AssetField: Assets Mining and Reconfiguration in Ground Feature Plane Representation

Yuanbo Xiangli[1*], Linning Xu[1*], Xingang Pan[3,4], Nanxuan Zhao[5], Bo Dai[2✉], Dahua Lin[1,2]
[1] The Chinese University of Hong Kong [2] Shanghai AI Laboratory
[3] Max Planck Institute for Informatics [4] Nanyang Technological University [5] Adobe Research
{xy019,xl020,dhlin}@ie.cuhk.edu.hk xingang.pan@ntu.edu.sg
nanxuanzhao@gmail.com daibo@pjlab.org.cn

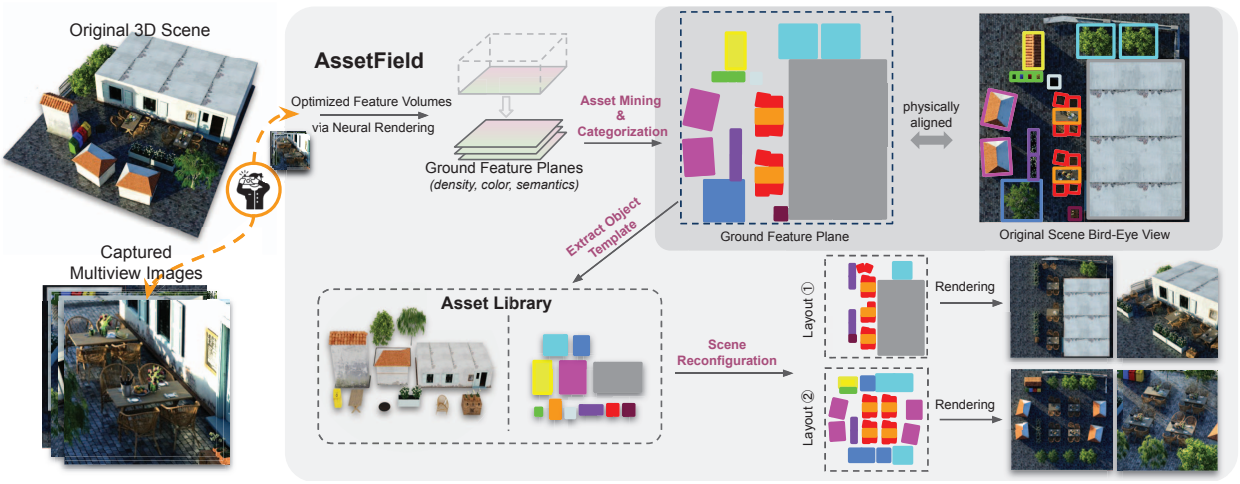

Figure 1: Man-made environments are often characterized by repetitive scene objects, *e.g.* tables, chairs, and trees. *AssetField* represents these environments with a set of informative ground feature planes aligning with the physical ground, from which neural representations of scene objects are extracted and grouped into categories. The proposed mechanism allows users to manipulate and compose assets directly on the ground feature plane and produces high-quality rendering on novel scene configurations.

Abstract

*Both indoor and outdoor environments are inherently structured and repetitive. Traditional modeling pipelines keep an asset library storing unique object templates, which is both versatile and memory efficient in practice. Inspired by this observation, we propose AssetField, a novel neural scene representation that learns a set of object-aware ground feature planes to represent the scene, where an asset library storing template feature patches can be constructed in an unsupervised manner. Unlike existing methods which require object masks to query spatial points for object editing, our ground feature plane representation offers a natural visualization of the scene in the bird-eye view, allowing a variety of operations (*e.g. *translation, duplication, deformation) on objects to configure a new scene. With the template feature patches, group editing is enabled for scenes with many recurring items to avoid repetitive work on object individuals. We show that AssetField not only achieves competitive performance for novel-view synthesis but also generates realistic renderings for new scene configurations.*

1. Introduction

The demand for bringing our living environment into a virtual realm continuous to increase these days, with example cases ranging from indoor scenes such as rooms and restaurants, to outdoor ones like streets and neighborhoods. Apart from the realistic 3D rendering, real-world applications also require flexible and user-friendly editing of the scene. Use cases can be commonly found in interior design, urban planning etc. To save human labor and expense, users need to frequently visualize different scene configurations before finalizing a plan and bringing it to reality, like shown in Fig.1. For their interests, a virtual environment offering versatile editing choices and high rendering

quality is always preferable. In these scenarios, objects are primarily located on a horizontal plane like ground, and can be inserted to/deleted from the scene. Translation along the plane and rotation around the vertical axis are also common operations. Furthermore, group editing becomes essential when scenes are populated with recurring items (*e.g*. substitute all chairs with stools and remove all sofas in a bar).

While recent advances in neural rendering [27, 3, 44, 28] offer promising solutions to producing realistic visuals, they struggle to meet the aforementioned editing demands. Specifically, traditional neural radiance field (NeRF)-based methods such as [47, 26, 3] encode an entire scene into a single neural network, making it difficult to manipulate and composite due to its implicit nature and limited network capacity. Some follow-up works [41, 16] tackle object-aware scene rendering in a bottom-up fashion by learning one model per object and then performing joint rendering. Another branch of methods learn object radiance fields using instance masks [42], object motions [46], and image features [39, 21] as clues but are scene-specific, limiting their applicable scenarios. Recently, some approaches have attempted to combine voxel grids with neural radiance fields [23, 44, 28] to explicitly model the scene. Previous work [23] showed local shape editing and scene composition abilities of the hybrid representation. However, since the learned scene representation is not object-aware, users must specify which voxels are affected to achieve certain editing requirements, which is cumbersome, especially for group editing. Traditional graphical workflows build upon an asset library that stores template objects, whose copies are deployed onto a 'canvas' by designers, then rendered by some professional software (*e.g*. interior designers arrange furniture according to floorplans). This practice significantly saves memory for large scene development and offers users versatile editing choices, which inspires us to resemble this characteristic in neural rendering.

To this end, we present *AssetField*, a novel neural representation that bears the editing flexibility of traditional graphical workflows. Our method factorizes a 3D neural field into a ground feature plane and a vertical feature axis. As illustrated in Fig. 1, the learned **ground feature plane** is a 2D feature plane that is visually aligned with the bird-eye view (BEV) of the scene, allowing intuitive manipulation of individual objects. It is also able to embed multiple scenes into scene-specific ground feature planes with a shared vertical feature axis, rendered using a shared MLP. The learned ground feature planes encode scene density, color and semantics, providing rich clues for object detection and categorization. We show that assets mining and categorization, and scene layout estimation can be directly performed on the ground feature planes. By maintaining a cross-scene *asset library* that stores template objects' ground feature patches, our method enables versatile editing at *object-level*, *category-level*, and *scene-level*.

In summary, AssetField 1) learns a set of explicit ground feature planes that are intuitive and user-friendly for scene manipulation; 2) offers a novel way to discover assets and scene layout on the informative ground feature planes, from which one can construct an asset library storing feature patches of object templates from multiple scenes; 3) improves group editing efficiency and enables versatile scene composition and reconfiguration and 4) provides realistic renderings on new scene configurations.

2. Related Works

Neural Implicit Representations and Semantic Fields. Since the introduction of neural radiance fields [27], many advanced scene representations have been proposed[24, 44, 28, 11, 24, 10, 28], demonstrating superior performance in terms of quality and speed for neural renderings. However, most of these methods are semantic and content agnostic, and many assume sparsity to design a more compact structure for rendering acceleration [24, 11, 28]. We notice that the compositional nature of a scene and the occurrence of repetitive objects within can be further utilized, where we can extract a reusable asset library for more scalable usages, similar to those adopted in the classical modeling pipeline.

A line of recent neural rendering works has explored the jointly learning a semantic fields along with the original radiance field. Earlier works use available semantic labels [50] or existing 2D detectors for supervision [22]. The realized semantic field can enable category or object-level control. More recently, [39, 21] explore the potential of distilling self-supervised 2D image feature extractors [8, 2, 14] into NeRF, and showcasing their usages of support local editing. In this work, we target an orthogonal editing goal where the accurate control of high-level scene configuration and easy editing on object instances is desired.

Object Manipulation and Scene Composition. Traditional modeling and rendering pipelines [5, 7, 33, 34, 35, 17] are vastly adopted for scene editing and novel view synthesis in early approaches. For example, Karsch *et al*. [17] propose to realistically insert synthetic objects into legacy images by creating a physical model of the scene from user annotations of geometry and lighting conditions, then compose and render the edited scene. Cossairt *et al*. [12] consider synthetic and real objects compositions from the perspective of light field, where objects are captured by a specific hardware system. [49, 19, 18] consider the problem of manipulating existing 3D scenes by matching the objects to cuboid proxies/pre-captured 3D models.

These days, several works propose to tackle object-decomposite rendering under the context of newly emerged neural implicit representations [27]. Ost *et al*. [31] target dynamic scenes and learn a scene graph representation that encodes object transformation and radiance at each node,

which further allows rendering novel views and re-arranged scenes. Kundu *et al*. [22] resort to existing 3D object detectors for foreground object extraction. Sharma *et al*. [36] disentangles static and movable scene contents, leveraging object motion as a cue. Guo *et al*. [16] propose to learn object-centric neural scattering functions to implicitly model per-object light transportation, enabling scene rendering with moving objects and lights. Neural Rendering in a Room [41] targets indoor scenes by learning a radiance field for each pre-captured object and putting objects into a panoramic image for optimization. While these methods need to infer object from motion, or require one model per object, ObjectNeRF [43] learns a decompositional neural radiance field, utilizing semantic masks to separate objects from the background to allow editable scene rendering. uORF [45] performs unsupervised discovery of object radiance fields without the need for semantic masks, but requires cross-scene training and is only tested on simple synthetic objects without textures.

3. AssetField

In this work, we primarily consider a branch of real-world application scenarios that require fast and high-quality rendering of scenes whose configuration is subject to change, such as interior design, urban planning and traffic simulation. In these cases, objects are mainly placed on some dominant horizontal plane, and is commonly manipulated with insertion, deletion, translation on the horizontal plane, and rotation around the vetical axis, etc.

We first introduce our ground feature plane representation in Sec. 3.1 to model each neural field. Sec. 3.2 describes the process of assets mining with the inferred the ground feature plane. We further leverage the color and semantic feature planes to categorize objects in an unsupervised manner, which is illustrated in Sec. 3.3. Finally, Sec. 3.4 demonstrates the construction of a cross-scene asset library that enables versatile scene editing.

3.1. Ground Feature Plane Representation

Ground plane has been commonly used for indoor and outdoor scene modeling [36, 13, 32]. We adopt a similar representation to parameterize a 3D neural field with a 2D ground feature plane $\mathcal{M}$ of shape $L \times W \times N$, and a globally encoded vertical feature axis $\mathcal{H}$ of shape $H \times N$, where N is the feature dimension. A query point at coordinate (x, y, z) is projected onto $\mathcal{M}$(plane) and $\mathcal{H}$(axis) to retrieve its feature values m and h via bilinear/linear interpolation:

$$m = \text{Interp}(\mathcal{M}, (x, y)), h = \text{Interp}(\mathcal{H}, z), \qquad (1)$$

which are then combined and decoded into the 3D scene feature via a MLP decoder. Concretely, a 3D radiance field is parameterized by a set of ground feature planes

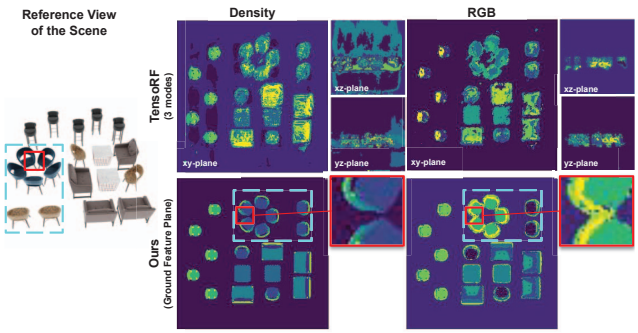

Figure 2: TensoRF with full $3D$ factorization produces noisy feature planes; our ground plane representation yields informative features that clearly illustrated scene contents and layout after discretization, especially in the density field. Red boxes: two spatially close objects can be clearly separated on the density plane but not the RGB plane. Blue boxes: objects with similar geometry but different appearance can be distinguished on the RGB plane but not the density plane.

$\mathcal{M}$=$(\mathcal{M}_\sigma, \mathcal{M}_c)$, and vertical feature axes $\mathcal{H}$=$(\mathcal{H}_\sigma, \mathcal{H}_c)$, for the density and color fields respectively. The retrieved feature values m=(m_σ, m_c) and h=(h_σ, h_c) are then combined and decoded into point density σ and view-dependent color c values by two decoders Dec_σ, Dec_{rgb}. Points along a ray $\mathbf{r}$ are volumetrically integrated following [27]:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i \left(1 - \exp\left(-\sigma_i \delta_i\right)\right) c_i, \qquad (2)$$

where $T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j)$, and supervised by the 2D image reconstruction loss with $\sum_{\mathbf{r}} (\|\hat{C}(\mathbf{r}) - C(\mathbf{r})\|_2^2)$, where $C(\mathbf{r})$ is the ground truth pixel color.

Such neural representations are beneficial to our scenarios. Firstly, the ground feature planes are naturally aligned with the BEV of the scene, mirroring the human approach to high-level editing and graphic design, where artists and designers mainly sketch on 2D canvas to reflect a 3D scene. Secondly, the globally encoded vertical feature axis encourages the ground feature plane to encode more scene information, which aligns better with scene contents. Thirdly, this compact representation is more robust when trained with sparse view images, where the full 3D feature grids are easy to overfit under insufficient supervision, producing noisy values, as depicted in Fig. 2.

3.2. Assets Mining on Ground Feature Plane

For the ease of demonstration, let us first consider a simplified case where objects are scattered on an invisible horizontal plane, as in Fig. 3 (a). We start from modeling the radiance field, where a set of ground features planes $\mathcal{M}$=$(\mathcal{M}_\sigma, \mathcal{M}_c)$ describing scene density and color are inferred following the formulation in Sec. 3.1. It can be observed that $\mathcal{M}_\sigma$ tends to exhibit sharper object boundaries

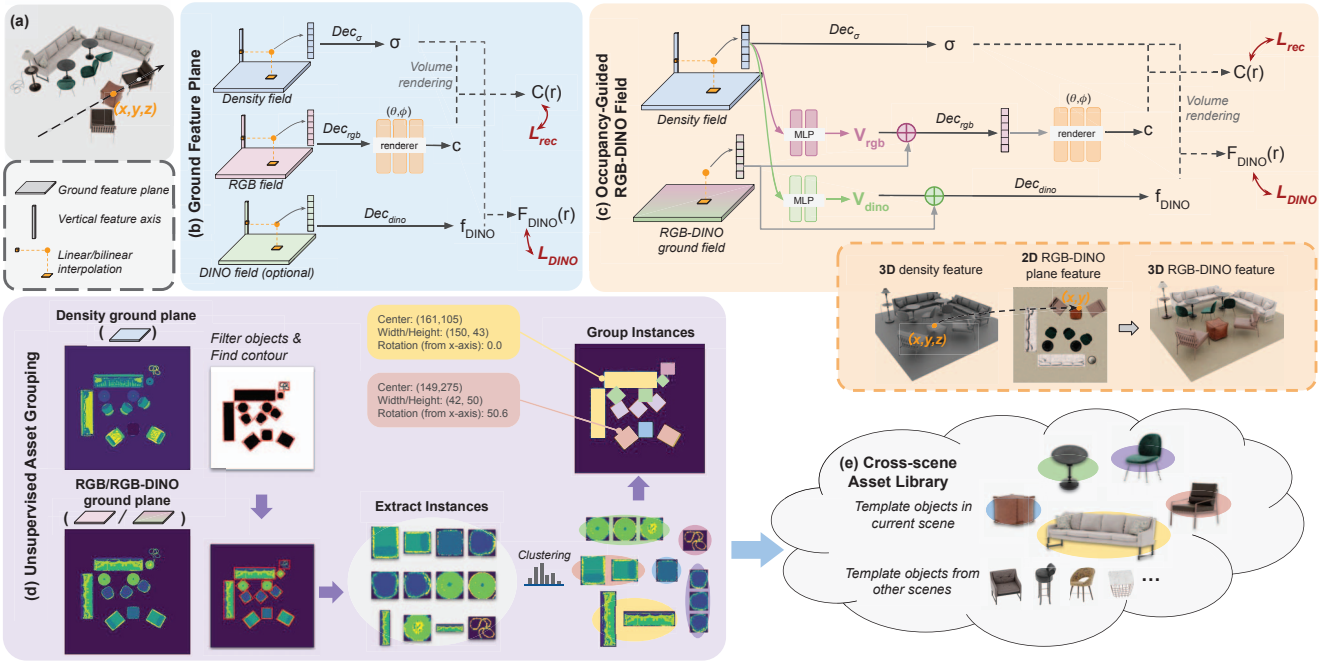

Figure 3: Overview of *AssetField*. (a) We demonstrate on a scene without background for clearer visuals. (b) The proposed ground feature plane representation factorizes a neural field into a horizontal feature plane and a vertical feature axis. (c) We further integrate color and semantic field into a 2D neural plane, which is decoded into 3D-aware features with the geometry guidance from scene density. The inferred RGB-DINO plane is rich in object appearance and semantic clues whilst being less sensitive to vertical displacement between objects, on which we can (d) detect assets and grouping them into categories. (e) For each category, we select a template object and store its density and color ground feature patches into the asset library. A cross-scene asset library can be construct by letting different scenes fit there own ground feature planes whilst sharing the same vertical feature axes and decoders/renderers.

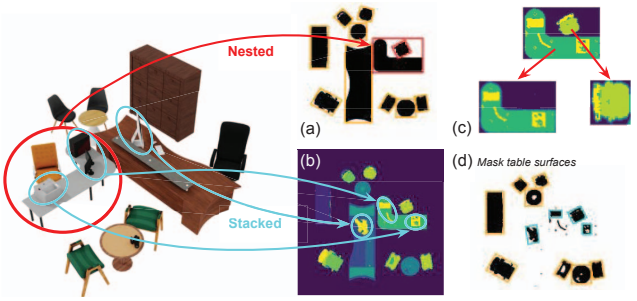

Figure 4: (a) Nested structure can be separated by (c) identify the enclosed chair then set its value to background feature for table patch. (b) Items placed on top of a surface can be detected by (d) another round of filtering that treats table surface as background.

compared to the color feature plane, as shown in the red boxes in Fig. 2. This could be attributed to the mechanism of neural rendering (Eq. 2), where the model firstly learns a clean and accurate density field to guide the learning of the color field. We therefore prefer to use $\mathcal{M}_\sigma$ for assets mining. In the example scene, the feature plane is segmented into two clusters with K-means [25] to obtain a binary mask of the objects. Contour detection [37, 6] is then applied to locate each object, resulting in a set of bounding box. Note that the number of clusters can be customized according to the objects users want to highlight. In more complex scenarios where objects are arranged in a hierarchical structure, (*e.g.* computer - table - floor), the clustering step can be repeated to gradually unpack the scene, as illustrated in Fig. 4. With the bounding boxes, a collection of object neural representations $\mathcal{P}=\{(p_\sigma, p_c)\}$ can be obtained, which are the enclosed feature patches on $\mathcal{M}_\sigma$ and $\mathcal{M}_{rgb}$. To address complex real-world scenes, we take inspiration from previous works [21, 39] that models a DINO [9] field to guide the learning a semantic-aware radiance field. Similarly, we can learn a *separate* DINO ground feature plane $\mathcal{M}_{dino}$ to provide more explicit indications of object presence. As AssetField models a set of separate fields, object discovery can be conducted on any field that offers the most distinctive features in a scene dependent manner.

3.3. Unsupervised Asset Grouping

Despite being versatile, users can only interact with individual instances in $\mathcal{P}$ from the learned ground planes, whereas group editing is also a desirable feature in real-world applications, especially when objects of the same category need to be altered together. While the definition of

object category can be narrow or broad, here we assume that objects with close appearance and semantics are analogues and use RGB and semantic feature plane for assets grouping. A case where the density features fail to distinguish two visually different objects is highlight in Fig. 2.

Occupancy-Guided RGB-DINO field. As our goal is to "self-discover" assets from neural scene representation, there is no extra prior on object category to regularize scene features. 3D voxel-based methods such as those described in [24, 44], may learn different sets of features to express the same objects, as grid features are independently optimized. Such issue can be alleviated by our proposed neural representation, where the ground feature plane $\mathcal{M}$ is constrained by the globally shared vertical feature axis $\mathcal{H}$. Concretely, given two identical objects i, j placed on a horizontal flat surface, the same feature chunk on $\mathcal{H}$ will be queried during training, which constraints their corresponding feature patches p_i and p_j to be as similar as possible so that they can be decoded into the same set of 3D features. However, such constraint no longer holds when there is a vertical displacement among identical objects (*e.g.* one the ground and one on the table), where different feature chunks on $\mathcal{H}$ are queried, leading to divergent p_i and p_j.

To learn a more object-centric ground feature plane rich in color and semantics clues, we propose to integrate the color and semantic fields by letting them share the same set of ground feature planes, denoted by $\mathcal{M}_{rgb\text{-}dino}$. Instead of appending a vertical feature axis, here we use scene density features to guide the decoding of $\mathcal{M}_{rgb\text{-}dino}$ into 3D-aware features, as illustrated in Fig. 3(c). It can be interpreted as $\mathcal{M}_\sigma$ and $\mathcal{H}_\sigma$ fully capture the scene geometry, while $\mathcal{M}_{rgb\text{-}dino}$ captures the 'floorplan' of scene semantics layouts and appearances. For a query point at (x, y, z), its retrieved density feature m_σ and h_σ are mapped to a color feature v_{rgb} and a semantic feature v_{dino} via two MLPs, which are then decoded into scene color c and semantic f_{dino} along with the RGB-DINO plane feature $m_{rgb\text{-}dino} = \text{Interp}(\mathcal{M}_{rgb\text{-}dino}, (x, y))$ via Dec_{rgb} and Dec_{dino}.

Assets Grouping and Template Matching. On the inferred RGB-DINO ground feature plane, we then categorize the discovered objects by comparing their RGB-DINO feature patches enclosed in bounding boxes. However, due to the absence of object pose information, pixel-wise comparison is not ideal. Instead, we compare the distributions of color and semantic features among patches. To do this, we first discretize $\mathcal{M}_{rgb\text{-}dino}$ with clustering (*e.g.* K-means), which results in a set of labeled object feature patches $\mathcal{K}$. The similarity between two object patches $k_i, k_j \in \mathcal{K}$ are measured by the Jensen-Shannon Divergence over the distribution of labels, denoted by $\text{JSD}(k_i||k_j)$. Agglomerative clustering [29] is then performed using JS-divergence as the distance metric. The number of clusters can be set by inspecting training views, and can be flexibly adjusted to fit users' desired categorization granularity.

With scene assets grouped into categories, a *template* object can be selected from each cluster either randomly or in a user-defined manner. We can further extract scene layout in BEV by computing the relative pose between the template object and its copies, *i.e.* to optimize a rotation angle θ that minimizes the pixel-wise loss between the RGB-DINO feature patches of the template and each copy with $\theta^* = \text{argmin}_\theta \sum_i^N ||\hat{p}_i - R_\theta(p)_i||_2^2$ for $p \in \mathcal{P}_{rgb\text{-}dino}$, where $\hat{p}$ is the template RGB-DINO feature patch, R_θ rotates the input feature patch by θ.

3.4. Cross-scene Asset Library

Following the proposed framework, a scene can be represented with (1) a set of template feature patches $\mathcal{P}$=$\{(p_\sigma, p_{rgb})\}$, (2) a layout describing object position and pose in the BEV, (3) the shared vertical feature axes $\mathcal{H} = (\mathcal{H}_\sigma, \mathcal{H}_{rgb})$, and (4) MLP decoders Dec_σ, Dec_{rgb}, which enables versatile scene editing at object-, category-, and scene-level. The newly configured scenes can be directly rendered without retraining. An optional template refinement step is also allowed. Examples are given in Sec. 4.

Previous work [24] demonstrates that voxel-based neural representations support multi-scene modeling by learning different voxel embeddings for each scene whilst sharing the same MLP renderer. However, it does not support cross-scene analogue discovery due to the aforementioned lack of constraints issue, whereas in reality, objects are not exclusive to a scene. Our proposed neural representation has such potential to discover cross-scene analogues by also sharing the vertical feature axes $\mathcal{H}$ among different scenes. Specifically, (1) if the user decides to construct a cross-scene asst library from the beginning, AssetField can be trained jointly on the target scenes by letting them share the same $\mathcal{H}$. (2) if the user later wants to expand the current asset library, the new scene can be modeled by optimizing its own set of ground feature planes whilst fixing $\mathcal{H}$ from the existing scene(s). Consequently, we can construct a cross-scene asset library storing template feature patches, and continuously expand it to accommodate new ones.

4. Experiment

In this section, we first describe our experiment setup, then evaluate AssetField on novel view synthesis both quantitatively and qualitatively, demonstrating its advantages in asset mining, categorization, and editing flexibility. More training details and ablating results of hyperparameters (*e.g.* the number of clusters, the pairing of plane feature, and axis feature) are provided in supplementary.

4.1. Experimental Setup

Dataset. A synthetic dataset is created for evaluation. We compose 10 scenes resembling common man-made envi-

	Scene1			*Scene2*			*Scene3*			*Scene4*		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
NeRF	32.977	0.969	0.067	35.743	0.967	0.051	32.521	0.959	0.058	34.212	0.964	0.072
TensoRF	35.751	0.990	0.057	**38.184**	**0.995**	**0.027**	36.933	0.994	0.034	**37.795**	**0.993**	**0.059**
S-AssetField	36.471	0.992	0.049	36.856	0.993	0.037	36.753	0.994	0.038	37.445	0.990	0.065
I-AssetField	**36.526**	**0.992**	**0.047**	37.271	0.994	0.035	**37.249**	**0.995**	**0.032**	37.716	0.991	0.060

Table 1: Quantitative comparison on test views for the 4 scenes in Fig. 7. We report PSNR(↑), SSIM(↑) [40] and LPIPS(↓) [48] for evalution. The **best** and second best results are highlighted.

ronments such as conference room, living room, dining hall and office. Each scene contains objects from 3∼12 categories with a fixed light source. For each scene, we render 50 views with viewpoints sampled on a half-sphere, among which 40 are used for training and the rest for testing. We demonstrate flexible scene manipulation with AssetField on both the synthetic and real-world data, including scenes from Mip-NeRF 360 [4], DONeRF [30], and ObjectNeRF [42]. We also show manipulation results on city scenes collected from Google Earth Studio [1].

Implementation. We use NeRF [27] and TensoRF [11] as baselines to evaluate the rendering quality of the original scenes. For a fair comparison, all methods are implemented to model an additional DINO field. Specifically, (1) NeRF is extended with an extra head to predict view-independent DINO feature [2] in parallel with density. (2) For TensoRF, we additionally construct the DINO field which is factorized along 3 directions the same as its radiance field. (3) **S(tandard)-AssetField** separately models the density, RGB, and DINO fields. (4) **I(ntegrated)-AssetField** models the density field the same as S-AssetField, and an integrated RGB-DINO ground feature plane. Both S-AssetField and I-AssetField adopt outer-product to combine ground plane features and vertical axis features, following [11]. The resolution of feature planes in TensoRF baseline and AssetField are set to 300×300. Detailed model adaptation can be found in the supplementary. We train NeRF for $200k$ iterations, and $50k$ iterations for TensoRF and AssetField using Adam [20] optimization with a learning rate set to $5e^{-4}$ for NeRF and 0.02 for TensoRF and AssetField.

4.2. Results

Novel View Rendering. We compare S-AssetField and I-AssetField with the adapted NeRF [27] and TensoRF [11] as described above. Quantitative results are provided in Tab. 1. It is noticeable that AssetField's ground feature plane representation (*i.e.* xy-z) achieves comparable performance with TensoRF's 3-mode factorization (*i.e.* xy-z, xz-y,yz-x), indicating the suitability of adopting ground plane representations for such scenes. Our method also inherits the merit of efficient training and rendering from grid-based methods. Compared to NeRF, our model converges 40x faster at training and renders 30x faster at inference.

Our ground feature plane representation is also effective on most NeRF datasets. Below we show test view PSNR on NeRF-Synthetic [27] and 360-v2 [4] scenes. The idea of having a vertical feature axis shared by different ground locations is to encourage the ground feature plane to encode more scene information. Compared to the full 3D representation, despite it may slightly decrease performance on scenes with large variations along the vertical direction, the derived ground feature plane can support assets mining much better.

	NeRF-Synthetic		*360-v2*	
	Chair	Ficus	Garden	Room
TensoRF	34.80	33.37	24.27	30.91
S-AssetField	34.77	33.27	23.97	30.73
I-AssetField	34.82	33.34	24.13	31.20

Table 2: Quantitative comparison (PSNR) on test views of example scenes from NeRF-Synthetic [27] and 360-v2 [4].

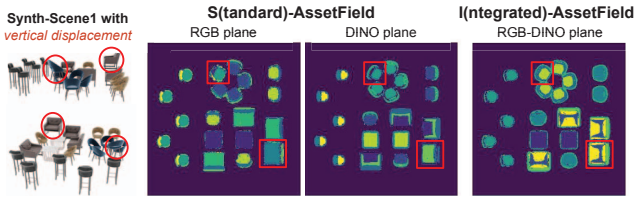

Figure 5: The RGB-DINO ground feature plane from I-AssetField yields consistent features for analogues with vertical displacement, whereas S-AssetField infers different set of features due to the lack of constraints.

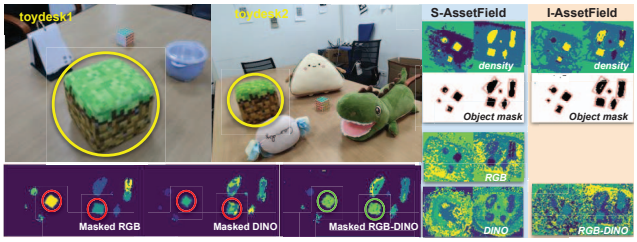

Figure 6: Multi-scene learning on the Toydesk dataset [42]. As real-world scenes usually exhibit noisier color and density features, we apply the object mask obtained from the density plane before categorization. The common object between scenes (yellow) can be correctly clustered with I-AssetField's occupancy-guided RGB-DINO plane features (green) whilst the independently modeled neural planes by S-AssetField fails (red).

Object Detection and Categorization. In Fig. 2 we already showed an example of the ground feature planes learned by AssetField compared to the xy-planes learned by TensoRF. While TensoRF produces noisy and less informative feature planes that is unfriendly for object discovery in the first place, AssetField is able to identify and categorize most of the scene contents, as shown in Fig. 7 (b). Furthermore, I-AssetField is more robust to vertical displacement, as shown in Fig. 5.

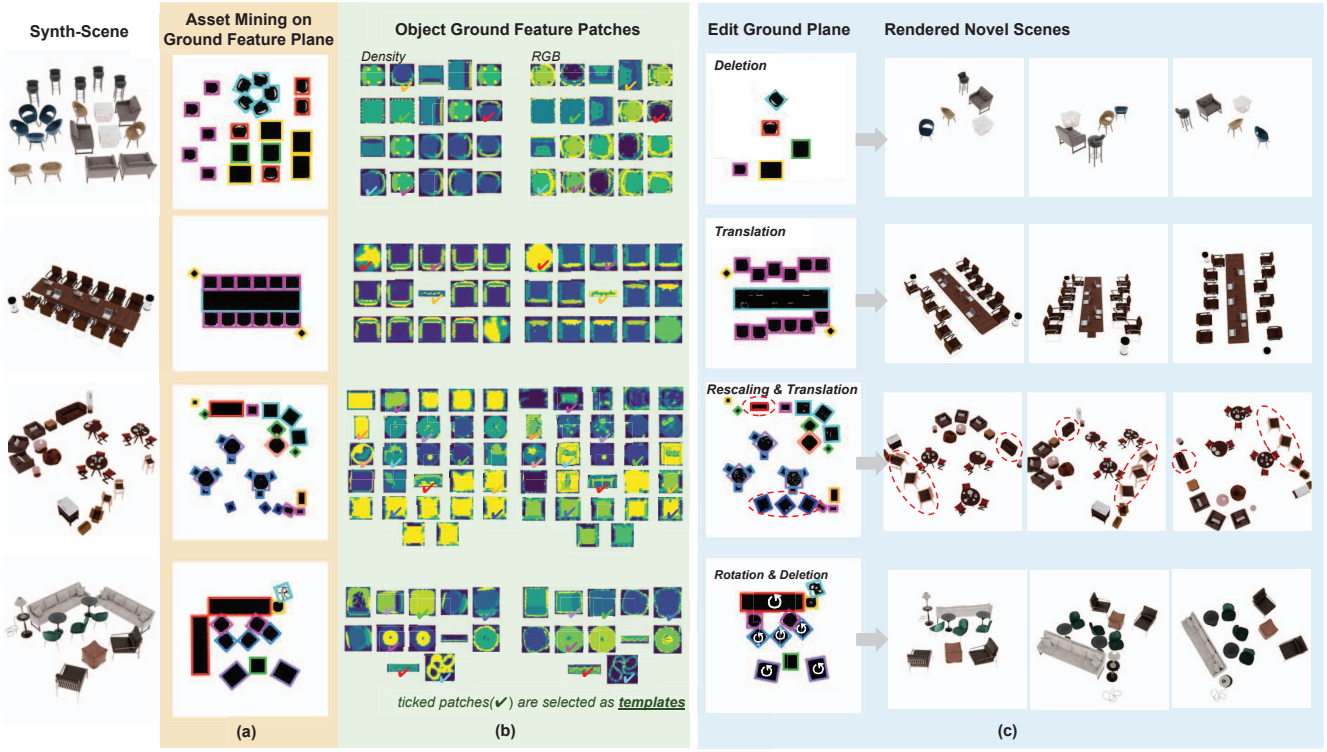

Figure 7: Results of assets mining and scene editing with *I-AssetField* on synthetic scenes. (a) Our approach learns informative density and RGB-DINO ground feature planes that support object detection and categorization. (b) With joint training, an asset library can be constructed by storing ground *feature plane patches* of the radiance field (we show label patches here for easy visualization). (c) The proposed ground plane representation provides an explicit visualization of the scene configuration, which can be directly manipulated by users. The altered ground feature planes are then fed to the global MLP renderer along with the shared vertical feature axes to render the novel scenes. Operations such as object removal, translation, rotation and rescaling are demonstrated on the right.

Recall that I-AssetField is able to identify object analogues *across different scenes*, to demonstrate such ability, we jointly model the two toy desk scenes from [42] by letting them share the same vertical feature axes and MLPs as described in Sec. 3.4. The inferred feature planes are showed in Fig. 6. Since the coordinate systems of these two scenes are not aligned with the physical world, we perform PCA [15] on camera poses such that the xy-plane is expanded along the ground/table-top. However, we cannot guarantee their table surfaces are at the same height, meaning that vertical displacement among objects is inevitable. I-AssetField is able to infer similar RGB-DINO feature values for the common cube plush (yellow circle), whilst the independently learned RGB/DINO planes in S-AssetField are affected by the height difference.

Scene Editing. Techniques on 2D image manipulation can be directly applied to ground feature planes. Fig. 7 shows that AssetField supports a variety of operations, such as object removal, insertion, translation and rescaling. Scene-level reconfiguration is also intuitive by composing objects' density and color ground feature patches. In particular, I-AssetField associates the RGB-DINO field with space occupancy, producing more plausible warping results. Fig. 8 demonstrates a case of topology deformation, where the blue bottle's density field is warped to the region of the brown bottle, while keeping their RGB(-DINO) feature unchanged. Results show that I-AssetField successfully preserves object structure and part semantics, whereas

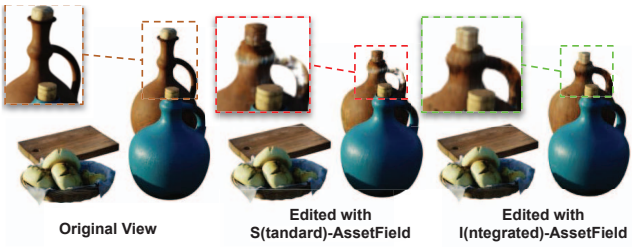

Figure 8: Density warping from the blue bottle to the region of the brown one. S-AssetField loses the structure of the brown bottle in terms of part semantics, while I-AssetField gives plausible editing result with appropriate structure transfer.

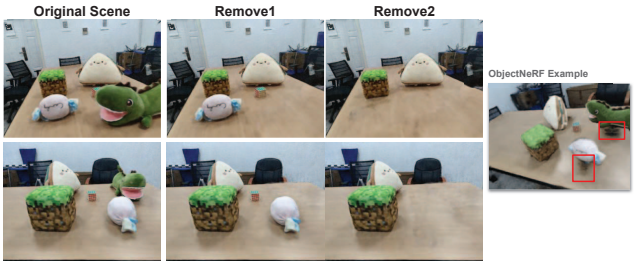

Figure 9: Object removal on toydesk [42]. Objects are first identified on each ground feature plane then substituted by the table feature patches. We simply 'crop' a feature patch from the table region and 'paste' it on to the object regions. Note that our method can also remove the shadow along with the object, whereas ObjectNeRF [42] cannot and leaves a black hole on the table-top (red).

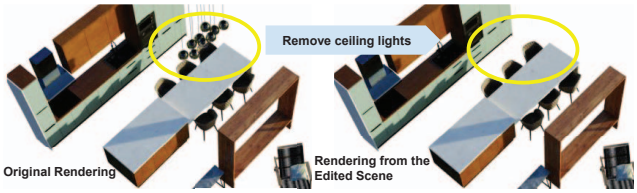

Figure 10: Expanding the 2D ground plane back to 3D feature grids, explicit control on full 3D space is allowed. We remove the ceiling light by setting the density grids as zero at the target region.

Figure 11: Example editings on real-world scenes [4] and indoor scenarios [30]. We use RGB-DINO plane for assets discovery.

S-AssetField fails to render the cork correctly.

Fig. 9 demonstrates an example where objects on the table are removed, which is realized by substituting the original object feature patches with the table feature patches. Note that our method can also remove the shadow along

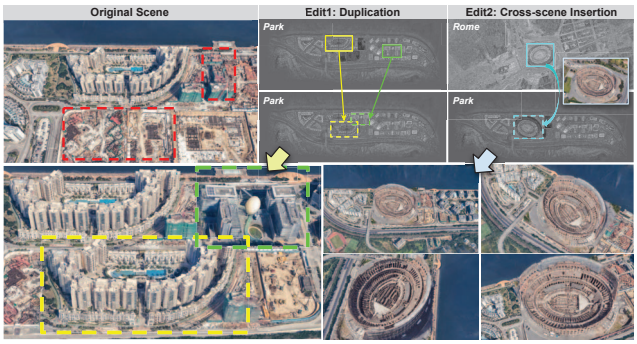

Figure 12: Editing two city scenes collected from Google Earth ©2023 Google. AssetField is versatile where users can directly operate on the ground feature plane, supporting both within-scene and cross-scene editing with realistic rendering results.

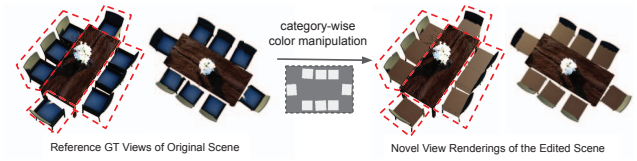

Figure 13: We apply batch-wise color changing for all instances of the chair, by replacing the template RGB feature map solely.

with the object, whereas ObjectNeRF [42] cannot eliminate the shadow and leaves a black hole on the table-top. Nevertheless, for cases where objects on top is larger (*e.g.* a chair being pushed in) or of the same size as the bottom object (*e.g.* a stack of boxes), it is necessary to lift the 2D ground plane to a full 3D voxel grid to distinguish. Despite the convenience of ground feature plane representation, it does not directly support manipulating overlapping/stacked objects. However, one can expand the ground feature plane back to 3D feature grids with its pairing vertical feature axis, and control the scene in the conventional way as described in [24]. An example is given in Fig. 10.

Fig. 11 shows AssetField's editing capability on real-world datasets [4, 42, 30]. Additionally, on a self-collected city scene from Google Earth, we find a construction site and complete it with different nearby buildings (within-scene editing), even borrow Colosseum from Rome (cross-scene editing). Results are shown in Fig 12.

Group Editing and Scene Reconfiguration. Recall that a template object can be selected for each asset category to substitute all its instances in the scene (on the ground feature planes). Consequently, we are allowed to perform group editing like changing the color of a specific category as depicted in Fig. 13. Scene-level reconfiguration is also intuitive, where users can freely compose objects from the *asset library* on a neural 'canvas' to obtain a set of new ground feature planes, as demonstrated in Fig. 14. The environments or containers (*e.g.* the floor or an empty house) can also be considered as a special asset category, where

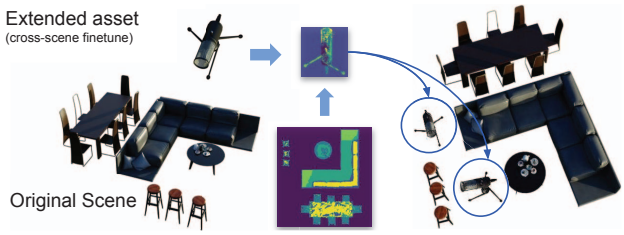

Figure 14: We expand the asset library from the living room with the newly included assets *mics* from [27]. The template of mics is in the shared latent space with the living room and can thus naturally composed together for rendering.

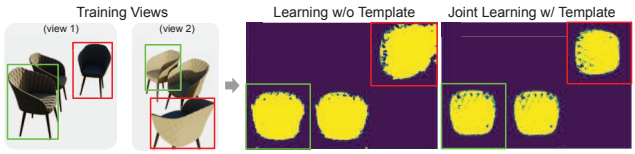

Figure 15: Feature plane refinement. The object template, when trained among all instances within the scene, produces more accurate feature map compared to the isolated ones.

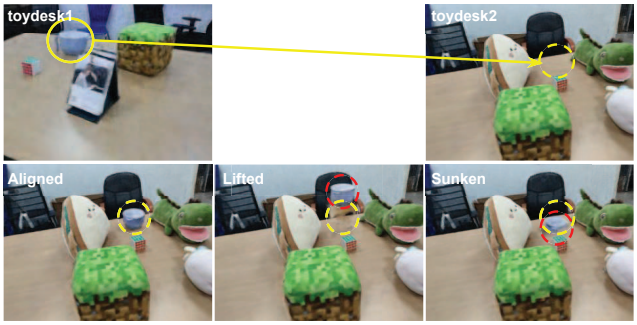

Figure 16: We insert the plastic bowl from toydesk1 to toydesk2 [42]. Vertical translation of the bowl is realized by jointly modeling the aligned/lifted/sunken-toydesk1 with toydesk2 and compose the new scene with feature patches inferred at different scene elevation.

small objects (*e.g.* furniture) can be placed into the container to deliver immersive experience. The final scene can be composited with summed density value and weighted color, as has been discussed in [38].

Template Refinement. Grid-based neural fields are sensitive to training views with insufficient point supervision, leading to noisy and inaccurate feature values. Appearance differences caused by lighting variation, occlusion, etc., interferes the obtaining of a clean template feature patch. An example can be found in Fig. 15. Due to imbalanced training view distribution, the chair in the corner receives less supervision, resulting in inconsistent object feature patch within a category. Such issue can be alleviated with a following-up *template refinement* step. With the inferred scene layout and the selected object templates (Sec. 3.3). We propose to replace all instances $p \in \mathcal{P}$ with their representative category template $\hat{p}$ and optimize this set of feature patches to reconstruct the scene instead of the full ground planes. Consequently, the template feature patch integrates supervisions from all instances in the scene to overcome appearance variations and sparse views.

Failure Cases. As our proposed ground feature plane representation targets scenes widespread along a horizontal plane, the flexibility of moving objects vertically is restricted because of the shared vertical feature axes. In Fig. 16 we show an example of (cross-scene) insertion of an object to different heights. In this experiment, we jointly model *4* scenes, namely toydesk2, *aligned*-toydesk1, *lifted*-toydesk1 and *sunken*-toydesk1. Specifically, the camera poses of *lifted*/*sunken*-toydesk1 is shifted up/down to represent toydesk1 at different elevation. The new scene is then composed using the bowl's feature patches inferred at each height variation. However, since the feature patch of the bowl is entangled with the table, such strategy results in artifacts where part of the table is also lifted, as shown in the middle of Fig. 16. Alternatively, one may choose to expand the ground plane into 3D and directly exchange the feature voxels along the vertical direction as in Fig. 10; or incorporate a new elevation map that is used to be subtracted by the z values when indexing the corresponding z embedding.

5. Discussion and Conclusion

We present AssetField, a novel framework that mines assets from neural fields. We adopt a ground feature plane representation to model scene density, color and semantic fields, on which assets mining and grouping can be directly conducted. The novel occupancy-guided RGB-DINO feature plane enables cross-scene asset grouping and the construction of an expandable neural asset library, enabling a variety of intuitive scene editing at object-, category- and scene-level. Extensive experiments are conducted to show the easy control over multiple scenes and the realistic rendering results given novel scene configurations. However, AssetField still suffer from limitations like: separating connected objects in the scene; handling stacked/overlapped objects; and performing vertical translations. Rendering quality might also be compromised due to complex scene background in real-world. More limitations are discussed in the supplementary. We believe the proposed representation can be further explored for the manipulation and construction of large-scale scenes, *e.g.*, by following floorplans or via a programmable scheme like *procedural modeling*.

Acknowledgment This project is funded in part by Shanghai AI Laboratory (P23KS00020, 2022ZD0160201), CUHK Interdisciplinary AI Research Institute, and the Centre for Perceptual and Interactive Intelligence (CPII) Ltd under the Innovation and Technology Commission (ITC)'s InnoHK.

References

[1] Google earth studio. https://earth.google.com/studio/. 6

[2] Shir Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel. Deep vit features as dense visual descriptors. *arXiv preprint arXiv:2112.05814*, 2021. 2, 6

[3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5460–5469, 2021. 2

[4] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 6, 8

[5] Michael Bleyer, Christoph Rhemann, and Carsten Rother. Patchmatch stereo - stereo matching with slanted support windows. In *BMVC*, 2011. 2

[6] G. Bradski. 4

[7] Adrian Broadhurst, Tom Drummond, and Roberto Cipolla. A probabilistic framework for space carving. *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, 1:388–393 vol.1, 2001. 2

[8] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9650–9660, 2021. 2

[9] Mathilde Caron, Hugo Touvron, Ishan Misra, Herv'e J'egou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9630–9640, 2021. 4

[10] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *arXiv*, 2021. 2

[11] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022. 2, 6

[12] Oliver S Cossairt, Shree K. Nayar, and Ravi Ramamoorthi. Light field transfer: global illumination between real and synthetic objects. *ACM SIGGRAPH 2008 papers*, 2008. 2

[13] Terrance DeVries, Miguel Angel Bautista, Nitish Srivastava, Graham W Taylor, and Joshua M Susskind. Unconstrained scene generation with locally conditioned radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14304–14313, 2021. 3

[14] Zhiwen Fan, Peihao Wang, Yifan Jiang, Xinyu Gong, Dejia Xu, and Zhangyang Wang. Nerf-sos: Any-view self-supervised object segmentation on complex scenes. *arXiv preprint arXiv:2209.08776*, 2022. 2

[15] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901. 7

[16] Michelle Guo, Alireza Fathi, Jiajun Wu, and Thomas A. Funkhouser. Object-centric neural scene rendering. *ArXiv*, abs/2012.08503, 2020. 2, 3

[17] Kevin Karsch, Varsha Hedau, David A. Forsyth, and Derek Hoiem. Rendering synthetic objects into legacy photographs. *Proceedings of the 2011 SIGGRAPH Asia Conference*, 2011. 2

[18] Natasha Kholgade, Tomas Simon, Alexei Efros, and Yaser Sheikh. 3d object manipulation in a single photograph using stock 3d models. *ACM Transactions on Graphics (TOG)*, 33(4):1–12, 2014. 2

[19] Young Min Kim, Niloy J Mitra, Dong-Ming Yan, and Leonidas Guibas. Acquiring 3d indoor environments with variability and repetition. *ACM Transactions on Graphics (TOG)*, 31(6):1–11, 2012. 2

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015. 6

[21] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. In *Advances in Neural Information Processing Systems*, volume 35, 2022. 2, 4

[22] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas J Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic neural fields: A semantic object-aware neural scene representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12871–12881, 2022. 2, 3

[23] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *ArXiv*, abs/2007.11571, 2020. 2

[24] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *NeurIPS*, 2020. 2, 5, 8

[25] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inf. Theory*, 28:129–136, 1982. 4

[26] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7206–7215, 2020. 2

[27] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2, 3, 6, 9

[28] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 2

[29] Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms. *ArXiv*, abs/1109.2378, 2011. 5

[30] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H Mueller, Chakravarty R Alla Chaitanya, Anton Kaplanyan, and Markus Steinberger. Donerf: Towards real-time rendering of compact neural radiance fields using depth

oracle networks. In *Computer Graphics Forum*, volume 40, pages 45–59. Wiley Online Library, 2021. 6, 8

[31] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2855–2864, 2021. 2

[32] Avishkar Saha, Oscar Alejandro Mendez Maldonado, Chris Russell, and R. Bowden. Translating images into maps. *2022 International Conference on Robotics and Automation (ICRA)*, pages 9200–9206, 2022. 3

[33] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016. 2

[34] Johannes L. Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, 2016. 2

[35] Steven M. Seitz and Charles R. Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35:151–173, 1997. 2

[36] Prafull Sharma, Ayush Tewari, Yilun Du, Sergey Zakharov, Rares Ambrus, Adrien Gaidon, William T Freeman, Fredo Durand, Joshua B Tenenbaum, and Vincent Sitzmann. Seeing 3d objects in a single image via self-supervised static-dynamic disentanglement. *arXiv preprint arXiv:2207.11232*, 2022. 3

[37] Satoshi Suzuki and Keiichi Abe. Topological structural analysis of digitized binary images by border following. *Comput. Vis. Graph. Image Process.*, 30:32–46, 1985. 4

[38] Jiaxiang Tang, Xiaokang Chen, Jingbo Wang, and Gang Zeng. Compressible-composable nerf via rank-residual decomposition. *ArXiv*, abs/2205.14870, 2022. 9

[39] Vadim Tschernezki, Iro Laina, Diane Larlus, and Andrea Vedaldi. Neural Feature Fusion Fields: 3D distillation of self-supervised 2D image representations. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2022. 2, 4

[40] Zhou Wang, Alan Conrad Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13:600–612, 2004. 6

[41] Bangbang Yang, Yinda Zhang, Yijin Li, Zhaopeng Cui, S. Fanello, Hujun Bao, and Guofeng Zhang. Neural rendering in a room. *ACM Transactions on Graphics (TOG)*, 41:1 – 10, 2022. 2, 3

[42] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *International Conference on Computer Vision (ICCV)*, October 2021. 2, 6, 7, 8, 9

[43] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 13759–13768, 2021. 3

[44] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5491–5500, 2022. 2, 5

[45] Hong-Xing Yu, Leonidas J. Guibas, and Jiajun Wu. Unsupervised discovery of object radiance fields. In *International Conference on Learning Representations*, 2022. 3

[46] Wentao Yuan, Zhaoyang Lv, Tanner Schmidt, and S. Lovegrove. Star: Self-supervised tracking and reconstruction of rigid objects in motion with neural rendering. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13139–13147, 2021. 2

[47] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *ArXiv*, abs/2010.07492, 2020. 2

[48] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018. 6

[49] Youyi Zheng, Xiang Chen, Ming-Ming Cheng, Kun Zhou, Shi-Min Hu, and Niloy J Mitra. Interactive images: Cuboid proxies for smart image manipulation. *ACM Trans. Graph.*, 31(4):99–1, 2012. 2

[50] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J Davison. In-place scene labelling and understanding with implicit scene representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15838–15847, 2021. 2