Event-Driven Storytelling with Multiple Lifelike Humans in a 3D Scene

Supplementary Material

In this supplementary material, we provide additional details not covered in the main text (Section A). We also present further experimental results (Section B) and visualization of our user study (Section C).

A. Further Details

A.1. Runtime Logic

Algorithm 1 illustrates the high-level runtime logic of our framework. In the preprocessing stage (Line 4-6), our framework extracts the scene graph $\mathcal{G}$ from the 3D scene $\mathcal{S}$ and generates a scene description $\mathcal{D}$ using the scene describer. The generated scene description $\mathcal{D}$ is then used repeatedly by the narrator and event parser during runtime. In the main runtime loop (Line 7-22), the framework first checks if a new event is required. A new event is created only when there are characters that are not assigned to any ongoing event. If a new event is required (Line 9-13), the narrator generates a new event by determining who should be involved among those characters and what activity they should perform. The event parser then parses the generated event, and our framework assigns the event and its parsed information (target position p_i, orientation d_i, and action label a_i) to the characters involved in the event. After the behavior planning of the action planning module, the motion synthesis module advances the characters' motions respectively, based on their assigned events (Line 15-21). If a character is on the move to its target position, the motion synthesis module periodically updates the character's collision-free path to follow using the windowed cooperative A^* algorithm [10]. A character's motion is advanced by synthesizing the next frame of the motion using the motion matching algorithm [2] based on its current state and assigned action label. The state of each character is maintained internally to manage the progress of the assigned event and to determine the type of motion to synthesize. For example, if a character is approaching the target position (*approaching* state), locomotion following the planned path is synthesized. But if a character is during an interaction after reaching the target position (*interacting* state), a corresponding interaction motion is synthesized according to the assigned action label. In our framework implementation, we define five states: *idle*, *approaching*, *interacting*, *transition_in* (standing to sitting), and *transition_out* (sitting to standing).

A.2. Scene Graph Construction

In the preprocessing stage, the scene describer generates a textual description $\mathcal{D}$ of the scene $\mathcal{S}$ based on the 3D scene graph. In this section, we detail the construction of the scene graph below.

Algorithm 1 High-Level Runtime Logic of the Framework

```
 1: Required: 3D scene $\mathcal{S}$, characters $\mathcal{C}$
 2: Optional: user instructions $\mathcal{T}$
 3:
 4: Create the 2D grid map of $\mathcal{S}$
 5: Extract the scene graph $\mathcal{G}$ from $\mathcal{S}$
 6: Generate a scene description $\mathcal{D}$          ▷ scene describer
 7: while Framework is running do
 8:     Check if a new event is required
 9:     if A new event is required then
10:         Generate a new event                         ▷ narrator
11:         Parse the event                              ▷ event parser
12:         Allocate the event to the associated characters
13:     end if
14:
15:     for Each character $c_i$ in the scene do
16:         if $c_i$ is on the move to the target position then
17:             Update $c_i$'s collision-free path
18:         end if
19:         Advance $c_i$'s motion
20:         Update $c_i$'s state
21:     end for
22: end while
```

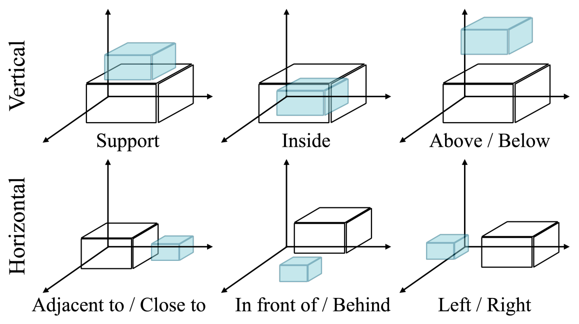


Figure 10. Spatial relationships used in the scene graph construction.

To construct a scene graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ from the segmented objects in the 3D scene $\mathcal{S}$, we follow the automated scene graph construction pipeline proposed in [7], but with a more simplified list of spatial relationships. At first, we initialize the nodes $\mathcal{V}$ with the segmented objects in the 3D scene. For each object, we compute the z-axis aligned 3D bounding box $b_i = \{p_i^1, p_i^2, ..., p_i^8\} \in \mathbb{R}^{8\times 3}$ of the object, where the p_i^j $(j \in \{1, ..., 8\})$ is a vertex composing the b_i, and estimate the orientation $d_i \in \mathbb{R}^2$ using the geometric heuristics proposed in [11]. After the nodes are initialized, we traverse the nodes and compute their spatial relationships to construct the edges $\mathcal{E}$. The spatial relationships are categorized into two types: *vertical* and *horizontal* relationships, the full list of which is provided in Figure 10. To avoid the explosion of the number of edges, we first determine the support level of each object based on the *Support* relationships, and limit the spatial relationships based on the support level. Starting from the support level zero objects, which are directly supported by the floor, if an object $\mathcal{O}_i$ is supported by another object $\mathcal{O}_j$, the support level of $\mathcal{O}_i$ is defined as the support level of $\mathcal{O}_j$ plus one. Those objects that are not supported by any other objects are defined as the hangable objects. We allow the horizontal relationships to be computed only between objects with the same support level, and compute *Above/Below* relationships only for the hangable objects. All the spatial relationships are heuristically computed based on the relative distances and orientations of the 3D bounding boxes of the objects. For further details of the spatial relationship computation, please refer to our released code.

A.3. High-level Action Planning Module

A.3.1. Scene Describer

In our system, the scene describer takes a scene graph, extracted from the 3D scene and converted into JSON format, as input and transforms it into a context-centric scene description. We provide object cluster information to help the scene describer better recognize regional context from the given 3D scene. For object clustering, we apply the DBSCAN algorithm [3] to objects present in the scene. The distance between objects is computed in 3D space as the distance between their bounding boxes. The key parameters of the DBSCAN algorithm, eps and minimum samples required to form a dense region, are set to 1.0 and 2, respectively. In Figure 11, we present examples of how our scene describer extracts key interesting areas from unseen scenes.

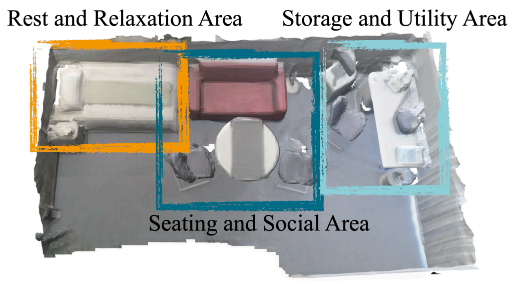


Figure 11. Key area extraction on MPH8 from PROX dataset [4].

A.3.2. Narrator

The narrator performs multi-agent behavior planning on a given scene based on semi-narrative events. The narrator generates new events only for characters not assigned to an 'ongoing' event in the current scene. If the LLM fails to follow this rule correctly, it receives feedback identifying characters that should not be included in the event and regenerates a corrected event based on this feedback. If all characters are engaged in ongoing events, the narrator does not generate new events and waits until a character completes their event.

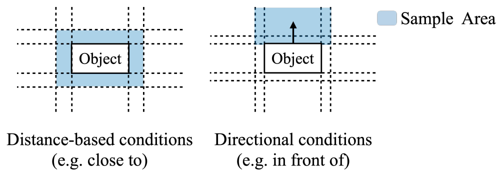


Figure 12. Semantic area representation examples.

A.3.3. Event Parser

The event parser utilizes programming-structured prompts and the area-conditioned position sampling method to parse events into low-level information. In the programming-structured prompt approach, we enable the event parser to use the following functions as spatial reasoning tools.

- `get_object_supporting(anchor)`
- `get_objects_supported_by(anchor)`
- `get_objects_in_front_of(anchor)`
- `get_objects_behind(anchor)`
- `get_objects_left_of(anchor)`
- `get_objects_right_of(anchor)`
- `get_objects_close_to(anchor)`
- `get_objects_associated_with(anchor)`
- `get_objects_between(anchor_1, anchor_2)`
- `get_closest_object(anchor)`
- `get_intersected_area(area_1, area_2)`
- `get_distance_between(object_1, object_2)`
- `is_object_occupied(object)`
- `is_object_of_label(object)`

Using these functions, the event parser can more easily retrieve objects and determine the appropriate area for character target position sampling based on the retrieved objects.

Our area-conditioned position sampling method enables the LLM to process a character's target position at a semantic level. To achieve this, the event parser is provided with the following area retrieval functions.

- `get_area_to_interact_with(object)`
- `get_area_to_sit_on(object)`
- `get_area_adjacent_to(object)`
- `get_area_close_to(object)`
- `get_area_in_front_of(object)`
- `get_area_behind(object)`
- `get_area_left_of(object)`
- `get_area_right_of(object)`
- `get_area_between(object_1, object_2)`
- `get_area_aligned_with(object_1, object_2)`

As shown in Figure 12, the event parser can meaningfully represent a character's target position without directly handling coordinate-level representations. Once an area is specified, the exact coordinates are sampled from within the area. The specific area size represented by each semantic expression, such as *close to*, is controlled by user hyperparameters.

A.4. Low-level Motion Synthesis Module

Our framework requires generating various types of motion to represent characters' daily life activities, including path-following locomotion, human-scene interaction motions, and human-human interaction motions. To efficiently cover these diverse motion types and generate stable motions in an online manner, we implement the motion synthesis module using the motion matching algorithm [2]. Our motion synthesis module utilizes SMPL-X [9] to represent character bodies and synthesize character animations at a frame rate of 30 fps.

A.4.1. Motion Database

Prior to motion synthesis, our framework defines a set of action labels, and we construct separate motion databases corresponding to each action label. Specifically, motions for daily life activities such as locomotion, drinking, eating, and laptop usage are collected from the AMASS dataset [8] and Mixamo [1]. Human-scene interaction motions like sitting and lying down are gathered from the SAMP dataset [5]. Human-human interaction motions, including chatting, hugging, and handshaking, are sourced from the Inter-X dataset [12]. All motions are downsampled initially to align with the 30 fps. Each action label has a dedicated motion database, allowing efficient database searching and the use of distinct matching features tailored to the characteristics of each action.

A.4.2. Motion Matching Details

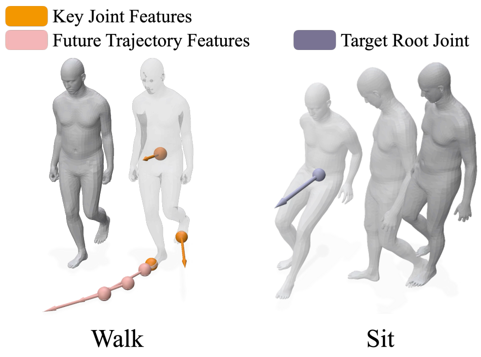


Figure 13. Matching features example.

Our motion synthesis module utilizes the following matching features:

- **Keyjoint Positions**: Positions of key joints J expressed in the character's local frame ($\mathbb{R}^{3J}$).
- **Keyjoint Velocities**: Velocities of key joints J in the local frame ($\mathbb{R}^{3J}$).
- **Future Positions**: Ground-projected 2D positions of the future trajectory (at 10, 20, and 30 frames ahead) in the character's local frame ($\mathbb{R}^6$).
- **Future Directions**: Ground-projected 2D facing directions of the future trajectory (at 10, 20, and 30 frames ahead) in the character's local frame ($\mathbb{R}^6$).
- **Relative Position**: 2D relative position of the character with respect to a specified target position ($\mathbb{R}^2$).
- **Relative Velocity**: 2D relative velocity of the character concerning a specified target position ($\mathbb{R}^2$).
- **Relative Direction**: 2D relative direction of the character towards a specified target direction ($\mathbb{R}^2$).
- **Target Root Height**: Height of the character's target root position ($\mathbb{R}^1$).

Keyjoint positions, keyjoint velocities, future positions, and future directions are all represented local to the character's root (pelvis) and facing direction.

As shown in Figure 13, when generating locomotion along a defined path, we employ keyjoint positions, keyjoint velocities, future positions, and future directions as matching features and `pelvis`, `spine3`, `right_foot` and `left_foot` as keyjoints. For human-scene and human-human interactions, we utilize relative position, velocity, and direction as primary matching features, with an additional target root height feature specifically included for human-scene interactions to ensure accurate sitting positions. For in-place activities such as eating and drinking, matching relies solely on keyjoint positions and velocities with `pelvis`, `spine3`, `right_wrist`, `left_wrist`, `right_foot` and `left_foot` as keyjoints. The pose vector structure and next-frame generation process for actual animation follow the methodologies presented in [6].

A.5. Benchmark

A.5.1. Test Scenes

In Figure 14, 15, and 16, we provide visualizations of our test scenes used in our benchmark. Each scene is designed to include two to three separate areas with distinct contextual meaning. The House scene is approximately $51.57m^2$ in size and was created by placing 23 objects from 14 different object categories. The Office scene is approximately $160.2m^2$ and includes 51 objects from 11 different object categories. The Restaurant scene is approximately $72.25m^2$ and consists of 39 objects from 11 different object categories.

A.5.2. Test Settings

Here, we provide additional details of our benchmark test settings. To address the randomness inherent in LLMs, we repeat each test case five times and average the results. We also set the temperature parameter, which affects output

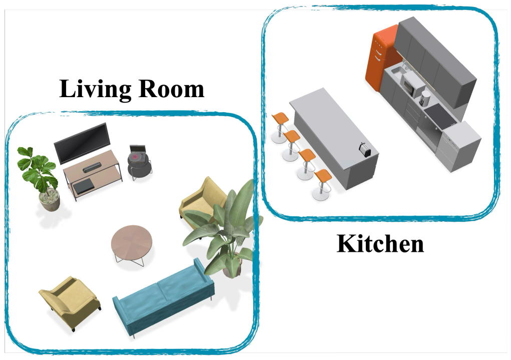


Figure 14. House scene.

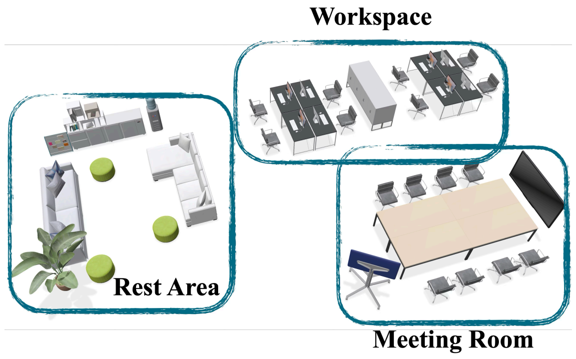


Figure 15. Office scene.

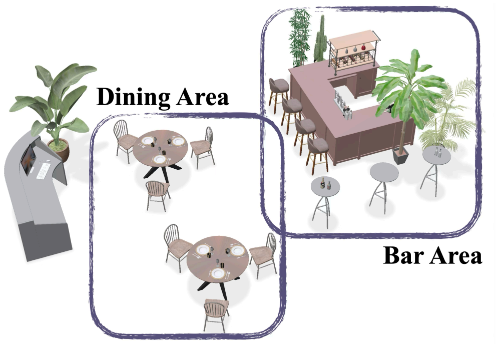


Figure 16. Restaurant scene.

variability, to 0.1 throughout all experiments. Such a low temperature value generally makes LLM responses more deterministic. The scene descriptions are pre-generated in five versions for each test scene using GPT-4o, and the corresponding version with the matching index of trials is fed into the action planning module, such that the nth trial observes nth scene description. The LLM planning module also observes input prompts with examples. We prepare a set of examples for each tag, and the system dynamically selects examples with the tags of the current test case. None of the examples include test scenes, ensuring that the LLM performs the benchmark in unseen environments.

B. Additional Results

B.1. Additional Benchmark Results

In Table 3 we present additional benchmark results that were not included in the main text. The results from the additional LLM models further confirm that our approach achieves the best overall performance in scene-aware multi-agent planning.

B.2. Experiment using Vision-Language Model

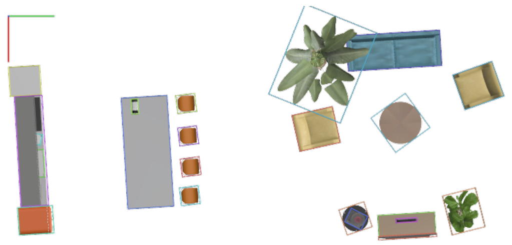


Figure 17. An example of a top-view image used in our VLM-based approaches.

We additionally conduct experiments on planning methods using vision-language models (VLMs).

Vision-based Description First, we evaluate how well a VLM utilizes visual information to generate high-quality scene descriptions. In the *Vision-based Description* approach, we maintain the existing planning pipeline but replace the scene description input for both the narrator and event parser with a vision-generated scene description. To achieve this, we modify the scene describer, which previously generated descriptions based on scene graphs. Instead, as shown in Figure 17, the updated scene describer generates detailed descriptions using top-view images along with object labels and position information.

Vision-based Planning In the *Vision-based Planning* approach, the narrator and event parser perceive scene information through visual inputs rather than textual scene descriptions. To enable this, we replace the scene description previously provided as input with a top-view image along with object labels and position information, allowing the system to perform planning based on visual data.

Benchmark results for VLM-based approaches Table 4 presents the benchmark results for the *Vision-based Description* and *Vision-based Planning* approaches described earlier. For these experiments, we use GPT-4o and GPT-4o Mini as foundation models capable of processing visual information. The benchmark settings remain the same as in original experiments.

Model	Llama-3.1-405B				Llama-3.3-70B				DeepSeek-V3			
Metrics	Total	OA	RC	SS	Total	OA	RC	SS	Total	OA	RC	SS
Ours	**0.66 (0.82)**	0.74 (0.84)	**0.71 (0.88)**	0.52 (0.68)	**0.74 (0.88)**	**0.8 (0.94)**	0.8 (0.95)	0.6 (0.77)	**0.83 (0.98)**	0.83 (0.96)	**0.87 (1.0)**	**0.82 (0.96)**
w/o Event	0.6 (0.71)	0.46 (0.58)	0.68 (0.77)	**0.6 (0.72)**	0.66 (0.83)	0.65 (0.79)	0.68 (0.86)	**0.65 (0.79)**	0.82 (0.95)	**0.89 (0.97)**	0.84 (0.95)	0.75 (0.9)
Object List	0.36 (0.71)	0.38 (0.69)	0.25 (0.79)	0.45 (0.7)	0.33 (0.65)	0.4 (0.72)	0.11 (0.49)	0.44 (0.68)	0.4 (0.81)	0.42 (0.81)	0.19 (0.76)	0.5 (0.79)
Scene Graph	0.65 (0.83)	**0.78 (0.88)**	0.68 (0.92)	0.52 (0.68)	0.72 (0.85)	0.68 (0.9)	**0.84 (0.92)**	0.57 (0.71)	0.76 (0.98)	0.83 (1.0)	0.73 (0.96)	0.76 (0.95)

Table 3. Additional benchmark result for test cases with *object arrangement reasoning (OA)*, *regional context reasoning (RC)*, and *scene state reasoning (SS)* tags.

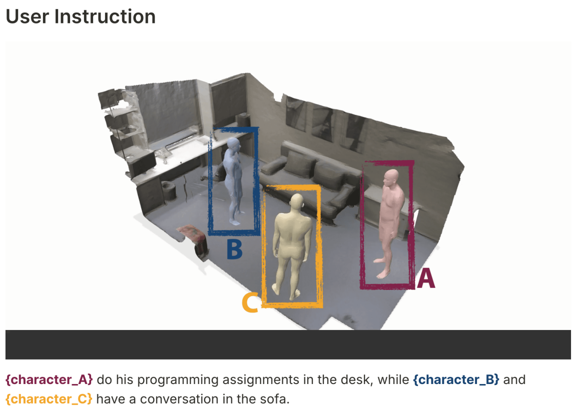
Model	GPT-4o				GPT-4o mini			
Metrics	Total	OA	RC	SS	Total	OA	RC	SS
Ours	**0.9 (0.98)**	**0.93 (0.99)**	**0.9 (0.98)**	**0.92 (0.98)**	**0.74 (0.96)**	**0.72 (0.95)**	**0.72 (0.97)**	**0.78 (0.93)**
Vision-based Description	0.68 (0.97)	0.66 (0.98)	0.63 (0.96)	0.76 (0.95)	0.48 (0.92)	0.41 (0.93)	0.47 (0.91)	0.53 (0.86)
Vision-based Planning	0.67 (0.91)	0.69 (0.92)	0.52 (0.87)	0.65 (0.86)	0.38 (0.86)	0.43 (0.86)	0.15 (0.81)	0.52 (0.84)

Table 4. Additional benchmark results for planning methods using vision-language models.

As shown in Table 4, vision-based planning methods perform far worse than our text-based approach. This suggests that more refined methodologies are needed to achieve effective planning through vision-based scene understanding. Further exploration in this area could lead to improvements in future work.

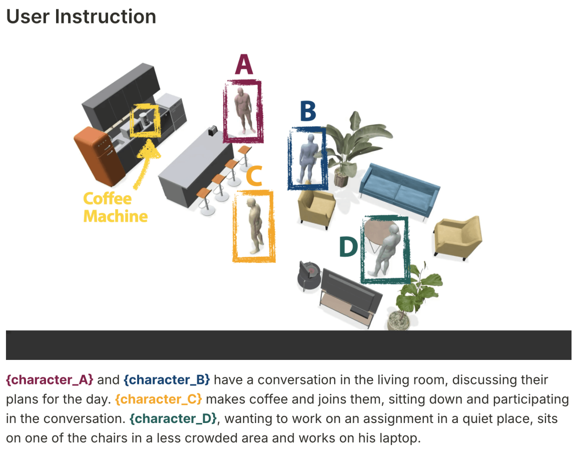
C. User Study

In this section, we present the test scenarios used in the user study, as shown in Figures 18, 19, 20, and 21. For each scenario, users are provided with full videos and snapshots of results generated from different ablation settings. They visually examine these results to identify any misrepresented events in the user instruction and ultimately select the outcome they find most accurate. We provide all full videos used in the user study in the supplementary video.



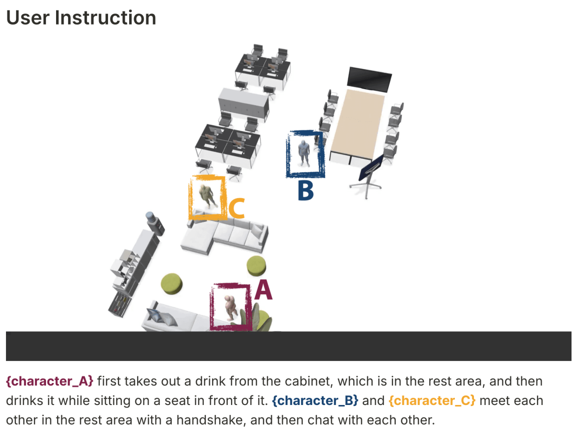
{character_A} do his programming assignments in the desk, while {character_B} and {character_C} have a conversation in the sofa.

Figure 18. Test scenario employed in the user study for the MPH11 scene.



{character_A} and {character_B} have a conversation in the living room, discussing their plans for the day. {character_C} makes coffee and joins them, sitting down and participating in the conversation. {character_D}, wanting to work on an assignment in a quiet place, sits on one of the chairs in a less crowded area and works on his laptop.

Figure 19. Test scenario employed in the user study for the House scene.



{character_A} first takes out a drink from the cabinet, which is in the rest area, and then drinks it while sitting on a seat in front of it. {character_B} and {character_C} meet each other in the rest area with a handshake, and then chat with each other.

Figure 20. Test scenario employed in the user study for the Office scene.

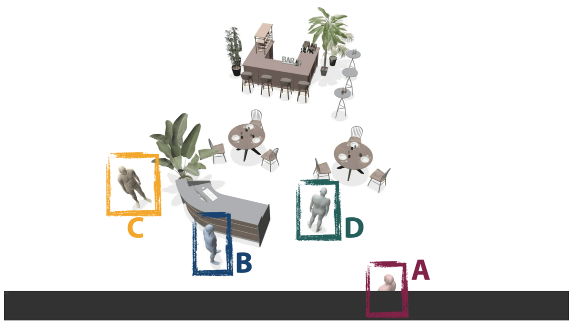
References

[1] Adobe. Mixamo. 3

[2] Simon Clavet et al. Motion matching and the road to next-gen animation. In *Proc. of GDC*, page 4, 2016. 1, 3

[3] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, pages 226–231, 1996. 2

User Instruction



{character_A} and {character_B} eat dinner separately, sitting at different tables in the dining area. {character_C} takes a phone call in front of the reception desk, and then joins {character_A}'s table to eat dinner together. On the other hand, {character_D} dances for a while next to a table in the bar area, then joins {character_B}'s.

Figure 21. Test scenario employed in the user study for the Restaurant scene.

[4] Mohamed Hassan, Vasileios Choutas, Dimitrios Tzionas, and Michael J Black. Resolving 3d human pose ambiguities with 3d scene constraints. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2282–2292, 2019. 2

[5] Mohamed Hassan, Duygu Ceylan, Ruben Villegas, Jun Saito, Jimei Yang, Yi Zhou, and Michael J Black. Stochastic scene-aware motion prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11374–11384, 2021. 3

[6] Daniel Holden, Oussama Kanoun, Maksym Perepichka, and Tiberiu Popa. Learned motion matching. *ACM Transactions on Graphics (ToG)*, 39(4):53–1, 2020. 3

[7] Baoxiong Jia, Yixin Chen, Huangyue Yu, Yan Wang, Xuesong Niu, Tengyu Liu, Qing Li, and Siyuan Huang. Sceneverse: Scaling 3d vision-language learning for grounded scene understanding. In *European Conference on Computer Vision*, pages 289–310. Springer, 2024. 1

[8] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. Amass: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5442–5451, 2019. 3

[9] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed AA Osman, Dimitrios Tzionas, and Michael J Black. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10975–10985, 2019. 3

[10] David Silver. Cooperative pathfinding. In *Proceedings of the aaai conference on artificial intelligence and interactive digital entertainment*, pages 117–122, 2005. 1

[11] Tomu Tahara, Takashi Seno, Gaku Narita, and Tomoya Ishikawa. Retargetable ar: Context-aware augmented reality in indoor scenes based on 3d scene graph. In *2020 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 249–255, 2020. 2

[12] Liang Xu, Xintao Lv, Yichao Yan, Xin Jin, Shuwen Wu, Congsheng Xu, Yifan Liu, Yizhou Zhou, Fengyun Rao, Xingdong Sheng, et al. Inter-x: Towards versatile human-human interaction analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22260–22271, 2024. 3