DM-NERF: 3D SCENE GEOMETRY DECOMPOSITION AND MANIPULATION FROM 2D IMAGES

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we study the problem of 3D scene geometry decomposition and manipulation from 2D views. By leveraging the recent implicit neural representation techniques, particularly the appealing neural radiance fields, we introduce an object field component to learn unique codes for all individual objects in 3D space only from 2D supervision. The key to this component is multiple carefully designed loss functions to enable every 3D point, especially in non-occupied space, to be effectively optimized without 3D labels. In addition, we introduce an inverse query algorithm to freely manipulate any specified 3D object shape in the learned scene representation. Notably, our manipulation algorithm can explicitly tackle key issues such as object collisions and visual occlusions. Our method, called DM-NeRF, is among the first to simultaneously reconstruct, decompose, manipulate and render complex 3D scenes in a single pipeline. Extensive experiments on three datasets clearly show that our method can accurately decompose all 3D objects from 2D views, allowing any interested object to be freely manipulated in 3D space such as translation, rotation, size adjustment, and deformation.

1 INTRODUCTION

In many cutting-edge applications such as mixed reality on mobile devices, users may desire to virtually manipulate objects in 3D scenes, such as moving a chair or making a flying broomstick in a 3D room. This would allow users to easily edit real scenes at fingertips and view objects from new perspectives. However, this is particularly challenging as it involves 3D scene reconstruction, decomposition, manipulation, and photorealistic rendering in a single framework (Savva et al., 2019).

A traditional pipeline firstly reconstructs explicit 3D structures such as point clouds or polygonal meshes using SfM/SLAM techniques (Ozyesil et al., 2017; Cadena et al., 2016), and then identifies 3D objects followed by manual editing. However, these explicit 3D representations inherently discretize continuous surfaces, and changing the shapes often requires additional repair procedures such as remeshing (Alliez et al., 2002). Such discretized and manipulated 3D structures can hardly retain geometry and appearance details, resulting in the generated novel views to be unappealing. Given this, it is worthwhile to design a new pipeline which can recover continuous 3D scene geometry only from 2D views and enable object decomposition and manipulation.

Recently, implicit representations, especially NeRF (Mildenhall et al., 2020), emerge as a promising tool to represent continuous 3D geometries from images. A series of succeeding methods (Boss et al., 2021; Chen et al., 2021; Zhang et al., 2021c) are rapidly developed to decouple lighting factors from structures, allowing free edits of illumination and materials. However, they fail to decompose 3D scene geometries into individual objects. Therefore, it is hard to manipulate individual object shapes in complex scenes. Recent works (Stelzner et al., 2021; Zhang et al., 2021b; Kania et al., 2022; Yuan et al., 2022; Tschernezki et al., 2022; Kobayashi et al., 2022; Kim et al., 2022; Benaim et al., 2022; Ren et al., 2022) have started to learn disentangled shape representations for potential geometry manipulation. However, they either focus on synthetic scenes or simple model segmentation, and can hardly extend to real-world 3D scenes with dozens of objects.

In this paper, we aim to simultaneously recover continuous 3D scene geometry, segment all individual objects in 3D space, and support flexible object shape manipulation such as translation, rotation, size adjustment and deformation. In addition, the edited 3D scenes can be also rendered from novel views. However, this task is extremely challenging as it requires: 1) an object decom-

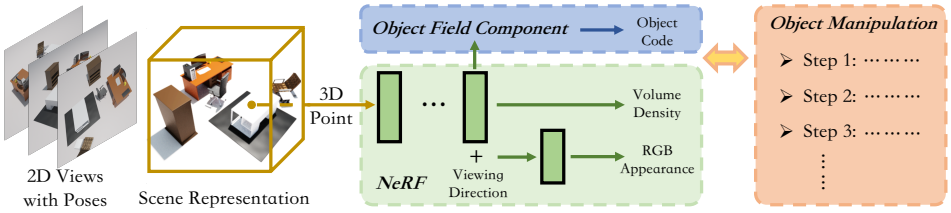


Figure 1: The general workflow of DM-NeRF. NeRF (green block) is used as the backbone. We propose the object field and manipulation components as illustrated by blue and orange blocks.

position approach amenable to continuous and implicit 3D fields, without relying on any 3D labels for supervision due to the infeasibility of collecting labels in continuous 3D space; 2) an object manipulation method agreeable to the learned implicit and decomposed fields, with an ability to clearly address visual occlusions inevitably caused by manipulation.

To tackle these challenges, we design a simple pipeline, **DM-NeRF**, which is built on the successful NeRF, but able to **d**ecompose the entire 3D space into object fields and freely **m**anipulate their geometries for realistic novel view rendering. As shown in Figure 1, it consists of 3 major components: 1) the existing radiance field to learn volume density and appearance for every 3D point in space; 2) the object field which learns a unique object code for every 3D point; 3) the object manipulator that directly edits the shape of any specified object and automatically tackles visual occlusions.

The **object field** is the core of DM-NeRF. This component aims to predict a one-hot vector, *i.e.,* object code, for every 3D point in the entire scene space. However, learning such code involves critical issues: 1) there are no ground truth 3D object codes available for full supervision; 2) the number of total objects is variable and there is no fixed order for objects; 3) the non-occupied (empty) 3D space must be taken into account, but there are no labels for supervision as well. In Section 3.1, we show that our object field together with multiple carefully designed loss functions can address them properly, under the supervision of color images with object 2D masks only.

Once the object field is well learned, our **object manipulator** aims to directly edit the geometry and render novel views when specifying the target objects, viewing angels, and manipulation settings. A naïve method is to obtain explicit 3D structures followed by manual editing and rendering, so that any shape occlusion and collision can be explicitly addressed. However, it is extremely inefficient to evaluate dense 3D points from implicit fields. To this end, as detailed in Section 3.2, we introduce a lightweight inverse query algorithm to automatically edit the scene geometry.

Overall, our pipeline can simultaneously recover 3D scene geometry, decompose and manipulate object instances only from 2D images. Extensive experiments on multiple datasets demonstrate that our method can precisely segment all 3D objects and effectively edit 3D scene geometry, without sacrificing high fidelity of novel view rendering. Our key contributions are:

- We propose an object field to directly learn a unique code for each object in 3D space only from 2D images, showing remarkable robustness and accuracy over the commonly-used individual image based segmentation methods.
- We propose an inverse query algorithm to effectively edit specified object shapes, while generating realistic scene images from novel views.
- We demonstrate superior performance for 3D decomposition and manipulation, and also contribute the first synthetic dataset for quantitative evaluation of 3D scene editing. Our code and dataset are available at `https://github.com/xxx` for future research in the community.

We note that recent works (Yang et al., 2021; Ost et al., 2021; Wu et al., 2022) address the similar task as ours. However, ObjectNeRF (Yang et al., 2021) only decomposes a foreground object, NSG (Ost et al., 2021) focuses on decomposing dynamic objects, and ObjectSDF (Wu et al., 2022) only uses semantic label as regularization. None of them directly learns to segment multiple 3D objects as ours. A few works (Norman et al., 2022; Kundu et al., 2022; Fu et al., 2022) tackle panoptic segmentation in radiance fields. However, they fundamentally segment objects in 2D images followed by learning a separate radiance field for each object. By comparison, our method learns to directly segment all objects in the 3D scene radiance space, and it demonstrates superior accuracy and robustness than 2D object segmentation methods such as MaskRCNN (He et al., 2017) and Swin Transformer (Liu et al., 2021b), especially when 2D object labels are noisy during training, as detailed in Section 4.

2 RELATED WORK

Explicit 3D Representations: To represent 3D geometry of objects and scenes, voxel grids (Choy et al., 2016), octree (Tatarchenko et al., 2017), meshes (Kato et al., 2018; Groueix et al., 2018), point clouds (Fan et al., 2017) and shape primitives (Zou et al., 2017) are widely used. Although impressive progress has been achieved in shape reconstruction (Yang et al., 2019a; Xie et al., 2019), completion (Song et al., 2017), generation (Lin et al., 2018), and scene understanding (Tulsiani et al., 2018; Gkioxari et al., 2019), the quality of these representations are inherently limited by the spatial resolution and memory footprint. Therefore, they are hard to represent complex 3D scenes.

Implicit 3D Representations: To overcome the discretization issue of explicit representations, coordinate based MLPs have been recently proposed to learn implicit functions to represent continuous 3D shapes. These implicit representations can be generally categorized as: 1) signed distance fields (Park et al., 2019), 2) occupancy fields (Mescheder et al., 2019), 3) unsigned distance fields (Chibane et al., 2020; Wang et al., 2022), 4) radiance fields (Mildenhall et al., 2020), and 5) hybrid fields (Wang et al., 2021). Among them, both occupancy fields and signed distance fields can only recover closed 3D shapes, and are hard to represent open geometries. These representations have been extensively studied for novel view synthesis (Niemeyer et al., 2020; Trevithick & Yang, 2021) and 3D scene understanding (Zhang et al., 2021a; Zhi et al., 2021a;b). Thanks to the powerful representation, impressive results have been achieved, especially from the neural radiance fields and its succeeding methods. In this paper, we also leverage the success of implicit representations, particularly NeRF, to recover the geometry and appearance of 3D scenes from 2D images.

3D Object Segmentation: To identify 3D objects from complex scenes, existing methods generally include 1) image based 3D object detection (Mousavian et al., 2017), 2) 3D voxel based detection methods (Zhou & Tuzel, 2018) and 3) 3D point cloud based object segmentation methods (Yang et al., 2019b). Given large-scale datasets with full 3D object annotations, these approaches have achieved excellent object segmentation accuracy. However, they are particularly designed to process explicit and discrete 3D geometries. Therefore, they are unable to segment continuous and fine-grained shapes, and fail to support geometry manipulation and realistic rendering. With the fast development of implicit representation, it is desirable to learn object segmentation for implicit surfaces. To the best of our knowledge, this paper is among the first to segment all 3D objects of implicit representations for complex scenes, only with color images and 2D object labels for supervision.

3D Scene Editing: Existing methods of editing 3D scenes from images can be categorized as 1) appearance editing and 2) shape editing. A majority of works (Sengupta et al., 2019; Boss et al., 2021; Zhang et al., 2021c; Chen et al., 2021) focus on lighting decomposition for appearance editing. Although achieving appealing results, they cannot separately manipulate individual objects. A number of recent works (Munkberg et al., 2021; Liu et al., 2021a; Jang & Agapito, 2021; Stelzner et al., 2021; Guandao Yang et al., 2021) start to learn disentangled shape representations for potential geometry manipulation. However, they can only deal with single objects or simple scenes, without being able to learn unique object codes for precise shape manipulation and novel view rendering. In addition, there are also a plethora of works (Tewari et al., 2020; Niemeyer & Geiger, 2021; Dhamo et al., 2021; Alaluf et al., 2022) on generation based scene editing. Although they can manipulate the synthesized objects and scenes, they cannot discover and edit objects from real-world images.

3 DM-NERF

Given a set of L images for a static scene with known camera poses and intrinsics $\{(\mathcal{I}_1, \boldsymbol{\xi}_1, \boldsymbol{K}_1) \cdots (\mathcal{I}_L, \boldsymbol{\xi}_L, \boldsymbol{K}_L)\}$, NeRF uses simple MLPs to learn the continuous 3D scene geometry and appearance. In particular, it takes 5D vectors of query point coordinates $\boldsymbol{p} = (x, y, z)$ and viewing directions $\boldsymbol{v} = (\theta, \phi)$ as input, and predicts the volume density σ and color $\boldsymbol{c} = (r, g, b)$ for point $\boldsymbol{p}$. In our pipeline, we leverage this vanilla NeRF as the backbone to learn continuous scene representations, although other NeRF variants can also be used. Our method aims to decompose all individual 3D objects, and freely manipulate any object in the 3D scene space. To achieve this, we design an object field component to parallelly learn an object code for every query point $\boldsymbol{p}$, together with an object manipulator to edit the learned radiance fields and object fields.

3.1 OBJECT FIELDS

Object Field Representation: As shown in Figure 2, given the input point $\boldsymbol{p}$, we model the object field as a function of its coordinates, because the object signature of a 3D point is irrelevant to the

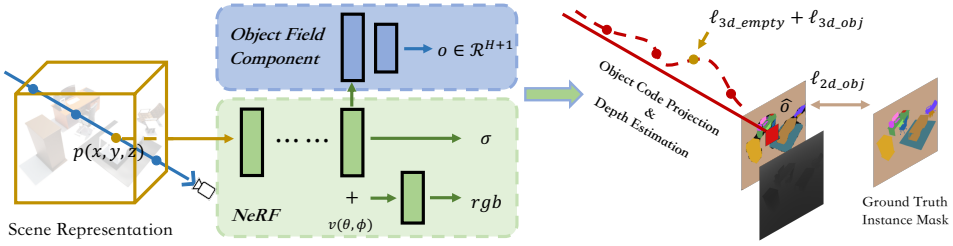


Figure 2: The architecture of our pipeline. Given a 3D point $\boldsymbol{p}$, we learn an object code through three loss functions: $\ell_{2d_obj}/\ell_{3d_empty}/\ell_{3d_obj}$ in Eqs 5&8&9, using 2D and 3D supervision signals.

viewing angles. The object field is represented by a one-hot vector $\boldsymbol{o}$. Basically, this one-hot object code aims to accurately describe the object ownership of any point in 3D space.

However, there are two issues here: 1) the total number of objects in 3D scenes is variable and it can be 1 or many; 2) the entire 3D space has a large non-occupied volume in addition to solid objects. To tackle these issues, we define the object code $\boldsymbol{o}$ as $H+1$ dimensional, where H is a predefined number of solid objects that the network is expected to predict in maximum. We can safely choose a relative large value for H in practice. The last dimension of $\boldsymbol{o}$ is particularly reserved to represent the non-occupied space. Notably, this dedicated design is crucial for tackling occlusion and collision during object manipulation discussed in Section 3.2. Formally, the object field is defined as:

$$\boldsymbol{o} = f(\boldsymbol{p}), \quad \text{where } \boldsymbol{o} \in \mathcal{R}^{H+1} \tag{1}$$

The function f is parameterized by a series of MLPs. If the last dimension of code $\boldsymbol{o}$ is 1, it represents the input point $\boldsymbol{p}$ is non-occupied or the point is empty.

Object Code Projection: Considering that it is infeasible to collect object code labels in continuous 3D space for full supervision while it is fairly easy and low-cost to collect object labels on 2D images, we aim to project the object codes along the query light ray back to a 2D pixel. Since the volume density σ learned by the backbone NeRF represents the geometry distribution, we simply approximate the projected object code of a pixel $\hat{\boldsymbol{o}}$ using the sampling strategy and volume rendering formulation of NeRF. Formally, it is defined as:

$$\hat{\boldsymbol{o}} = \sum_{k=1}^{K} T_k \alpha_k \boldsymbol{o}_k, \quad \text{where} \quad T_k = exp(-\sum_{i=1}^{k-1} \sigma_i \delta_i), \quad \alpha_k = 1 - exp(-\sigma_k \delta_k) \tag{2}$$

with K representing the total sample points along the light ray shooting from a pixel, σ_i representing the learned density of the ith sample point, δ_k representing the distance between the $(k+1)^{th}$ and k^{th} sample points. From this projection formulation, we can easily obtain 2D masks of 3D object codes given the pose and camera parameters of any query viewing angles.

Object Code Supervision: As shown in the right part of Figure 2, having projected 2D object predictions at hand, we choose 2D images with object annotations for supervision. However, there are two issues: 1) The number and order of ground truth objects can be very different across different views due to visual occlusions. For example, as to the same 3D object in space, its object annotation in image #1 can be quite different from its annotation in image #2. Therefore, it is non-trivial to consistently utilize 2D annotations for supervision. 2) The 2D annotations only provide labels for 3D solid objects, as non-occupied 3D space is never recorded in 2D images. Therefore, it is impossible to directly supervise non-occupied space, *i.e.*, the last dimension of $\hat{\boldsymbol{o}}$, from 2D annotations. These issues make adaptions of existing 2D methods ineffective, *e.g.*, Mask-RCNN (He et al., 2017) or Swin Transformer (Liu et al., 2021b), fundamentally because they do not consider the consistency between 3D and 2D, but just segment objects on individual images.

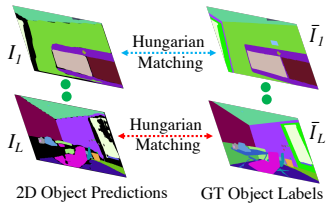


Figure 3: Illustration of 2D object matching and supervision for ℓ_{2d_obj}.

To tackle the first issue, we use the Optimal Association and Supervision strategy proposed by 3D-BoNet (Yang et al., 2019b). As illustrated in Figure 3, assuming we generate L images of 2D object predictions $\{I_1 \dots I_l \dots I_L\}, I_l \in \mathcal{R}^{U \times V \times (H+1)}$ and have the corresponding L images of 2D ground truth object labels $\{\bar{I}_1 \dots \bar{I}_l \dots \bar{I}_L\}, \bar{I}_l \in \mathcal{R}^{U \times V \times T}$, in which H is the predefined number of objects and T represents the number of ground truth objects. Note that the ground truth number of objects in each image is usually different, but here we use the same T to avoid an abuse of notation.

For each pair, we firstly take the first H dimensions (solid object predictions) of I and reshape them to be $M \in \mathcal{R}^{N \times H}$, where $N = U \times V$, while the last dimension is never used at this stage. Likewise, $\bar{I}$ is reshaped to be $\bar{M} \in \mathcal{R}^{N \times T}$. Then, M and $\bar{M}$ are fed into Hungarian algorithm (Kuhn, 1955) to associate every ground truth 2D object mask with a unique predicted 2D object mask, according to Soft Intersection-over-Union (sIoU) and Cross-Entropy Score (CES) (Yang et al., 2019b). Formally, the Soft Intersection-over-Union (sIoU) cost between the hth predicted object mask and the tth ground truth object mask in the lth pair is defined as follows:

$$\boldsymbol{C}_{h,t}^{sIoU} = \frac{-\sum_{n=1}^{N}(M_h^n * \bar{M}_t^n)}{\sum_{n=1}^{N} M_h^n + \sum_{n=1}^{N} \bar{M}_t^n - \sum_{n=1}^{N}(M_h^n * \bar{M}_t^n)} \tag{3}$$

where M_h^n and $\bar{M}_t^n$ are the nth values of M_i and $\bar{M}_t$. The Cross-Entropy Score (CES) between M_h and $\bar{M}_t$ is formally defined as:

$$\boldsymbol{C}_{h,t}^{CES} = -\frac{1}{N}\sum_{n=1}^{N}\left[\bar{M}_t^n \log M_h^n + (1-\bar{M}_t^n)\log(1-M_h^n)\right] \tag{4}$$

After the optimal association based on $(\boldsymbol{C}_{h,t}^{sIoU} + \boldsymbol{C}_{h,t}^{CES})$, we reorder the predicted object masks to align with the T ground truth masks, and then we directly minimize the cost of all ground truth objects in every pair. The final loss ℓ_{2d_obj} is defined by averaging across all L image pairs.

$$\ell_{2d_obj} = \sum_{l=1}^{L}(sIoU_l + CES_l), \quad \text{where} \quad sIoU_l = \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{C}_{t,t}^{sIoU}),\ CES_l = \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{C}_{t,t}^{CES}) \tag{5}$$

To tackle the second issue, we turn to supervise the non-occupied object code in 3D space with the aid of estimated surface distances. In particular, given a specific query light ray on which we sample K 3D points to compute the projected 2D object code $\hat{\boldsymbol{o}}$, we simultaneously compute an approximate distance d between camera center and the surface hit point along that query light:

$$d = \sum_{k=1}^{K} T_k \alpha_k \delta_k, \quad \text{where} \quad T_k = exp(-\sum_{i=1}^{k-1}\sigma_i\delta_i), \quad \alpha_k = 1 - exp(-\sigma_k\delta_k) \tag{6}$$

where K represents the total sample points along light ray, σ_i is the learned density, and δ_k is the distance between $(k+1)^{th}$ and k^{th} sample points, as same as Equation 2.

As shown in Figure 4, once we have the surface distance d at hand, we can easily know the relative position between every k^{th} sample point and the surface point s along the light ray. Naturally, we can then identify the subset of sample points surely belonging to empty space as indicated by green points, the subset of sample points near the surface as indicated by red points, and the remaining subset of sample points behind the surface as indicated by black points. Such geometric information provides critical signals to supervise empty space, *i.e.*, the last dimension of object code $\boldsymbol{o}$. Note that, the sample points behind the surface may not surely belong to empty space, and therefore cannot be used as supervision signals. We use the following kernel functions to obtain a surfaceness score s_k and an emptiness score e_k for the kth sample point with the indicator function represented by $\mathbb{1}()$.

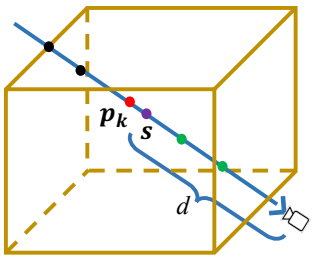


Figure 4: Empty points identification.

$$s_k = exp\big(-(d_k - d)^2\big), \quad e_k = (1 - s_k) * \mathbb{1}(d - \Delta d - d_k > 0) \tag{7}$$

where d_k represents the distance between camera center and the kth sample point, and Δd is a hyperparameter to compensate the inaccuracy of estimated surface distance d. Note that, the indicator function is defined as: $\mathbb{1}() = 1$ if $d - \Delta d - d_k > 0$, and otherwise $\mathbb{1}() = 0$. It is used to mask out the sample points behind surface point during loss calculation. Given the emptiness and surfaceness scores for the total K sample points along the ray, we use the simple log loss to supervise the last dimension of object code denoted as o_k^{H+1}.

$$\ell_{3d_empty} = -\frac{1}{K}\sum_{k=1}^{K}\Big(e_k * log(o_k^{H+1}) + s_k * log(1 - o_k^{H+1})\Big) \tag{8}$$

Since there should be no solid object at all in the empty space, we apply the following ℓ_{3d_obj} loss on the first H dimensions of the object code to push them to be zeros, being complementary to the existing ℓ_{2d_obj} loss. The hth dimension of the kth sample point's object code is denoted by o_k^h.

$$\ell_{3d_obj} = -\frac{1}{K}\sum_{k=1}^{K}\Big(e_k * \sum_{h=1}^{H} log(1-o_k^h)\Big) \tag{9}$$

To sum up, we firstly project object codes in 3D space back to 2D pixels along light rays using volume rendering equation, and then use optimal association strategy to compute the loss value with 2D object labels only. In addition, we introduce the key emptiness and surfaceness scores for points in 3D space with the aid of estimated surface distances. These unique scores are used to supervise the non-occupied 3D space. The whole object field is jointly supervised by:

$$\ell = \ell_{2d_obj} + \ell_{3d_empty} + \ell_{3d_obj} \tag{10}$$

3.2 Object Manipulator

If we want to manipulate a specific 3D object shape (*e.g.,* translation, rotation and size adjustment), how does the whole scene look like in a new perspective after manipulation, assuming the object code and manipulation matrix are precomputed from users' interactions (*e.g.,* click, drag or zoom)?

Intuitively, there could be two strategies: 1) firstly project 3D scene into 2D images, and then edit objects in 2D space. 2) firstly edit objects in 3D space, and then project into 2D images. Compared with the first strategy which would inevitably incur inconsistency across multiple images due to the independent edits on individual views, the latter is more favourable. Remarkably, our object field component can support the latter strategy. Regarding such a manipulation task on implicit fields, the core question is: how do we edit the codes $\sigma/\boldsymbol{c}/\boldsymbol{o}$ of every sample point along the query light ray, such that the generated novel view exactly shows the new appearance? This is nontrivial as:

- First, we need to address potential collisions between objects during manipulation. This is quite intuitive, thanks to our special design of emptiness score in the last dimension of object code $\boldsymbol{o}$.
- Second, due to visual occlusions, object codes behind surface points may not be accurate as they are not sufficiently optimized. By comparison, the projected object code $\hat{\boldsymbol{o}}$ along a light ray tends to be more accurate primarily because we have ground truth 2D labels for strong supervision.
- At last, we need a systematic procedure to update the codes with the known manipulation information. To this end, we design an inverse query approach.

Inverse Query: We design a creative inverse query approach to address all above issues, realizing the favourable strategy: *editing in 3D space followed by 2D projection*. In particular, as shown in Figure 5, for any 3D sample point $\boldsymbol{p}_k$ along a specific query light ray, given the *target* (*i.e.*, to-be-edited) object code $\boldsymbol{o}_t$ and its manipulation settings: relative translation $\Delta\boldsymbol{p}$=$(\Delta x, \Delta y, \Delta z)$, rotation matrix $\mathbf{R}^{3\times3}$, and scaling factor t>0, we firstly compute an inverse 3D point $\boldsymbol{p}_{k'}$, and then evaluate whether $\boldsymbol{p}_k$ and $\boldsymbol{p}_{k'}$ belong to the target object, and lastly decide to whether edit the codes or not. Formally, we introduce Inverse Query Algorithm 1 to conduct a single light ray editing and rendering for object shape manipulation. Naturally, we can shoot a bunch of rays from any novel viewing angles to generate images of manipulated 3D scenes.

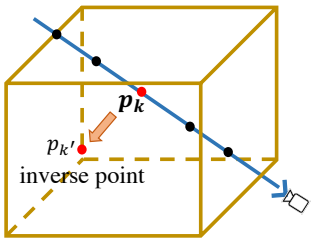


Figure 5: Inverse points computation.

3.3 Implementation

To preserve the high-quality of image rendering, our loss in Equation 10 is only used to optimize the MLPs of object field branch. The backbone is only optimized by the original NeRF photo-metric loss (Mildenhall et al., 2020). The whole network is end-to-end trained from scratch. The single hyper-parameter for our object field Δd is set as 0.05 meters in all experiments.

4 Experiments

4.1 Datasets

DM-SR: To the best of our knowledge, there is no 3D scene dataset suitable for quantitative evaluation of geometry manipulation. Therefore, we create a **s**ynthetic dataset with 8 different and complex indoor **r**ooms, called DM-SR. For each scene, we generate the following 5 groups of images:

- Group 1 (w/o Manipulation): Color images and 2D object masks at 400×400 pixels are rendered from viewpoints on the upper hemisphere. We generate 300 views for training.

Algorithm 1 Our Inverse Query Algorithm to manipulate the learned implicit fields. (1) $\boldsymbol{o}_t$ is the target object code, one-hot with $H+1$ dimensions. $\{\Delta\boldsymbol{p}, \mathbf{R}^{3\times3}, t>0\}$ represent the manipulation information for the target object. (2) $\{\boldsymbol{p}_1 \cdots \boldsymbol{p}_k \cdots \boldsymbol{p}_K\}$ represent the K sample points along a specific query light ray $\boldsymbol{r}$. Note that, we convert all object codes into hard one-hot vectors for easy implementation.

Input:
- The target object code $\boldsymbol{o}_t$, manipulation information $\{\Delta\boldsymbol{p}, \mathbf{R}^{3\times3}, t\geq 0\}$;
- The sample points $\{\boldsymbol{p}_1 \cdots \boldsymbol{p}_k \cdots \boldsymbol{p}_K\}$ along a specific query light ray $\boldsymbol{r}$;

Output:
- The final pixel color $\bar{\boldsymbol{c}}$ rendered from the query ray $\boldsymbol{r}$ after manipulation;
- The final pixel object code $\bar{\boldsymbol{o}}$ rendered from the query ray $\boldsymbol{r}$ after manipulation;

Preliminary step:
- Obtain the projected pixel object code $\hat{\boldsymbol{o}}$ of light ray $\boldsymbol{r}$ before manipulation;

NOTE: The loop below shows how to edit per sample point in 3D space.

for $\boldsymbol{p}_k$ in $\{\boldsymbol{p}_1 \cdots \boldsymbol{p}_k \cdots \boldsymbol{p}_K\}$ **do**
- Compute the inverse point $\boldsymbol{p}_{k'}$ for $\boldsymbol{p}_k$: $\boldsymbol{p}_{k'} = (1/t)\mathbf{R}^{-1}(\boldsymbol{p}_k - \Delta\boldsymbol{p})$;
- Obtain the codes $\{\sigma_k, \boldsymbol{c}_k, \boldsymbol{o}_k\}$ for the point $\boldsymbol{p}_k$;
- Obtain the codes $\{\sigma_{k'}, \boldsymbol{c}_{k'}, \boldsymbol{o}_{k'}\}$ for the inverse point $\boldsymbol{p}_{k'}$;
- *Tackle visual occlusions:*
  if $\boldsymbol{o}_k = \boldsymbol{o}_t$ and $\boldsymbol{o}_k \neq \hat{\boldsymbol{o}}$ do: $\boldsymbol{o}_k \leftarrow \hat{\boldsymbol{o}}$
  Note: the target object is behind the surface but will be manipulated;
- Obtain new implicit codes $\{\bar{\sigma}_k, \bar{\boldsymbol{c}}_k, \bar{\boldsymbol{o}}_k\}$ for $\boldsymbol{p}_k$ after manipulation:
  if $\boldsymbol{o}_k \neq \boldsymbol{o}_t$ and $o_k^{H+1} \neq 1$ and $\boldsymbol{o}_{k'} = \boldsymbol{o}_t$ do: collision detected, EXIT.
  if $\boldsymbol{o}_k \neq \boldsymbol{o}_t$ and $\boldsymbol{o}_{k'} = \boldsymbol{o}_t$ do: $\{\bar{\sigma}_k, \bar{\boldsymbol{c}}_k, \bar{\boldsymbol{o}}_k\} \leftarrow \{\sigma_{k'}, \boldsymbol{c}_{k'}, \boldsymbol{o}_{k'}\}$;
  if $\boldsymbol{o}_k = \boldsymbol{o}_t$ and $\boldsymbol{o}_{k'} = \boldsymbol{o}_t$ do: $\{\bar{\sigma}_k, \bar{\boldsymbol{c}}_k, \bar{\boldsymbol{o}}_k\} \leftarrow \{\sigma_{k'}, \boldsymbol{c}_{k'}, \boldsymbol{o}_{k'}\}$;
  if $\boldsymbol{o}_k = \boldsymbol{o}_t$ and $\boldsymbol{o}_{k'} \neq \boldsymbol{o}_t$ do: $\{\bar{\sigma}_k, \bar{\boldsymbol{c}}_k, \bar{\boldsymbol{o}}_k\} \leftarrow \{0, \mathbf{0}, \mathbf{0}\}$;
  if $\boldsymbol{o}_k \neq \boldsymbol{o}_t$ and $\boldsymbol{o}_{k'} \neq \boldsymbol{o}_t$ do: $\{\bar{\sigma}_k, \bar{\boldsymbol{c}}_k, \bar{\boldsymbol{o}}_k\} \leftarrow \{\sigma_k, \boldsymbol{c}_k, \boldsymbol{o}_k\}$;

After the above *for loop*, every point $\boldsymbol{p}_k$ will get new implicit codes $\{\bar{\sigma}_k, \bar{\boldsymbol{c}}_k, \bar{\boldsymbol{o}}_k\}$.

NOTE: The step below shows how to project edited 3D points to a 2D image.

According to volume rendering equation, the final pixel color and object code are:
- $\bar{\boldsymbol{c}} = \sum_{k=1}^{K} \bar{T}_k \bar{\alpha}_k \bar{\boldsymbol{c}}_k, \quad \bar{\boldsymbol{o}} = \sum_{k=1}^{K} \bar{T}_k \bar{\alpha}_k \bar{\boldsymbol{o}}_k$
  where $\bar{T}_k = exp(-\sum_{i=1}^{k-1} \bar{\sigma}_i \delta_i), \quad \bar{\alpha}_k = 1 - exp(-\bar{\sigma}_k \delta_k)$.

- Group 2 (Translation Only): One object is selected to be translated along x or y axis with $\sim 0.3m$.
- Group 3 (Rotation Only): One object is selected to be rotated around z axis with about 90 degrees.
- Group 4 (Scaling Only): One object is selected to be scaled down about $0.8\times$ smaller.
- Group 5 (Joint Translation/Rotation/Scaling): One object is selected to be simultaneously translated about $\sim 0.3m$, rotated about 90 degrees, scaled down about $0.8\times$ smaller.

For each group, 100 views are generated for testing at the same viewpoints.

Replica: Replica (Straub et al., 2019) is a reconstruction-based 3D dataset of high fidelity scenes. We request the authors of Semantic-NeRF (Zhi et al., 2021a) to generate (180 training / 180 testing) color images and 2D object masks with camera poses at 640×480 pixels for each of 7 scenes. Each scene has 59∼93 objects with very diverse sizes.

ScanNet: ScanNet (Dai et al., 2017) is a large-scale challenging real-world dataset. We select 8 scenes (∼10 objects in each one) for evaluation. Each scene has ∼3000 raw images with 2D object masks and camera poses, among which we evenly select 300 views for training and 100 for testing.

4.2 Baseline and Metrics

Scene Decomposition. The most relevant work to us is ObjectNeRF (Yang et al., 2021), but its design is vastly different: 1) It requires a point cloud as input for voxelization in training, but we do not. 2) It needs GT bounding boxes of target objects to manually prune point samples during editing, but we do not need any annotations in editing. 3) It only learns to binarily segment the foreground object and background, by pre-defining an Object Library in training and editing. However, our object code is completely learned from scratch. This means that ObjectNeRF is not comparable due to the fundamental differences. Note that, the recent Semantic-NeRF (Zhi et al., 2021a) is also not comparable because it only learns 3D semantic categories, not individual 3D objects. In fact, we notice that all recent published relevant works (Yang et al., 2021; Zhi et al., 2021a; Wu et al., 2022; Kundu et al., 2022; Norman et al., 2022; Fu et al., 2022) do not directly tackle and evaluate multiple 3D object segmentation in literature. In this regard, we use the powerful Mask-RCNN (He et al., 2017) and Swin Transformer (Liu et al., 2021b) as baselines.

Since we train our DM-NeRF in scene-specific fashion, for fairness, we also fine-tune Mask-RCNN and Swin Transformer models (pretrained under Detectron2 Library) on every single scene. In particular, we carefully fine-tune both models using up to 480 epochs until convergence with learning rate $5e^{-4}$ and then pick up the best models on the testing split of each scene for comparison.

Object Manipulation. Since there is no method that can directly manipulate objects in continuous radiance fields, we adapt the recent Point-NeRF (Xu et al., 2022) for comparison, because it can recover both explicit 3D scene point clouds and novel views. Adaptation details are in Appendix.

Metrics: We use the standard PSNR/SSIM/LPIPS scores to evaluate color image synthesis (Mildenhall et al., 2020), and use AP of all 2D test images to evaluate 3D scene decomposition. Note that object classification and panoptic segmentation are irrelevant in this paper.

4.3 3D Scene Decomposition

Training with 100% Accurate 2D Labels: We evaluate the performance of scene decomposition on 3 datasets. For DM-SR dataset, we evaluate our method and baselines on images of Group 1 only, while the images of Groups 2/3/4/5 are used for object manipulation. For every single scene in these datasets, using 100% accurate 2D object ground truth labels, we train a separate model for our method, and fine-tune separate models for Mask-RCNN (MR) and Swin Transformer (SwinT).

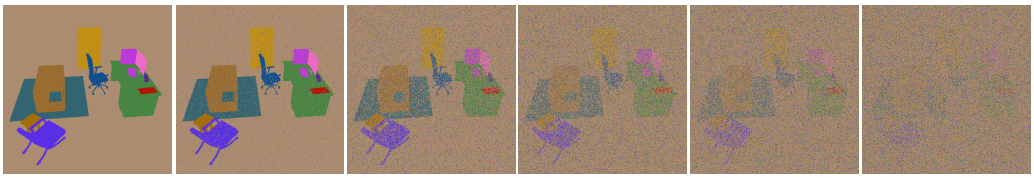
Analysis: Table 1 shows that our method, not surprisingly, achieves excellent results for novel view rendering thanks to the original NeRF backbone. Notably, our method obtains nearly perfect object segmentation results across multiple viewing points of complex 3D scenes in all three datasets, clearly outperforming baselines. Figure 7 shows that our results have much sharper object boundaries thanks to the explicit 3D geometry applied in our object field.

	DM-SR Dataset							Replica Dataset						ScanNet Dataset					
	Novel View Synthesis			Decomposition				Novel View Synthesis			Decomposition			Novel View Synthesis			Decomposition		
	PSNR↑	SSIM↑	LPIPS↓	MR	SwinT	**Ours**		PSNR↑	SSIM↑	LPIPS↓	MR	**Ours**		PSNR↑	SSIM↑	LPIPS↓	MR	**Ours**
Bathroom	44.05	0.994	0.009	93.81	98.89	100.0							0010_00	26.82	0.809	0.381	83.90	94.82
Bedroom	48.07	0.996	0.009	97.92	98.85	100.0	Office_0	40.66	0.972	0.070	74.05	82.71	0012_00	29.28	0.753	0.389	86.90	98.86
Dinning	42.34	0.984	0.028	98.85	97.81	99.66	Office_2	36.98	0.964	0.115	73.41	81.12	0024_00	23.68	0.705	0.452	69.87	93.25
Kitchen	46.06	0.994	0.014	92.04	98.81	100.0	Office_3	35.34	0.955	0.078	72.91	76.30	0033_00	27.76	0.856	0.342	88.70	97.02
Reception	42.59	0.993	0.008	98.81	95.75	100.0	Office_4	32.95	0.921	0.172	74.76	70.33	0038_00	29.36	0.716	0.415	96.01	99.17
Rest	42.80	0.994	0.007	98.89	94.50	99.89	Room_0	34.97	0.940	0.127	78.67	79.83	0088_00	29.37	0.825	0.386	69.06	83.59
Study	41.08	0.987	0.026	96.86	97.88	98.86	Room_1	34.72	0.931	0.134	78.38	92.11	0113_00	31.19	0.878	0.320	98.59	98.67
Office	46.38	0.996	0.006	97.83	96.87	100.0	Room_2	37.32	0.963	0.115	77.58	84.78	0192_00	28.19	0.732	0.376	96.95	99.40
Average	44.17	0.992	0.013	96.87	97.42	**99.80**	Average	36.13	0.949	0.116	75.68	**81.03**	Average	28.21	0.784	0.383	86.25	**95.60**

Table 1: Quantitative results on three datasets. The metric for object decomposition is $AP^{0.75}$.

Robustness to Noisy 2D Labels: Since our method inherently has multi-view consistency while the 2D segmentation methods do not, it is expected that our method has better robustness to inaccurate and noisy 2D labels in training. To validate this advantage, we conduct the following experiments on DM-SR dataset. Particularly, as illustrated in Figure 6, we randomly assign incorrect object labels to different amounts of image pixels of all training images (10%/50%/70%/80%/90%), and then our method and baselines are all trained with these noisy 2D labels.

Analysis: As shown in Table 2, it can be seen that our method still achieves an excellent object segmentation score ($AP^{0.75}$ = 74.08) on testing/novel views, even though 80% of 2D labels are incorrect in training. By contrast, both baselines fail catastrophically once more than 50% of labels are noisy in training. Basically, this is because our dedicated losses $\ell_{2d_obj}/\ell_{3d_empty}/\ell_{3d_obj}$ explicitly take into account object geometry consistency across multi-views, thus allowing the estimated 3D object codes to be resistant against wrong labels, whereas the existing 2D object segmentation methods only independently process single images, being easily misled by wrong labels in training. This clearly demonstrates the remarkable robustness of our method.



0% Noisy Label | 10% Noisy Labels | 50% Noisy Labels | 70% Noisy Labels | 80% Noisy Labels | 90% Noisy Labels

Figure 6: Examples of 2D object labels with different levels of noise.

	10% Noisy Labels			50% Noisy Labels			70% Noisy Labels			80% Noisy Labels			90% Noisy Labels		
	MR	SwinT	**Ours**	MR	SwinT	**Ours**	MR	SwinT	**Ours**	MR	SwinT	**Ours**	MR	SwinT	**Ours**
Bathroom	98.93	98.96	99.81	54.52	62.94	99.63	1.86	7.71	99.02	1.89	7.78	58.09	1.79	7.11	9.69
Bedroom	98.61	98.85	100.0	75.41	92.94	100.0	3.00	3.96	100.0	2.96	3.18	82.83	2.95	2.75	4.25
Dinning	95.57	97.88	98.41	46.52	43.50	85.48	1.40	1.05	81.91	1.51	1.09	63.50	1.36	0.96	14.44
Kitchen	98.82	98.81	100.0	84.93	93.60	100.0	4.09	8.42	100.0	4.02	5.04	51.87	4.04	4.69	1.80
Reception	80.96	91.00	100.0	32.78	42.11	100.0	1.03	6.27	100.0	0.73	1.96	100.0	0.22	1.64	37.63
Rest	93.58	94.50	99.64	52.75	51.99	99.32	1.63	1.57	99.33	2.46	2.02	66.74	1.63	1.69	11.11
Study	93.07	97.94	98.58	49.03	60.04	97.97	1.44	6.24	98.03	1.50	6.06	72.62	1.40	5.48	40.72
Office	95.09	97.00	100.0	66.28	69.11	100.0	4.13	2.96	100.0	2.61	3.94	97.00	2.66	2.48	0.00
Average	94.33	96.87	**99.56**	57.78	64.53	**97.80**	2.32	4.77	**97.29**	2.21	3.88	**74.08**	2.00	3.35	**14.96**

Table 2: Quantitative object decomposition $AP^{0.75}$ scores on noisy DM-SR dataset.

	Translation				Rotation			
	PSNR↑	SSIM↑	LPIPS↓	$AP^{0.9}$ ↑	PSNR↑	SSIM↑	LPIPS↓	$AP^{0.9}$ ↑
Point-NeRF (on Study)	29.85	0.933	0.128	-	24.51	0.918	0.140	-
Ours (on Study)	32.71	0.975	0.046	94.04	27.94	0.961	0.055	90.10
Point-NeRF	-	-	-	-	-	-	-	-
Ours	**33.94**	**0.975**	**0.033**	**89.33**	**31.94**	**0.969**	**0.038**	**85.68**
Ablation 1: Ours (w/o ℓ_{3d})	32.84	0.967	0.048	87.26	30.38	0.945	0.090	82.46
Ablation 2: Ours (w/o VO)	33.54	0.970	0.045	86.93	30.57	0.953	0.076	82.43
	Scale				Joint			
	PSNR↑	SSIM↑	LPIPS↓	$AP^{0.9}$ ↑	PSNR↑	SSIM↑	LPIPS↓	$AP^{0.9}$ ↑
Point-NeRF (on Study)	26.12	0.929	0.121	-	21.46	0.908	0.142	-
Ours (on Study)	30.23	0.959	0.054	92.40	27.26	0.955	0.057	86.92
Point-NeRF	-	-	-	-	-	-	-	-
Ours	**33.40**	**0.971**	**0.037**	**86.05**	**30.65**	**0.965**	**0.045**	**81.70**
Ablation 1: Ours (w/o ℓ_{3d})	31.84	0.959	0.062	83.31	29.95	0.947	0.088	77.36
Ablation 2: Ours (w/o VO)	32.43	0.964	0.054	76.33	29.85	0.951	0.075	74.71

Table 3: Quantitative results of object manipulation and ablation studies on DM-SR dataset.

4.4 3D Object Manipulation

In this section, we directly use our model trained on the images of Group 1 to test on the remaining images of Groups 2/3/4/5 in DM-SR dataset. In particular, with the trained model, we feed the known manipulation information of Groups 2/3/4/5 into Algorithm 1, generating images and 2D object masks. These (edited) images and masks are compared with the ground truth 2D views. For comparison, we carefully train Point-NeRF models in scene-specific fashion and apply the same manipulation information to its learned point clouds followed by novel view rendering.

Analysis: Table 3 shows that the quality of our novel view rendering is clearly better than Point-NeRF (Xu et al., 2022), although it decreases after manipulation compared with non-manipulation in Table 1, primarily because the lighting factors are not decomposed and the illumination of edited objects shows discrepancies. However, the object decomposition is still nearly perfect, as also shown for deformation manipulation in Figure 8. More results are in Appendix.

4.5 Ablation Study

To evaluate the effectiveness of our key designs of object field and manipulator, we conduct two ablation studies. 1) We only remove the loss functions $(\ell_{3d_empty} + \ell_{3d_obj})$ which are jointly designed to learn correct codes for empty 3D points, denoted as w/o ℓ_{3d}. 2) During manipulation, we remove the step "Tackle visual occlusions" in Algorithm 1, denoted as w/o VO. As shown in Table 3, both of our loss functions for empty space regularization and visual occlusion handling step are crucial for accurate 3D scene decomposition and manipulation. More ablation results are in Appendix.

5 Discussion and Conclusion

We have shown that it is feasible to simultaneously reconstruct, decompose, manipulate and render complex 3D scenes in a single pipeline only from 2D views. By adding an object field component into the implicit representation, we successfully decompose all individual objects in 3D space. The decomposed object shapes can be further freely edited by our visual occlusion aware manipulator.

One limitation is the lack of decomposing lighting factors, which is non-trivial and left for future work. In addition, manipulation of originally occluded objects or parts may produce artifacts due to the inaccuracy of learned radiance fields, although these artifacts can be easily repaired by applying simple heuristics such as continuity of surfaces or using better neural backbones.

REFERENCES

Yuval Alaluf, Ron Mokady, and Amit H Bermano. HyperStyle: StyleGAN Inversion with HyperNetworks for Real Image Editing. *CVPR*, 2022.

Pierre Alliez, Mark Meyer, and Mathieu Desbrun. Interactive geometry remeshing. *ACM TOG*, 2002.

Sagie Benaim, Frederik Warburg, Peter Ebert, and Christensen Serge. Volumetric Disentanglement for 3D Scene Manipulation. *arXiv:2206.02776*, 2022.

Mark Boss, Varun Jampani, Raphael Braun, Ce Liu, Jonathan T. Barron, and Hendrik. Neural-PIL: Neural Pre-Integrated Lighting for Reflectance Decomposition. *NeurIPS*, 2021.

Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian D. Reid, and John J. Leonard. Past, Present, and Future of Simultaneous Localization and Mapping: Towards the Robust-Perception Age. *IEEE Transactions on Robotics*, 2016.

Wenzheng Chen, Joey Litalien, Jun Gao, Zian Wang, Clement Fuji Tsang, Sameh Khamis, Or Litany, and Sanja Fidler. DIB-R++: Learning to Predict Lighting and Material with a Hybrid Differentiable Renderer. *NeurIPS*, 2021.

Julian Chibane, Aymen Mir, and Gerard Pons-Moll. Neural Unsigned Distance Fields for Implicit Function Learning. *NeurIPS*, 2020.

Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction. *ECCV*, 2016.

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. *CVPR*, 2017.

Helisa Dhamo, Fabian Manhardt, Nassir Navab, and Federico Tombari. Graph-to-3D: End-to-End Generation and Manipulation of 3D Scenes Using Scene Graphs. *ICCV*, 2021.

Haoqiang Fan, Hao Su, and Leonidas Guibas. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. *CVPR*, 2017.

Xiao Fu, Shangzhan Zhang, Tianrun Chen, Yichong Lu, Lanyun Zhu, Xiaowei Zhou, Andreas Geiger, and Yiyi Liao. Panoptic NeRF: 3D-to-2D Label Transfer for Panoptic Urban Scene Segmentation. *3DV*, 2022.

Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh R-CNN. *ICCV*, 2019.

Thibault Groueix, Matthew Fisher, V G Kim, B C Russell, and M Aubry. A Papier-Mache Approach to Learning 3D Surface Generation. *CVPR*, 2018.

Guandao Yang, Serge Belongie, Bharath Hariharan, and Vladlen Koltun. Geometry Processing with Neural Fields. *NeurIPS*, 2021.

Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. *IEEE International Conference on Computer Vision*, pp. 2961–2969, 2017.

Wonbong Jang and Lourdes Agapito. CodeNeRF: Disentangled Neural Radiance Fields for Object Categories. *ICCV*, 2021.

Kacper Kania, Kwang Moo Yi, Marek Kowalski, Tomasz Trzciński, and Andrea Tagliasacchi. CoNeRF: Controllable Neural Radiance Fields. *CVPR*, 2022.

Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D Mesh Renderer. *CVPR*, 2018.

Mira Kim, Jaehoon Ko, Kyusun Cho, Junmyeong Choi, and Daewon Choi. AE-NeRF: Auto-Encoding Neural Radiance Fields for 3D-Aware Object Manipulation. *arXiv:2204.13426*, 2022.

Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing NeRF for Editing via Feature Field Distillation. *NeurIPS*, 2022.

Harold W. Kuhn. The Hungarian Method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic Neural Fields: A Semantic Object-Aware Neural Scene Representation. *CVPR*, 2022.

Chen-Hsuan Lin, Chen Kong, and Simon Lucey. Learning Efficient Point Cloud Generation for Dense 3D Object Reconstruction. *AAAI*, 2018.

Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing Conditional Radiance Fields. *ICCV*, 2021a.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *ICCV*, 2021b.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. *CVPR*, 2019.

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. *ECCV*, 2020.

Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3D Bounding Box Estimation Using Deep Learning and Geometry. *CVPR*, 2017.

Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting Triangular 3D Models, Materials, and Lighting From Images. *arXiv:2111.12503*, 2021.

Michael Niemeyer and Andreas Geiger. GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields. *CVPR*, pp. 11453–11464, 2021.

Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision. *CVPR*, 2020.

M Norman, Andrea Simonelli, Lorenzo Porzi, Samuel Rota, Matthias Nießner, and Peter Kontschieder. AutoRF: Learning 3D Object Radiance Fields from Single View Observations. *CVPR*, 2022.

Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural Scene Graphs for Dynamic Scenes. *CVPR*, 2021.

Onur Ozyesil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A Survey of Structure from Motion. *Acta Numerica*, 26:305–364, 2017.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. *CVPR*, 2019.

Zhongzheng Ren, Aseem Agarwala, Bryan Russell, and Alexander G Schwing. Neural Volumetric Object Selection. *CVPR*, 2022.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. *ICCV*, 2019.

Soumyadip Sengupta, Jinwei Gu, Kihwan Kim, Guilin Liu, David W. Jacobs, and Jan Kautz. Neural Inverse Rendering of an Indoor Scene from a Single Image. *ICCV*, 2019.

Shuran Song, Fisher Yu, Andy Zeng, Angel X. Chang, Manolis Savva, and Thomas Funkhouser. Semantic Scene Completion from a Single Depth Image. *CVPR*, 2017.

Karl Stelzner, Kristian Kersting, and Adam R. Kosiorek. Decomposing 3D Scenes into Objects via Unsupervised Volume Segmentation. *arXiv:2104.01148*, 2021.

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, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, and Hauke M Strasdat. The Replica Dataset : A Digital Replica of Indoor Spaces. *arXiv:1906.05797*, 2019.

Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs. *ICCV*, 2017.

Ayush Tewari, Mohamed Elgharib, Gaurav Bharaj, Florian Bernard, Hans-Peter Seidel, Patrick Perez, Michael Zollhofer, and Christian Theobalt. StyleRig: Rigging StyleGAN for 3D Control over Portrait Images. *CVPR*, 2020.

Alex Trevithick and Bo Yang. GRF: Learning a General Radiance Field for 3D Representation and Rendering. *ICCV*, 2021.

Vadim Tschernezki, Iro Laina, Diane Larlus, and Andrea Vedaldi. Neural Feature Fusion Fields: 3D Distillation of Self-Supervised 2D Image Representations. *3DV*, 2022.

S Tulsiani, S Gupta, D Fouhey, A A. Efros, and J Malik. Factoring Shape, Pose, and Layout from the 2D Image of a 3D Scene. *CVPR*, 2018.

Bing Wang, Zhengdi Yu, Bo Yang, Jie Qin, Toby Breckon, Ling Shao, Niki Trigoni, and Andrew Markham. RangeUDF: Semantic Surface Reconstruction from 3D Point Clouds. *arXiv:2204.09138*, 2022.

Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. *NeurIPS*, 2021.

Qianyi Wu, Xian Liu, Yuedong Chen, Kejie Li, Chuanxia Zheng, Jianfei Cai, and Jianmin Zheng. Object-Compositional Neural Implicit Surfaces. *ECCV*, 2022.

Haozhe Xie, Hongxun Yao, Xiaoshuai Sun, Shangchen Zhou, Shengping Zhang, and Xiaojun Tong. Pix2Vox: Context-aware 3D Reconstruction from Single and Multi-view Images. *ICCV*, 2019.

Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-NeRF: Point-based Neural Radiance Fields. *arXiv*, 2022.

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. *ICCV*, 2021.

Bo Yang, Stefano Rosa, Andrew Markham, Niki Trigoni, and Hongkai Wen. Dense 3D Object Reconstruction from a Single Depth View. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019a.

Bo Yang, Jianan Wang, Ronald Clark, Qingyong Hu, Sen Wang, Andrew Markham, and Niki Trigoni. Learning Object Bounding Boxes for 3D Instance Segmentation on Point Clouds. *NeurIPS*, 2019b.

Yu-jie Yuan, Yang-tian Sun, Yu-kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. NeRF-Editing: Geometry Editing of Neural Radiance Fields. *CVPR*, 2022.

Cheng Zhang, Zhaopeng Cui, Yinda Zhang, Bing Zeng, Marc Pollefeys, and Shuaicheng Liu. Holistic 3D Scene Understanding from a Single Image with Implicit Representation. *CVPR*, 2021a.

Jiakai Zhang, Xinhang Liu, Xinyi Ye, Fuqiang Zhao, Yanshun Zhang, Minye Wu, Yingliang Zhang, Lan Xu, and Jingyi Yu. Editable free-viewpoint video using a layered neural representation. *ACM Transactions on Graphics*, 40(4), 2021b. ISSN 15577368.

Xiuming Zhang, Pratul P. Srinivasan, Boyang Deng, Paul Debevec, William T. Freeman, and Jonathan T. Barron. NeRFactor: Neural Factorization of Shape and Reflectance Under an Unknown Illumination. *SIGGRAPH Asia*, 2021c.

Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J Davison. In-Place Scene Labelling and Understanding with Implicit Scene Representation. *ICCV*, 2021a.

Shuaifeng Zhi, Edgar Sucar, Andre Mouton, Iain Haughton, Tristan Laidlow, and Andrew J. Davison. iLabel: Interactive Neural Scene Labelling. *arXiv:2111.14637*, 2021b.

Yin Zhou and Oncel Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. *CVPR*, 2018.

Chuhang Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 3D-PRNN: Generating Shape Primitives with Recurrent Neural Networks. *ICCV*, 2017.

A APPENDIX FOR REBUTTAL

A.1 QUALITATIVE RESULTS OF 3D SCENE DECOMPOSITION AND MANIPULATION IN SECTIONS 4.3 & 4.4

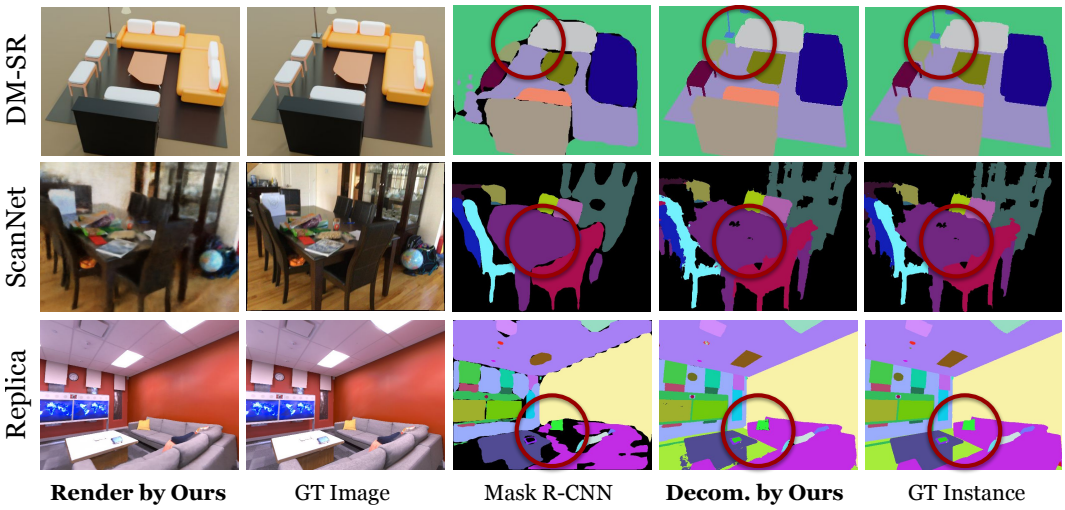


Figure 7: Qualitative results of our method and the baseline on three datasets: DM-SR, Replica and ScanNet. The dark red circles highlight the differences.

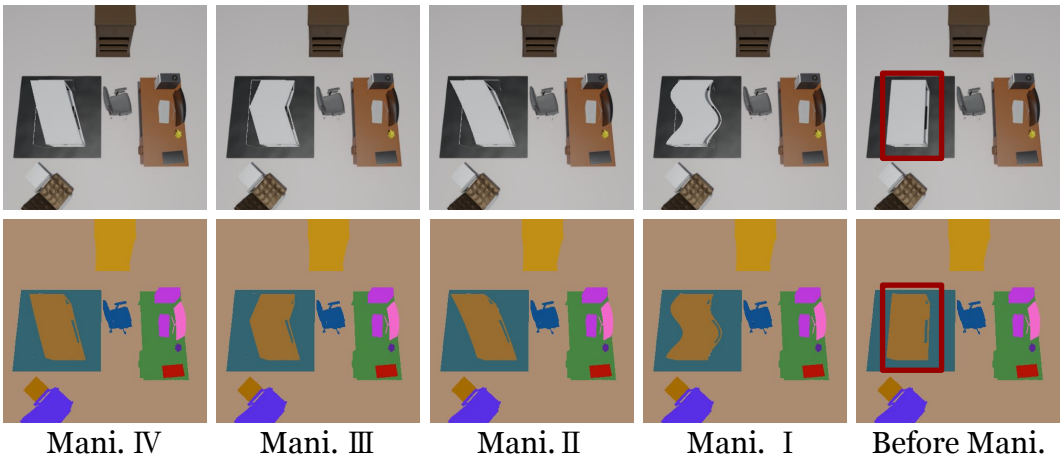
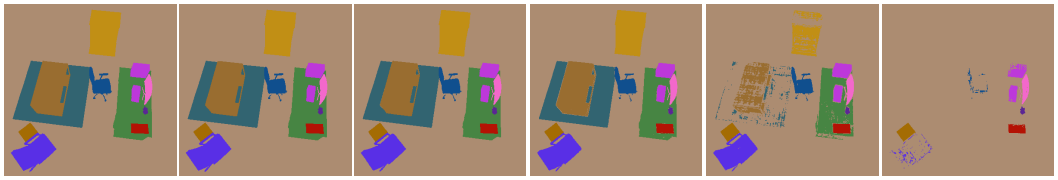


Figure 8: Qualitative results of our method for object deformation manipulation on DM-SR dataset. The dark red boxes highlight the target table to be manipulated.

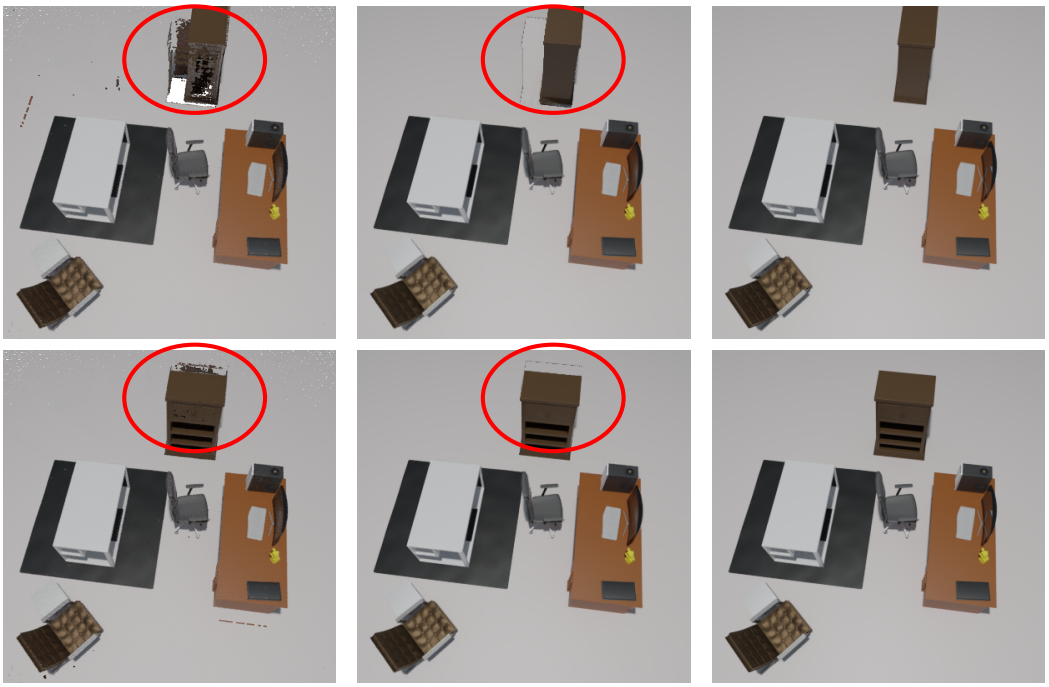
A.2 QUALITATIVE RESULTS OF OUR METHOD TRAINED ON NOISY 2D LABELS



Predicted Object Masks (0% Noisy Label) | Predicted Object Masks (10% Noisy Labels) | Predicted Object Masks (50% Noisy Labels) | Predicted Object Masks (70% Noisy Labels) | Predicted Object Masks (80% Noisy Labels) | Predicted Object Masks (90% Noisy Labels)

Figure 9: Estimated object masks at novel views from our models trained with different amount of noisy labels. Our method can infer satisfactory results, even though 80% of 2D labels are incorrect during training, demonstrating the robustness of our method.

A.3 QUALITATIVE RESULTS OF OUR METHOD AND POINT-NERF FOR OBJECT MANIPULATIONS



Point-NeRF | **Ours** | Ground Truth

Figure 10: Qualitative results of novel view rendering after manipulating 3D objects. It can be seen that our method obtains clearly sharper object shapes after manipulation in 3D space, whereas the baseline Point-NeRF shows obvious artifacts such as holes, primarily because its manipulation is conducted on explicit 3D point clouds followed by neural rendering, but our manipulation is conducted in continuous neural radiance space and therefore has fine-grained results.

A.4 REPAIRING ARTIFACTS

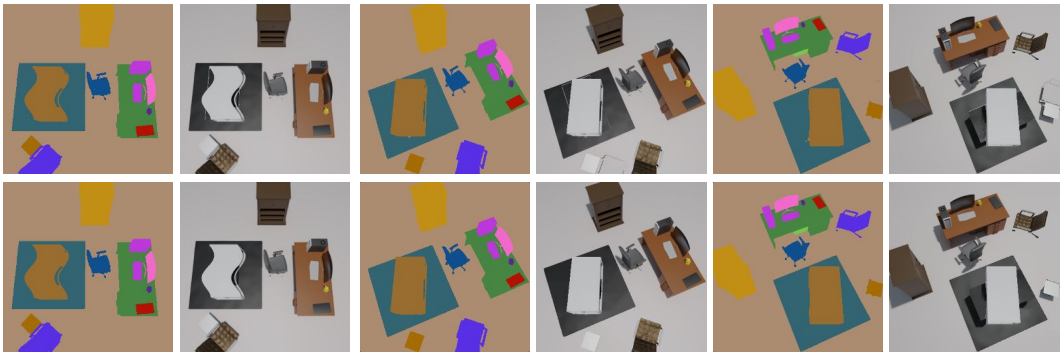


Figure 11: Artifacts Refinements.

A.5 PANOPTIC SEGMENTATION

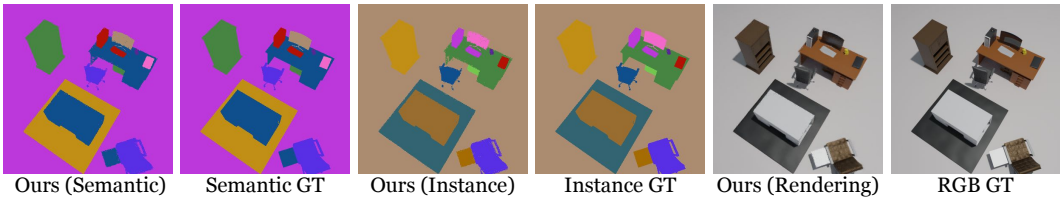


Figure 12: Qualitative results of our method for panoptic segmentation on DM-SR dataset. We insert an extra semantic branch into our current DM-NeRF. Due to the limited time, only 100K iterations are trained for this visualization.

A.6 SUPERVISION FROM MASK-RCNN

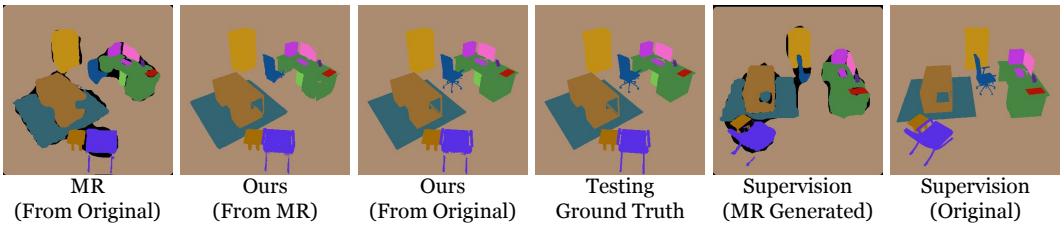


Figure 13: Qualitative results of our method for scene decomposition on DM-SR dataset. We use the instance labels generated by Msak-RCNN as the supervision signals when we train our DM-NeRF.

A.7 COMPUTATION

We typically train 200K iterations in ∼15 hours on each scene (∼0.27s for each iteration) with the batch size of 3072 rays, which uses ∼24GB GPU memory. In contrast, the original NeRF (rendering only) needs ∼0.22s for each iteration). During the inference of joint decomposition and rendering, our DM-NeRF costs ∼9.3s per image with the batch size of 4096 rays using ∼5GB GPU memory. For joint decomposition, manipulation and rendering, ∼23.4s is required for each image and ∼9GB GPU memory is needed when the batch size is set as 4096 rays. To just render a image from a novel view, the original NeRF and Point-NeRF need ∼7.8s and ∼8.2s, respectively. All training and testing are operated on a single Nvidia GeForce RTX 3090 card.