PLACEIT3D: Language-Guided Object Placement in Real 3D Scenes

Ahmed Abdelreheem[2,*] Filippo Aleotti[1] Jamie Watson[1] Zawar Qureshi[1] Abdelrahman Eldesokey[2]
Peter Wonka[2] Gabriel Brostow[1,3] Sara Vicente[1] Guillermo Garcia-Hernando[1]
[1]Niantic Spatial [2]KAUST [3]UCL

https://nianticlabs.github.io/placeit3d/

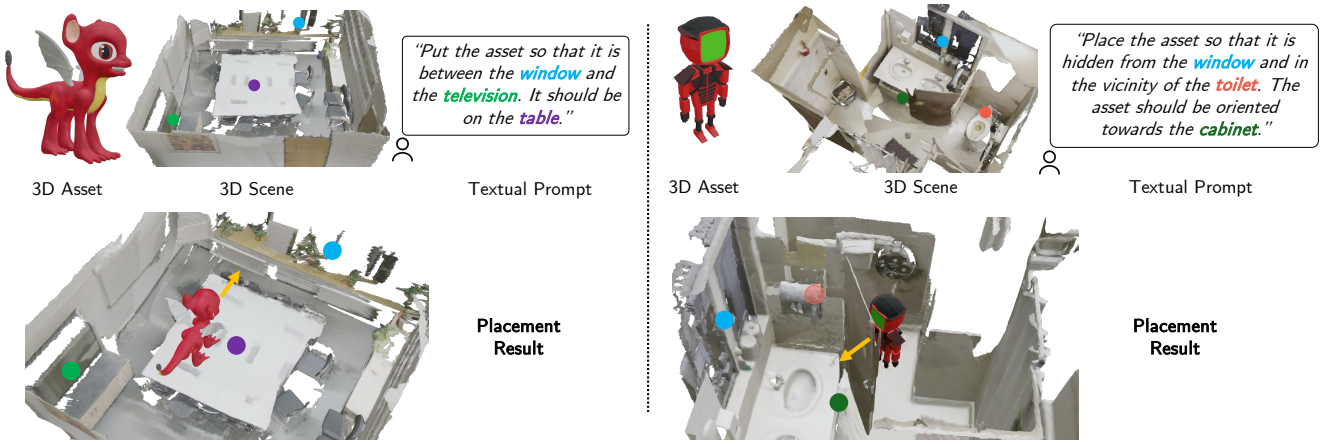

Figure 1. **Language-guided 3D Object Placement in Real 3D Scenes**: Given a text prompt, the task is to find a valid placement for an asset, requiring semantic and geometric understanding of the scene, the asset's shape, and spatial relationships. Colored dots show referenced objects (for visualization only, not given to the model), and the yellow arrow indicates the predicted frontal direction.

Abstract

We introduce the task of Language-Guided Object Placement in Real 3D Scenes. Given a 3D reconstructed point-cloud scene, a 3D asset, and a natural-language instruction, the goal is to place the asset so that the instruction is satisfied. The task demands tackling four intertwined challenges: (a) one-to-many ambiguity in valid placements; (b) precise geometric and physical reasoning; (c) joint understanding across the scene, the asset, and language; and (d) robustness to noisy point clouds with no privileged metadata at test time. The first three challenges mirror the complexities of synthetic scene generation, while the metadata-free, noisy-scan scenario is inherited from language-guided 3D visual grounding. We inaugurate this task by introducing a benchmark and evaluation protocol, releasing a dataset for training multi-modal large language models (MLLMs), and establishing a first nontrivial baseline. We believe this challenging setup and benchmark will provide a foundation for evaluating and advancing MLLMs in 3D understanding.

1. Introduction

At two to three years old, neurotypical children learn to follow two-step instructions like *"Get your shoes and put them on the shelf"* [42]. These tasks may appear simple, yet children need time to grasp basic vocabulary and to learn the physical affordances of both 3D objects and scene layout. Perhaps AIs could obtain similar capabilities.

In this paper, we focus on the novel task of *language-guided 3D object placement in a reconstructed real 3D scene*. As in the shoe example, the goal is to find a valid placement of the object among multiple configurations that satisfy the instruction. As shown in Figure 1, the placement must also respect the physical constraints of the space and of the 3D asset. Excelling at this task would unlock applications such as instructing a robot, through language, to move a real object to a new location. It is also relevant to augmented reality (AR). For instance, a shopper wearing an AR headset could use natural-language commands to anchor a virtual product on a real-world surface or reposition digital décor anywhere in the room.

*Work done during an internship at Niantic.

MLLMs have recently shown strong performance on language-guided 3D scene understanding tasks, including visual question answering [11, 22, 23], visual grounding [25, 62], and synthetic scene generation [18, 44, 55]. We study language-guided 3D asset placement in reconstructed scenes, a problem closest to grounding and to synthetic scene generation, yet distinct in that it requires addressing *all* of the following challenges simultaneously:

i. *Ambiguity of solutions*. 3D visual grounding typically admits a single correct match, whereas 3D placement is inherently one-to-many: multiple placements can satisfy the instruction [6, 19]. This complicates both benchmarking and data construction, which must accommodate multiple valid answers.

ii. *Intrinsic 3D reasoning*. Many constraints are geometric and cannot be resolved from 2D projections alone. For example, *"place the asset in between the chair and the table, hidden from the window"* requires reasoning about free space and spatial relationships in 3D.

iii. *No privileged information at test time*. In contrast with synthetic scene generation, we require just the reconstructed 3D scene. We are not given layouts [38, 44], scene graphs [61], object properties, or clean geometry [24, 55] to aid with making a prediction.

iv. *Joint reasoning over scene, asset, and language*. The asset's size and shape restrict feasible placements; given the same scene and instruction, a large object has fewer valid locations than a small one. Among the valid options, the model must follow the user's stated intent rather than default to common sense priors [36, 56].

The highlight of this paper is the introduction of a challenging novel task, which we call PLACEIT3D: language-guided 3D placement on real scenes. To the best of our knowledge, no existing benchmarks, datasets, or methods directly address this problem. To advance research in this area, we make three key contributions, summarized here:

- We introduce PLACEIT3D-benchmark for language-guided placement with 3,500 evaluation examples, each consisting of a real ScanNet scene [15], a PartObjaverse-Tiny asset [54], and a guiding prompt. Our evaluation protocol accounts for placement ambiguity, enabling fair comparison across methods.
- We present PLACEIT3D-dataset, a large-scale training dataset with 100,505 training and 2,566 validation examples, each annotated with all valid placements that can be used for training MLLMs. Like the benchmark, it uses ScanNet scenes and PartObjaverse-Tiny assets.
- We propose PLACEWIZARD, a proto-method for this task built on recent 3D LLMs [25]. It uses a modified form of spatial aggregation, an asset encoder, and rotation prediction, outperforming existing baselines.

2. Related Work

3D and Large Language Models. 3D LLMs [35] are a subclass of MLLMs that jointly process point clouds/meshes and text. They typically align 3D features to the language space with lightweight projection layers or cross-attention adapters, mirroring strategies from image-grounded MLLMs (VLMs) [5, 9, 13, 27, 29, 30, 32]. For grounding and QA, 3D-LLM [22] fuses point clouds with 2D image features; Reason3D [25] routes a pre-trained point encoder through a Q-Former [31] and a decoder guided by both the 3D input and the LLM output. ScanReason [62] interleaves grounding and reasoning during inference. However, none of these systems addresses language-guided object placement, which requires reasoning about free space, spatial relations, and asset properties. We introduce a 3D LLM baseline designed to model these factors.

3D object placement and scene generation. Early 3D layout work posed furniture arrangement as optimization over ergonomic or aesthetic constraints, sampling layouts that satisfy manually encoded rules and user constraints [36, 56]. Data driven methods replaced hand tuning with spatial priors learned from example scenes, yielding *placement masks* that reflect one-to-many valid locations [6, 8, 19]. Neural methods extend these priors to full indoor synthesis with scene graphs, transformers, or diffusion models [18, 38, 47, 49, 51, 53, 61], and hybrid systems mix learned priors with explicit geometric constraints [18, 55]. [14] pioneered the use of language as a control signal for 3D scene generation. Later systems parse text into spatial constraints or scene graphs to infer feasible regions [6–8, 34]. Recent LLM and VLM based methods broaden this paradigm [20, 44, 47, 55]. When it comes to real-world (*i.e.* non-synthetic) scenes, most existing methods are image-based, predicting plausible 2D placement regions [33, 39, 40, 60, 63]. RoboPoint [58] extends this by performing language-guided placement in images and then lifting the results to 3D using depth. Concurrent work FirePlace [24] focuses on synthetic, clean environments. In contrast, we tackle language-guided placement in reconstructed real-world scenes, which requires intrinsic 3D reasoning, no privileged test-time annotations, and joint understanding of scene geometry, object shape, and instruction intent. Inspired by optimization methods, our dataset and benchmark ensure geometric feasibility for all candidate placements.

Datasets for language-guided 3D tasks. Most language and 3D datasets based on ScanNet [15] focus on tasks like text grounding [1, 2, 10, 26], VQA [4], captioning [12], and instruction following [59]. Larger real-scene corpora [3, 50, 64] and synthetic embodied benchmarks [37, 41, 52] expand this scope. Yet none address language-guided object placement with its one-to-many valid solutions and explicit valid placement enumeration; our ScanNet-based benchmark and dataset aim to fill this gap.

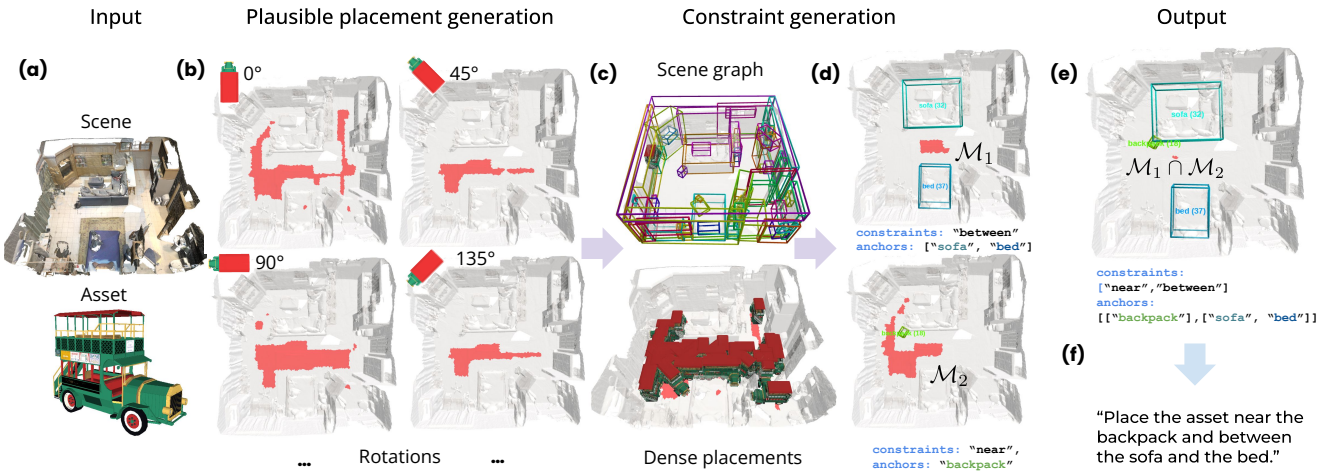

Figure 2. **PLACEIT3D-dataset creation**. Given a scene and an asset as input **(a)** the goal is to create a prompt **(f)** and corresponding mask $\mathcal{M}$ of valid placements **(e)**. We start by finding the set of points which are physically plausible placements, shown in red in **(b)**. We consider eight equally spaced rotation angles, which condition the valid placements. For this example, angle $0°$ has more valid placements than $45°$. To generate the language constraints, we use the ground truth scene graph **(c)**. Object anchors are selected from the scene graph and combined with relationship types to create a constraint and corresponding validity mask **(d)**. The different placement constraints are combined in the final output by intersecting the validity masks **(e)** given a mask of valid *dense* placements. Based on each selected set of anchors and constraint relationships, a natural language prompt is created using templates (please, see supplemental for more details).

3. Language-Guided 3D Object Placement

We introduce the task of language-guided 3D object placement on 3D reconstructed scenes. Given the point cloud of scene, a 3D asset, and text describing where the asset should be placed in the scene, the goal is to find a valid position and orientation for the asset that is physically plausible and adheres to the language prompt.

This task is inherently ambiguous because, in general, multiple valid placements exist. The multiple placements in Fig. 2 (c) demonstrate this ambiguity and illustrate the complexity of our task when compared with related tasks like object grounding, which typically has a single solution.

Simplifying assumptions. Given the ambiguity and complexity of our task, we make some simplifying assumptions to make the problem tractable. First, we assume the vertical orientation of the scene is fixed and given by the Z-axis. We also assume we know the vertical orientation of the asset as well as its frontal direction. The asset is always placed on a horizontal surface, and only the yaw angle is considered, *i.e*. rotation around the vertical axis.

3.1. Physical plausibility and language constraints

Valid placement in this task must satisfy a core set of common constraints. The first, **physical plausibility**, requires that the object does not intersect the scene mesh and rests on a surface. This constraint is language-independent and always enforced.

Beyond physical plausibility, the language instruction specifies how the object should be placed in the scene. Placement is often relative to an "anchor", a named object instance that must be inferred at test time. In practice, "language constraints" capture both semantic and physical aspects of the placement. These language constraints are organized into three distinct groups:

Spatial constraints: These constraints specify the object's location relative to one or more scene anchors. This group includes: (i) *near* and *adjacent*: the object is positioned within a specified distance from an anchor. (ii) *on*: the object should directly rest on top of an anchor. (iii) *between*: the object must be placed between two anchors. (iv) *above* and *below*: the object is located above or below an anchor.

Rotational constraint: This constraint focuses on the orientation of the object relative to scene anchors. The object is positioned so that it faces the anchor.

Visibility constraints: The object is either within the anchor's line of sight (*visible*) or hidden from it (*not visible*).

A candidate placement is considered a valid placement if and only if it simultaneously satisfies every constraint in that prompt.

Prompt creation:We generate language prompts using a template-based system, where each constraint is expressed through predefined sentence templates. A random subset of constraints and anchor objects from ScanNet annotations is sampled for each prompt. Prompts that cannot be satisfied due to conflicting or overly restrictive constraints, such as "place the asset on the table and below the desk", are discarded during a verification step. Template details are provided in the supplemental material.

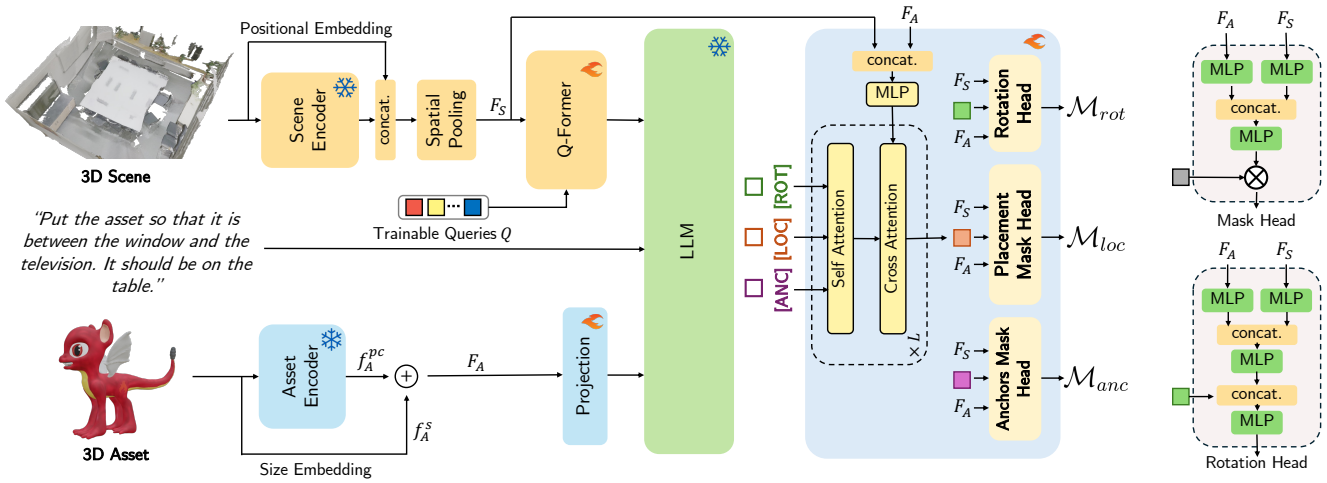

Figure 3. **PLACEWIZARD overview.** A point encoder extracts features from the 3D scene, which are then complemented with positional embeddings. Spatial pooling reduces feature dimensions, and a Q-Former merges the pooled features with trainable queries Q (Section 4.1). The asset is encoded into a single vector by using a pretrained asset encoder followed by max-pooling (Section 4.2). This vector together with a size embedding is passed to a projection layer that aligns the features with the LLM space. The LLM take as input (i) the output of the Q-Former, (ii) the text prompt, and (iii) the projected asset features and predicts three special tokens [ANC], [LOC] and [ROT]. A transformer based decoder takes as input the features associated with the three special tokens and the pooled scene features and performs a few self and cross attention operations (Section 4.3). Three heads produce the final outputs: $\mathcal{M}_{loc}$ the valid placement mask; $\mathcal{M}_{anc}$ an auxiliary mask that localizes the object anchors; and $\mathcal{M}_{rot}$ a mask indicating which rotation angles are valid at each location.

3.2. PLACEIT3D-benchmark

Each benchmark example consists of a 3D scene mesh, a 3D asset, and a language prompt comprising one or more 3D placement constraints. A placement method takes this triplet as input and predicts a placement, defined by a 3D translation vector $\mathbf{t}$ and a yaw angle α. Our evaluation protocol checks whether each constraint and all constraints collectively are satisfied.

3.2.1. Checking validity of each 3D constraint

We use a rule-based system to verify if a predicted placement satisfies 3D constraints, using ground truth oriented bounding boxes from ScanNet for anchor objects. Each constraint type is evaluated based on its specific properties, often allowing small deviations via thresholds.

Physical plausibility: We use `trimesh` [16] to check for intersections between the object and the scene and whether the object is placed on a surface.

Spatial constraints: For the *near* and *adjacent* constraints, we compute the distance from the posed asset to the anchor object. We check the *on*, *above*, and *below* relationships by comparing the value of the z-coordinate of the placed object with the z-coordinate of the anchor object. For the *between* relationship, we check if the placed asset is close to a line connecting the centers of the two anchor objects.

Rotational constraint: We compute a cone around the frontal vector of the posed asset and check that the anchor object intersects with that cone.

Visibility constraint: To determine visibility from a given anchor, we render the object and scene from a camera at the anchor point facing the object, and check if any pixels in the image correspond to the object.

More details are available in the supplemental material.

3.2.2. Benchmark metrics

To evaluate placement performance, we compute metrics that capture constraint validity overall and by subgroup:

- **Global Constraint Accuracy:** The percentage of all constraints (across all groups) that are correctly satisfied over the entire dataset. It provides a holistic measure of the overall placement quality.
- **Complete Placement Success:** The percentage of perfect valid placements, where every constraint—including physical plausibility—is satisfied. This is a strict metric that reflects the robustness of the placement method under full constraint satisfaction.
- **Language Adherence Success:** The percentage of placements that satisfy all language-based 3D constraints, excluding physical plausibility. It measures whether the model fully adheres to the language instructions.
- **Subgroup Metrics:** In addition to the overall metrics, we report accuracies across constraint groups.

3.2.3. Benchmark statistics

The benchmark contains 3,500 evaluation examples, combining a total of 142 different scenes from ScanNet [15] and 20 different assets from the PartObjaverse-Tiny

dataset [54]. Statistics for the number of constraints and type of constraints are shown in Table 1.

Constraints per sample	#	Type	#
One	900	Spatial	4,208
Two	1,871	Rotational	1,503
Three or more	729	Visibility	1,210

Table 1. Benchmark statistics for the number and types of language constraints per sample. Physical plausibility is evaluated in all samples and thus excluded from this table.

3.3. PLACEIT3D-dataset: Training dataset

Although our benchmark protocol allows offline method evaluation, we need a practical, less computationally costly approach to create a large-scale dataset for training, especially for obtaining the full set of valid placements.

Here we describe PLACEIT3D-dataset, our training dataset for the task of guided placement. The dataset consists of 100,505 training examples, sourced from 565 distinct ScanNet scenes and 20 unique assets. It includes a total of 83,530 spatial constraints, 34,445 rotational constraints, and 18,746 visibility constraints. Among these examples, 65,586 contain a single constraint, 26,395 have two constraints, and 8,524 include three or four constraints. Used throughout this paper, this training dataset is a subset, for practical purposes, of the PLACEIT3D-dataset-full corpus we are also sharing. PLACEIT3D-dataset-full has $\sim$4M examples: the 565 scenes x 140 objects x 50 prompts.

Dataset parametrization We denote the point cloud of the scene as $\mathbf{X} \in \mathbb{R}^{N\times 6}$, where each point $\mathbf{x}_i, i \in \{0, ..., N-1\}$ contains the 3D position for that point, as well as color information. Given a scene, an asset, and a prompt, we represent the set of valid ground truth placements for the asset as a binary mask $\mathcal{M}$ defined over the point cloud of the scene, associating a label $m_i \in \{0, 1\}$ to each 3D point $\mathbf{x}_i$. For each point i with label $m_i = 1$, *i.e.* a valid placement, we also define a binary mask over a discretized set of yaw angles indicating if the angle is valid for that specific location: $\alpha_i = \{\alpha_i^y \in \{0, 1\} | y = 0, ..., 7\}$, where each y corresponds to a $45°$ interval. Note that there is a fixed transform between the parametrizations used in the benchmark and the training dataset. While for the benchmark we parametrize the position of the center of the asset, for the training set, we consider contact points between the scene geometry and the asset's bottom surface.

Computing valid placement masks We create the valid placement masks $\mathcal{M}$ by using a combination of the rule-based system defined above and a few approximations to make it more efficient. More details on the approximations are available in the supplemental material. We treat each constraint independently, obtaining a valid mask per constraint $\mathcal{M}_c$ with $c \in \mathcal{C}$, where $\mathcal{C} = \{\text{physical}, \text{spatial}, \text{rotational}, \text{visibility}\}$. The final mask is given by the intersection of all the constraints that apply to that example, so

$$\mathcal{M} = \bigcap_{c\in\mathcal{C}} \mathcal{M}_c. \quad (1)$$

For the physical plausibility constraint we use a set of heightmaps to capture the different horizontal surfaces of the scene. We then compute the asset height and footprint and, for each point on a horizontal surface, check if the placement is valid. For the visibility constraint we use the same procedure as the benchmark, but use two approximations for efficiency: the asset is replaced by its bounding box, and a fixed rotation angle is used.

4. PLACEWIZARD: Method Description

Background. We briefly introduce Reason3D [25] as our method builds upon it. Given a textual prompt and a colored point cloud $\mathbf{X} \in \mathbb{R}^{N\times 6}$ as input, Reason3D performs dense 3D grounding, finding all the points in the point cloud that satisfy the prompt. A point encoder [45] extracts features $F_X \in \mathbb{R}^{N\times d}$ from the input point cloud, where d is the feature dimension. These are aggregated into superpoints [28] obtaining superpoint features $F_S \in \mathbb{R}^{M\times d}$, with $M \ll N$, reducing the overall complexity.

Next, the superpoint features F_s are projected into the embedding space of an LLM via a Q-Former block [31]. This model updates the learnable query vectors Q, resulting in Q'. From Q' and the input text, the LLM generates a response containing two special tokens, namely [LOC] and [SEG]. These tokens guide the model in two stages: coarse localization followed by precise mask prediction.

In practice, the Reason3D method uses a single token, [LOC], for datasets that contain small scenes, such as ScanNet, since hierarchical subdivision is not required. We will describe their method using this simplified version.

Finally, the last-layer embeddings associated to [LOC] are first projected via an MLP and then given as input to the Mask Decoder, which performs cross-attention [48] with F_s. The decoder produces an object-level binary segmentation mask over superpoints, which is upsampled into $\mathcal{M}_{loc} \in \{0, 1\}^N$ to provide a segmentation mask on the full point cloud.

Figure 3 provides an overview of our method. In the following subsections, we detail our approach and emphasize the key modifications to the Reason3D architecture necessary for addressing *language-guided placement instead* of standard 3D visual grounding.

4.1. Scene encoding

Similarly to Reason3D, we use the point encoder from [45] to extract features $F_X \in \mathbb{R}^{N \times d}$ from the 3D scene. We use an additional positional embedding feature $F_X^{pos} \in \mathbb{R}^{N \times d^*}$, for points in the point cloud, encoding their location, which is concatenated with the previous features.

Spatial pooling. Reason3D uses Superpoints [28] to reduce computational complexity and memory usage by pooling individual point features into a single feature per superpoint. Although effective for their task, this coarse representation limits performance for our placement task.

For example, Superpoints will generally cluster all points belonging to horizontal or vertical surfaces –such as floors, tabletops and walls– into single Superpoints, which is clearly undesirable for accurate 3D placement of assets. To address this, we instead use uniform spatial pooling to aggregate features. Specifically, we use farthest point sampling [21] to select M center points, then assign each point in the cloud to its nearest center based on Euclidean distance. This approach keeps the method computationally efficient while maintaining sufficient granularity for accurate asset placement. The level of granularity reflects a trade-off between computational cost and the model's ability to reason over fine-grained geometric details. This trade-off is evident in Table 2, where comparing row A with row B shows that finer sampling enables better spatial reasoning. Please see supplementary material for a visualization.

Our spatially aggregated features F_S are passed as input to the Q-Former block [31], which also takes as input a set of trainable queries and learns to project the features into the LLM embedding space.

4.2. Asset encoding

When compared with other tasks, our language-guided placement task has an additional input, the 3D asset point cloud. We encode the asset using a Point-BERT encoder [57] trained on the Objaverse [17] dataset. This encoder predicts a sequence of feature vectors that are max-pooled to obtain a single feature embedding.

Encoding the scale of the input asset is essential to facilitate a valid placement. Since the asset encoder assumes a normalized point cloud in a unit sphere, we separately encode the size of the asset by taking the asset's dimensions in the X, Y, and Z axes. The F_A feature for the asset is a combination of the asset encoding and scale embeddings and is projected to the LLM space using an MLP.

4.3. Placement decoder

We instruct our LLM to output three special tokens, namely a [LOC] token, an [ANC] token, and a [ROT] token. The features associated with the three special tokens are passed as input to the decoder, where they undergo a few self-attention layers. These are followed by a few cross-attention layers between the updated token features and the asset features F_A and the pooled scene features F_S.

Each individual head takes the feature of the associated token after attention, the asset feature F_A, and the scene feature F_S and predicts the corresponding output. The **Placement Mask Head** takes the [LOC] token embedding and predicts $\mathcal{M}_{loc} \in [0,1]^N$, a mask over the scene point cloud encoding the regions where the input asset can be placed satisfying the input prompt. The **Rotation Head** takes the [ROT] token embedding and predicts $\mathcal{M}_{rot} \in [0,1]^{N \times 8}$ indicating for each point in the point cloud, the validity of a discretized set of rotation angles. Finally, the **Anchors Mask Head** takes the [ANC] token and predicts $\mathcal{M}_{anc} \in [0,1]^N$, a mask encompassing the masks of all the anchor objects. This is used only as an auxiliary task, to help the network identifying anchors in the prompt. The head architectures are depicted on the right side of Figure 3.

4.4. Losses

We use a combination of Binary Cross Entropy (BCE) and Dice [43] losses when comparing a ground truth mask $\bar{\mathcal{M}}$ with a predicted mask $\mathcal{M}$, so

$$\mathcal{L}_{seg}(\bar{\mathcal{M}}, \mathcal{M}) = \text{BCE}(\bar{\mathcal{M}}, \mathcal{M}) + \text{Dice}(\bar{\mathcal{M}}, \mathcal{M}). \quad (2)$$

The loss for the rotation prediction is given by

$$\mathcal{L}_{rot} = \text{BCE}(\bar{\mathcal{M}}_{rot}, \mathcal{M}_{rot}), \quad (3)$$

where $\bar{\mathcal{M}}_{rot} \in \mathcal{M}_{rot} \in \{0,1\}^{N \times 8}$ is the ground truth indicator mask for valid rotation angles, per point in the scene point cloud.

The loss for the LLM is a cross-entropy loss, comparing the ground truth text $\bar{Y}$ with the predicted text Y: $\mathcal{L}_L = \text{CE}(\bar{Y}, Y)$. Note that the ground truth text $\bar{Y}$ for our task, follows a simple format, *e.g.* "Sure, it is [LOC][ANC][ROT]", since the LLM is not required to predict articulated responses or explain placement decisions. Instead, the information useful for placement should be encoded in the embeddings for the special tokens. Finally, our total loss is defined as

$$\mathcal{L} = \mathcal{L}_{seg}(\bar{\mathcal{M}}_{loc}, \mathcal{M}_{loc}) + \mathcal{L}_{rot} + \mathcal{L}_{seg}(\bar{\mathcal{M}}_{anc}, \mathcal{M}_{anc}) + \mathcal{L}_L. \quad (4)$$

4.5. Inference

At inference time, our method takes the network predictions for placement, $\mathcal{M}_{loc}$, and rotation, $\mathcal{M}_{rot}$, and extracts a single valid placement by finding the point in the point cloud with the maximum value in $\mathcal{M}_{loc}$: $\hat{\mathbf{x}} = \text{argmax}_{m \in \mathcal{M}_{loc}} m$. We apply a fixed offset to point $\hat{\mathbf{x}}$, half the asset height, to get the predicted 3D translation vector $\hat{t}$. This is due to the differences in parametrization between the training dataset and the benchmark. To predict the rotation angle, we use $\mathcal{M}_{rot}^{\hat{\mathbf{x}}} \in [0,1]^8$, which encodes the validity of discretized rotations for $\hat{\mathbf{x}}$. The predicted angle $\hat{\alpha}$ is obtained by taking the argmax over this vector.

Method ablation							Subgroup metrics				Global metrics		
Name	Spatial aggr.	Pos. emb.	Asset encoder	Anchor pred.	Rot pred.	Decode asset	Physical	Spatial	Rotational	Visibility	Language adherence success	Global constraint accuracy	Complete placement success
Baseline — OpenMask3D [46] + rules							**61.6**	28.6	6.5	53.4	21.8	29.2	11.7
Baseline — OpenMask3D [46] + LLM							5.8	35.3	10.5	**61.5**	18.4	26.7	1.6
A — Reason3D [25]	Superpoints	–	text	–	–	–	53.9	37.5	6.6	57.0	18.1	44.8	13.2
B	uniform	–	text	–	–	–	56.3	47.4	8.6	56.1	18.4	48.9	10.1
C	uniform	✓	text	–	–	–	58.6	49.7	7.9	59.1	20.0	50.4	10.9
D	uniform	✓	PointBert	–	–	–	57.9	47.1	7.5	58.5	17.2	49.3	9.6
E	uniform	✓	PointBert	✓	–	–	57.5	54.3	10.0	58.8	22.2	42.5	12.3
F	uniform	✓	PointBert	✓	✓	–	55.6	50.8	14.7	57.8	20.8	51.0	11.4
G — PLACEWIZARD	uniform	✓	PointBert	✓	✓	✓	58.8	**56.6**	**17.3**	61.2	**25.9**	**54.9**	**15.0**

Table 2. **Quantitative results:** We compare our full method with variations where some components are removed. The results validate our design choices, and they show improvements over OpenMask3D [46] with rule-based and LLM-based asset placement and Reason3D [25].

5. Experiments

We validate our method PLACEWIZARD for the task of language-guided object placement on the benchmark described in Section 3.2. Our metrics, described in Section 3.2.2, measure prediction validity. All values are percentages, where higher is better. Implementation details are in the supplemental material.

5.1. Quantitative results

In the absence of prior work on language-guided 3D object placement in real scenes, we implemented two baselines by integrating OpenMask3D [46], an open vocabulary grounding method, with two different placement strategies: (1) a rule-based optimization approach, similar to the one used during training data generation in Section 3.3, and (2) an LLM-based system that uses GPT-4o, with the estimated scene graph provided as part of the input prompt. We evaluate both baselines against our proposed method. Since OpenMask3D requires explicit object queries, we use ground truth anchor descriptions (rather than full placement instructions) to retrieve relevant regions. Due to its frequent failure to accurately detect floor regions, we substitute in ground truth floor masks, while other anchor objects are selected based on the highest similarity scores. Once anchor masks are obtained, asset placement is performed either by the rule-based strategy or inferred via the LLM.

Table 2 presents both baseline comparisons and ablation results. Our method, row G, consistently outperforms both baselines across all overall evaluation metrics. The LLM-based system often produces physically implausible placements, primarily because it lacks direct access to the 3D geometry of the scene. In contrast, the rule-based system, which leverages both asset and scene meshes, can produce more plausible placements, albeit at the cost of expensive collision checks during inference. By comparison, our method's end-to-end design eliminates the need for such costly test time operations, making it more scalable for large and complex scenes. Additionally, we observe that our method more accurately follows the user's language instructions than either baseline. Finally, the relatively low scores across all methods under the strictest evaluation metric, Complete Placement Success, which requires both physical plausibility and full adherence to all language constraints, highlight the inherent difficulty of the task.

5.1.1. Ablations

Table 2 also shows results for different ablations of our method. We start with an adaptation of the Reason3D [25] model to our task. One by one, we incrementally modify it using our novel components. Each row in the table introduces a single new modification, as compared to the previous row. We evaluate and report the model's performance until we reach PLACEWIZARD, our final method. All models are trained on our training dataset PLACEIT3D-dataset. For the methods that do not predict rotation (rows A, B, C, D, and E) we set the predicted rotation angle to 0. We describe the different variants below.

A. The asset dimensions are encoded in text and provided as part of the prompt: "*The asset dimensions are X Y Z cm*", where X, Y, and Z are integer values in cm.

B. This variant uses our proposed uniform spatial pooling approach instead of the original superpoints pooling.

C. Positional embedding features F_X^{pos} for points in the point cloud are added to the scene encoding.

D. We incorporate the asset encoder instead of only providing the asset dimensions in the text prompt to the LLM.

E. We introduce the anchor prediction auxiliary loss. In [1], predicting anchor objects leads to better 3D visual grounding. We find that this holds for our task as well.

F. We introduce a rotation prediction head, enabling the model to predict also the rotation mask $\mathcal{M}_{rot}$.

G. This variant defines our final method, where the asset feature F_A is added as an additional input to the placement decoder. This integration helps the decoder reason more effectively about the asset's geometry in relation to the scene.

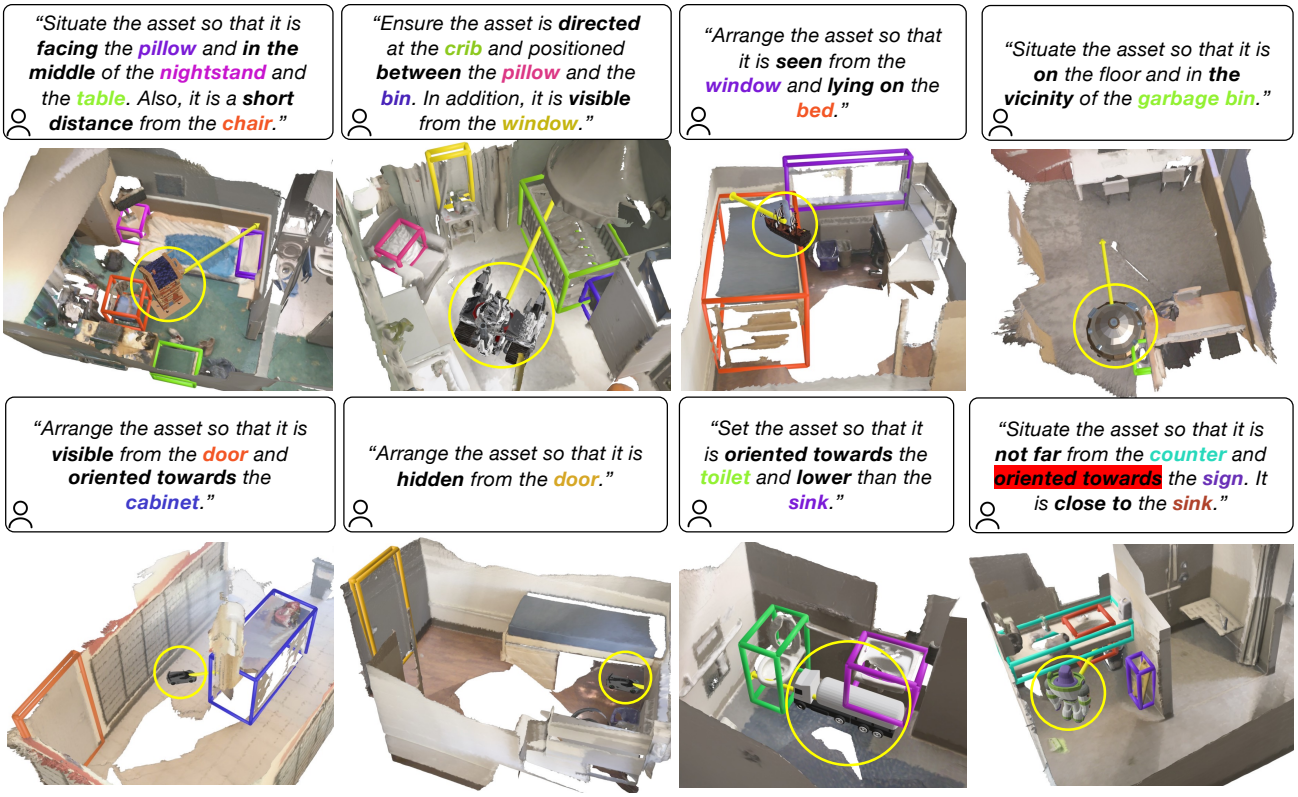

Figure 4. **Qualitative benchmark results**. Colored highlights indicate anchors referenced in the textual prompts (predictions are generated entirely from point clouds, with anchor information provided only as text). The asset position is marked with a yellow circle, and a yellow arrow denotes the frontal orientation. Our method successfully follows language instructions and meets the specified constraints. The top-right example illustrates a placement that satisfies constraints but slightly intersects with the scene mesh. The bottom-right example demonstrates a failure case where one constraint is not met (highlighted in red).

The results in Table 2 validate our design choices. Using spatial aggregation instead of superpoints improves over almost all metrics (compare row B with row A). The inclusion of the anchor prediction head as an auxiliary sub-task also improves performance (row E vs row D). Finally, the use of our rotation head combined with passing the asset encoding as input to the decoder gives our final best-performing method (row G, which we use in the qualitative results).

5.2. Qualitative Results

In Figure 4, we show the results of our method PLACEWIZARD on benchmark examples, demonstrating its ability to follow language instructions and satisfy constraints. While most placements are accurate, some cases exhibit minor intersections with the scene mesh or constraint failures.

6. Limitations and Future Work

Our novel task formulation currently has several limitations. First, we focus exclusively on placing objects on horizontal surfaces. Extending this to support arbitrary contact points would enable broader applications, such as hanging a clock on a vertical wall. Second, our dataset and method do not address inconsistencies in language guidance, where instructions may not align with the actual scene. Additionally, both the dataset and the benchmark rely on synthetic rule-based optimization without human verification. This limits annotation quality, especially in edge cases, and consequently affects both prediction accuracy and evaluation reliability. Despite these limitations, we believe our work lays the groundwork for further research in this area. Our method can also be seen as a specialist model, as it is trained and evaluated solely on the guided placement task. Exploring how to integrate this task into a more generalist framework remains an important direction for future work.

7. Conclusion

We introduced a new task, benchmark, and dataset for language-guided object placement in real 3D reconstructed scenes, connecting natural language understanding with spatial reasoning over both scenes and assets. The benchmark is designed to reflect the inherent ambiguity of placement, where multiple valid solutions are possible. We also proposed a baseline method built on recent advances in 3D LLMs, supported by ablation studies that highlight the impact of key architectural and design choices. Benchmark results show considerable room for improvement, and we hope this work provides a foundation for advancing research in 3D spatial reasoning.

8. Acknowledgments

We thank Jakub Powierza and Stanimir Vichev for their help with the experimental infrastructure. This work was partially supported by funding from King Abdullah University of Science and Technology (KAUST) – Center of Excellence for Generative AI, under award number 5940.

References

[1] Ahmed Abdelreheem, Kyle Olszewski, Hsin-Ying Lee, Peter Wonka, and Panos Achlioptas. Scanents3d: Exploiting phrase-to-3d-object correspondences for improved visio-linguistic models in 3d scenes. In *WACV*, 2024. 2, 7

[2] Panos Achlioptas, Ahmed Abdelreheem, Fei Xia, Mohamed Elhoseiny, and Leonidas Guibas. Referit3d: Neural listeners for fine-grained 3d object identification in real-world scenes. In *ECCV*, 2020. 2

[3] Sergio Arnaud, Paul McVay, Ada Martin, Arjun Majumdar, Krishna Murthy Jatavallabhula, Phillip Thomas, Ruslan Partsey, Daniel Dugas, Abha Gejji, Alexander Sax, Vincent-Pierre Berges, Mikael Henaff, Ayush Jain, Ang Cao, Ishita Prasad, Mrinal Kalakrishnan, Michael Rabbat, Nicolas Ballas, Mido Assran, Oleksandr Maksymets, Aravind Rajeswaran, and Franziska Meier. Locate 3d: Real-world object localization via self-supervised learning in 3d, 2025. 2

[4] Daichi Azuma, Taiki Miyanishi, Shuhei Kurita, and Motoaki Kawanabe. Scanqa: 3d question answering for spatial scene understanding. In *CVPR*, 2022. 2

[5] Wenxiao Cai, Yaroslav Ponomarenko, Jianhao Yuan, Xiaoqi Li, Wankou Yang, Hao Dong, and Bo Zhao. Spatialbot: Precise spatial understanding with vision language models. *arXiv 2406.13642*, 2024. 2

[6] Angel Chang, Manolis Savva, and Christopher D Manning. Learning spatial knowledge for text to 3d scene generation. In *EMNLP*, 2014. 2

[7] Angel Chang, Will Monroe, Manolis Savva, Christopher Potts, and Christopher D Manning. Text to 3d scene generation with rich lexical grounding. In *ACL-IJCNLP*, 2015.

[8] Angel X Chang, Mihail Eric, Manolis Savva, and Christopher D Manning. Sceneseer: 3d scene design with natural language. *arXiv*, 2017. 2

[9] Boyuan Chen, Zhuo Xu, Sean Kirmani, Brain Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. SpatialVLM: Endowing vision-language models with spatial reasoning capabilities. In *CVPR*, 2024. 2

[10] Dave Zhenyu Chen, Angel X Chang, and Matthias Nießner. Scanrefer: 3d object localization in rgb-d scans using natural language. In *ECCV*, 2020. 2

[11] Sijin Chen, Xin Chen, Chi Zhang, Mingsheng Li, Gang Yu, Hao Fei, Hongyuan Zhu, Jiayuan Fan, and Tao Chen. Ll3da: Visual interactive instruction tuning for omni-3d understanding reasoning and planning. In *CVPR*, 2024. 2

[12] Zhenyu Chen, Ali Gholami, Matthias Nießner, and Angel X Chang. Scan2cap: Context-aware dense captioning in rgb-d scans. In *CVPR*, 2021. 2

[13] An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. SpatialRGPT: Grounded spatial reasoning in vision language model. In *NeurIPS*, 2024. 2

[14] Bob Coyne and Richard Sproat. Wordseye: An automatic text-to-scene conversion system. In *SIGGRAPH*, 2001. 2

[15] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 2, 4

[16] Dawson-Haggerty et al. trimesh, 2019. 4

[17] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *CVPR*, 2022. 6

[18] Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. Layoutgpt: Compositional visual planning and generation with large language models. In *NeurIPS*, 2023. 2

[19] Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. Example-based synthesis of 3d object arrangements. *ACM TOG*, 2012. 2

[20] Rao Fu, Zehao Wen, Zichen Liu, and Srinath Sridhar. Anyhome: Open-vocabulary generation of structured and textured 3d homes. In *ECCV*, 2024. 2

[21] Meng Han, Liang Wang, Limin Xiao, Hao Zhang, Chenhao Zhang, Xiangrong Xu, and Jianfeng Zhu. Quickfps: Architecture and algorithm co-design for farthest point sampling in large-scale point clouds. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023. 6

[22] Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 3D-LLM: Injecting the 3D world into large language models. In *NeurIPS*, 2023. 2

[23] Haifeng Huang, Zehan Wang, Rongjie Huang, Luping Liu, Xize Cheng, Yang Zhao, Tao Jin, and Zhou Zhao. Chat-scene: Bridging 3d scene and large language models with object identifiers. In *NeurIPS*, 2024. 2

[24] Ian Huang, Yanan Bao, Karen Truong, Howard Zhou, Cordelia Schmid, Leonidas Guibas, and Alireza Fathi. FirePlace: Geometric Refinements of LLM Common Sense Reasoning for 3D Object Placement. In *CVPR*, 2025. 2

[25] Kuan-Chih Huang, Xiangtai Li, Lu Qi, Shuicheng Yan, and Ming-Hsuan Yang. Reason3D: Searching and reasoning 3d segmentation via large language model. In *3DV*, 2025. 2, 5, 7, 13, 14

[26] 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 *ECCV*, 2024. 2

[27] Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. Lisa: Reasoning segmentation via large language model. In *CVPR*, 2024. 2

[28] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*, 2018. 5, 6, 14

[29] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. Llava-onevision: Easy visual task transfer. *arXiv 2408.03326*, 2024. 2

[30] Feng Li, Renrui Zhang, Hao Zhang, Yuanhan Zhang, Bo Li, Wei Li, Zejun Ma, and Chunyuan Li. Llava-next-interleave: Tackling multi-image, video, and 3d in large multimodal models. *arXiv 2407.07895*, 2024. 2

[31] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, 2023. 2, 5, 6

[32] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2024. 2

[33] Liu Liu, Zhenchen Liu, Bo Zhang, Jiangtong Li, Li Niu, Qingyang Liu, and Liqing Zhang. Opa: object placement assessment dataset. *arXiv 2107.01889*, 2021. 2

[34] Rui Ma, Akshay Gadi Patil, Matthew Fisher, Manyi Li, Sören Pirk, Binh-Son Hua, Sai-Kit Yeung, Xin Tong, Leonidas Guibas, and Hao Zhang. Language-driven synthesis of 3d scenes from scene databases. *ACM TOG*, 2018. 2

[35] Xianzheng Ma, Yash Bhalgat, Brandon Smart, Shuai Chen, Xinghui Li, Jian Ding, Jindong Gu, Dave Zhenyu Chen, Songyou Peng, Jia-Wang Bian, Philip H Torr, Marc Pollefeys, Matthias Nießner, Ian D Reid, Angel X. Chang, Iro Laina, and Victor Adrian Prisacariu. When llms step into the 3d world: A survey and meta-analysis of 3d tasks via multi-modal large language models, 2024. 2

[36] Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. Interactive furniture layout using interior design guidelines. *ACM TOG*, 2011. 2

[37] Aishwarya Padmakumar, Jesse Thomason, Ayush Shrivastava, Patrick Lange, Anjali Narayan-Chen, Spandana Gella, Robinson Piramuthu, Gokhan Tur, and Dilek Hakkani-Tur. Teach: Task-driven embodied agents that chat. In *AAAI*, 2022. 2

[38] Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. Atiss: Autoregressive transformers for indoor scene synthesis. In *NeurIPS*, 2021. 2

[39] Ram Ramrakhya, Aniruddha Kembhavi, Dhruv Batra, Zsolt Kira, Kuo-Hao Zeng, and Luca Weihs. Seeing the unseen: Visual common sense for semantic placement. In *CVPR*, 2024. 2

[40] Aditya Sharma, Luke Yoffe, and Tobias Höllerer. Octo+: A suite for automatic open-vocabulary object placement in mixed reality. In *AIXVR*, 2024. 2

[41] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *CVPR*, 2020. 2

[42] R. Siegler, J. Saffran, E. Gershoff, N. Eisenberg, and J. DeLoache. *How Children Develop*. Macmillan Learning, 2020. 1

[43] Carole Helene Sudre, Wenqi Li, Tom Kamiel Magda Vercauteren, Sébastien Ourselin, and M. Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *MICCAI workshop*, 2017. 6

[44] Fan-Yun Sun, Weiyu Liu, Siyi Gu, Dylan Lim, Goutam Bhat, Federico Tombari, Manling Li, Nick Haber, and Jiajun Wu. Layoutvlm: Differentiable optimization of 3d layout via vision-language models. In *CVPR*, 2025. 2

[45] Jiahao Sun, Chunmei Qing, Junpeng Tan, and Xiangmin Xu. Superpoint transformer for 3d scene instance segmentation. In *AAAI*, 2023. 5, 6

[46] Ayça Takmaz, Elisabetta Fedele, Robert W. Sumner, Marc Pollefeys, Federico Tombari, and Francis Engelmann. OpenMask3D: Open-Vocabulary 3D Instance Segmentation. In *NeurIPS*, 2023. 7

[47] Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. Diffuscene: Denoising diffusion models for generative indoor scene synthesis. In *CVPR*, 2024. 2

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 5

[49] Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X Chang, and Daniel Ritchie. Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM TOG*, 2019. 2

[50] Tai Wang, Xiaohan Mao, Chenming Zhu, Runsen Xu, Ruiyuan Lyu, Peisen Li, Xiao Chen, Wenwei Zhang, Kai Chen, Tianfan Xue, Xihui Liu, Cewu Lu, Dahua Lin, and Jiangmiao Pang. Embodiedscan: A holistic multi-modal 3d perception suite towards embodied ai. In *CVPR*, 2024. 2

[51] Qiuhong Anna Wei, Sijie Ding, Jeong Joon Park, Rahul Sajnani, Adrien Poulenard, Srinath Sridhar, and Leonidas Guibas. Lego-net: Learning regular rearrangements of objects in rooms. In *CVPR*, 2023. 2

[52] Jianing Yang, Xuweiyi Chen, Nikhil Madaan, Madhavan Iyengar, Shengyi Qian, David F. Fouhey, and Joyce Chai. 3d-grand: A million-scale dataset for 3d-llms with better grounding and less hallucination. *arXiv 2406.05132*, 2024. 2

[53] Xiuyu Yang, Yunze Man, Junkun Chen, and Yu-Xiong Wang. Scenecraft: Layout-guided 3d scene generation. In *NeurIPS*, 2024. 2

[54] Yunhan Yang, Yukun Huang, Yuan-Chen Guo, Liangjun Lu, Xiaoyang Wu, Lam Edmund Y., Yan-Pei Cao, and Xihui Liu. Sampart3d: Segment any part in 3d objects. *arXiv 2411.07184*, 2024. 2, 5

[55] Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, Chris Callison-Burch, Mark Yatskar, Aniruddha Kembhavi, and Christopher Clark. Holodeck: Language guided generation of 3d embodied ai environments. In *CVPR*, 2024. 2

[56] Lap-Fai Yu, Sai Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F Chan, and Stanley J Osher. Make it home: automatic optimization of furniture arrangement. *ACM TOG*, 2011. 2

[57] 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 *CVPR*, 2022. 6

[58] Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics. In *CoRL*, 2024. 2

[59] Haochen Zhang, Nader Zantout, Pujith Kachana, Zongyuan Wu, Ji Zhang, and Wenshan Wang. Vla-3d: A dataset for 3d semantic scene understanding and navigation. In *RSS Workshop*, 2024. 2

[60] Hengshuang Zhao, Xiaohui Shen, Zhe Lin, Kalyan Sunkavalli, Brian Price, and Jiaya Jia. Compositing-aware image search. In *ECCV*, 2018. 2

[61] Yang Zhou, Zachary While, and Evangelos Kalogerakis. Scenegraphnet: Neural message passing for 3d indoor scene augmentation. In *CVPR*, 2019. 2

[62] Chenming Zhu, Tai Wang, Kai Chen, and Xihui Liu. Scanreason: Empowering 3d visual grounding with reasoning capabilities. In *ECCV*, 2024. 2

[63] Sijie Zhu, Zhe Lin, Scott Cohen, Jason Kuen, Zhifei Zhang, and Chen Chen. Topnet: Transformer-based object placement network for image compositing. In *CVPR*, 2023. 2

[64] Ziyu Zhu, Xiaojian Ma, Yixin Chen, Zhidong Deng, Siyuan Huang, and Qing Li. 3d-vista: Pre-trained transformer for 3d vision and text alignment, 2023. 2

PLACEIT3D: Language-Guided Object Placement in Real 3D Scenes

Supplementary Material

9. Additional details on the training dataset

9.1. Training dataset creation

We give some details on the training set creation, particularly how the physically plausible constraint and visibility constraint are computed.

Spatial constraints Each constraint uses geometric criteria on 3D oriented bounding boxes and is governed by the following parameters. We use the same values both in the training dataset and the evaluation benchmark:

- **"near"**: asset in a proximity of the anchor object. The threshold distance is proportional to the size of the room (1%).
- **"adjacent"**: asset close to the anchor object. We set a tolerance distance of 3 cm.
- **"above" / "below"**: asset vertically aligned above / below the anchor object. Vertical Intersection over Minimum (IoM) $>=$ 0.5 and a minimum of 1 cm above/below the anchor.
- **"on"**: resting on top of the anchor object, considering vertical stacking and size constraints: vertical IoM $>=$ 0.5 and a tolerance for vertical distance of 1 cm.
- **"between"**: Determines if the asset object lies between two anchor objects in both xy and z planes. Parameters: between IoM (0.5) in projection planes (xy and z). Overlap threshold (0.3): maximum IoM that ensures the asset does not overlap excessively with either object. Distance threshold: filters anchors beyond 1.5 m

Rotational constraint. The "facing" constraint determines which objects an asset is oriented towards in a 3D scene by evaluating directional alignment, proximity, and spatial overlap. It uses the asset's front direction to identify candidate objects within its field of view. We use a maximum distance threshold of 2 meters, an angular threshold of 30 degrees and an IoM for lateral overlap of 0.5.

Physically plausible constraint The first constraint that we consider is whether an object can physically be placed at a particular location in a scene. To compute valid placements in a scene efficiently, we make use of a heightmap based representation, where we raycast the mesh from above using a grid of rays with a predefined resolution, and store all points of intersection. Next, we create a set of heightmaps, where each cell represents a different ray, each layer represents a different intersection per ray, and the value is the height of the intersection point. We construct the first heightmap using the intersection points with the minimum height per ray. Each subsequent heightmap will contain either the next intersection point for each ray, or if there are no remaining points for a cell, will contain the maximum intersection point. Additionally, we raycast each asset from above to obtain an asset heightmap and 2D bounding box per possible rotation. Given these heightmaps, we check for physical plausibility by:

- Extracting overlapping patches of the mesh heightmap, with patch size equal to the asset bounding box
- Extract minimum height and maximum height of the heightmap for each patch, using the asset heightmap to generate a mask. If this differs by more than 10cm, this point is not valid
- Check that the asset can also fit in the Z direction using the next heightmap - is the asset height for this cell less than the height of the next surface

If these 2 conditions are true, we deem a location to be physically plausible. Finally, we generate labels for the mesh vertices by assigning them the labels of their nearest location in the heightmap.

Visibility constraint Our visibility constraint determines whether an asset is visible from a specific location in the scene, which corresponds to one of the object anchors. To evaluate this, we use mesh rendering. To assess visibility efficiently, we first place the asset in a physically feasible position. Instead of rendering the full asset mesh, we approximate it using a simple cuboid with the same dimensions as the asset bounding box, reducing computational overhead, we also consider only 1 rotation for the asset. The virtual camera's position is then determined by computing the centroid of the vertices associated with the anchor. The camera center is set to the vertex within the anchor that is closest to this centroid, and the camera is oriented to face the asset.

We then render the scene and check whether any pixels from the asset's bounding cuboid appear in the rendered image. This process is repeated for all valid asset placements across all scenes. The virtual camera locations correspond to TVs, doors, and windows. When multiple instances of the same object class exist in a scene, we select the largest instance.

We use a virtual camera with a field of view (FOV) of 60° and we render images at a resolution of 64×64 pixels.

In the benchmark, we follow the same procedure as stated above for generating the training data with 2 differences: we render the original asset mesh instead of the

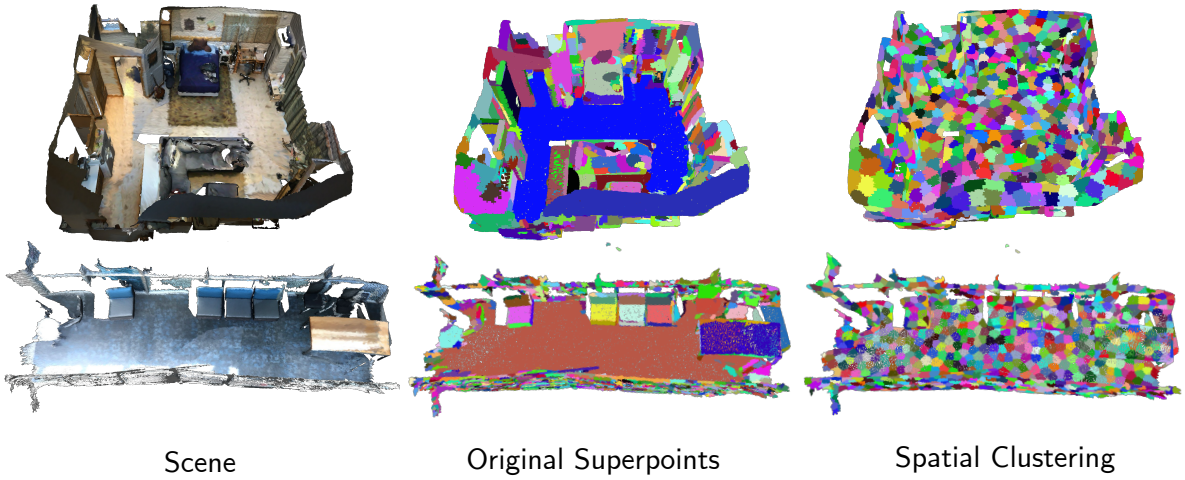

Figure 5. Comparison of our spatial pooling vs the superpoints used in [25]. Our regions are more local and more adequate to the task of object placement.

cuboid and render images at a resolution of 256×256 pixels.

9.2. Templates for prompts

We report the templates used to generate placement instructions.

```
relationships:
  - name: plausible
    templates:
    - in a plausible location
    - in a sensible location
    - in a reasonable spot
    - in a suitable position
    - in a feasible area
    - somewhere stable within the scene
    - at a steady spot in the scene
    - in a secure location within the scene
    - in a firm position in the scene
    - in an area that suits the scene's layout
  - name: adjacent
    templates:
      - adjacent to the {anchor_class}
      - next to the {anchor_class}
      - beside the {anchor_class}
      - right beside the {anchor_class}
      - alongside the {anchor_class}
      - abutting the {anchor_class}
  - name: between
    templates:
      - between the {anchor1_class} and the
          {anchor2_class}
      - in the space between the {anchor1_class}
          and the {anchor2_class}
      - positioned between the {anchor1_class} and
          the {anchor2_class}
      - in the middle of the {anchor1_class} and
          the {anchor2_class}
  - name: facing
    templates:
      - facing the {anchor_class}
      - directed at the {anchor_class}
      - pointing towards the {anchor_class}
      - oriented towards the {anchor_class}
      - looking at the {anchor_class}
      - angled toward the {anchor_class}
      - turned towards the {anchor_class}
  - name: near
    templates:
      - near the {anchor_class}
      - close to the {anchor_class}
      - in the vicinity of the {anchor_class}
      - not far from the {anchor_class}
      - within reach of the {anchor_class}
      - a short distance from the {anchor_class}
  - name: on
    templates:
      - on the {anchor_class}
      - resting on the {anchor_class}
      - placed on the {anchor_class}
      - sitting on the {anchor_class}
      - lying on the {anchor_class}
  - name: above
    templates:
      - above the {anchor_class}
      - over the {anchor_class}
      - higher than the {anchor_class}
      - up above the {anchor_class}
  - name: below
    templates:
      - below the {anchor_class}
      - under the {anchor_class}
      - beneath the {anchor_class}
      - underneath the {anchor_class}
      - lower than the {anchor_class}
      - situated under the {anchor_class}
      - right below the {anchor_class}
  - name: is_visible
    templates:
      - visible from the {anchor_class}
      - in view of the {anchor_class}
      - within sight of the {anchor_class}
      - seen from the {anchor_class}
      - not obstructing the view to the
          {anchor_class}
```

```
    - keeping the view to the {anchor_class}clear
    - positioned to avoid blocking the
        {anchor_class}
    - allowing an unobstructed view of the
        {anchor_class}
- name: not_visible
  templates:
    - not visible from the {anchor_class}
    - out of sight of the {anchor_class}
    - hidden from the {anchor_class}
    - obstructing the view to the {anchor_class}
    - blocking the view to the {anchor_class}
    - in the way of the {anchor_class}
    - preventing a clear view of the
        {anchor_class}
```

Dataset Examples In Figures 7 and 8, we provide examples from our proposed dataset.

10. PlaceWizard Implementation Details

We conduct all our experiments using eight NVIDIA Tesla A100 GPUs, with a training batch size of 28 per single gpu. Following Reason3D, we utilize the AdamW optimizer with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$, a weight decay of 0.05, and a linear warm-up strategy for the learning rate during the initial 1000 steps gradually increasing it from 10^{-8} to 10^{-4} followed by a cosine decay schedule. We train for 50 epochs. We also use a pretrained FlanT5XL model, keeping most of its pre-trained weights frozen, except for adapting the weights of the newly added tokens, as similarly done in Reason3D. For spatial pooling, we employ 1024 groups for each ScanNet scan.

11. Visualization of superpoints

Figure 5 shows the difference between the superpoints [28] used in Reason3D [25] and our proposed spatial pooling. While [28] generates large clusters, such as for the floor, our method produces clusters at a finer granularity.

12. Further Qualitative Results

In Figure 6 we show the confidence scores predicted by our model for the spatial clusters in two example scenes from our dataset.

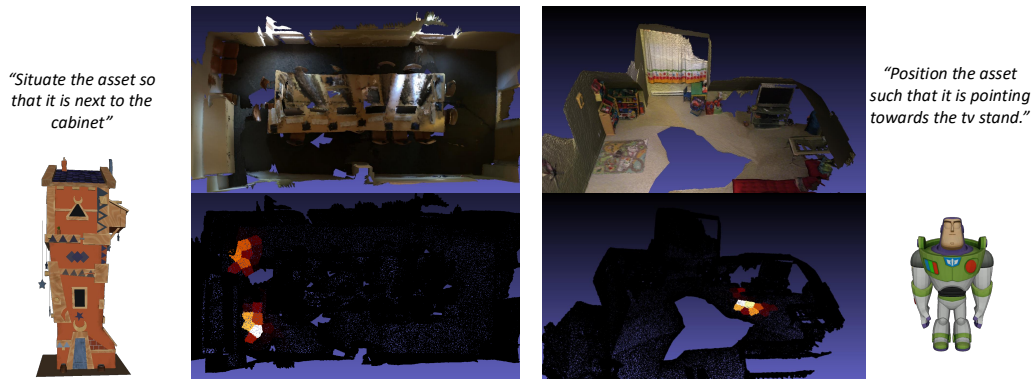

Figure 6. Heatmap visualization of the predicted confidence scores by our model for spatial clusters in two examples from our dataset, across two scenes, given different assets and textual prompts. Warmer colors indicate higher confidence regions for asset placement, with white representing the highest confidence.

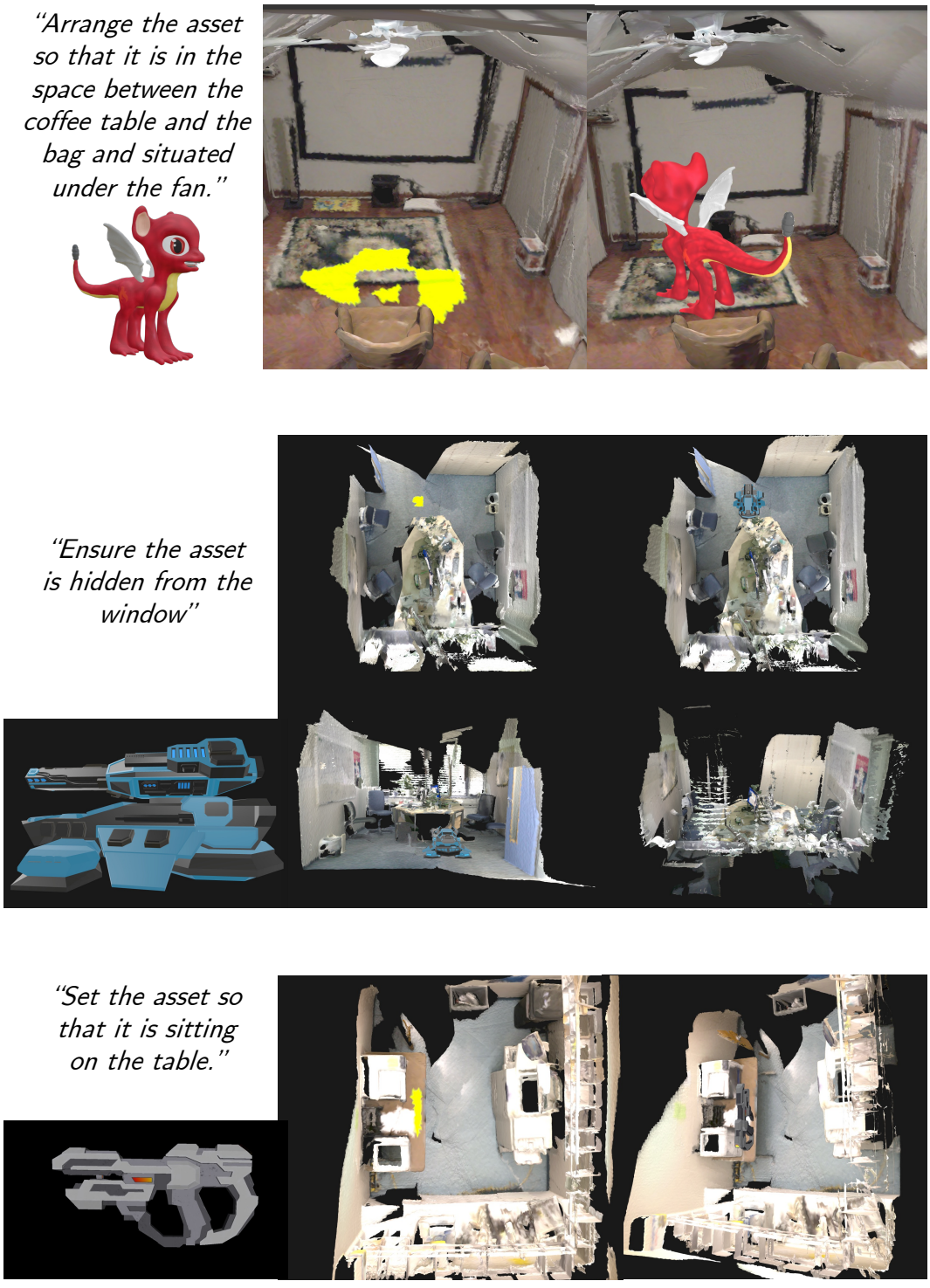

Figure 7. Examples from our proposed dataset illustrating prompts with different constraints, along with the corresponding placement mask and a sample placed asset.

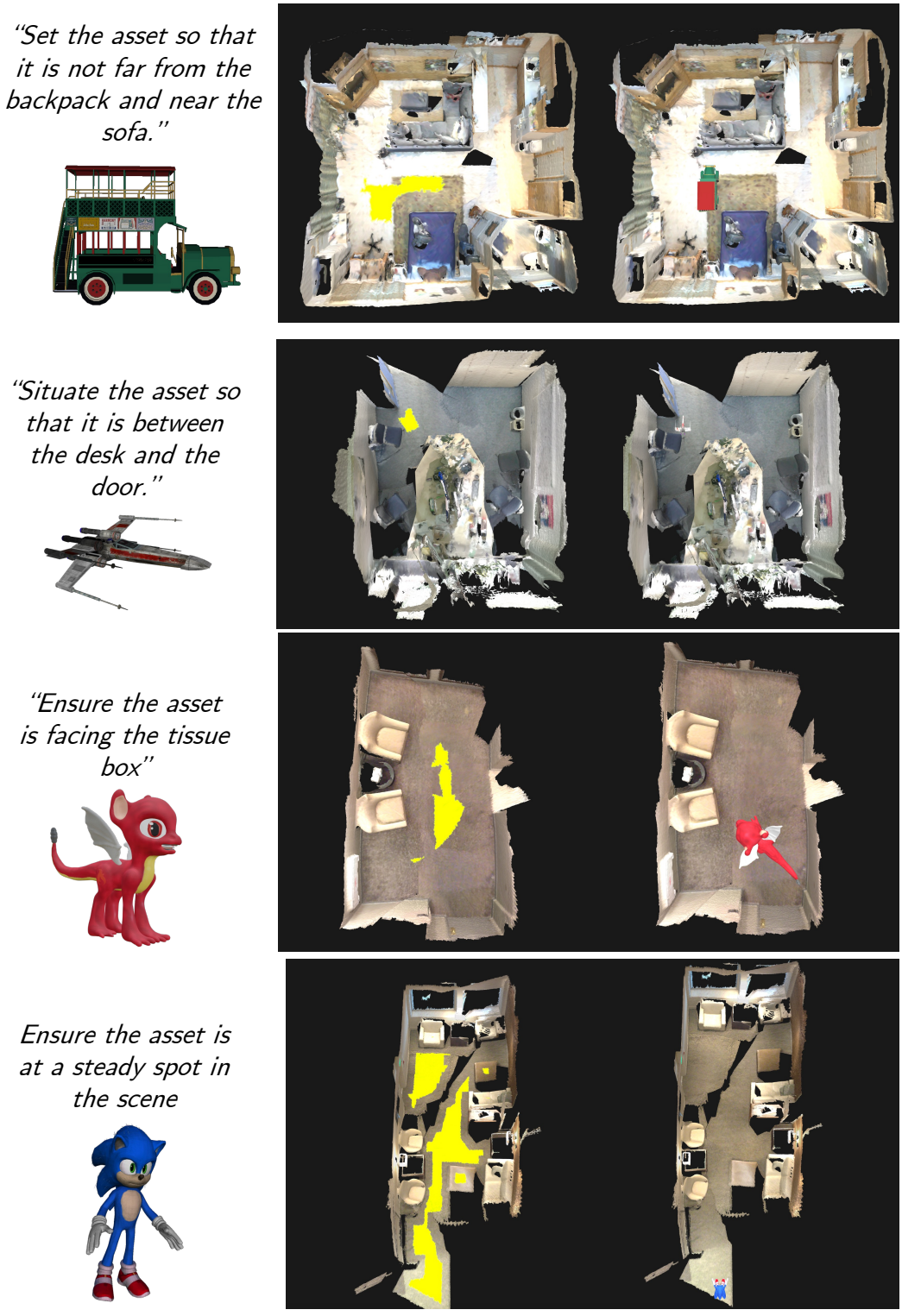

Figure 8. Examples from our proposed dataset illustrating prompts with different constraints, along with the corresponding placement mask and a sample placed asset.