

Towards a Training Free Approach for 3D Scene Editing

Vivek Madhavaram¹ Shivangana Rawat² Chaitanya Devaguptapu²
Charu Sharma¹ Manohar Kaul²

madhavaram.vardhan@research.iit.ac.in

¹ Machine Learning Lab, IIIT Hyderabad, India ² Fujitsu Research India

Abstract

Text driven diffusion models have shown remarkable capabilities in editing images. However, when editing 3D scenes, existing works mostly rely on training a NeRF for 3D editing. Recent NeRF editing methods leverages edit operations by deploying 2D diffusion models and project these edits into 3D space. They require strong positional priors alongside text prompt to identify the edit location. These methods are operational on small 3D scenes and are more generalized to particular scene. They require training for each specific edit and cannot be exploited in real-time edits. To address these limitations, we propose a novel method, *FreeEdit*, to make edits in training free manner using mesh representations as a substitute for NeRF. Training-free methods are now a possibility because of the advances in foundation model's space. We leverage these models to bring a training-free alternative and introduce solutions for insertion, replacement and deletion. We consider insertion, replacement and deletion as basic blocks for performing intricate edits with certain combinations of these operations. Given a text prompt and a 3D scene, our model is capable of identifying what object should be inserted/replaced or deleted and location where edit should be performed. We also introduce a novel algorithm as part of *FreeEdit* to find the optimal location on ground-ing object for placement. We evaluate our model by comparing it with baseline models on a wide range of scenes using quantitative and qualitative metrics and showcase the merits of our method with respect to others. Project page: https://vivekmadhavaram.github.io/FreeEdit_page/

1. Introduction

Existing 3D editing approaches [15, 34, 43] rely heavily on computationally expensive, trained-based methods. Advancements in foundation models have opened up new possibilities for training-free approaches in various 3D understanding tasks [14, 16, 40]. Despite these developments,

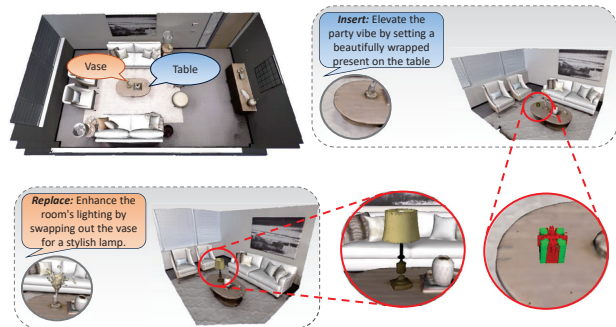


Figure 1. **Illustration of FreeEdit for inserting and replacing objects in a complex 3D scene:** Queries in blue and orange illustrate the insertion and replacement prompts provided as input by the user, respectively.

the task of scene editing has yet to fully leverage such advancements. Existing 3D editing approaches predominantly rely on training-based methods that require training a Neural Radiance Field (NeRF) for each new edit and scene.

The ability to edit large-scale three-dimensional (3D) environments has emerged as a pivotal area recently, driven by applications across Virtual Reality (VR) [12, 36], Augmented Reality (AR) [11, 26], interior designing [9, 21] among others [23, 27]. The ability to dynamically modify and enhance 3D scenes in real time is crucial for creating immersive experiences and personalizing spaces to individual preferences.

Most of the existing [15, 34, 43] methods that focus on editing 3D scenes predominantly focus on object-centric modifications or broad scene-level changes such as ambiance adjustments, often neglecting the nuanced requirements of detailed, fine-grained editing. Specifically, the insertion/replacement of objects within complex, multi-object 3D scenes remains inadequately supported. Moreover, these approaches typically require retraining for each new edit, which is both time-consuming and resource-intensive, making them ill-suited for quick or multiple iterative edits.

To address these challenges, we present *FreeEdit*¹, a *training free* approach for 3D scene editing. Our work pri-

¹The name is inspired by the training-free nature of our approach.

marily focuses on text-guided scene manipulation tasks, including insertion, replacement, and deletion. These fundamental operations serve as building blocks for a wide range of 3D editing tasks. Our main goal is to provide an efficient alternative to expensive training-based methods, leveraging the recent improvements in foundation models to enable near real-time edits within complex scenes comprising numerous objects.

FreeEdit operates solely based on text instructions and utilizes mesh representations, enabling efficient editing of room-sized 3D scenes. While NeRF-based representations excel in realistic rendering, our primary focus is on efficiency while maintaining acceptable quality, particularly when editing scenes with multiple objects where NeRF’s complexity and need for retraining become bottlenecks. Unlike most NeRF-based editing methods that focus on single-object or small-scale scenes, we target large-scale, room-sized 3D scenes.

Inspired by training-free approaches for various scene understanding tasks [14, 16, 40], FreeEdit employs a modular approach for inserting and replacing objects in large-scale 3D scenes (as shown in Figure 1). It supports iterative and real-time edits while eliminating the need for explicit location supervision when inserting objects, instead inferring locations based on user-provided textual instructions. By integrating open-vocabulary segmentation and 3D diffusion models, FreeEdit broadens the range of recognizable and insertable objects in any given scene, offering a versatile solution for 3D scene editing even when specific 3D assets are unavailable.

Our contributions: To the best of our knowledge, we are the first to introduce a training-free approach for 3D scene editing. *i)* We present a framework that enables text-driven object insertion, replacement and deletion in 3D scenes without the prerequisite of training. *ii)* Our framework is specifically designed to handle large-scale 3D scenes with multiple objects, offering unprecedented flexibility and efficiency in scene customization without the need for positional priors. *iii)* We introduce an algorithm as part of FreeEdit to find the optimal location to place the primary object in a scene with minimal intersection. *iv)* While FreeEdit is primarily designed for inserting and replacing objects in 3D scenes, by the nature of its design, it also supports interactive object translation, rotation, iterative insertion and object deletion.

It’s important to note that while FreeEdit represents a step towards a more versatile, training-free 3D scene editing, its current implementation focuses on a specific set of object-level manipulations. We acknowledge that this approach may not encompass all possible types of edits. However, by efficient solutions for insertion, replacement, deletion, translation and rotation, FreeEdit lays a foundation for developments in training-free 3D editing techniques.

2. Related work

Text-driven Editing in 2D Text-driven editing methods gained attention due to editing objects/scenes based on text prompts, thereby reducing manual labor. Recent advancements in diffusion models have notably improved the photo-realism and diversity of generated content, prompting researchers to explore their applications. InstructPix2Pix [3] exploits cross-attention layers mentioned in Prompt-to-Prompt [16], which trains models using paired images of before/after edits. In contrast, MasaCtrl [4] introduces mutual self-attention in diffusion models. Imagic [20] and ObjectStitch [35] make complex semantic edits such as posture or compositional editing of multiple objects. GLIDE [29] edits images in a photorealistic manner using classifier-free guidance in masked portions. DiffEdit [7] generates implicit masks from text instruction.

3D Scene Editing: Despite advancements in text-driven editing in 2D scenes, few methods have extended this to 3D environments using NeRF-based rendering due to limited training data. They can be divided into three categories. *i)* Prior guided, *ii)* Geometry/Texture editing and *iii)* Text guided insertion.

Prior Guided: Some scene editing techniques necessitate human supervision for precise region identification. SKED [28] utilizes human sketches from multiple viewpoints to localize edit regions. Set-the-Scene [6] adopts a proxy scene layout to perform local and global iterative edits, ensuring scene coherence. InseRF [34] inserts objects in the scene using a bounding box. While effective, these methods require human intervention, limiting their scalability and automation potential.

Geometry/Texture Editing: Contrary to earlier methods, few works try to edit the scene without any prior guidance. DreamEditor [43] represents scenes as mesh-based neural fields and makes edits using text-to-image diffusion models in the edit region identified by the text encoder. RePaint-NeRF [42] semantically selects target objects and leverages a pre-trained diffusion model to guide NeRF models in editing 3D objects. ViCA-NeRF [10], Instruct 3D-to-3D [19] and SINE [2] make edits to the scene through multi-view consistent 3D texture editing using [3]. These methods are capable of changing the geometry and texture of existing objects but cannot insert new object into the scene.

Text-guided Object Insertion: The above methods are restricted to geometry/textural editing but cannot insert/replace things in scene. Instruct-NeRF2NeRF [15] addresses this by iterative editing a set of images to perform a single edit. Vox-E [33] learns a grid-based volumetric representation of images captured from 3D objects for making edits in 3D space. FusedRF [13] performs objects and scene compositing using single RF through distillation. Control-NeRF [22] edits and manipulates scene by inserting objects across different scenes or multiplying objects within same

scene. For every edit, it needs re-training, which takes ample time, making them unfit for real-time edits.

There are quite a few fast variants of NeRF that primarily are confined to scene reconstruction but not to scene edits. Also, they fail to process complex prompts that involve open vocabulary because they do not have semantic and global information about scene and depend on image edits. NeRF models face difficulty in converging the diverse edits from image to image based on complexity of natural prompt. Moreover, the main goal of this work is to present a training-free alternative for scene editing.

3. Method

This section presents our proposed text-guided 3D object insertion and replacement method. FreeEdit takes a 3D scene and a text prompt as inputs specifying which object should be inserted/replaced in a scene. Input text is processed for task categorization and entity extraction. Entity extraction carried out on text helps in object generation and finding the best suitable location in the scene. We apply transformations on the generated object for placement. As a result, our method produces a union of 3D scene mesh and a 3D object placed in the scene based on the text prompt.

The overview of our proposed method for object insertion is shown in Fig. 2. The method includes six major steps: *i*) Task classification and entity extraction from a given prompt using LLM, explained in Section 3.2, *ii*) Text-conditioned 3D mesh synthesis of a primary object (Section 3.3), *iii*) Extracting the grounding object from the 3D scene (Section 3.4), *iv*) Scaling the primary object with respect to the grounding object (Section 3.5), *v*) Identifying the optimal location where the object has to be placed in Section 3.6, and *vi*) Refine the placement of primary object at the identified location in Section 3.7. Similarly, last section (Section 3.8) explains the process of replacing objects.

3.1. Preliminaries

Diffusion Models Diffusion models [17] are the probabilistic models that gradually learn to generate data similar to the training data. They destroy training data slowly by adding noise during forward pass. In the reverse pass, they predict and remove the noise added between two consecutive steps during forward pass to generate the data. Once the model learns the denoising step, it can generate data from Gaussian noise in such a way that the generated output resembles data from the distribution on which it was trained.

Latent Diffusion Latent Diffusion [32] is a two-step process where initially, the encoder and decoder are trained to produce latent codes from data $Z = E(x)$ and reconstruct data from these latent codes $\tilde{x} = D(Z)$. Next, a diffusion model is trained to denoise latent representations for generating data, resulting in low computational costs with latent

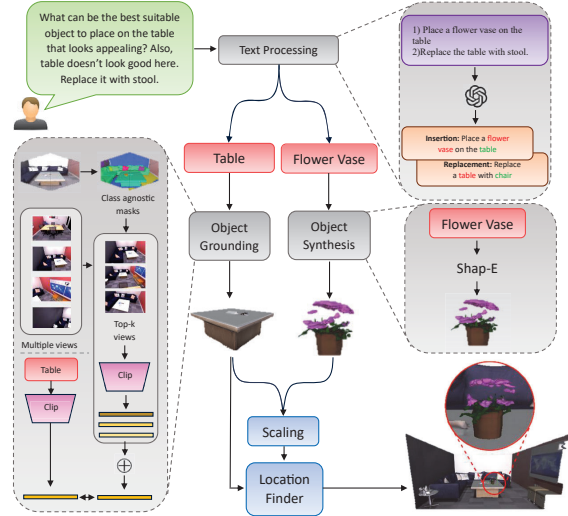


Figure 2. **FreeEdit**: Object insertion in a 3D scene. Given a text prompt, LLM classifies the task and extracts primary and grounding entities. Object synthesis for primary object is done by Shap-E. OpenMask3D does object grounding. Scaling of primary object is performed. Location finder computes an optimal location to place primary object on grounding object. Scaling and location finder (in blue) are not pre-trained models and run on the fly.

codes capturing the important details of data.

Signed Distance Function (SDF) Signed Distance Function (SDF) is a mathematical function that represents the shape of a 3D mesh. It maps the coordinates of a point to a scalar value $f(x) = D$ which is the orthogonal distance from the point x to the nearest point on the surface of the mesh. The sign of D determines the interiority of the point with respect to the surface boundary, negative sign for inside points, positive for points outside the mesh and 0 for points on the surface. Algorithms like Marching cubes [25] can be employed to construct the surface from SDF.

3.2. Task Classification and Entity Extraction

This section explains “What” needs to be inserted in a 3D scene and “Where”. To do this, we leverage GPT-4 [1], a Large Language Model (LLM) to categorize the task into Insertion/Replacement or Deletion. For insertions, the LLM identifies primary (“what”) and grounding (“where”) objects from the prompt. In replacement, it determines which object to replace and its substitute. LLMs excel at interpreting natural language prompts with open vocabularies, making them invaluable for providing accurate answers and relevant suggestions. GPT-4 can process complex paragraphs by breaking them into manageable tasks, categorizing them, and extracting necessary entities. It offers context-specific recommendations, such as suggesting appropriate dishes for a kitchen or suitable decorative items for a living room. Moreover, LLMs understand practical considerations, like plants requiring sunlight, and provide advice accordingly

(see Fig. 5, row 2). Examples of prompts are available in the Supplementary materials.

3.3. Object Synthesis

After identifying the primary object in the text prompt as in Section 3.2, it is used to generate a 3D object which should be placed in a scene. We use a pre-trained model for text to 3D generation mentioned in [18]. Shap-E is a transformer-based generative model built on [30] that produces a 3D mesh, conditioned on text or an image using latent diffusion. The role of this generative model is to synthesize any object as per requirement with modernized traits. It do not restrict the primary object to set of objects that can be retrieved from repository. This helps in designing primary object with diversified features whereas in object retrieval, it is limited to predefined object categories.

3.4. Object Grounding

Unlike previous methods, our method reduces the user interaction by eliminating process of creating bounding box or masks (process of explicitly mentioning the grounding object). Instead, from the given scene, we extract this object using an open vocabulary grounding mechanism. Open vocabulary object grounding provides a flexibility of localizing objects without restricting to predefined set of object classes and can be deployed for unseen objects. This process is good at compiling natural languages enhancing human communication. We use a pre-trained open vocabulary 3D instance segmentation model to ground the surface object in a 3D scene. OpenMask3D [38] computes feature representation of 3D objects aggregating CLIP [31] features from multi-view images and retrieves objects having high similarity with query features. The name of the grounding object obtained in Section 3.2 is embedded using CLIP. The mesh of object whose representation is similar to query embedding is retrieved.

3.5. Scaling: Primary vs Grounding Object

The primary object is generated in a different coordinate space, and the scene is in another space. As a result, the primary object is independent of the scene and has to be scaled accordingly to orient perfectly in the given scene. To solve this, we consider 2D latent diffusion. [32] generates images based on text using cross-attention layers in diffusion models. We generate a few images based on the text prompt using 2D diffusion, and these images are passed to [24]. Grounding DINO is a text guided transformer-based open-set object detector that detects the objects in the input image. We consider a set of images containing both objects so that we get the scale of one object with respect to the other. Bounding boxes (bb) of both objects are predicted in each image and the dimensions are extracted. We consider width primarily to compute the scale as width plays a major role

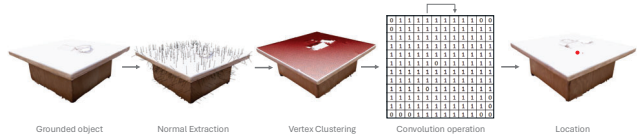


Figure 3. From grounded object on the left, vertices having normals parallel to the positive Z -axis are filtered. They are clustered based on density, visualized as red dots and a voxel grid is created. Convolution operation is performed on this voxel grid which finally gives the optimal location pointed with red dot.

in determining the space needed for placing. Scale (S) is the ratio of the width of bb of the primary object (W_p) to the width of bb of the grounding object (W_g).

$$S = \frac{W_p}{W_g} \quad (1)$$

Out of all the computed scales from generated images, we consider the minimum value and use it for scaling the primary object in a 3D scene as the results look realistic. Our model doesn't perform practical edits if the scale is of larger value. For more details, please refer supplementary.

3.6. Location Finder

To minimize user interaction, we propose a technique to obtain a location where the object should be placed without user intervention. For this, we first extract the vertex normals of the grounding object retrieved from Section 3.4. Please note that the our focus is only on surfaces that are parallel to the ground. We filter the vertices whose vertex normals are orthogonal to the ground, pointing towards the roof as shown in Fig. 3. Other vertices are omitted which eliminates the regions that are unfit for placing the objects leaving cavities on the surface. Filtered vertices are clustered using density-based clustering (DBSCAN). As a result, the vertices having nearly the same Z values are clustered together. Width of scaled primary object is divided into n voxels each of size s . Similarly, the each cluster of the grounding object containing filtered vertices is voxelized, each voxel having size s . This forms a grid with M rows and N columns on the surface of the grounding object. The value in each empty cell is filled with '0' and cells containing at least one vertex are marked with '1'.

A filter of size $n \times n$ is created and slithered over the grid. Convolution operation is performed on the grid, and average pooling is applied. The output of this operation is '1' if the average value exceeds the threshold; else, the output is '0' as shown in Fig. 4. Next, the filter is shifted by one cell till the entire grid is covered, and for each shift, similar operation is performed. The size of the resultant grid is $(M - n + 1) \times (N - n + 1)$ where M and N are the rows and columns of the voxel grid from previous level, respectively. This process will continue until there is no '1' in the output

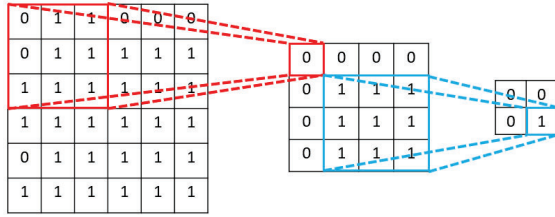


Figure 4. Level-0 voxel grid of size 6×6 . ‘1’ represents presence of at least one vertex. Filter of size 3×3 is slithered over level-0 to get level-1 with value ‘1’ if a threshold condition is met, else ‘0’. Similarly, level-2 is computed from level-1 for the final output.

grid or size of output grid is less than the filter size (after multiple levels). The value ‘1’ in the output grid specifies the location containing vertices. This process enables us to find locations where there is a feasibility to place primary object and out of all the viable locations, optimal location is selected that minimizes the intersection with prior objects in scene. Finding the best placement among all possible results is a non-trivial problem, considering the text prompt might not always have detailed instructions, multiple positions of placement might be correct. Keeping this in mind, we have developed our method in such a way that it finds the best spot by avoiding mesh intersections and preferring surfaces with fewer collisions among multiple options. As we repeat the process over multiple levels, the grounding surface is examined at multiple scales (obtaining global information) to locate the finest position, far from prior objects. Index of valid voxel from final grid is extracted and coordinates of prime location are calculated using

$$(x, y) = (x_0 + (x_{id} + levels - 0.5) * width, y_0 + (y_{id} + levels - 0.5) * width) \quad (2)$$

where x_0/y_0 are the minimum values along x/y -axis in cluster, x_{id}/y_{id} are the indices of voxel along x/y axis in final grid, $levels$ is number of levels in hierarchy and $width$ is voxel size. Neighboring vertices are considered to compute the Z coordinate. Experiments with different threshold values and different filter sizes are mentioned in Supplementary.

3.7. Refinement

Refinement is required to determine the best orientation of the primary object to be placed on the grounding object with minimal intersection. The base centroid of the primary object is calculated. Using this centroid, the object is translated to the location identified in Section 3.6. Now the object is rotated by certain angles at equal intervals and angle with least penetration percent (Eq.3) is considered to rotate the object. With this, the primary object is transformed to the desired location, and both meshes are merged.

3.8. Replacement

Our proposed method can also replace objects in a given scene mentioned through a text prompt. Given prompt is processed by LLM as mentioned in Section 3.2 to identify the existing object and replacing object. If a single object is mentioned in the text prompt, a similar object will replace the existing object. There is also a provision to input 3D mesh of replacing objects instead of generating new. The grounding object is extracted using [38], and its dimensions are recorded. This grounded object is deleted from the scene and cavities are inpainted as mentioned in 4.2. The replacing object is scaled according to the dimension of the grounding object so that it fits exactly in the place of the grounding object. After scaling, this mesh is transformed to the location of the grounded object and fused with the scene mesh. Detailed process is described in supplementary.

4. Experiments

In this section, we discuss the details of the datasets. We mention the baseline to compare our proposed method, the metrics used and both qualitative and quantitative results in Section 4.1. These evaluations are mainly to find the presence of intersections between primary and other objects located on grounding objects in a scene. Moreover, we provide functionalities that can be used to refine the results in Section 4.2.

Datasets We use ScanNet [8] and Replica [37] datasets for our experiments. ScanNet is a real-world dataset containing 1500+ indoor scenes with rich annotations. It has 1201 training scenes and 312 validation scenes and a few hidden test scenes. We consider 20 scenes randomly from the validation set that comprises different objects such as tables, chairs, couches, etc. Replica is synthetic dataset containing 18 high-resolution indoor scenes, each having semantic-level segmentation information. We consider 8 scenes mentioned in [38] for our experiments. Since, FreeEdit works on both synthetic and real-world datasets, we are sure that this can be applied to any type of dataset.

4.1. Results

We demonstrate the results of our method against the baseline methods. We consider two baselines for our main results. *i*) Instruct-NeRF2NeRF [15] is a recent text-guided 3D scene editing method. It uses an image-conditioned diffusion model to iteratively edit the images while optimizing the underlying scene. *ii*) We propose another baseline similar to FreeEdit. It is based on pre-trained models used in our pipeline, followed by scaling of the primary object. We do not consider location finder algorithm as it is our proposed algorithm. Instead, we place the object by default at the center of the grounding object in baseline. Other steps are similar to FreeEdit as discussed in Section 3.

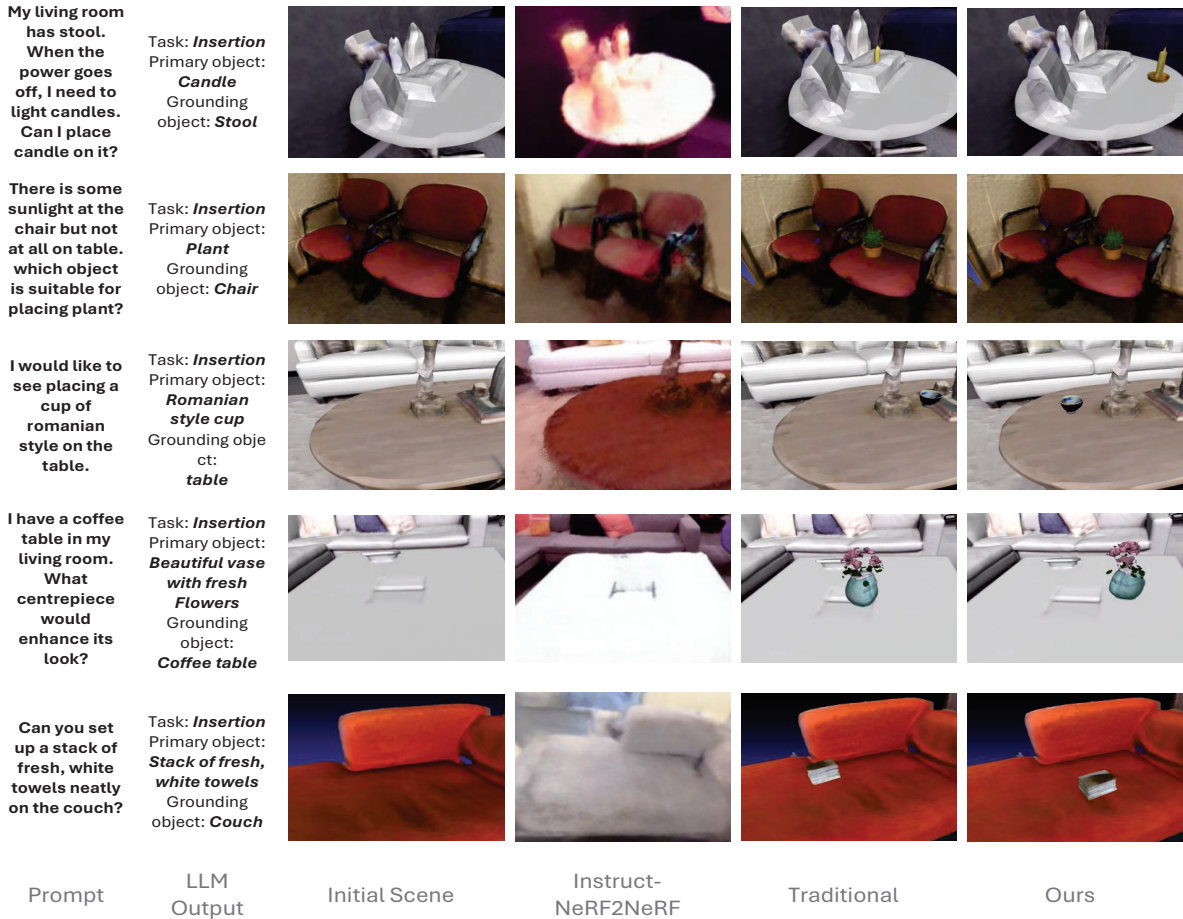


Figure 5. **Insertion.** (a) The input prompt, (b) Response from LLM, (c) The input scene, (d) Output from Instruct-NeRF2NeRF, (e) Results of our traditional baseline, (f) Output of FreeEdit.

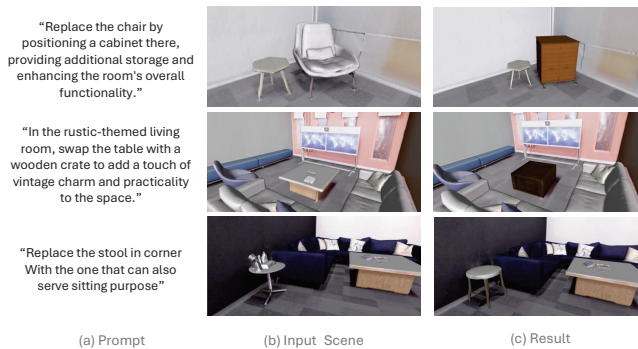


Figure 6. **Replacement.** (a) The input prompt, (b) Input scene before replacement, (c) Scene after replacement using FreeEdit.

Qualitative Results Our qualitative results for object insertion are shown in Figure 5. We consider five samples of scenes and text prompts. We show scenes after editing by our proposed FreeEdit, Instruct-NeRF2NeRF and our traditional baseline where primary object is inserted on ground-

ing object in each example. We can clearly observe that our method inserts the object at the best-suited location without any intersection with other objects, while the traditional baseline fails to avoid intersections. On the other hand, Instruct-NeRF2NeRF attempts to insert a new object in the scene, leading to global changes (rows 1,2,3) in the scene and modifying existing objects instead of inserting a new object(row 5). We can see the change in texture, color, etc. in Fig. 5 (d). This is because Instruct-NeRF2NeRF performs editing in 2D images and placing things without strong semantics leads to diverse edits in images. These edits may include variations in primary object or different placing locations and Instruct-NeRF2NeRF fails at converging these edits resulting in global edits. From Figure 5, please note (especially in examples 1,3,4) that, compared to other methods, FreeEdit outperforms the insertion job by avoiding intersections with prior objects and ignoring the uneven surface, which is not suitable for placing object. Subsequently, we show our qualitative results for object replacement in Figure 6. We can see that when a single object

Refinement	FreeEdit	Traditional
No	4.40%	8.21%
Yes	3.84%	7.45%

Table 1. Penetration percent. Bold represents the best result.

is mentioned in the text prompt (last row), a similar object will replace the existing object otherwise primary object replaces the grounding object. Our results indicate that our method smoothly replaces the existing object without discrepancies, intersections and distortions in the scene. For more examples, refer supplementary.

User Study Evaluation Next, we evaluate our results by conducting a small user study with 35 participants having research background with age in between the range of 19-30 years. We evaluate 8 test cases following the practice of [5, 43]. Our questionnaire includes input prompt, edited output of the baseline and our method and questions related to these methods. The input prompt guides the user to understand the edit operation performed. The outputs of our method and baseline are ordered randomly in each questionnaire and no pattern/clue was available whatsoever. It contains questions like *a*) “Is primary object placed correctly on the grounding object?”, *b*) “Rate the look and feel of scene”, and *c*) “Rate the orientation of primary object”. We requested people to rate the results on a scale of 1 – 5, with 5 being excellent. We collected responses and averaged the scores for each question. We found the scores to be **4.17** and 3.02 for question *a*) for FreeEdit and baseline, respectively. Similarly, *b*) and *c*) have scores of **3.86** and 2.78, and **4.17** and 3.16 for our method and traditional baseline (the higher the better). These results demonstrate that the quality of scenes edited by our method is better than the baseline in real-world.

Quantitative Results As no ground truth information available, we considered a quantitative metric motivated by [39, 41]. This metric calculates the percentage of primary object vertices intersecting with the scene mesh. Penetration percent metric is the count of primary object vertices having negative signed distance divided by the total number of primary object vertices. A point will have negative signed distance if it lies inside the mesh, positive if it lies outside the mesh and 0 if it lies on the mesh. Since we insert objects inside a closed scene, intersection results in vertices penetration out of the scene leading to positive sign for penetrated vertices.

$$Penetration\ percent = \frac{1}{N} \sum_{n=1}^N [dist(v_n, M) > 0] \quad (3)$$

where N is total number of vertices in primary object, v_n is the n^{th} vertex of primary object, M is scene mesh

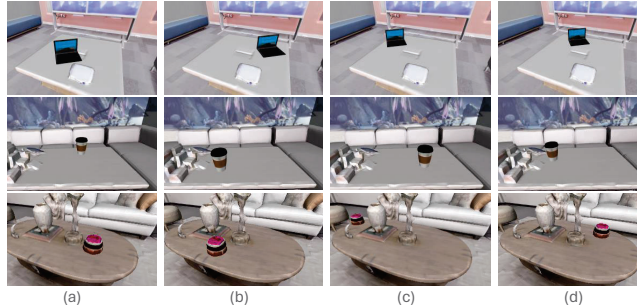


Figure 7. **Translation.** (a) The initial placement of primary object, (b), (c) and (d) The translation of the object to different points.

and $dist(x, y)$ is a Signed Distance Function (SDF). For this experiment, we consider 50 grounded objects from 28 scenes and generated 25 primary objects, such as a coffee cup, teapot, laptop, etc. With this, the whole evaluation set contains 1250 cases. We evaluate our proposed method against the traditional baseline on all the cases using the penetration percent metric. We observe that the average penetration percent value is reduced by nearly half from the baseline method in the proposed method as shown in Table 1. This indicates that the percent of vertices intersecting with scene mesh is reduced by significant margin using our proposed location finder algorithm even without refinement step. After refinement (Section 3.7), the percentage is further reduced for both the methods.

4.2. Manoeuvre

In this experiment, we provide an option for further refining the placement of object. We can translate, rotate, delete or iteratively add objects in a scene.

Translation If the position of the primary object has to be changed, we can select a point on the grounding object. Now the object will be translated to the point we select. If more than one point is selected, the mean of all these points is calculated, and the primary object is translated to the centroid point. Few examples are demonstrated in Fig. 7.

Rotation Similar to translation, we can rotate the placed object accordingly. We can rotate the object in a clockwise or anticlockwise direction from the top view by specifying the angle. Given the angle and direction, the primary object is rotated around the Z -axis at the placed location on the grounding object. Fig. 8 illustrates a few examples.

Deletion Given a text prompt, we can delete existing objects in a 3D scene as shown in Fig. 9. Based on the feed, the related object is grounded using OpenMask3D [38], and it is deleted from the scene. FreeEdit deletes other objects placed on the grounding object as well. Deletion of objects results in the cavity in the scene. A new surface is created connecting boundary vertices of cavity but the faces in the new mesh are much larger leading to inappropriate shading. This issue can be tackled by breaking down the larger faces

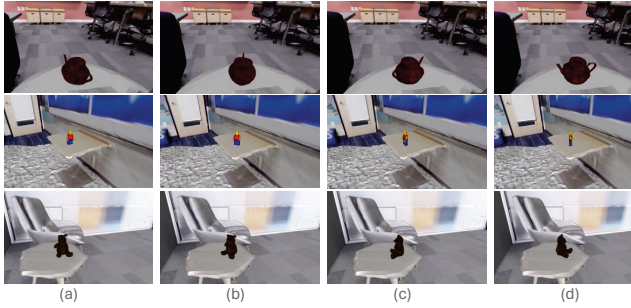


Figure 8. **Rotation.** (a) The initial placement of the primary object, (b) Primary object rotated by 45 degrees in clockwise direction, (c) Object rotation by 90 degrees in clockwise direction and (d) Object rotation by 60 degrees in anticlockwise direction.

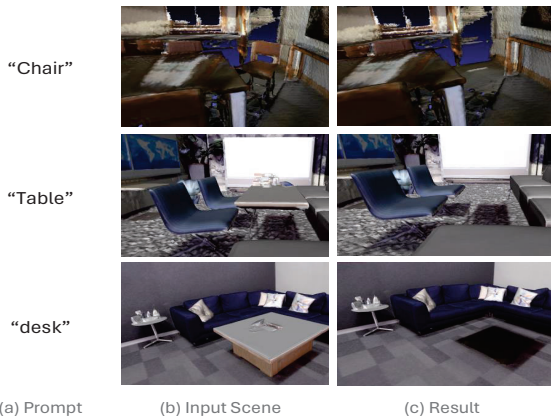


Figure 9. **Deletion.** (a) The text prompt, (b) The input scene, (c) Scene after deletion.

into small and each vertex feature can be computed considering nearest neighbors. Also note that appearance tailoring is out of the scope of this paper.

Inserting Objects Iteratively FreeEdit is capable of adding multiple objects iteratively. When our model is provided with prompts containing multiple primary objects, it adds them in a recursive manner, each object at a time. Initially, the first object is generated and placed on the grounding object, and the next object is processed. This process continues until either all the objects are added or there is insufficient space to place an object. The model makes sure that new object does not intersect with other objects. This ensures intersection-free placement of objects in an iterative fashion. Results are shown in the Fig. 10

Limitations and Future work: Following are some of the limitations of FreeEdit that present opportunities for future work. *i)* Object placement is currently restricted to flat surfaces. Incorporating physics simulations guided by LLMs could enable placement on non-flat surfaces. *ii)* The deletion of objects can result in illegitimate inpainting within cavities (as discussed in Section 4.2). *iii)* The system lacks

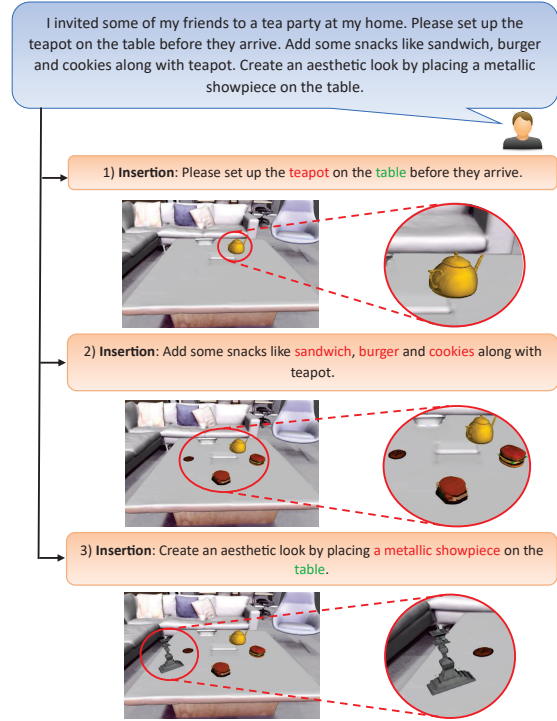


Figure 10. **Iterative Insertion.** Given a text prompt, objects are added onto grounding surface in iterative fashion.

the ability to place objects precisely in relation to existing objects (e.g., “in front of”). This limitation could be addressed by incorporating scene graphs to provide spatial relationship information. *iv)* Unrealistic object placement occurs when scaling values exceed certain thresholds. This issue can be mitigated by using appropriate threshold limits.

5. Conclusion

We introduced FreeEdit, a text-guided approach for adding and replacing objects in 3D scenes comprising multiple objects. Given a scene along with a prompt for editing, the proposed method extracts the entities and synthesizes the object to be inserted (or replaced), followed by identification of the object to anchor it to. The optimal position on the grounding object is automatically determined and the synthesized object is placed. To the best of our knowledge, our method is the first text-guided training-free approach to edit 3D scenes with multiple objects. We show how leveraging mesh-level representation for simple edits can be a potential alternative to training-based and time-consuming NeRF based approaches. We showcase the effectiveness of our approach through qualitative and quantitative comparisons with relevant baselines and existing approaches.

Acknowledgments: This work is supported by Fujitsu Research of India Private Limited. We also thank the participants for participating in the user study evaluation.

References

- [1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 3
- [2] Chong Bao, Yinda Zhang, Bangbang Yang, Tianxing Fan, Zesong Yang, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Sine: Semantic-driven image-based nerf editing with prior-guided editing field. In *The IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2023. 2
- [3] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023. 2
- [4] Mingdeng Cao, Xintao Wang, Zhongang Qi, Ying Shan, Xiaohu Qie, and Yinqiang Zheng. Masactrl: Tuning-free mutual self-attention control for consistent image synthesis and editing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22560–22570, October 2023. 2
- [5] Yukang Cao, Yan-Pei Cao, Kai Han, Ying Shan, and Kwan-Yee K. Wong. Dreamavatar: Text-and-shape guided 3d human avatar generation via diffusion models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 7
- [6] Dana Cohen-Bar, Elad Richardson, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Set-the-scene: Global-local training for generating controllable nerf scenes. *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 2912–2921, 2023. 2
- [7] Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. Diffedit: Diffusion-based semantic image editing with mask guidance. *arXiv preprint arXiv:2210.11427*, 2022. 2
- [8] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Habber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2017. 5
- [9] Zhang Dan. Study on interior decoration system design based on 3d scene modeling technology. In *2017 International Conference on Smart Grid and Electrical Automation (ICSGEA)*, pages 380–383, 2017. 1
- [10] Jiahua Dong and Yu-Xiong Wang. Vica-nerf: View-consistency-aware 3d editing of neural radiance fields. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 2
- [11] Shen Fangyang, Jia Jun, Lu Xuejun, and Qi Yue. 3d modeling and augmented reality. In *2010 4th International Universal Communication Symposium*, pages 185–192, 2010. 1
- [12] Daniele Giunchi, Stuart James, and Anthony Steed. 3d sketching for interactive model retrieval in virtual reality. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch-Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*, Expressive '18, New York, NY, USA, 2018. Association for Computing Machinery. 1
- [13] Rahul Goel, Dhawal Sirikonda, Rajvi Shah, and PJ Narayanan. Fusedrf: Fusing multiple radiance fields. *arXiv preprint arXiv:2306.04180*, 2023. 2
- [14] Qiao Gu, Alihusein Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, Aditya Agarwal, Corban Rivera, William Paul, Kirsty Ellis, Rama Chellappa, Chuang Gan, Celso Miguel de Melo, Joshua B. Tenenbaum, Antonio Torralba, Florian Shkurti, and Liam Paull. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning. In *ICRA*, 2024. 1, 2
- [15] Ayaan Haque, Matthew Tancik, Alexei Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 1, 2, 5
- [16] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022. 1, 2
- [17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020. 3
- [18] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023. 4
- [19] Hiromichi Kamata, Yuiko Sakuma, Akio Hayakawa, Masato Ishii, and Takuya Narihira. Instruct 3d-to-3d: Text instruction guided 3d-to-3d conversion. *arXiv preprint arXiv:2303.15780*, 2023. 2
- [20] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *Conference on Computer Vision and Pattern Recognition 2023*, 2023. 2
- [21] Peter Kán, Andrija Kurtic, Mohamed Radwan, and Jorge M. Loáiciga Rodríguez. Automatic interior design in augmented reality based on hierarchical tree of procedural rules. *Electronics*, 10(3), 2021. 1
- [22] Verica Lazova, Vladimir Guzov, Kyle Olszewski, Sergey Tulyakov, and Gerard Pons-Moll. Control-nerf: Editable feature volumes for scene rendering and manipulation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 4340–4350, January 2023. 2
- [23] Xiaoyu Li, Qi Zhang, Di Kang, Weihao Cheng, Yiming Gao, Jingbo Zhang, Zhihao Liang, Jing Liao, Yan-Pei Cao, and Ying Shan. Advances in 3d generation: A survey. *arXiv preprint arXiv: 2401.17807*, 2024. 1
- [24] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023. 4

- [25] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In Maureen C. Stone, editor, *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1987, Anaheim, California, USA, July 27-31, 1987*, pages 163–169. ACM, 1987. [3](#)
- [26] Mario Lorenz, Sebastian Knopp, Jisu Kim, and Philipp Klimant. Industrial augmented reality: 3d-content editor for augmented reality maintenance worker support system. In *2020 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 203–205, 2020. [1](#)
- [27] Lyndsay Mesjar, Karen Cross, Yang Jiang, and Josie Steed. The intersection of fashion, immersive technology, and sustainability: A literature review. *Sustainability*, 15(4), 2023. [1](#)
- [28] Aryan Mikaeili, Or Perel, Mehdi Safaee, Daniel Cohen-Or, and Ali Mahdavi-Amiri. Sked: Sketch-guided text-based 3d editing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14607–14619, 2023. [2](#)
- [29] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. In *International Conference on Machine Learning*, 2021. [2](#)
- [30] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022. [4](#)
- [31] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya 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*, 2021. [4](#)
- [32] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022. [3](#), [4](#)
- [33] Etai Sella, Gal Fiebelman, Peter Hedman, and Hadar Averbuch-Elor. Vox-e: Text-guided voxel editing of 3d objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 430–440, 2023. [2](#)
- [34] Mohamad Shahbazi, Liesbeth Claessens, Michael Niemeyer, Edo Collins, Alessio Tonioni, Luc Van Gool, and Federico Tombari. Insef: Text-driven generative object insertion in neural 3d scenes. *Arxiv*, 2024. [1](#), [2](#)
- [35] Yizhi Song, Zhifei Zhang, Zhe Lin, Scott Cohen, Brian Price, Jianming Zhang, Soo Ye Kim, and Daniel Aliaga. Object-stitch: Object compositing with diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18310–18319, June 2023. [2](#)
- [36] Andrew H. Stevens and Thomas Butkiewicz. Faster multi-beam sonar data cleaning: Evaluation of editing 3d point clouds using immersive vr. In *OCEANS 2019 MTS/IEEE SEATTLE*, pages 1–10, 2019. [1](#)
- [37] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. [5](#)
- [38] Ayça Takmaz, Elisabetta Fedele, Robert W. Sumner, Marc Pollefeys, Federico Tombari, and Francis Engelmann. OpenMask3D: Open-Vocabulary 3D Instance Segmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. [4](#), [5](#), [7](#)
- [39] Purva Tendulkar, Dídac Surís, and Carl Vondrick. Flex: Full-body grasping without full-body grasps. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. [7](#)
- [40] Yoad Tewel, Omri Kaduri, Rinon Gal, Yoni Kasten, Lior Wolf, Gal Chechik, and Yuval Atzmon. Training-free consistent text-to-image generation, 2024. [1](#), [2](#)
- [41] Dylan Turpin, Liquan Wang, Eric Heiden, Yun-Chun Chen, Miles Macklin, Stavros Tsogkas, Sven Dickinson, and Animesh Garg. Grasp’d: Differentiable contact-rich grasp synthesis for multi-fingered hands. In *European Conference on Computer Vision*, pages 201–221. Springer, 2022. [7](#)
- [42] Xingchen Zhou, Ying He, F Richard Yu, Jianqiang Li, and You Li. RePaint-NeRF: Nerf editing via semantic masks and diffusion models. In *IJCAI*, 2023. [2](#)
- [43] Jingyu Zhuang, Chen Wang, Liang Lin, Lingjie Liu, and Guanbin Li. Dreameditor: Text-driven 3d scene editing with neural fields. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–10, 2023. [1](#), [2](#), [7](#)