METASCENES: Towards Automated Replica Creation for Real-world 3D Scans

Supplementary Material

A. The METASCENES Dataset

A.1. Data Acquisition details

Small objects capturing METASCENES includes numerous small objects, a category that existing datasets [1, 15] often fail to capture effectively. We follow a structured approach to identify and capture small objects that may be difficult to locate within a scene. First, we manually curate a list of support objects—such as tables and shelves—that are likely to either support or contain small objects. Next, we utilize SAM [8] to generate 2D masks for these support objects. These masked images are then input into GPT-4V [18] to prompt potential small objects that may be positioned on or within these support objects. Finally, we employ YOLO-v8 [6] to detect and segment these small objects within the scene. The prompt used to guide GPT-4V in capturing small objects is presented in Tab. A1.

Object captions generation To generate detailed object captions that describe object attributes, we employ GPT-4V [18] for description prompting. The object captions are categorized into two types: *Object appearance*, which detail visual characteristics such as color, shape, and texture. *Physical attribute*, which cover attributes like physics properties, mass, friction and bounciness. These two types of captions comprehensive coverage of object features, enabling a nuanced understanding of each object's role within the scene. We show some examples in Tab. A2. The prompt used to guide GPT-4V in generating *physical attribute* captions is presented in Tab. A1.

Asset candidates curation To replace each object with simulatable 3D assets, our goal is to identify diverse, high-quality candidates that closely resemble the original objects. For each scanned object, we generate 10 asset candidates using a combination of methods: text-to-3D generation, image-to-3D generation, and text-to-3D retrieval. The models for generating these 10 candidates are detailed in Fig. A1. These candidates ensure a balance of variety and fidelity, offering multiple options for replacement that enhance realism and physical plausibility. We show additional qualitative examples of asset candidates in our METASCENES dataset in Fig. A5.

For texture optimization, we refine the UV unwrapping process to improve the handling of complex object shapes. Instead of using the open-source UV-Atlas tool, as adopted in Paint3D [20]. We employ Blender's Smart UV unwrapping to preprocess images. This approach generates a UV map with fewer fragments and greater stability, facilitating smoother and more effective texture optimization. This refinement is particularly beneficial for assets with intricate geometries, ensuring more consistent and visually appealing texture mapping.

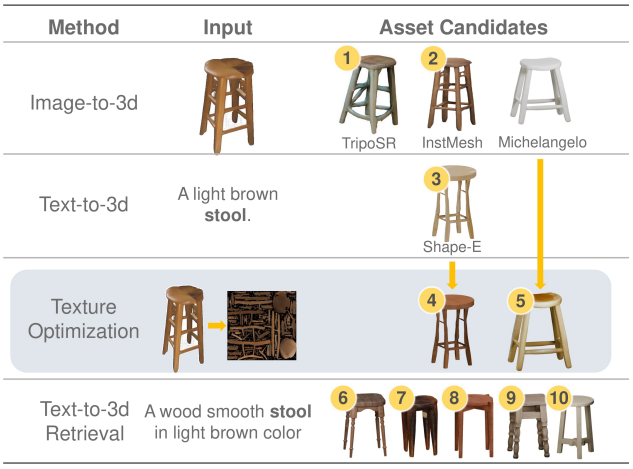


Figure A1. **Models for generate asset candidates.** For each object, we generate 10 asset candidates (labeled as 1–10 in the figure) by leveraging a combination of approaches: text-to-3D generation, image-to-3D generation, and text-to-3D retrieval.

A.2. Data Annotation and processing details

Human annotation We outline a typical annotation workflow that begins with a real-world scene represented as a point cloud. Annotators freely pan the camera to explore the entire scene, with an overlaid interface that remains synchronized with their view. The annotation process involves the following three sequential steps:

(i) **Selection**: Annotators select an object from the list of unannotated objects. Once an object is selected, a panel displays a list of candidate 3D assets corresponding to the object. Annotators are instructed to evaluate and identify the best-matching 3D asset based on visual and geometric similarity.

(ii) **Transformation**: The selected 3D asset is automatically integrated into the scene with a preprocessed scale and orientation. Annotators can then refine the placement by adjusting the asset's position, height, scale, and rotation to ensure accurate alignment with the point cloud and image.

(iii) **Ranking**: Annotators rank the remaining 9 candidate assets, identifying the top 2–5 objects that also closely match the real-world object. As shown in Fig. A2.

We recruited annotators to ensure the quality and accuracy

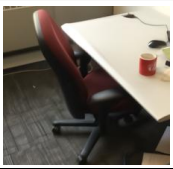
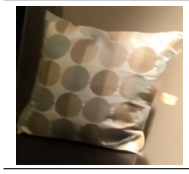
Table A1. **Prompts used in METASCENES.**

Purpose	Prompt
Small object capturing	You will be provided with an **image** containing a **label**. Your task is to carefully analyze the image and list the items present on the surface of the **label**. Please ensure that you only include items that are on its surface and not those nearby. If you think there is nothing on this **label**, please return an empty list. Each item should be described in a concise and accurate manner and returned in JSON format. Each item's JSON object should include the following fields: - item: The name of the object - color: The color of the object Example Output: If there is a black mouse pad and a red cup on the table, your output should be: `[{ 'item': 'mouse pad', 'color': 'black' }, { 'item': 'cup', 'color': 'red' }]` **Image**: *A real-world image containing a table.* **Label**: Table
Physical attribute	Given the following object **label** and its **size**, please output the **physics attributes** of the object in strict JSON format, including: Physics Properties: Classify the object into one of the following categories: Rigid Body (e.g., Table, Chair, Book, Ball, Cup, Box, Door) Cloth (e.g., T-shirt, Curtain, Tablecloth, Flag, Bed sheet, Towel, Pants) Soft Body (e.g., Jelly, Soft toy, Rubber ball, Cushion, Slime, Foam, Balloon) Mass: Estimate the mass of the object based on its label and bbox size. The mass value should be a float number. - For small objects (e.g., ball, book), the mass should be between 0.1 to 5.0. - For medium objects (e.g., table, chair), the mass should be between 5.0 to 50.0. - For large objects (e.g., building, vehicle), the mass should be above 50.0, depending on the object's real properties. Friction: Assign a friction value between 0 and 1 based on the object type. The friction value should be a float number: - 0.0: No friction (completely smooth, slides freely). - 0.1 - 0.3: Low friction (slight resistance, still easy to slide). - 0.4 - 0.6: Medium friction (noticeable resistance, sliding becomes difficult). - 0.7 - 1.0: High friction (almost no sliding, quickly stops after collision). - > 1.0: Super high friction (very high resistance, may "stick" together, preventing sliding). Bounciness: Assign an integer value of 0 or 1 to indicate whether the object bounces or not: - 0: Does not bounce. - 1: Bounces. Output Format: Please format your output strictly as JSON, ensuring that mass and friction are float values, and bounciness is an integer: `{ 'physics_attributes': 'category':{Rigid Body | Cloth | Soft Body}, 'mass': [float], 'friction': [float], 'bounciness':[int]}` **Object Label**: Chair **Object Size**: [1.2, 1.0, 0.6]

of the 3D scene reconstruction process. Annotators were instructed to follow these detailed guidelines: (i) *Object Matching.* Annotators were required to select 3D assets that closely align with the observed categories, shapes, and sizes of the objects in the scene. Accurate matching between the original objects and their replica creations is critical for maintaining realism. (ii) *Object Consistency.* For objects with uniform appearance across the scene, the same 3D asset must be consistently selected for replacement. (iii) *Spatial Accuracy.* Each object must be placed and oriented to match its position in the 3D point cloud and accompanying image as closely as possible. Annotators were instructed to avoid misplacements, such as collisions between objects or floating artifacts, to the greatest extent feasible.

To ensure the accuracy and reliability of the annotation results, we implemented a quality control process as follows: For each batch of annotated data, 10% of the samples are randomly selected for accuracy verification. If more than

Table A2. **Examples of object captions in METASCENES.** Note that 'Friction' assign a friction value between 0 and 1 based on the object type and 'Bounciness' assign an integer value of 1 or 0 to indicate whether the object bounces or not.

Image	Object Appearance	Physical Attributes
	A fabric and plastic soft office chair in red color.	• Rigid body • Mass: 20 kg • Friction: 0.5 • Bounciness: 0
	A fabric soft blanket in white color.	• Cloth • Mass: 10 kg • Friction: 0.3 • Bounciness: 0
	A fabric smooth pillow in multi-colored.	• Soft Body • Mass: 1 kg • Friction: 0.3 • Bounciness: 0
	A fabric soft stuffed animal in brown color.	• Soft Body • Mass: 0.5 kg • Friction: 0.3 • Bounciness: 1

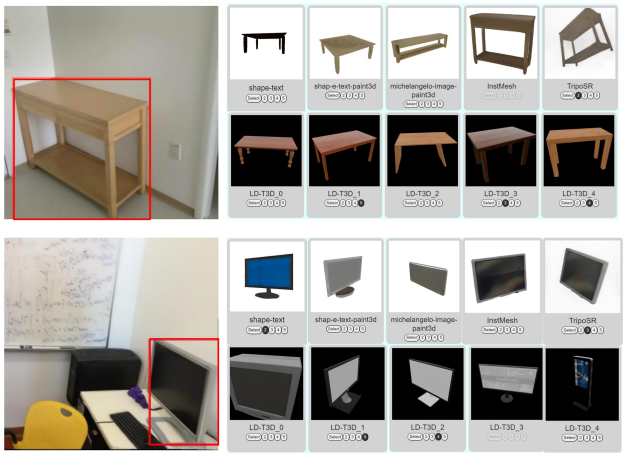


Figure A2. **Annotation interface of object ranking.** Once the best-match asset is selected, annotators are asked to rank the remaining 9 candidate assets.

98% of the inspected samples pass the reviewer's validation, the batch is deemed acceptable. Otherwise, the annotators are required to re-label the entire batch to address potential errors and meet the quality standards.

Physics-based Optimization We use Markov Chain Monte Carlo (MCMC) to traverse the non-differentiable solution space, optimizing the horizontal and vertical placement of objects to prevent issues like collisions or floating objects. See Algorithm 1 for the pseudo code. To quantify collisions for m objects in scene $\mathbb{S}$, we compute the collision loss as follows:

$$L = \sum_{i=1}^{m} \sum_{j=i+1}^{m} \text{IoU}(\text{BBox}(o_i), \text{BBox}(o_j)), \tag{A1}$$

where $\text{BBox}(\cdot)$ represents the 3D bounding box of object, and IoU denotes the *Intersection over Union* metric. The loss L aggregates the pairwise IoU values for all unique object pairs. This formulation allows the optimization process to iteratively minimize L, effectively reducing collisions and ensuring proper spatial arrangements in the scene.

Algorithm 1: MCMC Optimization Algorithm

Input : Scene $\mathbb{S}$ with m objects at their initial positions, where $\mathbb{S} = \{o_1, o_2, \ldots, o_m\}$
Output : Scene $\mathbb{S}$ with m objects at their optimized positions.

$n \leftarrow 0$ {Initialize MCMC step counter}
$T \leftarrow \{t_1, t_2, t_3, t_4\}$ {Set of possible movements along parameter axes}
$L_0 \leftarrow$ CalculateCollisionLoss($\mathbb{S}$) {Initial collision loss}
$L_{\text{min}} \leftarrow L_0$ {Track the minimum collision loss}
while $L_n > 0$ **and** $n <$ MaxStep **do**
  for $i = 1$ **to** m **do**
    Randomly select a movement $t \in T$ and apply it to object o_i
    if o_i remains within scene boundaries **then**
      Compute the new position for o_i
      $L_n^i \leftarrow$ CalculateCollisionLoss($\mathbb{S}$) {Collision loss after moving o_i}
      if $L_n^i < L_{\text{min}}$ **then**
        Update the position of o_i
        $L_{\text{min}} \leftarrow L_n^i$ {Update the minimum loss}
      else
        Revert the position of o_i
      end if
    end if
  end for
  $n \leftarrow n + 1$ {Increment the MCMC step counter}
end while

A.3. METASCENES statistics

We present histograms showing the distribution of object counts and object categories per scene in Fig. A6a

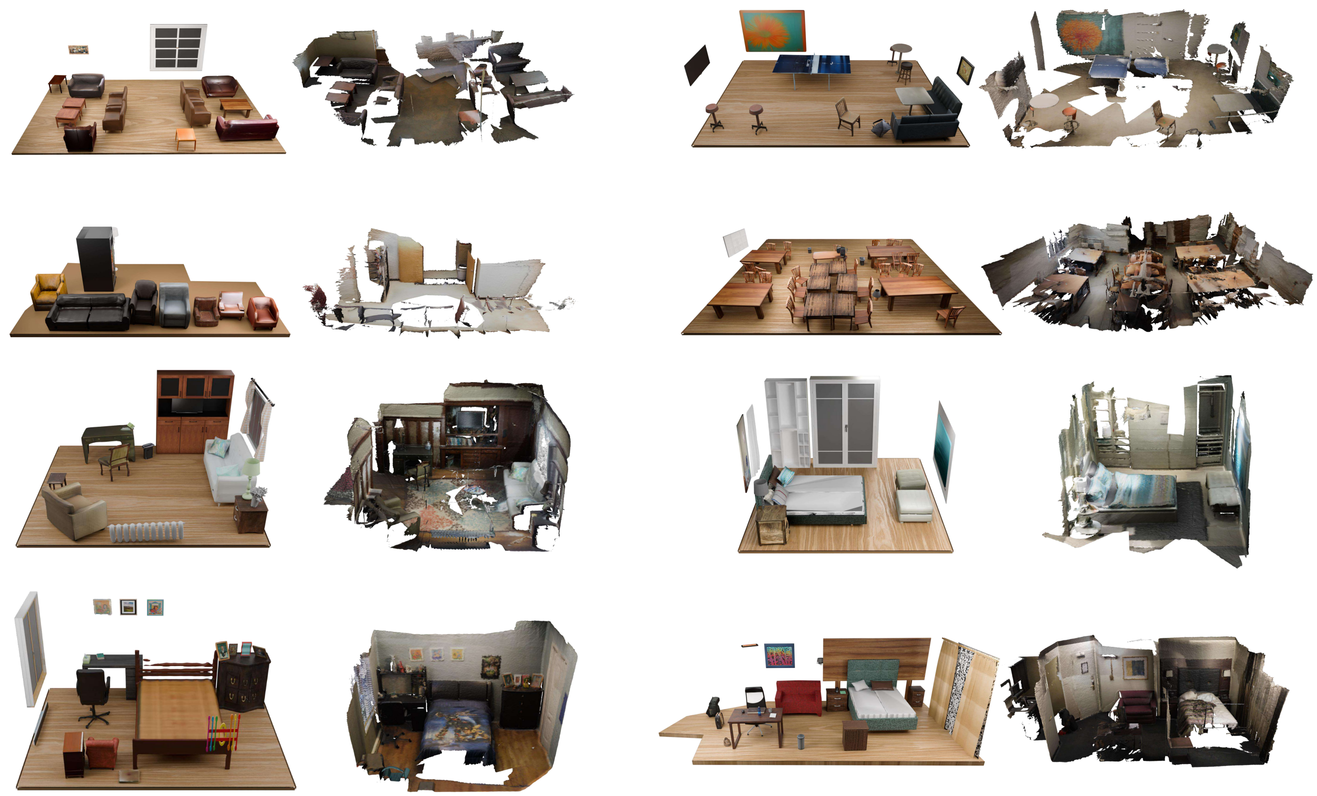


Figure A3. **Scene examples**. We compare the scenes in **METASCENES** (left) with its original 3D point cloud (right). Note that layouts are set to be invisible.

Figure A4. **Word cloud of object categories in METASCENES.** Font sizes indicate unique instance count per category.

and Fig. A6b. Additionally, we include a box plot illustrating the distribution of physical sizes (measured in volume, m^3) for the top 50 most frequent object categories in Fig. A7. Fig. A4 shows a word cloud visualization of categories in **METASCENES**, with the text font size representing the total count of unique object instances in each category. We see that our dataset contains a diverse set of object categories. Qualitative examples of scenes from our **METASCENES** dataset can be found in Fig. A3. For the efficiency of dataset creation, the end-to-end preprocessing of a scene with 39 preprocessed object candidates takes approximately 12 minutes. The time for object candidate creation depends on the reconstruction model used. Each annotator takes about 2 minutes to annotate one object.

B. Experiment Details

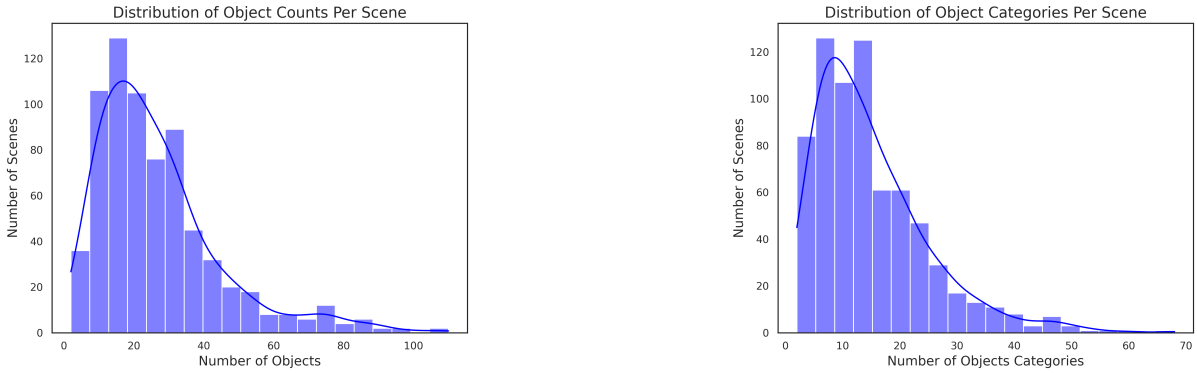
B.1. Automated Replica Creation

Model Training We train our optimal asset retrieval model using a training set of 600 scenes, which includes a total of 13125 objects. For point cloud encoding, we finetune the PointBERT pretrained on [16], and for image and text encoding, we utilized OpenCLIP. During training, we applied standard data augmentation techniques to the 3D point clouds, such as random point dropping, scaling, shifting, and rotational perturbations, to enhance model robustness.

Baselines We detail the setup for the comparative models, through two key components: *Optimal Asset Selection* and *Object Pose Alignment*.

(i) *Optimal Asset Selection.* We evaluate **METASCENES** against state-of-the-art multimodal alignment methods, as summarized in **??** in the main paper. For the Uni3D [22] baseline, we use OpenCLIP with the model configuration

Figure A5. **Overview of our asset candidates.** Note that "*" indicates texture optimization.



(a) **Distribution of Object Counts Per Scene.**

(b) **Distribution of Object Categories Per Scene.**

Figure A6. **Object statistics in METASCENES.**

"EVA02-E-14-plus" as the image and text encoder. This advanced Transformer-based model is pre-trained to reconstruct robust language-aligned visual features through masked image modeling, enabling strong cross-modal alignment capabilities. The Point-BERT [19] is pre-trained on the ModelNet40 dataset, while PointNet++ [11] is pre-trained on the SceneVerse [5] dataset. For the ACDC [2] framework, we employ CLIP and DINO-v2 [9] to identify the best-matching assets.

(ii) *Object Pose Alignment.* In the ACDC framework, we first utilize DINO-v2 to determine the optimal orientation of the asset. Once the best orientation is selected, we apply a render-and-compare method to adjust the asset's scale. Specifically, after identifying the optimal orientation, we scale the asset across a range of factors from 0.5 to 1.5 and render both the asset and the corresponding real-world object into the 2D image. The asset's scale is then determined by comparing the 2D bounding box sizes of the rendered asset and the real-world object in the 2D image, with the best-matching scale corresponding to the minimal discrepancy between the two boxes.

Metrics We detail the metrics used in our experiment as follows: Chamfer Distance (CD) measures the average distance between point clouds. Enhanced Chamfer Distance (ECD) extends CD by incorporating curvature and geometric features to better capture fine details. Bounding Box Intersection over Union (Bbox IoU) calculates the intersection over union for the 3D bounding boxes of the assets. Color

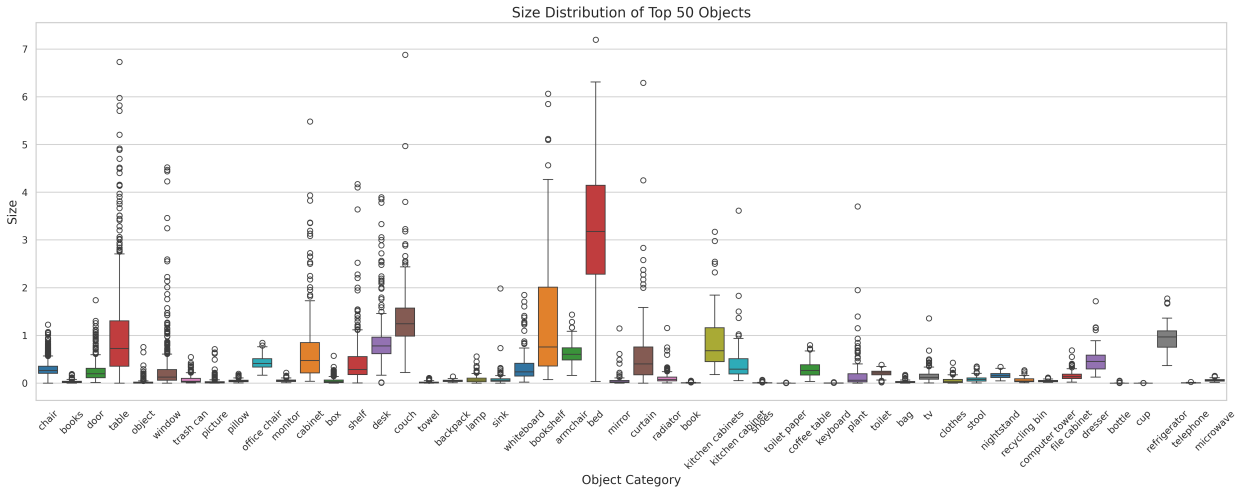


Figure A7. **Box plot of the physical size distribution.** This shows a wide range of object sizes, with the size distribution clearly highlighting a significant contrast between larger and smaller objects.

Figure A8. **Diverse results of the micro-scene synthesis.** The model is capable of generating varied layouts for the same large furniture.

Histograms (Color Hist) compute the Kullback-Leibler divergence between the color distributions of the selected and ground truth assets.

B.2. Micro-Scene Synthesis

Data Processing We preprocess METASCENES by dividing the rooms into micro-scenes. Each micro-scene contains one large object and several corresponding smaller objects placed on it. We retain the large object categories similar to those in 3D-FRONT, such as "sofa," "cabinet," and "table". For a small portion of objects with unknown categories, we classify them as "object". Additionally, we merge over 400 open-vocabulary object names into 60 categories: 25 for large objects and 43 for small objects, with 8 categories shared between them, as shown in Tab. A3. After processing, the micro-scene dataset consists of 1,012 micro-scenes and 773 object assets. The quantity distribution of each category in the preprocessed micro-scene dataset is illustrated in Fig. A10.

Model Setting In our setup, micro-scenes do not require the shape of the floor plan. Therefore, for all three models, *i.e.*, ATISS [10], DiffuScene [14], and PhyScene [17], we exclude the floor plan input and layout encoder. For DiffuScene and PhyScene, we set the maximum number of objects to 24, with the layout of the large furniture provided as the first object vector. The models generate the remaining 23 vectors, including the empty vectors. For ATISS, the model uses the layout of the large furniture as the first object and then sequentially predicts the layouts of the smaller objects. From the 1,012 processed scenes, we randomly select 803 for training and reserve the remaining 208 for testing.

Diverse Generation Results We present results with various large furniture pieces in **??**. In addition, we show diverse results for the same large furniture, specifically selecting a ta-

Table A3. **List of 60 categories in micro-scene synthesis.** The category for large furniture is marked in green and the category for small object is marked with underline. There are 8 categories shared between both groups.

alarm_clock	bag	basket	bathtub	bed	bin
book	bottle	box	bucket	cabinet	can
chair	clothing	coffee_table	computer	cooking_machine	counter
decoration	desk	dining_table	earphone	electronic_devices	end_table
food	instrument	kettle	keyboard	kitchenware	lamp
ledge	monitor	mouse	mouse_pad	mug	nightstand
object	organizer	phone	picture	pillow	plant
refrigerator	remote_control	round_table	shelf	sink	sofa
stool	table	tissue_paper	toilet	tool	towel
toy	tv	tv_stand	wardrobe	washing_machine	washing_stuff

ble. As shown in Fig. A8, the model is capable of generating varied layouts for the same large furniture.

Room Type - Object Category Relationship We train DiffuScene with a text embedding module, where the prompt includes both the large object's category and the room type. For example: "A counter in the **kitchen**". The text encoder from CLIP [12] is used to embed the prompt. During inference, we generate layouts with a fixed large object, specifically a table, while varying the room type, such as "A table in the **office**". We calculate the related small object's category distribution for each room type. The results in Fig. A11 demonstrate that the model has learned distinct category distributions for different room types. For example, "monitor" has the highest probability of appearing in "office", "cooking_machine" is most likely in "kitchen", and "bag" is most often found in "Bookstore/Library". These findings also validate the effectiveness of our **MetaScenes**.

B.3. Embodied Navigation in 3D scenes

Data and Simulation Setup We use the Habitat simulator for our data generation and simulation. For data generation, we convert all `glb` format files into the desired format in Habitat. To generate trajectories for training, we randomly sample a start position for the agent and a navigable target object except for walls. For each trajectory, we sample the ground-truth shortest path using PathFinder within the Habitat simulator. Therefore, each trajectory consists of the agent's start position and end position, the ground-truth shortest path, and the semantics of the target object. Then these trajectories will be used for training the navigation model. In the Habitat simulator, the agent's action space contains `move forward` (0.25m), `turn left` (30 degrees), and `turn right` (30 degrees).

Model and Training Details We use SPOC [4] as our shared model architecture, with SigLIP [21] image and text encoders. We use a 3-layer transformer encoder and decoder and a context window of 10. We evaluate the object navigation task for the SPOC model trained on the ProcTHOR, **MetaScenes**, and Both, within the AI Habitat environment. The dataset consists of 706 scenes which are randomly split into train/test on a 4:1 ratio. We randomly collect 100 trajectories from each training scene and 50 trajectories from each testing scene for train/test data. We train or fine-tune the model on our **MetaScenes** navigation data with a batch size of 256, a learning rate of 0.0001, and 70k training steps.

Quantitative Metrics Following Eftekhar *et al*. [3], we use quantitative metrics containing SR (Success Rate), EL (Episode Length), SEL (Success weighted by Episode Length), SPL (Success weighted by Path Length), and curvature. SR represents the proportion of correctly navigated trajectories with respect to all trajectories. EL indicates how many actions on average are needed to successfully navigate to the target object. SEL and SPL indicate the difference between the ground-truth path and the predicted path by the agent. A larger SEL or SPL value indicates a closer alignment between the ground truth path and the actual path. Curvature measures the smoothness of a trajectory, with larger curvature values indicating a less smooth path. Some qualitative examples of navigation are shown in Fig. A12. Regardless of whether the target object is seen at the beginning, the agent can navigate to the destination correctly.

Figure A9. **The configuration of UP AGV and its environment.** This includes the real-world scene, the scanned scene, and the digital replica.

Table A4. Comparison on VLN experiments with HSSD

Benchmark	Data Source	SR(%)↑	EL↓	Curvature↓	SEL↑	SPL↑
10 scenes from Replica CAD	HSSD	27.00	33.77	0.39	26.77	23.32
	METASCENES	32.00	33.71	0.46	31.56	26.91

Real-world Deployment We deploy the policy trained on METASCENES to a real-world Automated Guided Vehicle (AGV), called UP. For odometry estimation, the vehicle combines data from a 2D Lidar, IMU, and wheel speedometer. After receiving the predicted actions from the navigation policy based on the digital replica of the scene, we downsample these actions at approximately 0.5-meter intervals to create a sequence of local goals. UP plans a trajectory for each local goal and computes the corresponding linear and angular velocities using Dynamic Window Approach (DWA) algorithm, ensuring collision-free execution. The AGV configuration, the real-world scan, and its digital replica are shown in Fig. A9. We present navigation scenarios in Fig. A13, demonstrating that UP successfully reaches the target by transferring the policy in simulation to the real world.

Comparisons with Other Datasets We evaluate navigation models pre-trained on METASCENES and HSSD [7] using the replica-CAD [13] dataset in Tab. A4. We randomly selected 10 scenes in the replica-CAD dataset, and randomly sampled the starting point and target object in each scene, collecting 10 trajectories for testing. Finally, the two models are tested on these 100 trajectories and the metrics are calculated. The results confirm that pre-training with our dataset consistently yields superior performance, further verifying our scene quality claim.

References

[1] Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X Chang, and Matthias Nießner. Scan2cad: Learning cad model alignment in rgb-d scans. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. A1

[2] Tianyuan Dai, Josiah Wong, Yunfan Jiang, Chen Wang, Cem Gokmen, Ruohan Zhang, Jiajun Wu, and Li Fei-Fei. Acdc: Automated creation of digital cousins for robust policy learning. *arXiv preprint arXiv:2410.07408*, 2024. A5

[3] Ainaz Eftekhar, Kuo-Hao Zeng, Jiafei Duan, Ali Farhadi, Ani Kembhavi, and Ranjay Krishna. Selective visual representations improve convergence and generalization for embodied ai. *arXiv preprint arXiv:2311.04193*, 2023. A7

[4] Kiana Ehsani, Tanmay Gupta, Rose Hendrix, Jordi Salvador, Luca Weihs, Kuo-Hao Zeng, Kunal Pratap Singh, Yejin Kim, Winson Han, Alvaro Herrasti, et al. Spoc: Imitating shortest paths in simulation enables effective navigation and manipulation in the real world. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. A7

[5] 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 (ECCV)*, 2024. A5

[6] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Yolo by ultralytics. https://github.com/ultralytics/ultralytics, 2023. A1

[7] Mukul Khanna, Yongsen Mao, Hanxiao Jiang, Sanjay Haresh, Brennan Shacklett, Dhruv Batra, Alexander Clegg, Eric Undersander, Angel X Chang, and Manolis Savva. Habitat synthetic scenes dataset (hssd-200): An analysis of 3d scene scale and realism tradeoffs for objectgoal navigation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. A8

[8] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *International Conference on Computer Vision (ICCV)*, pages 4015–4026, 2023. A1

[9] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. A5

[10] Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. Atiss: Autoregressive transformers for indoor scene synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. A6

[11] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. A5

[12] Alec Radford, Jong Wook Kim, Chris Hallacy, A. Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021. A7

[13] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. A8

[14] Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. Diffuscene: Denoising diffusion models for gerative indoor scene synthesis. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. A6

[15] Qirui Wu, Sonia Raychaudhuri, Daniel Ritchie, Manolis Savva, and Angel X Chang. R3ds: Reality-linked 3d scenes for panoramic scene understanding. *arXiv preprint arXiv:2403.12301*, 2024. A1

[16] Le Xue, Ning Yu, Shu Zhang, Artemis Panagopoulou, Junnan Li, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, et al. Ulip-2: Towards scalable multimodal pre-training for 3d understanding. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 27091–27101, 2024. A4

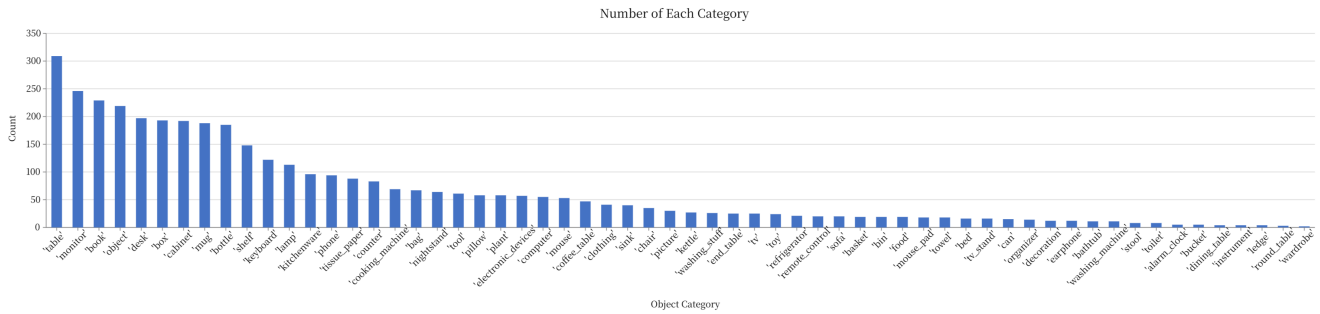


Figure A10. **Number of each category in preprocessed micro-scene dataset**.

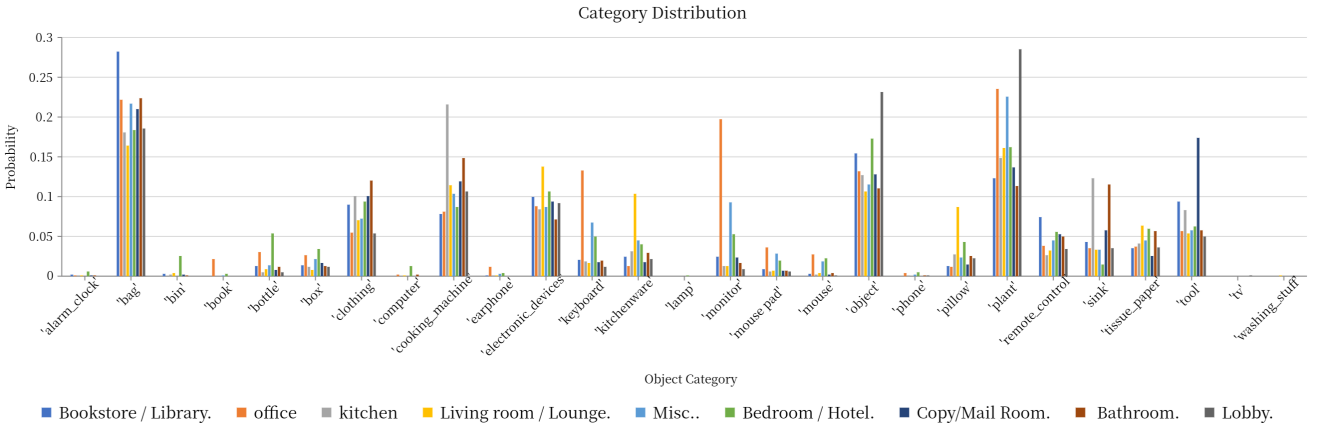


Figure A11. **Generated class distribution of different room types.** We generate the layout with the same large furniture using the prompt with different room types. Results show the model has learned different class distribution of different room types.

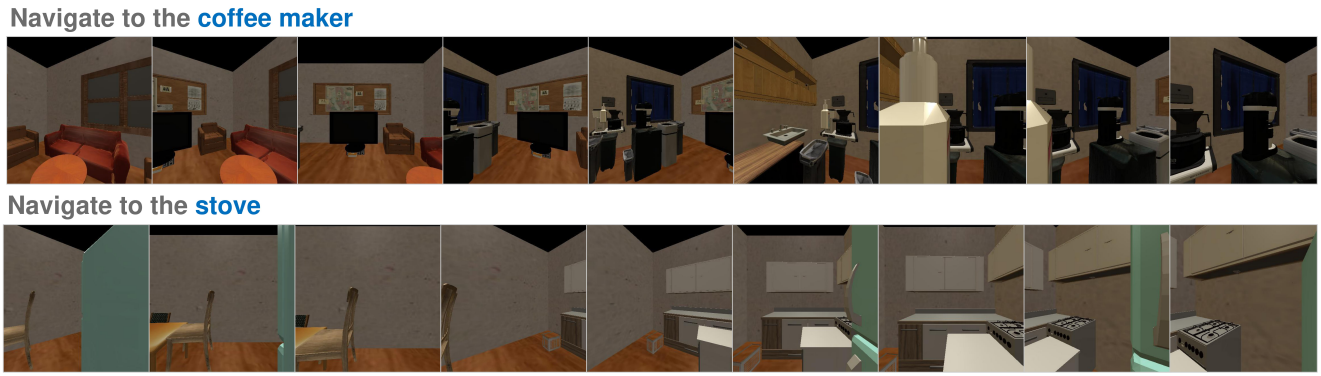


Figure A12. **Embodied Navigation.** Demonstration of the embodied agent performing goal-directed navigation in Habitat.

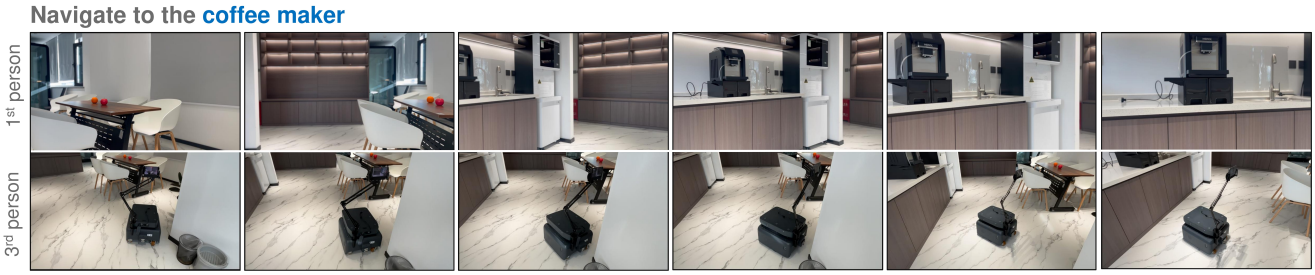


Figure A13. **Real-world transfer.** Demonstration of the embodied agent performing goal-directed navigation in the real world.

[17] Yandan Yang, Baoxiong Jia, Peiyuan Zhi, and Siyuan Huang. Physcene: Physically interactable 3d scene synthesis for embodied ai. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. A6

[18] Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. The dawn of lmms: Preliminary explorations with gpt-4v (ision). *arXiv preprint arXiv:2309.17421*, 2023. A1

[19] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. A5

[20] Xianfang Zeng, Xin Chen, Zhongqi Qi, Wen Liu, Zibo Zhao, Zhibin Wang, Bin Fu, Yong Liu, and Gang Yu. Paint3d: Paint anything 3d with lighting-less texture diffusion models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4252–4262, 2024. A1

[21] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11975–11986, 2023. A7

[22] Junsheng Zhou, Jinsheng Wang, Baorui Ma, Yu-Shen Liu, Tiejun Huang, and Xinlong Wang. Uni3d: Exploring unified 3d representation at scale. In *International Conference on Learning Representations (ICLR)*, 2024. A4