Supplementary for AssetField: Assets Mining and Reconfiguration in Ground Feature Plane Representation

Yuanbo Xiangli[1*], Linning Xu[1*], Xingang Pan[3], Nanxuan Zhao[4], 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] Adobe Research
{xy019,xl020,dhlin}@ie.cuhk.edu.hk xpan@mpi-inf.mpg.de
nanxuanzhao@gmail.com daibo@pjlab.org.cn

1. Overview

This supplementary is organized as follows: (1) In the first section we show some concrete examples of the useful asset library in industry when building indoor and outdoor environments. (2) The second part elaborate our training details, including model adaptation and hyper-parameter setting. (3) The third part includes ablation studies on the number of clusters to discretize ground feature plane; and alternations decode 2D ground feature into 3D. (4) Next we show some special cases which might require extra cautions or post-processing. (5) Finally we include some applications to demonstrate the flexibility of AssetField and its potential to cooperate with physical rendering pipelines. Video demos are also provided to show various editing effects using the asset library extracted by AssetField, spanning instance-level, category-level to scene-level manipulation.

2. Asset Library in Industry

Having an asset library is one of the key enabler to create large-scale environments in virtual presence. Man-made environments like indoor and urban scenes are comprised of highly structured and repetitive items. For example, an office is composed by groups of chairs and tables of different types, organized in grid or other regular patterns; a city is a structured combination of city blocks, roads, facilities and greening, where each block is constructed by buildings of different categories. It is a common practice in gaming industries to first create an asset library, then deploy asset instances in the scene per demand. Fig. 1 shows examples of asset libraries for rooms and city regions [2, 1, 5, 4]. On the right are asset instances grouped in categories. Note that in practice, the definition of category 'templates' and 'instances' different from projects to projects. For example, in Fig. 1 (b), sofas with the same shape but different textures are stored in the asset library; while in Fig. 1 (d) distinctive

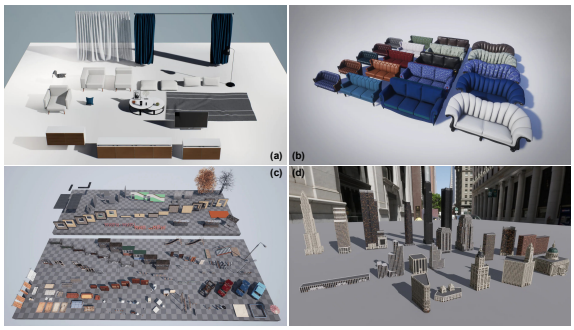

Figure 1. Examples of asset libraries used in virtual environments built with UE5[2]. (a) are assets used to construct a living room; (b) is a collections of sofa instances; (c) are assets in a city region; (d) is a collection of buildings templates.

landmarks are saved in the asset library as templates.

3. Training Details

We use NeRF [8] and TensoRF [3] as baselines to evaluate the rendering quality of the original scenes. Concretely, (1) For density field, both S(tandard)-AssetField and I(ntegrated)-AssetField uses 16 components (*i.e.* channel dimension); feature vectors sampled from ground feature planes $\mathcal{M}$ and vertical feature axes $\mathcal{H}$ are combined via outer product following [3]. S-AssetField separately models color and semantic fields using 48 components for each field, and also adopts outer product to recover 3D features in space. I-AssetField unifies color and semantic field into the RGB-DINO ground feature plane $\mathcal{M}_{rgb\text{-}dino}$, which has 48 components. The feature vectors sampled from $\mathcal{M}_{rgb\text{-}dino}$ and the mapped latent features v_{rgb} and v_{dino} are concatenated then decoded into RGB and DINO feature values. Decoders Dec_σ, Dec_{rgb}, Dec_{dino} are all single layer MLPs. (2) TensoRF baseline uses $6, 16, 16$ ten-

[2] https://www.unrealengine.com/en-US/unreal-engine-5

sor components for each XYZ-mode of density, color and DINO fields respectively, which results in similar amount of total components $(18, 48, 48)$ as AssetField $(16, 48, 48)$. (3) For NeRF baseline, the hidden feature before alpha prediction is transformed by a 2-layer MLP with ReLU activation to a 384-dim feature vector, which is then aggregated via volume rendering to output the pixel's DINO feature. To alleviate over-fitting and local minima issues in gradient descent, we utilize TV (total variation) loss on ground plane features for both AssetField and TensoRF. Specifically, TV loss weight is set to 0.1 for density plane, and 0.01 for RGB/DINO/RGB-DINO plane. We report PSNR, SSIM [13] and LPIPS (e^{-1}) [15] scores for novel view synthesis, and qualitatively show the editing results on various tasks.

On synthetic scenes (main-Fig.7), we clustered all ground feature planes into 10 clusters at assets mining step (main-Sec.3.2) using K-means [7]. During assets grouping, Agglomerative clustering [9] is adopted to group feature patches using JS-divergence [6] as the distance metric. The number of clusters is set by observing the training views, and set to be a bit higher than the observed number of categories. On real-world datasets we normally use 3 to 5 clusters to filter objects for assets mining; and 7 to 10 clusters for assets grouping.

4. Ablation

Number of Clusters. Recall that we first discretize the feature plane with K-means clustering to obtain a binary image and detect objects with the finding contour algorithm. We propose to perform this step on the density ground feature plane as it is cleaner and exhibits sharper object boundaries. In Fig. 2, we show the clustering results of using $\{2, 6, 10, 20\}$ clusters and also the results of clustering on the RGB-DINO feature plane. The scene demonstrated here is a restaurant, which contains various objects of different sizes and are organized in patterns. It can be noticed that, on the density ground plane, when using a small number of clusters (*e.g.* 2 clusters), some objects cannot be identified from the background, as shown in the 3rd row of Fig. 2. When the number of clusters increases, more objects can be discovered by the finding contour algorithm, whereas using 10 and 20 clusters give similar results. On the other hand, although the RGB-DINO feature plane can easily separate objects from the background, the produced feature map is not as sharp as the density plane. As a result, close objects are always attached to each other, making them inseparable on the feature plane, as illustrated in the 5th row of Fig. 2 .

Decoding Ground Feature Plane to 3D. Recall that features sampled from ground planes need to be combined with vertical information to decode 3D scene features. Naively, the plane and axis features can be directly concatenated and fed to decoders. We on the other hand follow the idea of [3], and combine them via outer-product to recover 3D scene information. Different from ours, Sharma *et al.* [10] directly use the vertical z-axis coordinate, then concatenate with the ground plane feature to obtain 3D features. In Tab. 1, we quantitatively show the novel view rendering quality of the aforementioned three different ways. In general, the performance on novel view synthesis is similar, where *Concat* appear to be inferior to others on more complex scenes (Scene3 and Scene4). Inspecting the learned ground feature planes, such as Fig. 3, it can be noticed that *Concat* gives fuzzy feature, leading to inseparable objects; $PE(z)$ tends to learn a separate feature space for boundaries, which makes objects indistinguishable if only the background region is masked. However, with an additional edge mask (left corner blue boxes), instances can still be isolated.

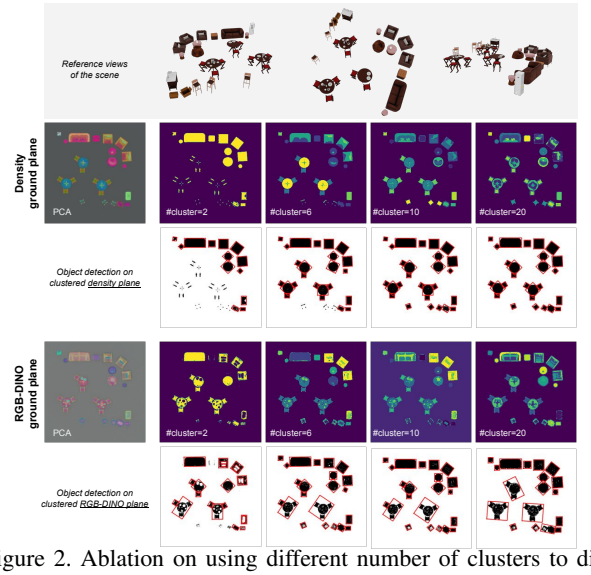

Figure 2. Ablation on using different number of clusters to discretize ground feature planes at object detection step. In general, the density plane is cleaner and sharper than the RGB-DINO plane; objects placed close to each other are very likely to be inseparable on the RGB-DINO plane (5th row). On the density plane, small objects might be clustered into background when using a small number of clusters; increasing the number of clusters can alleviate this issue (3rd row).

	Scene1			*Scene2*			*Scene3*			*Scene4*		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
Concat	36.398	0.991	0.051	37.135	0.994	0.036	32.552	0.974	0.124	32.960	0.978	0.227
PE(z)	36.158	0.989	0.048	37.846	0.996	0.029	37.385	0.995	0.030	36.899	0.987	0.085
Product	36.526	0.991	0.047	37.271	0.994	0.035	37.249	0.995	0.032	37.716	0.991	0.060

Table 1. Ablation on the 4 scenes (the same as in the main paper) using different ways to combine vertical information with ground plane features. (1) *Concat* concatenates z-axis features with ground plane features; (2) *PE(z)* concatenates the Fourier frequency encoded z-coordinates with ground plane features; (3) *Product* uses outer-product between z-axis features and ground plane features. We report PSNR(↑), SSIM(↑) [13] and LPIPS(↓) [15] for evaluation.

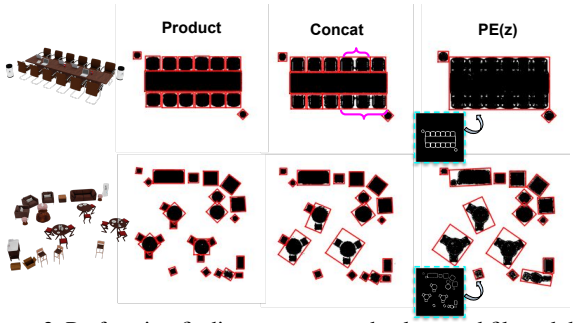

Figure 3. Performing finding contours on background filtered density ground feature plane learned with *(Outer-)Product*, *Concat* and *PE(z)*. *Concat* gives fuzzy feature plane where close objects are grouped together; $PE(z)$ tends to learn a feature for objects' edge, which makes objects indistinguishable if only the background region is masked. However, with an additional edge mask (also derived from clustering) as shown in the blue boxes, instances can still be isolated.

5. Special Cases and Potential Failures

Recall that in the main paper, we propose a pipeline for object detection which involves masking the background and performing contour detection on the binary image. While this strategy works on most scenes composed of non-overlapping objects with convex shapes, there are some exceptional cases that may cause failure at this step but can still be solved with delicate adjustment.

Floor Removal. Background can be handled in various ways. In the main paper, we use clustered DINO feature to separate objects from the background (*i.e.* floor). It can also be estimated from depth by first learning to model the scene to obtain a depth map; or from annotations like a mask or user scribble. Background is also separable by grouping the learned ground features into a few clusters (*e.g.*, 3 clusters) that roughly corresponds to foreground and background regions; and manually filter out the background, which is adopted in our experiments on real-world scenes (*e.g.* main-Fig.6, main-Fig.10). Fig. 4 shows an example scene from [12] where modeling with floor yields unwanted noise in the ground feature plane. Despite being disturbing, adding a simple clustering step can still distinguish it from objects. The filtered ground feature plane is then suitable for object detection afterward.

Nested Structure. Scene contents may not always be in convex shape from a top view. For example, in Fig. 5 (a), the orange chair is partially enclosed by a the gray table that is in an 'L'-shape. Since the 'L'-shaped table is not convex, the bounding box predicted by contour detection is not tightly aligned with the object edge. Consequently, it includes the chair in its patch. Suppose the table and chair are desired to be separated instead of being treated as a combo. In that case, the user can first identify the chair patch, then remove it by filling this region with background feature values, as demonstrated in Fig. 5 (c). Note that when reconfiguring the scene, the 'L'-shaped table patch need to be always placed at the back so it won't occlude any enclosed objects.

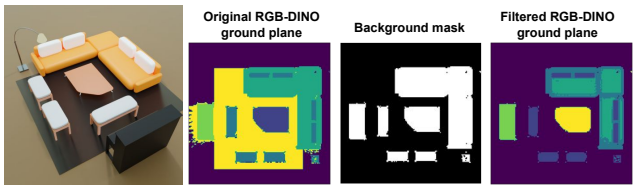

Figure 4. Modeling scene with floor results in noisy ground feature plane. The floor region can be identified on RGB-DINO feature plane via a clustering step. The filtered feature plane is then friendly to object detection.

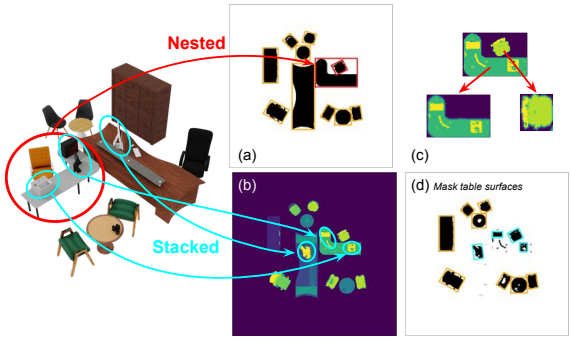

Figure 5. Special cases in object detection step. (a) Nested structure caused by non-convex shaped objects, which can be separated by (c) firstly identify the enclosed chair, then set its value to background feature to obtain solely the table patch. (b) Stacked structure where items are placed on top of another object's surface, which can be detected by (d) another round of filtering that treats the table-top as the background.

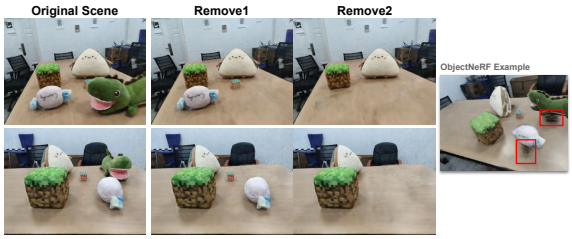

Figure 6. Object removal on toydesk [14]. Objects are first identified on each ground feature plane then substituted by the table feature patches. We simply 'crop' a feature patch from the table region and 'paste' it on to the object regions. Note that our method can also remove the shadow along with the object, whereas ObjectNeRF [14] cannot and leaves a black hole on the table-top (red).

Stacked Instances. Man-made scenes are actually in hierarchical structures. Fig. 5 (b)) shows an example where small objects are placed on top of a table, forming a hierarchical structure. If the user wants to also extract these

[4] https://www.blender.org/

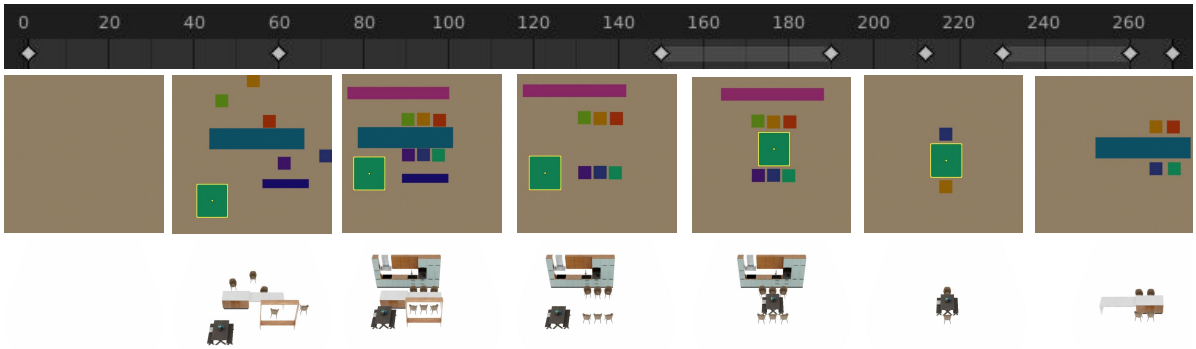

Figure 7. Scene manipulation on 2D canvas with Blender[4] animation tools. The first row is the *timeline* of the animation; the second row is the top-view of the scene (assets represented by colored boxes), resemble a 2D canvas. We can first configure the scene and plan the transformation of assets in Blender, then export the frame sequence as 'floorplans' and fill each colored box with object features (with warping or rescaling) extracted by AssetField. The last row is the direct rendering results correspond to the key frames (white dots in timeline). Using various interpolation methods provided by Blender animation tools, users can easily play with all kinds of transformations and configurations with smooth transitions. The integrated inference pipeline of Blender (or any other user-friendly 2D/3D softwares) and AssetField also enable investigations on the 3D consistent rendering results from arbitrary viewing angle.

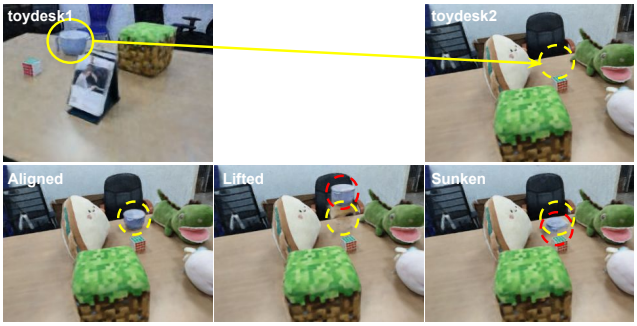

Figure 8. We insert the plastic bowl from toydesk1 to toydesk2 [14]. Vertical translation of the bowl is realized by jointly modeling the aligned/lifted/sunken-toydesk1 with toydesk2 and compose the new scene with feature patches inferred at different scene elevation.

small items as asset, one more step of filtering can be performed on the ground feature plane by treating table surfaces as the background, as shown in Fig. 5 (d). After separating the items from the table surface, the 'hole' left on table's feather patch can be filled with the average feature of the table instance. Fig. 6 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 with the object, whereas ObjectNeRF [14] 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.

Vertical Translation. As our proposed ground feature plane representation is specifically designed for scenes widespread along a horizontal plane, the flexibility of moving objects vertically is restricted because of the shared vertical feature axes. However, since height information is fully encoded into the scene density feature, an object instance can still appear at a different elevation given there is a lifted object instance *within its category*. User can directly replace current instance, or substitute its density feature patch with a lifted one. In Fig. 8 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. 8. Alternatively, one may choose to expand the ground plane into 3D and directly exchange the feature voxels along the vertical direction; or incorporate a new elevation map that is used to be subtracted by the z values when indexing the corresponding z embedding.

6. Miscs.

Asset Library Expansion. In real-world scenario, one might want to enlarge existing asset library with new objects or objects from other scenes, instead of jointly modeling the new and old scenes from scratch. To incorporate a new scene S_i, we fix $\mathcal{H}_0$ from the existing scene(s) and learn a set of exclusive ground plans $\mathcal{M}_i$ to represent S_i. Note that we need to scale the new scene S_i to be within

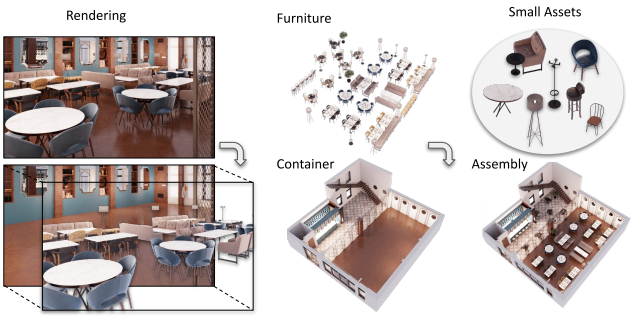

Figure 9. We separately learn the furniture asset field and container asset fields via the cross-scene learning, which can be assembled to form a realistic indoor scene.

the height range of the existing scene(s) since $\mathcal{H}_0$ is fixed. Contents vertically exceed the original scene bound will not be recovered. Furthermore, to make cross-scene learning more effective, we suggest to start the training on a diverse scene, or jointly train different scenes to obtain a informative $\mathcal{H}_0$ (*e.g.* avoid too much vertical empty space being encoded to z-axis feature), such that a new set of ground feature planes $\mathcal{M}_i$ of density, color and semantic field can be effectively decoded with $\mathcal{H}_0$ to fit new scenes.

Interactive Editing With AssetField, 3D scene editing can be achieved with simple manipulation on 2D planes. Here we show one example of how we enable the easy production of animated scenes with manipulated layouts. We first construct a blank canvas in Blender representing the empty ground plane, and create a "rectangle object" for each mined asset from the ground plane with the retrieved *center*, *width and height*, and *orientations*. Users can then easily manipulate each object and add animation to the layouts with all kinds of tools in Blender (*e.g.*, translation, rotation, deformation, scaling, *etc.*). A demo timeline in Blender is shown in Fig. 7. We can then obtain the configuration of each object at each timestamp, and use them to configure the ground feature planes with the placed asset templates. The learned renderer can then naturally produce the animated frames with changing layouts from arbitrary viewpoints. The demo video for this example is provided.

Scene Assembling. The environments or containers (*e.g.*, an empty house) can also be considered as a special category of assets, where small objects (*e.g.*, furniture) can be placed into to deliver immersive experience. Fig. 9 illustrates such idea by separately modeling these two types of assets. The final scene can then be composited with summed density value and weighted color, which can correctly handled occlusions between different objects [11].

[4] https://www.blender.org/

References

[1] North 3D. North sofa pack vol.1. https://www.unrealengine.com/marketplace/en-US/product/north-sofa-pack-vol. 1

[2] AndresAlv. Simplelivingroom2. https://www.unrealengine.com/marketplace/en-US/product/simplelivingroom2. 1

[3] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022. 1, 2

[4] Epic Games. City sample buildings. https://www.unrealengine.com/marketplace/en-US/product/city-sample-buildings. 1

[5] GeorgeShachnev. Post-soviet world. https://www.unrealengine.com/marketplace/en-US/product/post-soviet-world. 1

[6] Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Trans. Inf. Theory*, 37:145–151, 1991. 2

[7] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inf. Theory*, 28:129–136, 1982. 2

[8] 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. 1

[9] Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms. *ArXiv*, abs/1109.2378, 2011. 2

[10] 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. 2

[11] Jiaxiang Tang, Xiaokang Chen, Jingbo Wang, and Gang Zeng. Compressible-composable nerf via rank-residual decomposition. *arXiv preprint arXiv:2205.14870*, 2022. 5

[12] Bing Wang, Lujia Chen, and Bo-Hsiang Yang. Dm-nerf: 3d scene geometry decomposition and manipulation from 2d images. *ArXiv*, abs/2208.07227, 2022. 3

[13] 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. 2

[14] 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. 3, 4

[15] 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. 2