Supplementary Material for
SimRecon: SimReady Compositional Scene Reconstruction from Real Videos

The supplementary material is organized as follows:

- Section A provides more details on our Object-Centric Scene Representation, where we delve into the acquisition of various attributes.
- Section B presents the logic of the two core algorithms introduced in the main paper.
- Section C provides more experimental results.

A. Object-Centric Scene Representation

In our main paper, each object primitive o_i is a comprehensive entity defined by two categories of attributes: intrinsic attributes $\mathcal{A}_{\text{int},i}$ and relational attributes $\mathcal{A}_{\text{rel},i}$. The intrinsic attributes $A_{\text{int},i}$ are formally defined as:

$$A_{\text{int},i} = (A_{\text{spatial},i}, A_{\text{appr},i}, A_{\text{phys},i}). \tag{1}$$

In this section, we will provide a detailed explanation of how these attributes are obtained.

Spatial Attribute. The spatial attribute consists of the object's scale, rotation, and translation. Due to the lack of axis alignment in the reconstructed scene by Gaussian Splatting [2], the quality of the direct bounding box is suboptimal, making it difficult to accurately estimate the position and pose of objects. Therefore, we first perform axis alignment of the reconstructed scene prior to the acquisition of spatial attributes. In the axis alignment process, we first constructs a category-priority lookup table, where objects typically aligned with the room's orientation (e.g., a fridge or a painting) are assigned higher priority. Then we select the instance with the highest priority, largest point count, and visibility as the alignment reference object. Using all points of this instance, we compute its opacity-weighted centroid $\mathbf{c}_{\text{align}}$:

$$\mathbf{c}_{\text{align}} = \frac{\sum_i \alpha_i \mathbf{p}_i}{\sum_i \alpha_i}, \tag{2}$$

where $\mathbf{p}_i$ represents the point of the instance along with its opacity weight α_i. To obtain the principal axes, we conduct principal component analysis (PCA) by first computing the weighted covariance matrix:

$$\mathbf{C} = \frac{\sum_i \alpha_i (\mathbf{p}_i - \mathbf{c}_{\text{align}})(\mathbf{p}_i - \mathbf{c}_{\text{align}})^\top}{\sum_i \alpha_i} \tag{3}$$

and then performing eigen-decomposition:

$$\mathbf{C}\mathbf{v}_j = \lambda_j \mathbf{v}_j, \quad j = 1, 2, 3 \tag{4}$$

where $\mathbf{v}_j$ are the principal axes and λ_j are the corresponding eigenvalues. The principal axes form the matrix $\mathbf{P}_{\text{align}}$. Then the rotation matrix $\mathbf{R}_{\text{align}}$ is constructed by aligning $\mathbf{P}_{\text{align}}$ to the global axes. The 4×4 homogeneous transformation matrix can be obtained as follows:

$$\mathbf{T}_{\text{align}} = \begin{bmatrix} \mathbf{R}_{\text{align}} & -\mathbf{R}_{\text{align}}\mathbf{c}_{\text{align}} \\ \mathbf{0} & 1 \end{bmatrix} \tag{5}$$

which aligns the principal axes of the reference object to the global axes and translates its center to the original scene.

In the axis-aligned scene, we can further utilize the PCA method to estimate the pose of each instance. For each instance, in the newly aligned coordinate system, we first select the densest 95% of points $\mathcal{P}_{95}$ based on local density to reduce the influence of outliers. The axis-aligned bounding box of these points is computed as the spatial extent. A weighted PCA is again performed on $\mathcal{P}_{95}$ to obtain the principal direction $\mathbf{d}_{\text{pose}}$:

$$\mathbf{C}_{\text{pose}} = \frac{\sum_{i \in \mathcal{P}_{95}} \alpha_i (\mathbf{p}_i - \mathbf{c}_{\text{obj}})(\mathbf{p}_i - \mathbf{c}_{\text{obj}})^\top}{\sum_{i \in \mathcal{P}_{95}} \alpha_i}, \tag{6}$$

$$\mathbf{C}_{\text{pose}}\mathbf{d}_{\text{pose}} = \lambda_{\text{pose}}\mathbf{d}_{\text{pose}}, \tag{7}$$

To determine the canonical "pose" direction, among the six orthogonal directions $\{\pm\mathbf{e}_x, \pm\mathbf{e}_y, \pm\mathbf{e}_z\}$, the one most aligned with the vector from the object center $\mathbf{c}_{\text{obj}}$ to the origin is chosen by:

$$\mathbf{v}_{\text{pose}} = \arg \max_{\mathbf{e} \in \{\pm\mathbf{e}_x, \pm\mathbf{e}_y, \pm\mathbf{e}_z\}} |\langle \mathbf{e}, -\mathbf{c}_{\text{obj}} \rangle| \tag{8}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product. The estimated spatial extent provides the size $\mathbf{s}_i \in \mathbb{R}^3$, while the principal directions (along with $\mathbf{v}_{\text{pose}}$) define the rotation $\mathbf{R}_i \in SO(3)$, and the object center $\mathbf{c}_{\text{obj}}$ serves as the translation $\mathbf{t}_i \in \mathbb{R}^3$.

Physical Attribute. Physical attributes typically include the semantic label l_i, material mat_i, center of mass c_i, and

Table 1. Prompt used for physical attribute inference.

Assume you are an advanced vision-language model (VLM) with 3D perception capabilities. You are provided with a PNG image and object labels. Your task is to predict the following three physical attributes for the object: **Material**: Specify the material of the object in a hierarchical format (e.g., Fabric/Nylon, Metal/Steel, Plastic/Polycarbonate). **Mass Center**: Estimate the mass center of the object as a 3D offset vector relative to the object's geometric center. The offset should be expressed as a percentage of the object's scale along each axis, in the format (x, y, z), where: The x-axis points inward (into the image), the y-axis points to the right of the image, and the z-axis points upward following the right-hand rule. Each value should be a float between -1.0 and 1.0, representing the offset as a fraction of the object's size in that direction. **Mass**: Estimate the mass of the object in kilograms (kg). Carefully analyze the visual information from the image to make your prediction. Your output must be in strict JSON format as follows: { "material": "Fabric/Nylon", "mass center": [0.0, -0.11, 0.4], "mass": 0.35 } **Example Output:** { "material": "Plastic/Polycarbonate", "mass center": [0.12, 0.05, 0.30], "mass": 0.18 } Ensure that the material is described with both category and subcategory, the mass center is a 3-element array of floats (relative offsets), and the mass is a float in kilograms. Base your predictions on the appearance and inferred 3D structure of the object from the input image.

Table 2. Prompt used for relation attribute inference.

Assume you are an expert vision-language model (VLM) specialized in scene understanding. Your task is to analyze a given PNG image containing multiple objects with corresponding instance IDs and identify the relationships between these objects. There are only two types of relationships to consider: **Support**: One object physically supports another (e.g., a book on the table). **Attachment**: One object is attached to another (e.g., a painting on the wall). For each object in the image, ensure that it appears at least once in the output. Represent the relationships as a list of object label pairs and their relationship type. Your output must be in strict JSON format as follows: [{"object1": "label1", "object2": "label2", "relation": "support"}, {"object1": "label3", "object2": "label4", "relation": "adhesion"}] **Example Output:** [{"object1": "cup", "object2": "table", "relation": "support"}, {"object1": "painting", "object2": "wall", "relation": "attachment"}, {"object1": "book", "object2": "table", "relation": "support"}] Carefully analyze the spatial and physical context in the image to determine the most plausible relationships. Only use the two specified relation types. If an object has no clear relation, it should still appear in the output as a singleton relation (e.g., {"object1": "object", "object2": null, "relation": null}).

mass m_i essential for simulation. Among them, the semantic label l_i is obtained using the semantic reconstruction for each object. The remaining attributes are acquired via the vision-language model (VLM) inference, with the specific prompts detailed in Table 1.

Relation Attribute. In the main paper, we utilize the scene graph to correctly assemble objects in the simulator, and the construction of the scene graph relies on the relational attributes between objects. We categorize these relational attributes into two types: support relations (e.g., a cup placed on a table) and attachment relations (e.g., a painting hanging on a wall). These relations are inferred using a vision-language model (VLM). We provide the VLM with an image annotated with instance IDs of the objects in the image, and the VLM infers how the objects are structured using these two types of relations. The specific prompts are detailed in Table 2.

Algorithm 1 Active Viewpoint Optimization Logic

Input: Instance points $\mathbf{P}$, Relevant cameras $\mathcal{C}$
Output: Optimized transformation parameters R, T

Initialize optimization parameters: Rotation $R \leftarrow \mathbf{I}_{3\times3}$, Translation $T \leftarrow \mathbf{0}_3$, Optimizer
for $iteration \leftarrow 1$ to $max\_iterations$ **do**
  $\mathbf{P'} \leftarrow R\mathbf{P} + T$
  For each pixel $\mathbf{p}$ in the rendered image:
    Project $\mathbf{P'}$ onto the image plane using:
      $\mathbf{p} \leftarrow \pi(R\mathbf{P}+T, \mathcal{C})$, where π represents projection using camera intrinsics and extrinsics
    Sort points $\mathbf{p}_i \in \mathbf{P'}$ by depth:
      $\mathbf{p}_i \leftarrow \text{sort}(\mathbf{P'}, D(\mathbf{p}, v))$, where $D(\mathbf{p}, v)$ is the depth of transformed points at pixel $\mathbf{p}$ from viewpoint v
    Perform alpha blending for each point $\mathbf{p}_i$:
      $\alpha(\mathbf{p}, v) \leftarrow 1 - \prod_i (1 - \alpha_i(\mathbf{p}, v))$, where $\alpha_i(\mathbf{p}, v)$ is the Gaussian contribution of $\mathbf{p}_i$ at $\mathbf{p}$ from viewpoint v
    Normalize $\alpha(\mathbf{p}, v)$ to ensure values are in $[0, 1]$
  End For
  $L_{AVO}(v) \leftarrow -\sum_{\mathbf{p}\in\mathcal{P}_{obj}(v)} \alpha(\mathbf{p}, v)$
    $+\lambda_{depth} \frac{1}{|\mathcal{P}_{obj}(v)|} \sum_{\mathbf{p}\in\mathcal{P}_{obj}(v)} \big(D(\mathbf{p}, v)$
    $-d_{target}(s_i)\big)^2$
  where:
  $\mathcal{P}_{obj}(v)$: Pixels corresponding to the object rendered from view v
  $d_{target}(s_i)$: Target depth proportional to object size s_i
  λ_{depth}: Regularization weight for depth
  Update R, T using gradient descent
end for
return Optimized parameters R, T

Algorithm 2 Scene Graph Synthesizer Logic

Input: Global graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, Local subgraph $\mathcal{G}_k = (\mathcal{N}_k, \mathcal{E}_k)$
Output: Updated global graph $\mathcal{G}$

$\mathcal{N} \leftarrow \mathcal{N} \cup \mathcal{N}_k$
for all $e_{new} = (o_i, r_{new}, o_j) \in \mathcal{E}_k$ **do**
  $conflict \leftarrow$ **false**
  if $o_i \in \mathcal{N}$ **and** $o_j \in \mathcal{N}$ **then**
    for all $e_{old} = (o_i, r_{old}, o_m) \in \mathcal{E}$ **do**
      if `CheckConflict`(r_{new}, r_{old}) **then**
        $\mathcal{O}_{conflict} \leftarrow \{o_i, o_j, o_m\}$
        $v^*_{adj} \leftarrow \arg\min_{T_v} \Big(-\sum_{o_k\in\mathcal{O}_{conflict}} \sum_{g_p\in \text{GS}(o_k)} \Omega(g_p, v(T_v))\Big)$
        $\mathcal{E}_{adj} \leftarrow$ VLM-Infer(Image$(v^*_{adj}), \mathcal{O}_{conflict})$
        $\mathcal{E} \leftarrow (\mathcal{E} \setminus \{e_{old}\}) \cup \mathcal{E}_{adj}$
        $conflict \leftarrow$ **true**
        break
      end if
    end for
    if not $conflict$ **then**
      if `IsTransitive`(r_{new}) **and** `PathExists`$(\mathcal{G}, o_i, o_j, r_{new})$ **then**
        continue
      else if `IsShortcut`$(\mathcal{G}, e_{new})$ **then**
        continue
      end if
    end if
  end if
  if not $conflict$ **and** $e_{new} \notin \mathcal{E}$ **then**
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{e_{new}\}$
  end if
end for
return $\mathcal{G}$

Meanwhile, during the construction in the simulator, there may be failure cases where objects that should be supported become misaligned with the base support object across levels, due to the inaccurate position estimation of objects.For example, the estimated position error of the table may cause a book that originally is supported by the table to fall to the floor in the simulator, which could influence the correctness of the scene graph.

To ensure correct support relations despite estimated position errors, we compute the valid support surface S_{valid} of the supporting object O_s as:

$$S_{\text{valid}} = \{\mathbf{p} \in \partial O_s \mid \mathbf{n}_p \cdot \mathbf{e}_z > \tau\} \tag{9}$$

where $\mathbf{n}_p$ is the normal at point $\mathbf{p}$ and τ is a threshold. The bottom center of the supported object O_t is calculated as c_t. Then we project c_t onto S_{valid} to find the closest point c_s in the support surface and translate O_t horizontally:

$$c_s = \arg\min_{\mathbf{p}\in S_{\text{valid}}} \|\mathbf{p}_{xy} - c_{t,xy}\|, \tag{10}$$

$$O'_t = O_t + (c_{s,x} - c_{t,x}, c_{s,y} - c_{t,y}, 0). \tag{11}$$

This ensures the supported object is placed over the valid surface of the support object.

B. Algorithm Illustration

In this section, we present the logic of the two core algorithms introduced in the main paper, namely Active Viewpoint Optimization and Scene Graph Synthesizer, which are provided as Algorithm 1 and Algorithm 2, respectively.

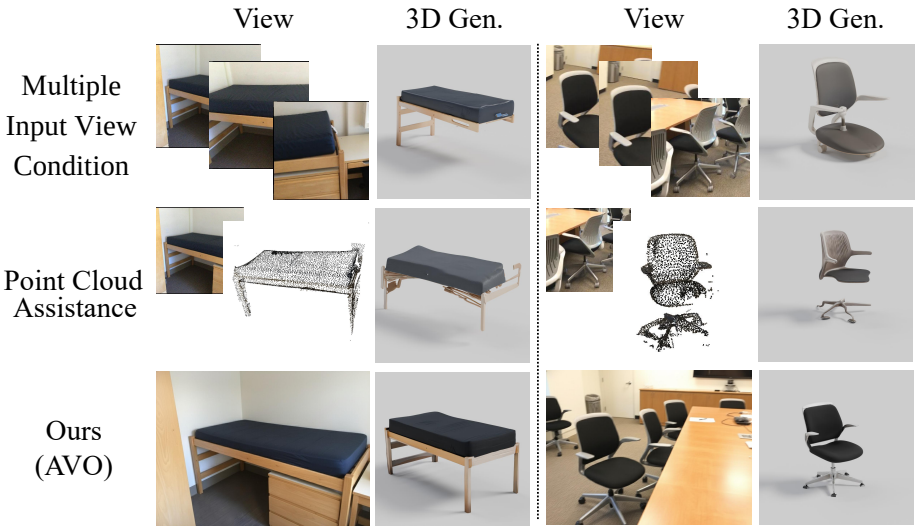

Figure 1. **Qualitative comparison of more 3D generation conditions.** We compare the single projected view condition by our Active Viewpoint Optimization (AVO) against two baselines: multiple input view condition and point cloud assistence.

Figure 2. **Randomization and editing results.** We conduct qualitative visualization for object-level randomization and scene-level editing along with language descriptions and edited relation attributes.

C. More Results

In this section, we present four additional experimental results to further demonstrate the effectiveness and robustness of our proposed approach.

3D Object Generation. In our main paper, we optimize the reconstructed scene using the Active Viewpoint Optimization (AVO) algorithm and then feed the optimized projected image to a 3D generative model for object completion. In this experimental section, we further investigate two more types of conditions for 3D generation: multi-view condition where multiple input images from different viewpoints are provided to the generative model, and 3D point cloud assisted generation where point cloud data is fed into the generative model. The qualitative results of these strategies are presented in Figure 1.

It can be observed that our AVO method apparently outperforms these two strategies. For multiple input view condition, generative models often struggle to effectively aggregate 3D information from multiple input views of arbitrary viewpoints. This limitation primarily arises because

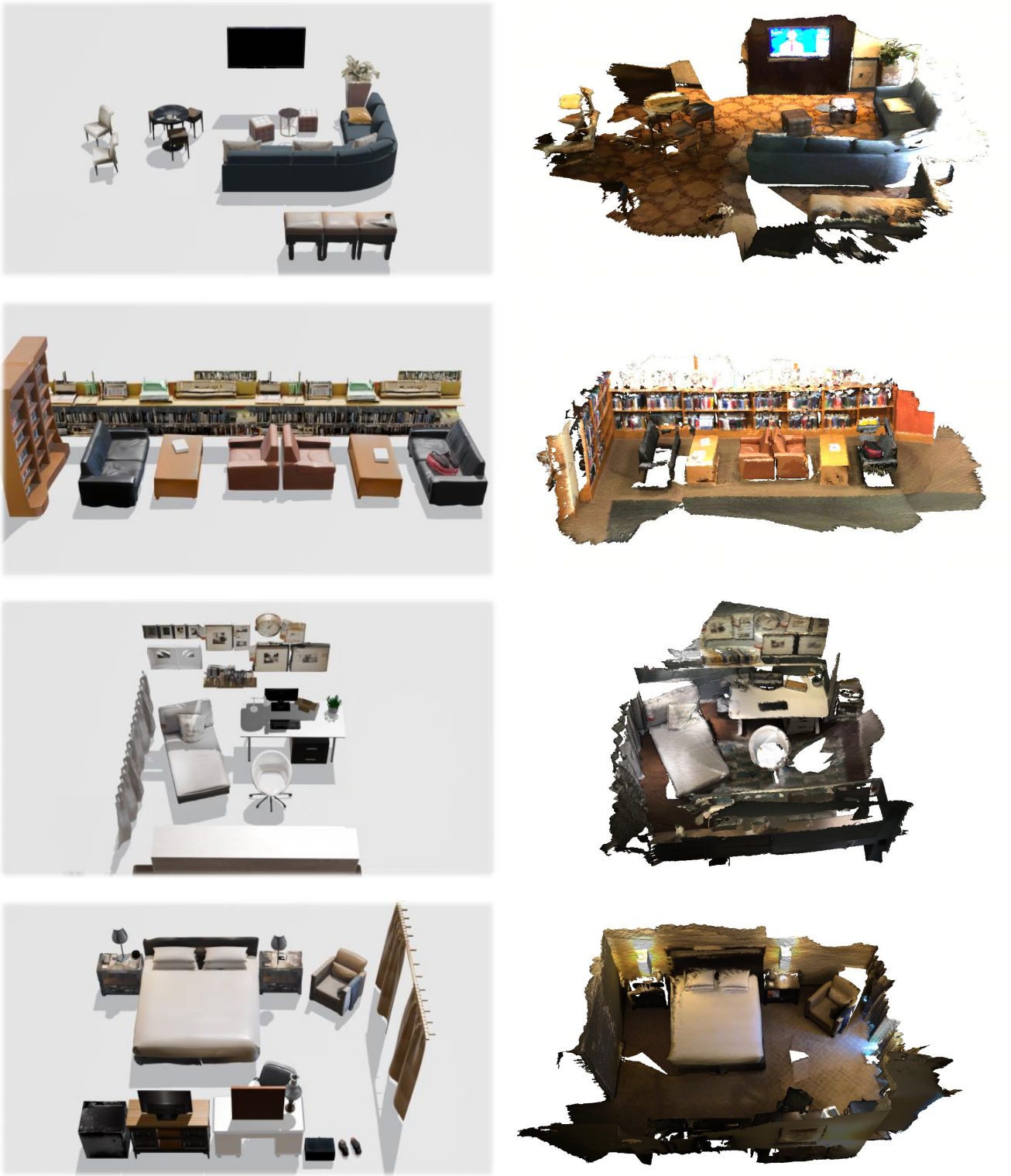

Figure 3. **Constructed scene visualization.** We compare the simulation-ready scenes created by our method (left) with its original 3D scan mesh provided by ScanNet [1] (right).

Figure 4. **Demonstration of goal-directed embodied navigation.**

naturally captured images tend to capture only a small, often redundant, portion of the object with ambiguous viewpoints, resulting in significantly low information density. Furthermore, since generative models are typically trained on well-curated, high-density few-view datasets, they lack the intrinsic capacity to effectively synthesize 3D geometry from such information-sparse inputs. Therefore, simply increasing the number of low-quality input views does not necessarily enhance the 3D generation capability. For point cloud assistance, the quality of generated objects is significantly compromised because point clouds collected in real-world environments are often sparse, noisy, and highly fragmented. Therefore, we conclude that selecting one or more projection images that provide the maximum 3D information gain as the most generalizable generation condition achieves the optimal 3D generation performance.

Scene Visualization. To further demonstrate the high fidelity and consistency of our method in scene-level scene reconstruction, we present more results of several scenes constructed using our method from ScanNet [1], as illustrated in Figure 3. We can observe that our method yields complete and independent object geometry, which offer significantly higher quality than holistic 3D scans and are inherently suitable for robot simulation.

Randomization and Editing. In embodied AI research, the diversity of training scene data is a core factor for ensuring the generalization capability of policy models. Our simulation-ready scenes, built with an underlying Scene Graph structure, inherently support this diversity. As demonstrated in Figure 2, by utilizing node editing and the addition or deletion of edges within the Scene Graph, we can effortlessly perform two levels of scene editing. At the object level, we achieve geometry and position randomization of individual instances, such as performing style randomization or plausible position adjustments on specific objects. At the scene level, we perform complex scene editing by leveraging the large language model (LLM) to edit the Scene Graph. Based on the updated Scene Graph, we can accurately determine the new positions of the added or moved objects and physically place them into the simulator, maintaining semantic and physical consistency.

Embodied Navigation. To demonstrate the high physical plausibility of our method in the simulator and its compatibility with embodied intelligence, we conducted an embodied navigation experiment within the scene. As shown in Figure 4, we deploy our constructed scene with the Habitat [3] simulator environment, and leverage NavFoM [4] for zero-shot object navigation with the language goal "Navigate to the backpack."

References

[1] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, pages 5828—-5839, 2017. 5, 6

[2] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 1

[3] Xavier Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Tsung-Yen Yang, Ruslan Partsey, Ruta Desai, Alexander William Clegg, Michal Hlavac, So Yeon Min, et al. Habitat 3.0: A co-habitat for humans, avatars and robots. *arXiv preprint arXiv:2310.13724*, 2023. 6

[4] Jiazhao Zhang, Anqi Li, Yunpeng Qi, Minghan Li, Jiahang Liu, Shaoan Wang, Haoran Liu, Gengze Zhou, Yuze Wu, Xingxing Li, et al. Embodied navigation foundation model. *arXiv preprint arXiv:2509.12129*, 2025. 6