A IMPLEMENTATION DETAILS

In this section, we describe the detailed architecture of each module in our framework, along with the training configurations.

A.1 Motion Diffusion Module

The motion diffusion module employs a Transformer encoder architecture [Vaswani et al. 2017] with 8 layers and 16 attention heads, which has proven highly effective in modeling sequential data. The input to the model consists of tokens representing the noised body joints and additional tokens for the condition information. To achieve auto-regressive generation, we fix the first two tokens of the current segment, copy the value of the last two frames from the last segment, and zero out the noise applied to them, during both training and sampling. In addition to the body joint tokens, four other tokens are introduced to incorporate the conditions, representing the scene, text, pelvis, and hand goal location, which provide crucial context for generating coherent and relevant motions. The details of these conditioning tokens will be discussed in the following three subsections. An embedding of diffusion timestep is added to the four conditioning tokens to incorporate temporal information into the model. All tokens undergo a positional encoding before being fed into the Transformer model.

A.2 Scene Encoder

The current scene voxel and its predictive counterpart are concatenated along the channel dimension, creating a unified 64-channel, 32x32 image. This image is segmented into 8x8 patches, which serve as input for a ViT [Dosovitskiy et al. 2021] consisting of 6 layers and 16 attention heads. The ViT processes these patches and produces a 512-dimensional feature vector. This vector is then used as the scene conditioning token in the motion diffusion module, ensuring context-aware motion synthesis.

When the target action involves interactions unrelated to the scene, such as drinking water from a bottle or talking on the phone, we mask the scene conditional token as all zeros. This approach is intended to prevent interference from scene information during the generation process of scene-independent interactions.

A.3 Frame-embedded Text Encoder

We employ the CLIP encoder [Radford et al. 2021] to convert raw text descriptions into 768-dimensional latent vectors. These vectors are then transformed into 512-dimensional vectors using an MLP model. Simultaneously, a sinusoidal positional encoder converts the frame number from an integer to a 512-dimensional vector, forming the frame embedding. We then add the text embedding to the frame embedding and pass the result through another MLP layer to obtain the final text conditional token, the input for our motion diffusion module.

During the sampling phase, at each step of autoregressive generation, the frame number input to the model increases as the number of generated frames increases. This aligns with the increased rate during training. In particular, by controlling the rate at which the frame number increases, we can adjust the total duration of the generated action. Due to the periodic nature of locomotion, such as walking, we set the frame number to zero during both the training and sampling processes for locomotion.

A.4 Goal Encoder

We train separate MLPs to embed locomotion and hand goals, resulting in embeddings for pelvic and hand goals, respectively. For locomotion tasks, we remove the vertical component of the goal location and retain only the two-dimensional horizontal coordinates as input for the model. In object-reaching tasks, the target coordinates of the hand serve as input for the model. We mask the pelvis or hand goal tokens as all zeros for actions that do not involve locomotion or hand reaching.

A.5 Autonomous Scheduler

Our scheduler model generates a value ranging from 0 to 1, which indicates whether the previously generated motion clip has completed its entire semantic motion. This value subsequently determines whether the current motion clip should maintain the existing semantic or initiate the first motion clip of a new semantic. Our Scheduler model utilizes a Transformer encoder with 3 layers, 8 attention heads, and a hidden dimension of 512. We leverage a model identical to the structure described in Section 3.4 to embed the current frame number and the given language instruction as a text conditioning token. This token, along with other motion frames, serves as the input to the Transformer encoder.

Due to the simplicity of this task, we train the scheduler model on the entire LINGO dataset for only 5 epochs. We use a batch size of 1024 and a learning rate of 0.0001, employing the Adam optimizer with its default parameter settings. The model converges effectively and demonstrates strong performance.

A.6 Training Configuration

The training of our motion diffusion module is conducted with a carefully selected set of hyperparameters to ensure optimal performance. We utilize a learning rate of 0.0001. The number of diffusion timesteps is fixed at 100, balancing computational feasibility and the quality of generated samples. In addition, we adopt a linear noise schedule, which gradually increases noise levels throughout the diffusion process. We use 4 NVIDIA A100 GPUs to train over 500 epochs with a batch size of 1024, ensuring sufficient exposure to the training data for convergence. These hyperparameter choices are informed by prior literature and empirical experimentation.

B LINGO DATASET

In this section, we elaborate on the recording process of the LINGO dataset and statistics of the LINGO dataset in detail.

B.1 How LINGO Dataset Is Produced

Producing a VR-assisted motion-captured dataset is a complex process that involves multiple people, specialized equipment, and custom software. In this section, we provide an overview of the key components and steps involved in this process.

B.1.1 People Participants. The MoCap process involves three main roles.

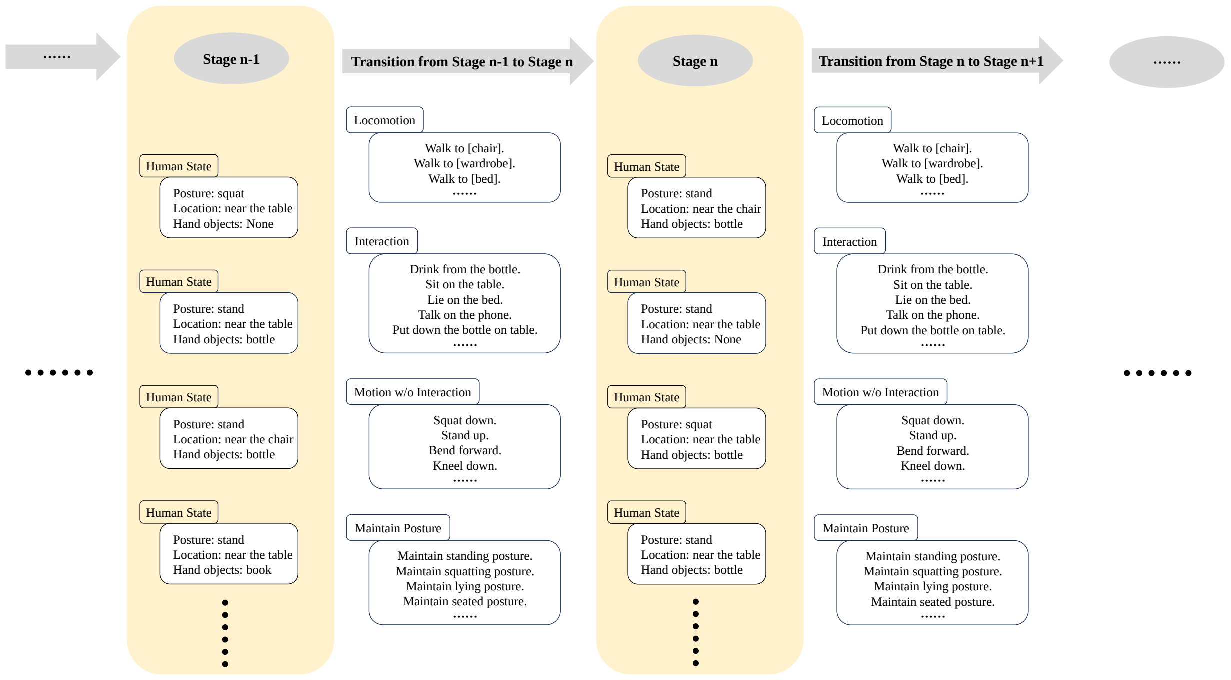

Fig. A1. **Motion Planner.** A Markov Chain generates the next instruction to guarantee plausible interaction and maintain a balanced distribution of motion types. The Motion Planner provides language instructions to the Actor.

Actor. The Actor performs the motions while wearing a MoCap suit and a VR headset.

Controller. The Controller provides the Actor with language descriptions of the motions to perform.

Assistant. The Assistant marks the frames when the Actor picks up or puts down hand-held objects, ensuring accurate synchronization between the motion data and the object interactions.

B.1.2 MoCap Add-on. We designed a custom Blender add-on to facilitate the MoCap process. This addon has three main functions. First, it displays the live motion of the Actor in the physical environment, superimposed on the virtual scene. This allows the production team to identify and correct errors such as penetration between the Actor's virtual representation and the scene objects or erroneous motion capture data. Second, the addon shows the Actor's VR headset view, which helps the team adjust the motion capture setup to suit the current motion type and Actor best. Third, the add-on provides a third-person view that follows the Actor's movement, similar to a third-person video game camera. This helps the Controller give orientation-related locomotion instructions, such as "walk to the right." The addon also links the VR and VICON systems, projecting rendered views and language instructions to the Actor and repeatedly aligning the real-world and virtual-world coordinates to maintain synchronization.

B.1.3 Motion Planner Add-on. Another custom addon, the Motion Planner (Figure A1), generates a sequence of instructions for the Actor to perform in the current scene as a Markov Chain. The input to the Motion Planner is a list of candidate interactions, their properties, and constraints. For example, some interactions may require the use of one or both hands or may have specific starting or ending positions. The Motion Planner considers these constraints and outputs a sequence of motion instructions that satisfy them. The Motion Planner also helps maintain a balanced distribution in the dataset. The Controller advances the instructions displayed in the Actor's VR headset by pressing a "go to next instruction" button.

B.1.4 Preparation. Before starting the MoCap process, the scene files are prepared in Blender. This involves selecting scene objects for the Actor to interact with and adding small interactable objects as needed. For sittable objects such as sofas and chairs, placeholders are placed in the physical environment to support the Actor during the capture session. The interaction types for each scene are specified and input to the Motion Planner. The VICON system is warmed up and calibrated to ensure accurate tracking. The VR headset is initialized by aligning the virtual world with the real world using a calibration procedure.

B.1.5 Motion Capturing. During the MoCap session, the Actor stands in a plausible location within the scene. The Controller checks that everything is ready and displays the first motion instruction in the Actor's headset. For interactive motions, such as picking up or manipulating objects, the Controller determines when the interaction is finished and advances to the next instruction. For grasping motions, the Assistant marks the time frames when the Actor grasps or releases the object, ensuring that these critical events are accurately recorded in the dataset and the object is attached to hands correctly. For locomotion, the Controller guides the Actor to the goal location using arrow keys, with specific direction-related information projected in the VR headset and recorded as annotations for the LINGO dataset.

During the MoCap process, the Actor performs the actions based on the language instructions projected on their VR headset. The VICON system tracks and records their body movements in real-time as the Actor moves and interacts within the physical space. Simultaneously, the captured motion data is instantly transmitted to the virtual scene, where the Actor's virtual body is rendered in real-time. This real-time reconstruction allows the Actor to see their virtual representation within the VR headset, creating a highly immersive and interactive experience.

B.1.6 Data Post-processing. After the MoCap session, the raw motion capture data is split into segments according to motion types. The Motion Planner Add-on accompanies each segment with a raw text annotation describing the performed motion. To enhance the richness and variety of annotations, GPT-4 is used to augment the raw text into multiple versions, providing alternative descriptions and additional context. We also double the data size by mirroring both the motions and the annotations.

B.2 Dataset Statistics

B.2.1 Interaction Types. The LINGO dataset covers 40 types of motion listed in Table A1, including non-interactive motions such as locomotion and maintaining posture. For interactive motions, the LINGO dataset contains interaction with static scene objects (*e.g.*, sit down, lie down) and small hand-held objects (*e.g.*, cellphone, gamepad). Figure 6 counts the number of occurrences of each motion type, and Figure A2 shows the distribution of the motion length. The detailed categorization is listed in Table A1.

B.2.2 Locomotion Goal Distribution. In the analysis of locomotion clips, we visualize the distribution of goal locations in Figure A3 represented in the canonical coordinate system of the first frame. This coordinate system is defined to align the character's initial orientation with the positive yaw direction. Using this consistent frame of reference, we can compare and study the relative positions of the goal locations across different clips. The plot represents the spatial distribution of the goal locations, with the origin (0, 0) corresponding to the character's starting position in the first frame. The plot's x-axis and y-axis represent the lateral and forward/backward directions, respectively, relative to the character's initial orientation.

B.2.3 Motion Length Distribution. Figure A2 presents the motion length distribution for various motion types in the LINGO dataset. Each violin represents a motion type, with the width of the violin indicating the density of data points at different motion lengths. The vertical axis measures the motion length, while the horizontal axis lists the motion types.

Table A1. **Motion types of LINGO.**

Motion Description	Motion Name
Move from one place to another by taking steps.	walk forward walk back walk front left walk front right
Change the orientation of the body.	turn left turn right
Change to a standing position.	stand up get up
Interact with hand-held objects.	pick up put down take photo turn on write type on read play[1] drink eat talk on listen to music brush teeth toss swing wave
Pick up and put down hand-held objects.	pick up put down
Stationary motions.	stand still maintain lie maintain sit maintain bend maintain kneel maintain squat
In-place motions.	bend forward straighten up kneel down squat down crawl
Interact with static scene objects.	sit down lie down punch kick wash take shower rotate play[2] type

[1] Play game and guitar. [2] Play drums and piano.

The motion lengths span from 1 to 12 seconds across all motion types. However, most of the data points lie between 2 and 6 seconds. The median motion length for most motion types is around 4-5 seconds. Some motions, like "walk forward," are close to a normal

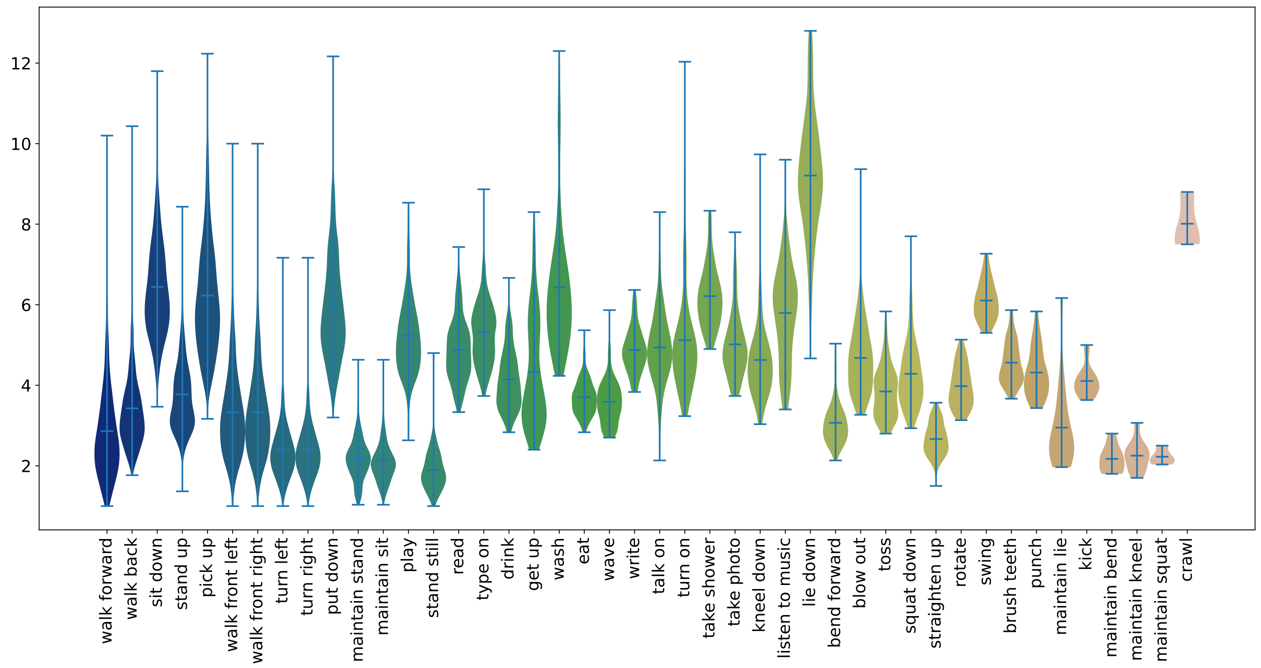

Fig. A2. **Motion length distribution of each motion type in LINGO dataset.**

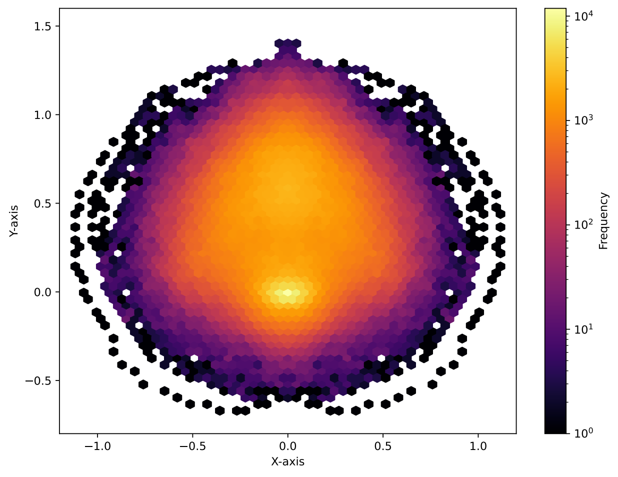

Fig. A3. **Distribution of goal locations for all locomotion clips in the local coordinate system of the first frame.** The character is aligned to initially face the y-axis direction. Unit: meter.

distribution, while others, such as "sit down" or "stand up," have a longer tail towards higher motion lengths due to the varied Actor preferences. Some motion types, such as "lie," have significantly longer motions. Motions with respect to locomotion, such as "walk forward" and "walk front left" have similar distributions, while interactive motions have distinct distributions compared to the rest.

B.2.4 Motion Occurance Count. Figure 6 displays the number of occurrences for each motion type in the LINGO dataset. The vertical axis represents the count on a logarithmic scale, while the horizontal axis lists the various motion types.

The number of occurrences varies across motion types. Locomotion-related motions have the highest number of occurrences, exceeding 1000 instances in the dataset. This is because locomotion occurs between two interaction motions. Most interaction motion types, such as "drink," "eat," and "wash," have around 200 occurrences or segments in the LINGO dataset.

C TASK PLANNER

We show a workable pipeline that leverages GPT4-o to break down complex instruction into sub-tasks.

```
  - prompt = "I need you to help me complete a task now. I will give you a target action. The target action is:"+action+"." You need to give in English a number of steps that I need to complete the target action. The steps should be as concise as possible without the need for irrelevant attributives. The details of the interaction action are not required. For example, there is no need to open the game console. The steps are divided into three categories: locomotion, grasp, and interaction, where locomotion only includes the movement of the person's position, Grasp only consists of the grabbing and touching of objects, and interaction includes people's operations on the appliance (such as listening to music with headphones, turning the door handle to open the door (excluding grabbing the door handle) )), indicate the hand when it comes to hand movements. Please complete this task according to some step examples I gave you. Example: "+str(text_list)+,"
```

output format: [{"step" :,"step_id":1,"category":},...], only output the final format, no other nonsense”
  - output = [{"step": "walk to the sofa", "step_id": 1, "category": "locomotion"}, {"step": "sit down on the sofa", "step_id": 2, "category": "locomotion"}, {"step": "pick up remote with left hand", "step_id": 3, "category": "grasp"}, {"step": "turn on TV with left hand", "step_id": 4, "category": "interaction"}, {"step": "watch TV", "step_id": 5, "category": "interaction"}]