rotation is unchanged. The reference may be a live scene object or a ghost entry in `REFERENCE_OBJECT_PLACEMENT`.

`align_objects_rotation(target_names, reference_name)` Copies the rotation of a reference object to each target, leaving positions unchanged.

`align_objects_transform(target_names, reference_name)` Copies both location and rotation from a reference, fully matching its pose.

`align_objects_along_axis(object_names, axis, alignment_type)` Aligns all listed objects along one axis (`x`, `y`, or `z`) to the bounding-box extent of the first object. The alignment type is `min`, `max`, or `center` (default). Useful for placing objects flush along a shared edge without manual coordinate lookup.

Relative Placement

`align_relative_position(target, anchor, example_target, example_anchor, mode)`
Places the target object relative to an anchor in the same spatial relationship that an example pair holds in the reference scene. This is the preferred tool for transferring object layouts from the reference clip to the novel scene without specifying any absolute coordinates. Three modes control how the offset is computed and applied.

- `translation_only`: the world-space offset between the example pair is applied directly to the anchor's position.
- `translation_with_anchor_rotation`: the offset is first expressed in the example anchor's local frame, then re-expressed in the anchor's local frame, so relative placement is preserved when the two anchors face different directions.
- `full` (default): as above, and additionally transfers the relative rotation so the target faces the anchor with the same orientation as the example pair.

Ghost reference objects in `REFERENCE_OBJECT_PLACEMENT` are supported for the anchor and example objects.

Spatial Grounding

Applied after alignment to enforce spatial plausibility.

`snap_to_ground(object_names)` Lowers each object until its bounding-box bottom face rests on the ground plane ($z = 0$), by subtracting the minimum bounding-box z from the object's location.

`snap_downwards(object_names)` Snaps each object onto the nearest surface below it using downward raycasting, ignoring self-intersections and the human mesh. The ray is cast from the bounding-box bottom center and iterated up to ten times to resolve layered geometry. Objects are not snapped when a valid upward-facing surface is found within 0.01 m of the bounding-box minimum. More general than `snap_to_ground`, used whenever the support surface is not the ground plane.

C.8 Discussion: LLM Backbone Sensitivity and Limitations

Hallucination. The LLM backbone can fail in two distinct ways, with very different severity. *Function-call-level* hallucination, e.g., invoking a nonexistent tool or supplying malformed arguments, is handled by the validation and retry logic described in Sec. E: an invalid call returns an error string that is fed back to the LLM within the same turn, and the placement loop ends early only after an error-free, self-evaluated completion, otherwise stopping when its iteration budget is exhausted (Sec. 4.3). The harder failure mode is a lack of genuine 3D spatial understanding: the LLM can be confidently wrong about geometric relationships, e.g., misjudging whether two objects actually overlap or how a rotation will affect contact, despite being handed structured numeric scene state. This is a reasoning error over correct inputs rather than an invalid action, so it cannot be caught by tool-execution safeguards by construction. It reflects a longstanding, open limitation of language models.

Scaling with Scene Complexity. A related concern is how the agent's context and reliability scale as the number of objects in a scene grows. Across our evaluated sequences, the human interacts with up to seven distinct objects within a single scene, already large relative to the object count typical of an individual HUMOTO clip. We count only objects the human actually interacts with; scenes may also contain additional, never-manipulated background objects, but we exclude these since their number can grow arbitrarily without being informative about the reasoning load the agent actually faces. A second axis is sequence length: memory accumulates over the full action chain, not just the objects present at any one time, and every subsequent Propose, Select, and Placement call conditions on it. Our evaluated sequences chain 7–12 sub-actions (mean 10.7), each appending a structured summary to the persistent memory string (Sec. C.6). Scaling substantially beyond this regime, i.e., scenes with many more simultaneously interacted objects or much longer action chains, has not been evaluated, and would likely stress both the LLM's context budget and the reliability of its spatial reasoning.

Controllability. InterAssembly's action-proposal step is not architecturally incompatible with goal conditioning: the agent already consumes natural-language context at every stage, so directing it toward a target activity (e.g., "wash the dishes") by seeding or biasing proposals with a text goal is technically feasible. In practice, however, how reliably the agent decomposes such a goal into a correct, executable action sequence is highly dependent on the LLM backbone and on the coverage of the underlying HOI database, rather than being a property our pipeline guarantees. We keep this outside the current work's focus and instead concentrate on establishing that coherent long-horizon assembly from short, atomic clips is possible at all, a prerequisite for reliable goal-driven control rather than a substitute for it. We leave a systematic study of controllability to future work.

Scalability and Video/World Models. A natural question is how a retrieval-based approach like ours relates to the rapid progress of video and world models that aim to directly generate long, coherent interactions(Bruce et al., 2024; NVIDIA et al., 2025). We see these as complementary, not competing: better 3D-grounded, physically correct long-horizon interaction data, exactly what InterAssembly is designed to produce, is training signal such models need, and as they mature they become an additional source of interaction priors or synthetic clips our HOI database can ingest.

Since our contribution is a systematic pipeline of swappable stages rather than a fixed algorithm, progress on either side of this loop directly improves what InterAssembly can assemble.

D EVALUATION

D.1 EVALUATION SETUP

Baseline Implementations. We compare against the official public releases of CHOIS (Li et al., 2024) (https://github.com/lijiaman/chois_release) and HOI-Diff (Peng et al., 2023) (https://github.com/neu-vi/HOI-Diff). We do not modify either codebase or its published model weights, and run each method's own released test/inference scripts and example inputs unchanged, using whatever input and output format that release's default configuration specifies.

CHOIS. Input: a text description of the action, the target object's geometry, the human's initial pose, and a sparse set of target object waypoints (a full planned path through a named 3D scene for long sequences). Output: a dense human motion sequence together with an object trajectory that reaches the given waypoints. We evaluate all 404 released example sequences, each exactly 120 frames at 30 fps (4.0 s).

HOI-Diff. Input: a text description of the action and the target object's geometry, drawn from the BEHAVE test split by default. Output: a dense human motion sequence together with a full object trajectory, both generated by the model itself; no object waypoints are supplied externally. We evaluate all 32 released example sequences, ranging 40–55 frames at 20 fps (2.0–2.75 s, mean 2.6 s). HOI-Diff's own paper separately reports a 196-frame maximum training/architecture length; the 40–55-frame figures here are the actual lengths of its released example outputs, not that architectural maximum.

CHOIS's default conditioning is therefore substantially richer than HOI-Diff's, since it is additionally given the human's initial pose and target object waypoints rather than generating object motion from text alone. We supply whichever of these each method's own published scripts specify rather than aligning them to a common protocol ourselves, since neither baseline's public release exposes a mode that takes only the conditioning InterAssembly uses (a scene and a retrieved reference motion, with no external text prompt; the text descriptions of the motions are produced internally by the pipeline).

Methods Not Compared. We also considered CG-HOI (Diller & Dai, 2024) and InterDiff (Xu et al., 2023) but did not include either as a running baseline, for different reasons. CG-HOI has no public code or model release, so it cannot be reproduced. InterDiff does release code, but it targets a different task: short-horizon motion *forecasting* rather than task-conditioned generation. Its input is the first several frames (10 in the default configuration) of an already-captured interaction sequence, comprising human pose, object pose, and contact, drawn from its own test split, with no text or task specification of any kind; its output is a short predicted continuation of that same observed interaction (25 frames by default), optionally refined by a physics-informed correction stage, with longer sequences produced by autoregressively chaining repeated short predictions. Because InterDiff has no mechanism to specify a target action, object, or scene and can only continue a real clip it is already given, it cannot be pointed at an arbitrary task the way CHOIS, HOI-Diff, or InterAssembly can, so a like-for-like comparison is not possible without repurposing it well outside its intended use.

On Ablations. InterAssembly's stages are coupled by hard data dependency rather than being independent additive terms: retrieval selects the clip that retargeting deforms, and retargeting produces the joint trajectory MaskControl conditions on. Removing a stage outright, e.g., feeding MaskControl an un-retargeted trajectory, does not yield a weaker but still-functioning pipeline to compare against; it produces joint trajectories that are inconsistent with the adapted object placements, so the next stage cannot consume them and the pipeline does not complete rather than completing with degraded quality. This rules out a stage-removal ablation table, and we say so directly rather than presenting one that would not measure what it claims to. What we do report instead: Sec. D.5 measures how much the adaptation and synthesis stages change the retrieved clip (evidence they do

substantive, non-trivial work, not evidence that a specific design choice within them is the best one); Sec. D.6 characterizes where the current design falls short and why, tied to a specific stage; and Sec. 7.1 compares end-to-end output quality against baselines that make different design choices for the closest comparable substeps. None of these substitutes for a controlled ablation, and we do not claim otherwise.

Fairness Considerations. We highlight three points relevant to interpreting the comparisons in Sec. 7. First, InterAssembly targets a problem, long-horizon, multi-object activity assembly from an HOI database of short clips, that neither CHOIS nor HOI-Diff attempts; since no existing method operates in this exact setting, a fully matched comparison protocol across the entire task is inherently limited, and we treat the comparisons here as measuring per-clip motion and contact fidelity against the closest available generative baselines, not the long-horizon assembly capability itself. Second, we run both baselines strictly as published, without adding or removing any component of their pipelines; when a baseline's published example requires sparse object waypoints as conditioning, we supply the version the authors provide rather than withholding it. InterAssembly receives no external text prompt, but this is not a poorer-information regime overall: internally, it consumes the full retrieved object and human trajectories, an LLM-generated action description, and target-scene geometry, richer per-step conditioning than a short text string, just not directly comparable to it. We call out the waypoint choice specifically because it is the one input a baseline's own released configuration could plausibly have omitted in favor of a harder, text-only setting, and we chose not to make that substitution ourselves. Third, for the human evaluation in Sec. 7, we select InterAssembly outputs that share objects with the corresponding baseline output (e.g., floor lamp vs. floor lamp, chair vs. chair, box vs. box), and for multi-object interactions we ensure at least one major interacted object is shared between the compared outputs. The one exception is the diversity portion of the perceptual study, where we show the full suite of InterAssembly-generated results, which can include object categories a baseline's example set does not cover (e.g., a mug); this is intentional, since that portion of the study is designed to assess the breadth of generated content rather than matched per-object fairness.

D.2 Interpreting the Quantitative Comparison

HOI-Diff and CHOIS are designed for short-horizon, single-interaction synthesis, while InterAssembly targets long-horizon, multi-step assembly. We include them as the closest available generative HOI baselines on per-clip motion and contact quality; the goal is not to claim dominance on every metric, but to show that assembled sequences remain competitive on motion fidelity while supporting a broader task class.

The settings in Tab. 1 are unmatched rather than controlled: the methods differ in input modality, dataset, sequence length, object count, skeleton, and frame rate, so the numbers describe where each system lands on its own task rather than an isolated improvement. In particular, HOI-Diff is evaluated at 20 fps against 30 fps for CHOIS and InterAssembly. Since our Jerk metric is not fully frame-rate normalized (Sec. D.4), reported Jerk rises at lower frame rates, which inflates both of HOI-Diff's Jerk values by roughly $1.5\times$ relative to a 30 fps evaluation. This factor does not change the qualitative picture: HOI-Diff still has the lowest human Jerk among generation methods despite the inflation, and its object Jerk ($158.73\,m/s^3$) remains more than an order of magnitude above InterAssembly's ($6.13\,m/s^3$) even after accounting for it.

Our attribution of InterAssembly's lower object jerk and penetration to inheriting captured object trajectories through ARAP retargeting (Sec. 7.1) identifies a likely factor rather than an isolated, ablated cause, since scene complexity, sequence length, object count, and frame rate also differ between the settings. Likewise, whether a motion model trained on interaction-aware data closes the human motion gap would need to be verified experimentally rather than assumed. Finally, the $1.24\times$ nearest-neighbor ratio for synthesized transitions is a conservative estimate, given the descriptor and segment-length differences discussed in Sec. D.5.

D.3 Sequence Selection and Compute Cost

The 15 sequences reported in Sec. 7 are selected from 60 candidates. Full motion synthesis and rendering is the most expensive stage per candidate, so we do not carry every planned candidate

through it: candidates are screened by an automatic geometric-feasibility filter using the scene- and motion-penetration signal underlying the Penetration metric (Sec. D.4), and we report results on the candidates that passed this filter and completed the full pipeline within the compute and rendering budget available for this evaluation. We do not have aggregate statistics for the candidates that did not reach this point, so the numbers in Tab. 1 characterize InterAssembly's output quality on the sequences we were able to fully process end-to-end, not an unconditional success rate over all 60 planned sequences.

Across all 60 candidates, the agentic planning stages (action proposal, candidate selection, and iterative placement) use approximately $82 in total LLM token cost (Gemini-3-flash-preview), averaging about $1.4 per candidate sequence. This figure covers only LLM tokens: it excludes the ARAP retargeting and MaskControl motion-synthesis compute, which we run on a single NVIDIA H200 GPU and which is the more expensive stage per candidate that passes the automatic feasibility filter. The low LLM cost reflects the pipeline's reliance on retrieval and geometric optimization rather than repeated LLM-driven generation, and suggests that scaling the candidate pool itself (before GPU-side synthesis) remains inexpensive; we do not have comparably granular GPU-time accounting to report per-candidate.

D.4 Evaluation Metrics

We evaluate generated animations along two axes: motion quality and interaction plausibility. The metrics below capture complementary properties; no single value should be interpreted in isolation, as each reflects a combination of motion characteristics and scene complexity.

Human and Object Motion. InterAssembly and CHOIS are evaluated at their native 30 fps; HOI-Diff at its native 20 fps where outputs are not resampled to a common rate. Human jerk is averaged over all body joints and all frames; object jerk uses the translation of the object's origin (not its vertices).

Foot Sliding penalizes unphysical ground-contact slippage. For each foot joint j whose height falls below a threshold H_j (ankles, $H_j = 8$ cm; toes, $H_j = 4$ cm), the per-joint sliding score is

$$\text{Slide}_j = \frac{1}{N_f} \sum_{t \in \mathcal{G}_j} \left\| \mathbf{p}^{xy}_{j,t+1} - \mathbf{p}^{xy}_{j,t} \right\|_2 \cdot \left(2 - 2^{\mathbf{p}^{z}_{j,t}/H_j}\right), \tag{7}$$

where $\mathbf{p}_{j,t}$ is the world-space position of joint j at frame t (after subtracting the estimated floor height, itself found by clustering low-velocity toe heights, following (Li et al., 2024)), $\mathcal{G}_j = \{t : \mathbf{p}^{z}_{j,t} < H_j\}$ is the set of ground-contact frames, and N_f is the sequence length; dividing by the full sequence length N_f rather than $|\mathcal{G}_j|$ means the score also reflects how much of the sequence involves ground contact at all, not only the severity of sliding when contact occurs. The exponential factor smoothly down-weights frames as the foot lifts from the surface. We average across the four foot joints and report the resulting per-sequence score in centimeters. Lower scores correspond to more natural locomotion. This formula also follows the BEHAVE evaluation protocol (Bhatnagar et al., 2022). Because the underlying per-frame displacement and floor-contact threshold are not themselves normalized by frame rate, this metric is not strictly frame-rate invariant: HOI-Diff's lower native frame rate (20 fps vs. 30 fps for CHOIS and InterAssembly) is expected to inflate its reported value somewhat independent of true motion quality.

Jerk measures temporal smoothness via the discrete rate of change of joint acceleration. For a joint trajectory $\mathbf{p}$ of length N_f at frame rate fps:

$$\mathbf{v}_t = (\mathbf{p}_{t+1} - \mathbf{p}_t) \cdot \text{fps}, \quad \mathbf{a}_t = (\mathbf{v}_{t+1} - \mathbf{v}_t) \cdot \text{fps}, \quad \text{Jerk} = \frac{1}{N_f - 3} \sum_{t=1}^{N_f - 3} \|\mathbf{a}_{t+1} - \mathbf{a}_t\|_2. \tag{8}$$

Velocity and acceleration are each properly rescaled by the frame rate to approximate physical units (m/s, m/s^2), but the final acceleration-difference term is not rescaled by a further factor of fps, so it approximates true jerk (m/s^3) divided by fps rather than true jerk itself. Consequently, Jerk as reported is not frame-rate invariant: at a fixed underlying motion quality, evaluating the same method at a lower frame rate would raise its reported Jerk. Since HOI-Diff is evaluated at 20 fps against 30 fps for CHOIS and InterAssembly (Sec. 7), both its human and object Jerk in Tab. 1 are

inflated by this effect; its human Jerk is nonetheless the lowest among generation methods (Sec. D.2). Smaller values indicate smoother motion sequences at a given frame rate.

Interaction Quality. **Penetration** evaluates geometric plausibility by measuring how deeply object surfaces intrude into the human body mesh. For each frame, we take every vertex of the object's own mesh (no additional resampling to a fixed density) as $\mathcal{P}_{\text{obj}}$ and compute:

$$\text{Pen}(t) = \max_{p \in \mathcal{P}_{\text{obj}}} d(p, \mathcal{M}_h)\,, \tag{9}$$

where $d(p, \mathcal{M}_h)$ is the unsigned distance from p to the human mesh $\mathcal{M}_h$, taken only over points p that test as inside $\mathcal{M}_h$ (zero otherwise). We report the mean over frames of this per-frame maximum penetration depth, in meters, converted to centimeters in Tab. 1; lower values indicate more physically valid interactions. This metric is one-sided: it detects object vertices inside the human mesh but not human vertices inside an object, so it can miss configurations where an object's interior is instead penetrated by the body.

Jerk is computed for both human and object trajectories. Foot Sliding is applied exclusively to human motion. Penetration quantifies human-object interaction quality. Because each metric conflates multiple factors, we recommend interpreting them jointly rather than optimizing any single score.

D.5 Retrieval-vs-Generation Similarity Analysis

A natural concern for a pipeline built on clip retrieval is how much of the output is genuinely synthesized versus replayed verbatim from the HOI database. We quantify this directly using data the pipeline already computes internally, across 14 of the 15 sequences reported in Sec. 7 (7–12 actions each, spanning objects including pans, knives, vases, chairs, lamps, side tables, and trash cans); one sequence is excluded for missing intermediate logs. Figure 6 and Fig. 7 render a subset of these sequences for qualitative inspection, not a separate set.

At each action step, InterAssembly retrieves the best-matching clip, rigidly aligns it to the current scene, and adapts it to the actual object placement via the interaction-mesh retargeting step described in Sec. 5; for 10 of the 14 sequences it additionally synthesizes inbetween transitions bridging consecutive actions (the remaining 4 use direct pose interpolation and are excluded from the transition analysis). We measure the divergence introduced at each of these steps relative to the retrieved clip, before any adaptation.

Adaptation reshapes the retrieved clip. We compute the mean pelvis-centered joint displacement between the aligned (pre-adaptation) and adapted (post-retargeting) human trajectory, both over all 52 Mixamo skeleton joints and over 10 anatomical landmarks (hips, head, hands, feet, forearms, shins) to avoid finger-joint noise. We similarly compute the divergence in relative object motion, i.e., translation and rotation after removing the frame-0 offset, which isolates changes to the motion pattern itself rather than the change in absolute scene placement (always large, since the scene differs). Table 3 reports mean joint divergence of 5.9 cm (all joints) and 5.3 cm (landmarks), with the largest per-sequence mean reaching 10.3 cm, and mean object position divergence of 19.6 cm and rotation divergence of 4.4°, reaching up to 182.8 cm and 90.3° in individual segments. The per-joint breakdown is consistent with genuine physical adaptation rather than noise: feet and toes change most (8.3–8.8 cm), since floor contact must be re-grounded to the current scene regardless of the retrieved clip's original floor level, followed by distal hand joints ($\sim$7 cm), consistent with reaching toward the object's actual placement in the new scene.

Retrieval alone does not explain the match. We also report the pipeline's own structural scene-similarity score S_{struct} (Sec. 4.2), an exponentiated pairwise layout cost between the current scene and each candidate clip's source scene, evaluated at the moment of retrieval. Its wide distribution (0.73 ± 0.24, ranging down to 0.001) shows that retrieval followed by LLM selection routinely picks clips from scene configurations that differ substantially from the target, rather than only succeeding when a near-identical layout happens to already exist in the database. Even for the sequence with the highest mean score ($S_{\text{struct}} = 0.95$), the adaptation step still measurably reshapes the clip (5.0 cm joint divergence), confirming that retrieval is finding semantically appropriate motion, not pre-matched geometry that requires no further work.

Table 3: **Retrieval-vs-generation similarity analysis** (Sec. D.5). Divergence and distance metrics are measured relative to the retrieved database clip; larger values indicate more adaptation or synthesis beyond retrieval. S_{struct} is the scene-layout similarity score at the moment of retrieval; a wide distribution indicates the retrieved clips are not limited to near-identical layouts.

Metric	InterAssembly
Joint divergence, all joints (cm)	5.9 ± 4.7
Joint divergence, landmarks (cm)	5.3 ± 4.1
Object position divergence (cm)	19.6 ± 38.9
Object rotation divergence (°)	4.4 ± 13.4
Inbetween / aligned-retrieval NN-distance ratio	$1.24\times$
Scene-layout similarity at retrieval S_{struct}	0.73 ± 0.24

Inbetween transitions are not in the database. The strongest evidence against retrieval-playback comes from the synthesized inbetween segments. For each inbetween segment, and, as a calibration baseline, for each aligned (unadapted) retrieved action segment, we compute its nearest-neighbor distance to all 735 clips in the HOI database, using a pelvis-centered mean-joint-position descriptor averaged over the segment. The calibration confirms the descriptor is discriminative: the minimum aligned-retrieval distance is 0.01, corresponding to a segment that is, correctly, recovered as an almost verbatim database match. Every inbetween segment, in contrast, has a nearest-neighbor distance of at least 0.49, so none is a near-verbatim database match, and the mean inbetween distance (1.09) is $1.24\times$ the mean aligned-retrieval distance (0.88). Since this descriptor is a coarse mean-pose vector and inbetween segments are shorter than action segments (10–60 vs. 40–300 frames, which biases the comparison toward smaller measured distances), this ratio is a conservative estimate of how far the synthesized transitions actually are from anything already in the database.

Taken together, these results support the same conclusion from complementary angles: the scenes the pipeline retrieves for are frequently unlike the clip's original context, the adaptation step makes physically meaningful changes to fit the current scene, and the transitions bridging actions are measurably distinct from anything in the 735-clip HOI database. InterAssembly is not a retrieval-and-playback system; retrieval selects a semantically appropriate starting point, and the optimization and generation stages that follow do the work of making it fit.

D.6 Failure Case Analysis

We observe two main failure modes in InterAssembly, illustrated in Fig. 9, and additionally discuss artifacts arising from the lack of interaction-aware training data for motion synthesis. The first two stem from design choices that keep the pipeline modular and zero-shot, and all three point to well-defined extensions.

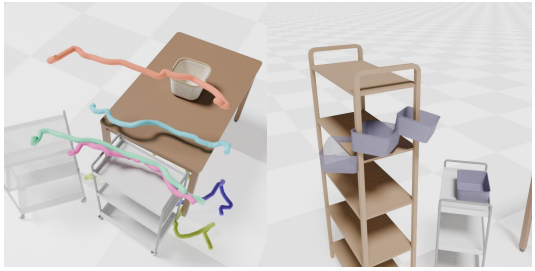


Figure 9: **Failure cases.** *Left*: limited scene awareness during motion synthesis; the body does not interact plausibly with an existing object that its major joints do not collide with. *Right*: residual penetration when an object must weave through another object.

Limited scene awareness during motion synthesis. Because the motion model is trained without scene or interaction data, its only direct scene signal is the SDF-based avoidance loss on the hands and feet, alongside the retargeted joint controls from the interaction mesh stage. Its awareness of the rest of the scene is therefore only as strong as these controls. Although we supply accurate retargeted motion, when the major controlled joints do not collide with an existing object, nothing in the synthesis stage signals that the rest of the body should respond to it, and the resulting motion can miss the intended interaction with that object (Fig. 9, left). Training the motion model on scene- and interaction-aware data, or extending the distance-field guidance beyond the hands and feet, would address this without changing the stages that precede it.

Collision-aware rather than collision-free placement. During iterative placement (Sec. 4.3), the agent receives penetration reports for both the approach path and the action trajectory, and colliding frames are correctly detected. These reports guide the agent's adjustments and keep penetration low overall (Tab. 1), but non-penetration is not enforced as a hard constraint in placement or retargeting, so the pipeline is collision-aware rather than collision-free. Residual penetration therefore remains in hard cases where an object must weave through another (Fig. 9, right). Such cases require a coordinated change to the trajectory within a narrow free space, which makes it especially hard for the agent to find an adjustment that is collision-free over the whole action. An explicit non-penetration term in the retargeting energy, or a lightweight physics-based projection after placement, would target these cases directly.

Artifacts from limited interaction-aware training data. Some artifacts arise at the synthesis stage because large-scale interaction-aware motion data is not yet available for training controllable motion models. The model we use is trained on body-only motion without objects, and on short, simple action descriptions rather than the compositional descriptions our planner produces. When conditioned on dense, interaction-driven joint controls from retargeting together with such descriptions, it therefore has no learned prior for how the body moves during manipulation, which introduces foot sliding and jitter. These effects account for many of the artifacts visible in our supplementary video, and are consistent with the foot sliding and jerk results in Tab. 1. They are orthogonal to the retrieval, adaptation, and retargeting stages that are the focus of this work. An additional refinement stage, such as foot-contact cleanup or physics-based motion tracking, or a motion model trained on interaction-aware data with richer text, would reduce them directly; since every stage of InterAssembly is swappable, such improvements can be adopted without changing the rest of the pipeline.

E Full Prompt Templates

Section C.5 summarizes the input/output structure shared by the agent's three LLM-driven stages (**Propose**, **Select**, and **Iterative Placement**); this section provides the complete, verbatim prompt templates for each, including system instructions, the user-prompt assembly, tool schemas, parsing/retry rules, and LLM sampling parameters. `{curly-brace}` placeholders denote values substituted at runtime from the scene state and memory string.

Shared Parsing and Retry Policy. All LLM calls in this pipeline use the same error-handling wrapper. Responses are parsed as JSON with LangChain's `JsonOutputParser`. On a JSON decode failure, the raw response is appended to the conversation with the instruction "Please respond with ONLY a valid JSON object"; this corrective loop runs for up to 5 attempts. Transient API errors (5xx, 429) trigger exponential back-off (base delay 2 s, doubled each attempt) for up to 5 retries. All three stages share the same model: `gemini-3-flash-preview` (Gemini 3 Flash Preview via Vertex AI), temperature 0.7, max tokens: model default.

E.1 Propose Prompt

System Prompt — Propose

```
You are an AI assistant that helps plan the next action in a scene sequence.
Given the history of actions and the current scene state, you need to propose:
1. Which objects should be interacted with in the next step
2. What the next action should be (a natural language description,
   can be empty if you're unsure)

Consider:
- Whether the current action is completed or needs continuation
- What objects are available in the scene
- The logical progression from previous actions
- Spatial relationships and object positions

IMPORTANT: The "interacted_objects" list must be ordered by importance:
- Primary objects that must be interacted with should come first
- Supporting objects (e.g., tables, surfaces) should be included but
  placed later in the list
- The order reflects priority: earlier objects are more critical for
```

```
  the action
- Object names should be in the exact form, e.g. if you see "table.001",
  use "table.001" in the exact form.

You can propose objects without specifying an action if you're unsure
what specific action to take. In that case, leave "next_action" as an
empty string, and the system will show you available actions involving
those objects for you to choose from.

Respond with a JSON object containing:
- "interacted_objects": a list of object names ordered by importance
  (most important first, can not be empty)
- "next_action": a natural language description of the proposed next
  action (can be empty string if unsure)
```

User Prompt Template — Propose

```
History of actions performed:
{memory}

Current scene information:
Objects in scene: {objects_in_scene}
Current object positions and relationships: Object positions are the
same as the last action in the history.

Based on this information, propose what objects should be interacted
with next and what the next action should be.
```

Output Schema. The parsed object must contain `interacted_objects` (list of strings) and `next_action` (string); missing keys default to an empty list / empty string.

E.2 SELECT PROMPT

Two wording notes on the verbatim prompt below, reproduced as actually sent: "goal of the task" in the Proposed Action bullet refers to the single next sub-action proposed by the preceding Propose call, not a persistent, multi-step activity goal, none is ever supplied to the agent (Sec. 3); and no step in this prompt takes human input, despite its internal use of "user" framing inherited from the calling harness's conversation-role convention.

System Prompt — Select

```
Your goal is to ground a high-level action proposal into a specific motion
sequence (candidate) while maintaining rigorous spatial consistency. You must
determine which objects from a candidate sequence should be "grounded" into
the scene, recognizing that spatial validation metrics are only provided for
a specific subset of those objects.

### Input Context

* **Scene History and State**: A record of previously executed actions and
  the current environment state, including object identifiers, global
  positions, and spatial relationships.
* **Proposed Objects**: The specific set of objects intended for manipulation,
  as output by the preceding Propose call.
* **Proposed Action**: The high-level intent or goal of the task.
* **Candidate Action Sequences**: A list of potential motion templates.
  Each candidate includes:
* **Motion/Setting Semantic Description**: Textual details about the object
  motion and layout.
* **Structural Similarity Score** $s \in [0, 1]$: Computed **exclusively
  for the Proposed Objects**.
* **Alignment Similarity Score** $a \in [0, 1]$: Computed **exclusively
  for the Proposed Objects**. Alignment similarity is computed by aligning
  the candidate scene with the current scene via one of the proposed objects
  as an anchor, and then computing the similarity of the remaining objects.
  The alignment similarity with human as anchor is utterly important, it is
  the stepping stone for the next step.

### Step-by-Step Execution Logic
```

```
**Step 1: Semantic Logic and Score Interpretation**
Select a candidate by evaluating how its motion and internal logic satisfy
the Proposed Action. Use the Structural Similarity Score and Alignment
Similarity Score as a confidence metric for the **Proposed Objects**. Be
aware that even with a high score, auxiliary objects in the candidate may
not align with the current scene.

Evaluate not just whether the proposed action superficially matches the
scene, but whether it logically supports a meaningful next step at this
point in the task. Critically consider the candidate action's semantic
intent and current scene context; sometimes a re-interpretation of the
action or a different next step (such as completing unfinished transitions
or adjusting previous objects) is more appropriate than directly following
the candidate proposal.

When reviewing action history, avoid repeatedly manipulating the same object
unless further interaction is clearly justified to accomplish the task's
objective.

When it comes to bringing new objects into the scene: If you notice that the
agent is repeatedly interacting with only a small or limited set of objects
across multiple steps, this may indicate a stagnation. In such cases, you
should proactively consider extending the scene by introducing new objects
if this helps the task move forward. However, before switching focus to new
objects, ensure that manipulation or transitions involving existing objects
are logically and fully resolved.

**Step 2: Object Pruning and Grounding Risk Assessment**
Classify every object in the selected candidate into one of two tiers:

1. **Tier 1: Proposed Objects (Anchors)**: The targets identified at
   the prior Propose step. Spatially "safe"
   to include.
2. **Tier 2: Auxiliary Objects (High Risk)**: Objects present in the
   candidate but absent from the proposal.
* **The Exclusion Rule**: If an auxiliary object is not strictly required
  for the motion logic of the Proposed Objects, **exclude it**.
* **The Necessity Exception**: Include an auxiliary object only if the
  action is physically impossible without it (e.g., a secondary plate
  required for a transfer task).

**Step 3: Functional Identity and Instance Mapping**
Reconcile the pruned list of objects with the scene using a
**Identity-Based Mapping** rule:

* **Lexical Prerequisite**: An object from the sequence can only map to a
  scene object if their names match after removing suffixes (e.g.,
  "plate.001" and "plate" share the base name "plate").
* **Instance Disambiguation (The Identity Test)**: Do not assume identical
  base names imply the same physical instance. Use the candidate's
  **Motion/Setting Semantic Description** and the **Action History** to
  judge:
* **Continuation**: Is this the same object from a previous step (e.g.,
  picking up the same plate)? If yes, map to the existing scene object.
* **Distinct Instance**: Does the candidate describe a different spatial
  role or a new entity (e.g., a second plate located on the other side
  of the table)? If the candidate implies a separate instance, or if
  mapping to an existing object would cause a logical/spatial conflict,
  treat it as a **New Object**. Map it to an empty string "".

* **Conflict Prevention**: Never map a sequence object to a scene object
  if it would result in "teleporting" an existing scene entity to an
  unverified location described in the candidate sequence.

**Step 4: Follow-Along Objects**
After completing object mapping, check whether any object currently in the
scene is *passively contained in or attached to* one of the mapped carrier
objects, and would therefore be physically carried along during this action
-- even though it has no independent motion entry in the retrieved sequence.
A typical example: items (spoon, knife, bowl) sitting inside a basket that
is being lifted and moved.

Identify follow-along candidates by cross-checking:
- The action history / scene context: is the follower known to be inside
  or attached to the carrier from a prior action?
- The intended action: does the carrier is in the "movable objects" list?
  If it is in that list, it means there is motion on the carrier and the
  follower will be carried along (even sometimes the motion is not
  significant).
```

```
For each identified follower, record:
{"scene_object_name": "retrieved_object_name"} where retrieved_object_name
is the carrier's key in selected_objects (i.e., its retrieved sequence name,
**not** the mapped scene name). If no follow-along relationships exist,
return an empty dict {}.

**Step 5: Natural Language Grounding**
Synthesize a rewritten action description. This text must reflect only the
verified objects and the specific instances selected. If a new instance was
identified (e.g., "a second plate"), the description should clearly
distinguish it from existing scene objects.

### Response Specification

Return a JSON object:

* `reasoning`: A technical justification for the candidate selection and
  mapping decisions. **Critically, explain how you distinguished between
  existing scene instances and new instances** based on the candidate's
  setting description and functional roles.
* `selected_candidate_index`: Integer (0-based).
* `rewritten_action`: A precise natural language description of the
  grounded motion.
* `selected_objects`: A filtered list of the exact object names from the
  sequence.
* `object_mapping`: A dictionary mapping sequence object names to either a
  scene object name or "".
* `follow_along_mapping`: A dictionary mapping each follower's scene object
  name to the retrieved carrier name it rides along with. Format:
  {"scene_follower_name": "retrieved_carrier_name"}. Empty dict {} if none.
```

User Prompt Template — Select

```
History of actions performed:
{memory}

Current scene information:
Objects in scene: {objects_in_scene}
Current object positions: Object positions are the same as the last
action in the history.

Proposed objects: {proposed_objects}
Proposed next action: {proposed_action}
{action_note}

Candidate actions:
{candidates}

Tasks:
1. Select the best candidate that matches the proposed action and fits
   the scene context.
2. Review the selected candidate's objects list and determine which
   objects are actually needed (remove redundant/background objects).
3. For each needed object, check if it maps to an existing object in the
   scene (consider type, location, and context from memory). If multiple
   objects of the same type exist, use spatial relationships and action
   history to determine the correct mapping.
4. Rewrite the action description to accurately reflect what the selected
   sequence actually does.

Return all information in a single JSON response with the fields specified
in the system prompt.
```

Output Schema. The parsed object must contain `reasoning`, `selected_candidate_index` (0-based integer), `rewritten_action`, `selected_objects` (list), `object_mapping` (dict), and `follow_along_mapping` (dict; defaults to {} if absent). If `selected_candidate_index` is out of range the first candidate is selected as a fallback.

E.3 Iterative Placement Prompt

As described in Sec. C.5, this stage runs in two rounds per action – pre-retargeting and post-retargeting – that share the same prompt skeleton. The two verbatim templates below highlight where their inputs differ.

Shared Preamble: Coordinate System and Description Format

The following block is concatenated immediately before the stage-specific task section in every placement system prompt (the tool-reference block described in the next subsection precedes this one, so the full order is tool reference, then this coordinate block, then the stage-specific task):

Shared Preamble — Coordinate System & Description Format

```
## Coordinate System: Right-handed coordinate system with assumption
that we are looking at the scene from the front view:
- X-axis: Left (-X) to Right (+X).
- Y-axis: Front (-Y) to Back (+Y).
- Z-axis: Down (-Z) to Up (+Z).
- Human Facing: Z-axis rotation should be considered when determining
  the facing direction of the human. A 0 degree z-axis rotation of the
  pelvis means facing "Front" (along the -Y axis) and a 90 degree
  z-axis rotation of the pelvis means facing "Right" (along the +X axis).

## Description Format:
- location: [x, y, z] in meters
- rotation: [rx, ry, rz] in euler angles in degrees
- bounding_box: {"min_corner": [x, y, z], "max_corner": [x, y, z]}
  in meters
- object motion: [[x, y, z, rx, ry, rz], ...] in meters and euler
  angles in degrees sampled every 10 frames (1/3 seconds) including
  the start and end frame.
- human location is represented by the x-y coordinates of the pelvis
  and the z-axis rotation of the pelvis for the direction it is facing.
```

Shared Preamble: Available Tool Functions

Before the coordinate-system block, a tool-reference section generated at runtime from the tools' Python docstrings is prepended, exposing the scene-manipulation operations described in Sec. C.7. The section ends with a best-practices note advising the agent to prefer `align_relative_position` and `snap_downwards` over manual coordinate edits.

Pre-Retargeting Round

The prompt below instructs the LLM that non-penetration is a hard rule, reproduced verbatim as actually sent. This instruction is not backed by a programmatic acceptance gate on our side: the placement loop still terminates once its fixed iteration budget (Sec. B) is reached regardless of whether the rule is satisfied, so in practice the mechanism is collision-aware rather than collision-free (Sec. D.6).

System Prompt — Pre-Retargeting (stage-specific)

```
## Your Task

Your task is to make **minimal adjustments** to the current positions of
`Movable Objects` so that the **reference motion can be reasonably
retargeted onto them**.

* **What "retargeting" means:** The reference motion is a recorded
  trajectory (position + rotation over time) of how the object moved in
  a prior, similar interaction. That trajectory will be *replayed starting
  from the pose you set here*. Your job is to ensure the starting pose
  makes that replay physically and semantically valid.
* **Goal:** Find the smallest adjustment to the current pose such that,
  when the reference motion is applied from that pose, the resulting
  interaction is reasonable -- the object doesn't clip through geometry,
  the human can reach it, and the motion makes sense in context.
* **What you do NOT need to do:** You do NOT need to exactly match the
  reference starting position. Full positional alignment is neither
```

```
  expected nor necessary. What matters is that the *type of
  configuration* (support surface, reachability, orientation relative to
  the human) is compatible with the motion.
* **Constraint:** Keep adjustments **small**. Do not relocate objects to
  a different area of the scene. Preserve the general layout from prior
  interactions.
* **Hard rule -- no penetration:** The starting pose and every frame of
  the resulting trajectory **must not penetrate any static object**. An
  object embedded in a wall, floor, table, or any other `Existing Object`
  -- even partially -- is never acceptable. This constraint takes priority
  over minimality: if avoiding penetration requires a larger adjustment,
  make it.
* **Scope:** You are refining the starting state (t=0) of objects that
  already exist in the scene. You are NOT placing new objects.

## Input

### General Information

**Interaction History:** Descriptions of history interactions and how
objects are placed and moved.

**Existing Objects:** Descriptions of all objects in the scene.
```json
{
    "object_name": {
        "description": "Detailed description of the object",
        "size": [sx, sy, sz],
        "position_and_relationship": {
            # High-level relational constraints (e.g., 'cup on table')
        }
    },
    ...
}
```

Movable Objects: A list of objects **already in the scene** whose
starting poses may need minor adjustment.
These are the objects the human will interact with (pick up, push, open,
etc.) during the action.
Other objects in the scene will NOT move during the action. Treat them
as **hard constraints** (obstacles or support surfaces).

Reference Action and Intended Action:
A description of the reference action that the human will perform after
the objects are placed. And a description of the intended action that
the human will perform after the objects are placed. The intended action
may differ from the reference action and you need to keep this in mind
when making the adjustment.

Reference Object Motion:
The full motion trajectory of the movable objects from the retrieved
reference interaction (a sequence of poses over time). For any
reference-related tool calls (e.g., `align_objects_location(a, ref-a)`),
the tool will use the **start** position (first frame) of the reference
trajectory. This is **not a target to match exactly** -- treat the start
frame as evidence of what spatial configuration made the reference motion
work. Extract from it the preconditions that matter (see Step 3).

Operation Information

Current Scene: {current_scene}
The current coordinates of all objects in the scene. For `Movable
Objects`, this also includes two trajectory fields and two separate
penetration reports:

* **`preceding_trajectory`** -- the interpolated path from the object's
  initial scene placement (`start` pose, recorded before any
  adjustment in this step) to the **current proposed starting pose**.
  This is the pre-action inbetween transition that will be played before
  the action begins. It must not clip through any static geometry.
* **`potential_trajectory`** -- the sequence of poses (position +
  rotation over time) that will result when the reference motion is
  applied from the current starting pose. Use this trajectory to
  anticipate and detect physical problems *before* committing to a pose.
  This trajectory is sampled every 10 frames (1/3 seconds) including
  the start and end frame, the detailed penetration report is provided
  in penetration_potential_trajectory.
* **`penetration_preceding_trajectory`** -- collision report covering
```

```
  only the `preceding_trajectory` segment (the pre-action approach).
  If this reports a collision, the proposed starting pose causes the
  approach path to clip through geometry.
* **`penetration_potential_trajectory`** -- collision report covering
  only the `potential_trajectory` segment (the action motion). If this
  reports a collision, the proposed starting pose causes the action
  itself to clip through geometry.

## Core Steps

1. **Identify Movable Assets:**
* Read the `Movable Objects` list. These are the **only** objects you
  may transform.
* **Rule:** You may NOT move objects outside `Movable Objects`. All
  `Existing Objects` are fixed constraints.

2. **Understand the Reference Motion:**
* Read `Intended Action` and `Reference Object Motion` together. Ask:
  *what spatial preconditions must hold at t=0 for the reference motion
  to remain valid when replayed in this scene?*
* Focus on:
  - **Support surface:** Is the object resting on the correct type of
    surface (table, floor, shelf)?
  - **Human reachability:** Is the object within arm's reach from the
    human's expected starting position?
  - **Orientation:** Does the object face the right direction so the
    human can grasp or interact with it as the motion describes?
  - **Relative layout:** If multiple movable objects interact with each
    other in the motion, their relative positions must be preserved.

3. **Check the Current Pose and Trajectory:**
* Compare the current pose of each movable object against the
  preconditions identified in Step 2.
* If the current pose already satisfies all preconditions, **make no
  adjustment** -- leave the object as-is.
* Only adjust what is strictly necessary to satisfy a violated
  precondition.
* **Inspect `penetration_preceding_trajectory` and the
  `preceding_trajectory`** for each movable object. This covers the
  path the object travels *to reach* the proposed starting pose from
  its initial placement. Identify penetrations and adjust accordingly.
* **Inspect `penetration_potential_trajectory` and the
  `potential_trajectory`** for each movable object. This covers the
  action motion that will play starting from the proposed pose.

4. **Plan the Minimal Fix:**
* Determine the smallest translation and/or rotation that resolves the
  violated precondition.
* Do NOT drift toward the reference absolute position if the current
  position already satisfies the preconditions.
* Preserve the existing relational context.
* Respect `Existing Objects` as hard constraints.
* **Penetration check (mandatory):** After planning an adjustment,
  verify that both penetration reports are clear.

5. **Refine and Validate:**
* Check `Last Operation History` for your past actions.
* Check `Evaluation Result` if provided.
* Ask: "If the reference motion trajectory is applied from this
  adjusted pose, will the result be physically plausible and
  semantically correct?" If no, apply a further minimal correction.

6. **Finalize:**
* Submit function calls to adjust the objects.
* Set `"finished": false` to verify intermediate steps.
* Set `"finished": true` ONLY when all necessary adjustments are
  complete. If you call any function at this round, we will set it to
  false automatically.

## Response Format

'''json
{
    "functions": [
        {"function": "function_name",
         "params": {"params1": "params1", ...}},
        ...
    ],
    "stop_on_error": true,
```

```
    "finished": false
}
```

User Prompt — Turn 1, Pre-Retargeting (static context)

```
Interaction History: {interaction_history}
Existing Objects: {existing_objects}
Movable Objects: {movable_objects}
Reference Action (the original action description from the retrieved
reference): {reference_action}
Intended Action (the actual action to be performed -- may differ from
the reference): {intended_action}
Reference Object Motion: {reference_object_placement}
```

User Prompt — Subsequent Turns, Pre-Retargeting (dynamic scene state, appended each iteration)

```
Last Operation History: {last_operation_history}
Current Scene: {current_scene}
```

POST-RETARGETING ROUND

The same caveat applies to this round's "Hard rule – no penetration" instruction below: it is not backed by a programmatic acceptance gate (see the note before the pre-retargeting round above).

System Prompt — Post-Retargeting (stage-specific)

```
## Your Task

Your task is to make **minimal corrections** to the final resting
positions of `Movable Objects` after the reference motion has been
retargeted and applied, and then **decide how the correction should be
integrated** back into the motion.

* **What happened before this step:** A reference motion trajectory was
  copied and retargeted onto the `Movable Objects`. The `Current Scene`
  shows where those objects landed as a result, along with the full
  executed trajectory.
* **Why corrections are needed:** The retargeted motion was derived from
  a different scene. Geometric differences (surface heights, container
  dimensions, relative distances) between the reference scene and the
  current scene can cause the objects to land in physically or
  semantically incorrect states.
* **Goal:** Identify landing errors, apply the smallest possible
  correction to restore the intended semantic end state, and declare
  how that correction should be applied to the motion.
* **Hard rule -- no penetration:** The corrected end pose **must not
  penetrate any static object**.
* **Scope:** You are correcting the end state (t=end) of a motion that
  has already been executed. You are NOT re-planning the motion or
  repositioning objects for a new action.

## Input

### General Information

**Interaction History:** Descriptions of history interactions and how
objects are placed and moved.

**Existing Objects:** Descriptions of all objects in the scene.
```json
{
    "object_name": {
        "description": "Detailed description of the object",
        "size": [sx, sy, sz],
        "position_and_relationship": {
            # High-level relational constraints describing what
            # relationships should hold
        }
    },
```
```

```
    ...
}
```

Movable Objects: The objects whose positions may need correction after motion retargeting.
Other objects in the scene are static. Treat them as **hard constraints** (obstacles or support surfaces).

Reference Action and Intended Action:
A description of the reference action that the human will perform after the objects are placed. And a description of the intended action that the human will perform after the objects are placed. The intended action may differ from the reference action.

Reference Object Motion:
The full motion trajectory of the movable objects from the retrieved reference interaction. Use this entire trajectory as context to interpret what type of relationship or interaction was achieved. You should assume the reference objects are placed at the ending position of these trajectories, which can be used for placement tools that require a reference object.

Operation Information

Current Scene: {current_scene}
The current coordinates of all objects **after** motion retargeting. For `Movable Objects`, this includes two trajectory fields that let you preview both mode options before committing:

* **`executed_trajectory`** -- The raw retargeted trajectory the object actually followed during the motion. This is exactly what the motion looks like if you choose **`"inbetween"` mode**: the motion stays as-is and a short bridge segment is added from its end to wherever you place the object now.

* **`retarget_trajectory`** -- The trajectory that *would* be computed if you choose **`"retarget"` mode**: the original motion shape is discarded and replaced with a new path from the before-action pose to the object's **current position in the scene**. This field updates every round as you adjust the object's position.

Use these two fields to reason about which mode produces a better object motion.

Core Steps

1. **Identify Movable Assets:**
* Read the `Movable Objects` list. These are the **only** objects you may transform.
* **Rule:** You may NOT move objects outside `Movable Objects`.

2. **Determine the Intended End Relationships:**
* Read `Intended Action` and `Reference Object Motion` together to extract the semantic end state for each movable object.
* Common relationships: resting on a surface, contained in a volume, in contact/adjacent, correct orientation for rest.
* The `position_and_relationship` field in `Existing Objects` is a strong signal.

3. **Diagnose the Landing Error Using the Trajectory:**
* For each movable object, compare its current position against the intended end relationship.
* **Inspect the executed trajectory** to understand how the object arrived at its current state:
  - **Persistent float:** trajectory shows hovering --> height error is a systematic retargeting offset.
  - **Geometry interference:** trajectory shows object passing through a static object --> lateral landing position may be wrong.
  - **Orientation drift:** trajectory shows progressive rotation away from expected rest orientation.
  - **Trajectory confirms valid landing:** make no adjustment.
* Classify the error: height mismatch, lateral drift, orientation error, or no error.

4. **Estimate Correction Magnitude:**
* Assess whether the correction is **large** (different surface, different container, major shift) or **small** (minor snap, height fix, orientation tweak). This informs the `adjust_mode` decision.

```
5. **Plan the Minimal Correction:**
* Fix only the diagnosed error. Do not adjust unbroken axes.
* **Penetration check (mandatory):** The corrected pose must not
  intersect any static geometry.

6. **Decide Adjust Mode (declare on final response only):**

Inspect both trajectory fields and decide how your correction should
be integrated:

* **`"retarget"`** -- Use when the `executed_trajectory` ends in the
  wrong place and the **motion path itself must change**. The code
  uses `retarget_trajectory` as the final motion.

* **`"inbetween"`** -- Use when the `executed_trajectory` motion shape
  is acceptable but the **landing point needs a small physical fix**.
  The code keeps `executed_trajectory` and adds a short bridge segment.

* **`"both"`** -- Use when the motion path needs to change **and** a
  further small settling correction is still needed after. A second
  placement call will follow.

* If you made **no adjustment**, use `"inbetween"` -- the bridge has
  zero displacement and is harmless.

7. **Refine and Validate:**
* Check `Last Operation History` and `Evaluation Result`.
* Ask: "Does the current configuration represent a physically valid
  instance of the intended end relationship?" If no, apply a further
  minimal correction.

8. **Finalize:**
* Submit function calls to correct the objects.
* Set `"finished": false` to verify intermediate steps.
* Set `"finished": true` ONLY when all corrections are complete.
  **When setting `finished: true`, you MUST include the `adjust_mode`
  field** declaring the mode for every movable object. If you call any
  function at this round, we will set `finished` to false automatically.

## Response Format

Intermediate rounds:
```json
{
    "functions": [
        {"function": "function_name",
         "params": {"params1": "params1", ...}},
        ...
    ],
    "stop_on_error": true,
    "finished": false
}
```

Final round (`finished: true`) -- `adjust_mode` is **required**:
```json
{
    "functions": [],
    "stop_on_error": true,
    "finished": true,
    "adjust_mode": {
        "object_name_1": "retarget",
        "object_name_2": "inbetween",
        "object_name_3": "both"
    }
}
```
```

User Prompt — Turn 1, Post-Retargeting (static context)

```
Interaction History: {interaction_history}
Existing Objects: {existing_objects}
Movable Objects: {movable_objects}
Reference Action (the original action description from the retrieved
reference): {reference_action}
Intended Action (the actual action to be performed -- may differ from
```

```
the reference): {intended_action}
Reference Object Motion: {reference_object_placement}
```

Note: if `follow_along` objects are present, the following note is appended to Turn 1:

Appended Note — Follow-Along Objects

```
Follow-Along Objects (passive riders -- do NOT reposition them, their
trajectory is derived automatically from the carrier):
{
  "scene_follower_name": "follows carrier: scene_carrier_name",
  ...
}
```

User Prompt — Subsequent Turns, Post-Retargeting

```
Last Operation History: {last_operation_history}
Current Scene: {current_scene}
```

Additionally, if a penetration is detected in the current scene, the following warning is appended:

Appended Warning — Penetration Detected

```
Please double check if the current scene is penetration valid.
Proceed with caution when there is a penetration warning.
```

SHARED DETAILS

Sampling Parameters and Loop Budget. Both the pre-retargeting and post-retargeting loops run for up to 10 iterations each; the `"both"`-mode secondary settling pass runs for up to 5 additional iterations.

Output Schema (tool-call format). Each turn the LLM must respond with a JSON object containing:

- `functions`: list of `{"function": name, "params": {...}}` dicts.
- `stop_on_error`: boolean; if `true`, the executor aborts on the first failing call.
- `finished`: boolean; the loop breaks only when the LLM sets this to `true` *and* no function call produced an error.
- `adjust_mode` (post-retargeting final turn only): dict mapping each movable object to `"retarget"`, `"inbetween"`, or `"both"`.

Each function call is dispatched by name through a registry; an unrecognised function name or a non-movable target returns an error string injected into the next turn's `last_operation_history`. When the LLM sets `finished: true` successfully, one additional self-evaluation turn is triggered with the instruction: "You have indicated that last placement should be the final placement. Therefore, for this run, your primary goal is to self-evaluate the entire placement process and ensure everything is satisfactory. If you find any issues, please call functions for corrections. Otherwise, return empty function call to indicate completion."

Penetration-Report Formatting. Penetration data is serialised per object as a dictionary with keys `penetration_preceding_trajectory` and `penetration_potential_trajectory` (pre-retargeting round) or `executed_trajectory` and `retarget_trajectory` (post-retargeting round). Each value is a human-readable string produced by SDF-based collision checking: `"no penetration"` if clear, otherwise a list of the penetrating static objects together with the maximum penetration depth (meters) and the frame indices where collisions occurred.

Validation Logic. After each function call batch, the executor checks whether the response JSON is well-formed and whether every `function` field resolves to a registered callable. Invalid calls return an error message; the loop continues so the LLM can self-correct. The loop terminates early (before `max_iterations`) only if the LLM returns `finished: true` with a successful (error-free) execution, followed by a clean self-evaluation turn.

`"both"`-Mode Secondary Settling Pass. When the post-retargeting round declares `adjust_mode = "both"` for an object, the system first recomputes that object's trajectory to the corrected end pose (equivalent to the `"retarget"` step), then runs a second short correction loop (up to 5 iterations) to apply any residual fine-grained settling that the recomputed trajectory may not have fully resolved. This pass uses the same user-prompt fields and tool schema as the post-retargeting round, with two differences: there is no `adjust_mode` field in the response (mode is already determined), and the scene description omits the trajectory-preview fields (`executed_trajectory` / `retarget_trajectory`) — only the final recomputed end pose is shown. The system prompt replaces the post-retargeting task description with a shorter settling-only variant whose key constraints are: *do not relocate the object to a different surface or semantic context; correct only residual physical errors (float, gap, tilt) at the current landing site*.