LaTeRF: Label and Text Driven Object Radiance Fields

Ashkan Mirzaei, Yash Kant, Jonathan Kelly, and Igor Gilitschenski

University of Toronto

Abstract. Obtaining 3D object representations is important for creating photo-realistic simulators and collecting assets for AR/VR applications. Neural fields have shown their effectiveness in learning a continuous volumetric representation of a scene from 2D images, but acquiring object representations from these models with weak supervision remains an open challenge. In this paper we introduce LaTeRF, a method for extracting an object of interest from a scene given 2D images of the entire scene and known camera poses, a natural language description of the object, and a small number of point-labels of object and non-object points in the input images. To faithfully extract the object from the scene, LaTeRF extends the NeRF formulation with an additional 'objectness' probability at each 3D point. Additionally, we leverage the rich latent space of a pre-trained CLIP model combined with our differentiable object renderer, to inpaint the occluded parts of the object. We demonstrate high-fidelity object extraction on both synthetic and real datasets and justify our design choices through an extensive ablation study.

Keywords: image-based rendering, neural radiance fields, 3D reconstruction, 3D computer vision, novel view synthesis

1 Introduction

Extracting a partially-occluded object of interest (OOI) from a 3D scene remains an important problem with numerous applications in computer vision and elsewhere. For example, using such methods enables the creation of faithful photo-realistic simulators for robotics, supports the creation of digital content (*e.g.* animated movies), and the building of AR/VR tools. Although 3D scene reconstruction from 2D images has been well explored [23,14,46,35,18], the ability to extract semantically consistent and meaningful objects from complex scenes remains an open challenge.

Early approaches [34,56] attempted to segment objects from real images, but could not extract their 3D geometry and did not consider the underlying image formation process. Recent work has attempted to disentangle object-centric representations from a scene by learning to decompose it into a scene graph and train object representations at the leaf nodes [27], or via the use of joint 3D semantic segmentation and neural volumetric rendering [55]. In [27], the

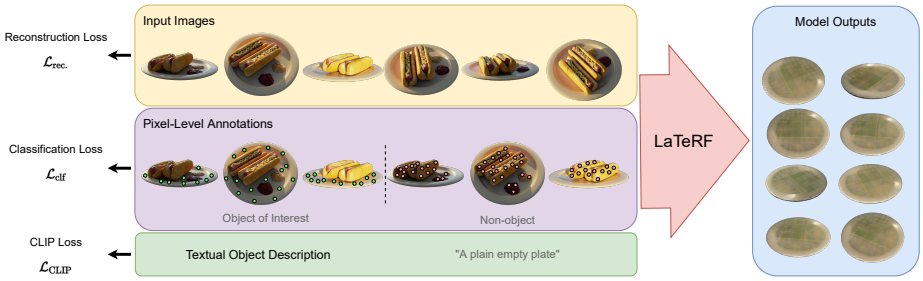

Fig. 1: An overview of LaTeRF showing the extraction of a plate. The method takes as input 2D images with camera parameters and poses, a few manually annotated pixel labels corresponding to the object, and labels pointing to the places in the images that do not belong to the object (foreground or background parts). LaTeRF is able to extract the object from the scene while also filling the missing details of the object invisible in the input views.

approach is specifically designed for automotive data and is not applicable to object instances from other categories, whereas [55] leads to non-photorealistic object extraction due to training constraints required for real-time deployment. Moreover, none of the works allows for reasoning about occlusions of the OOI in the scene. Approaches proposing unsupervised discovery of 3D objects from a single image [51,38] typically require optimizing a non-trivial loss function and does not work well with high-resolution real-world scenes. Other approaches [47] require large-scale prior category-level videos which are expensive to collect and might not generalize well to novel categories.

We propose LaTeRF, a novel object extraction method given a set of 2D images (with known camera parameters) containing the OOI mixed with other clutter, a textual description of the OOI, and a minimal set of pixel annotations distinguishing the object and non-object boundaries. More specifically, we assign an objectness probability to each point in the 3D space and use this to guide the disentanglement of the OOI and non-object parts in the scene during training. Once trained, we use an *objectness threshold* to filter points belonging to the OOI while ignoring non-object points. Moreover, we find that reasoning about occlusion is an important aspect of extracting objects without artifacts (such as holes, or clutter). Thus, we leverage CLIP [31], a large cross-modal vision-language model, for formulating an objective to fill in the occluded parts using supervision derived from a textual prompt. Combining these two loss functions we can perform high-fidelity object extraction from photo-realistic scenes. An overview of our approach is visualized in Figure 1. We evaluate LaTeRF on real-world and synthetic data, and study the effects of individual design choices in extensive ablation studies. We find that under challenging settings with high occlusions, using LaTeRF leads to 11.19 and 0.63 gains in PSNR and SSIM metrics respectively.

2 Related Work

3D object reconstruction: Understanding 3D structure from 2D input images has been a long-standing problem in computer vision. Several works have tried to learn category-specific information from a set of videos [10,47] and use this information to generalize to new instances within the same category. Recent work [17] captures objects from images with varying illumination and backgrounds, but is unable to inpaint potential missing parts and relies on having an object mask for every input image. Neural scene graphs have been utilized for scene decomposition into background and object instances for automotive data [27]. Another line of work involves recovering the 3D structure of a scene using a single image only and in an unsupervised manner [51,38], but it is limited to simple scenarios and lack the ability to perform well in complex scenes. Although these methods can recover the 3D shape and appearance of the objects with a few 2D inputs, most of them rely on training on a large set of either category-specific or general objects, and collecting such data can be expensive or dangerous for certain categories. We aim to overcome the need for prior training and instead employ a minimal set of test-time clues given by a human annotator to extract a photo-realistic object radiance field from the scene.

Representing Scenes with neural fields: Volumetric rendering of 3D objects and scenes via neural fields has attracted a lot of attention [41]. Among them, in the past few years and based on differentiable volume rendering [42,9], NeRF-style methods [23,53,25,33,28,49,50] have achieved impressive results in reconstructing high-quality 3D models and novel views learned from 2D inputs of bounded static scenes. The success of NeRF in capturing the details of the scenes is partly indebted to positional encoding [43,7] and periodic activations [36,5,37] that help to increase the model's capacity. The backbone of our method is a neural field[23,48] and we use it to learn the density and the view-dependent color for the scene and the object simultaneously based on the camera parameters and input images of the scene containing the object of interest. It is worthwhile to mention that our method is not limited to a specific neural field method and can be extended easily to faster [19,40,15] and better-quality NeRFs [2,24].

Semantic segmentation in 3D: Semantic segmentation in 3D has been studied using multi-view fusion-based representations [1,22,11,20,39,21,44,52] which require only 2D supervision when training, and a separate 3D mesh at testing time unlike implicit methods like ours. Recently, there have been promising attempts to recover 3D semantic maps from 2D inputs using NeRFs. NeSF [45] recovers the 3D geometry as density fields from posed RGB images and uses 2D semantic maps to train a semantic field to assign a probability of belonging to each of the semantic classes to every point in the space. Another proposed approach is to extend the NeRF MLP and add a view-independent semantic head to it to get the logits for each of the semantic classes [54,55]. Our idea is conceptually similar to these, and we propose using binary partitioning of 3D points into two categories (object of interest vs. non-object). In contrast to the previous works, we introduce a differentiable rendering scheme to only render

the object of interest using these objectness probabilities, which enables us to fill the visual gaps in the object.

Language-guided NeRFs: In the representation learning [3] literature, self-supervised approaches [6,8,26,12] are some of the most compelling settings due to the rather cheap availability of unlabelled data. CLIP [31] is a contrastive-based representation learning method that learns visual and textual feature extraction by looking at a massive set of diverse image-caption pairs scraped from the internet. Following the success of CLIP in finding the similarity of image and text samples, recent works have used it for image generation guided by language [32]. More freshly, fusing CLIP with NeRF has been explored. In [14], CLIP similarity of the rendered scene from different views is utilized to reduce the amount of data needed to train the model and alleviate the data-hungriness of NeRFs. This is based on the CLIP's ability to assign similar features to different views of a single object. CLIP-NeRF [46] makes use of the joint image-text latent space of CLIP and provides a framework to manipulate neural radiance fields using multi-modal inputs. Dream fields paper [13] uses the potential of CLIP in finding the similarity between a text and an image and optimizes a NeRF to increase the similarity of its different renders to a text prompt. This way, starting from a phrase, they come up with a 3D object closely representing the text.

Motivated by the results of these works in showing the possibility and benefits of using CLIP alongside neural radiance fields, we leverage the rich multi-modal feature space of the pre-trained CLIP model to give our object extractor semantic information as text and help it to inpaint the points of the object of interest that are invisible and obscured by other elements in the scene. Needless to say, we only use CLIP as a good current instance of a joint image-language model, and the results of our model can be improved with richer joint embedding functions in the future. The language module of our method lies closely with the dream fields paper [13] but we use it to generate 3D objects that are consistent with the ones provided in a scene.

3 Background

Neural Radiance Fields (NeRFs) [23] use a multi-layer perceptron (MLP) to implicitly capture the geometry and visual appearance of a 3D scene. A scene is encoded as a mapping between the 3D coordinates x and view directions d, to a view-dependent color c and view-independent density σ using an MLP $f : (x, d) \rightarrow (c, \sigma)$. For simplicity, for every point x and view direction d, we show the outputs of the MLP with $c(x, d)$ and $\sigma(x)$.

Consider a ray r with origin o and direction d characterized as $r(t) = o + td$ with near and far bounds t_n and t_f respectively. Similar to the original NeRF formulation [23], the rendering equation to calculate the expected color for the ray $C(r)$ is:

$$C(r) = \int_{t_n}^{t_f} T(t)\sigma\big(r(t)\big)c\big(r(t), d\big)\,\mathrm{d}t, \tag{1}$$

where $T(t) = \exp\left(-\int_{t_n}^{t} \sigma(r(s))\,\mathrm{d}s\right)$ is the transmittance. This integral is numerically estimated via quadrature. The interval between t_n and t_f is partitioned into N equal sections, and t_i is uniformly sampled from the i-th section. This way, the continuous ray from t_n to t_f is discretized and the estimated rendering integral can be obtained as:

$$\hat{C}(r) = \sum_{i=1}^{N} T_i\big(1 - \exp(-\sigma_i \delta_i)\big)c_i, \tag{2}$$

where $T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$ is the discretized estimation of the transmittance, and $\delta_i = t_{i+1} - t_i$ is the distance between two adjacent sample points on the ray. Note that for the sake of simplicity, $\sigma\big(r(t_i)\big)$ and $c\big(r(t_i), d\big)$ are being shown as σ_i and c_i respectively. The rendering scheme shown in Eq. 2 is differentiable, and allows training our MLP to minimize the L2 reconstruction loss between the estimated color $C(r)$ and the ground-truth color $C_{\mathrm{GT}}(r)$ via variants of the stochastic gradient descent algorithm (SGD):

$$\mathcal{L}_{\text{rec.}} = \sum_{r \in \mathcal{R}} \big\| C(r) - C_{\mathrm{GT}}(r) \big\|^2, \tag{3}$$

where $\mathcal{R}$ is a batch of rays sampled from the set of rays where their corresponding pixel is available in the training data. In this paper, the reconstruction loss $\mathcal{L}_{\text{rec.}}$ is used to support the consistency of the extracted object radiance field with the one shown in the original scene. The goal is to make the resulting object look similar to the one represented in the 2D input images of the scene, and not only generate an object within the same category.

4 Method

In this paper, we propose a simple framework to extract 3D objects as object radiance fields from a set of 2D input images of a scene containing the object of interest (OOI). Built on top of the recent advances in scene representation via neural fields [41,23], our method aims to softly partition the space into the object and the foreground/background by adding an object probability output to the original NeRF's MLP. The model is optimized concerning pixel-level annotations pointing to the object or the foreground/background, and a text prompt expressing the visual features and semantics of the object.

4.1 Objectness Probability

In this paper, we want to extract an OOI from a neural radiance field guided by a minimal set of human instructions. Our approach is to reason about the probability of each point in the space being part of the object. Recent work proposed to append a view-independent semantic classifier output [54] to the original NeRF architecture. Inspired by this, we extend the NeRF's MLP to

return an additional objectness score $s(x)$ for every point x in the space. The objectness probability $p(x)$ concerning the output of the MLP is then obtained as:

$$p(x) = \text{Sigmoid}\big(s(x)\big). \tag{4}$$

For an overview of the architecture of the MLP used in LaTeRF please refer to the supplementary material.

4.2 Differentiable Object Volume Renderer

The most straightforward approach to render the object using the NeRF model with the objectness probabilities is to apply a threshold on these probabilities and zero out the density of all points in the space with less than 0.5 objectness probabilities. In other words, for every point x, the new density function would be $\sigma'(x) = \sigma(x)\mathbb{1}\big(p(x) \geq 0.5\big)$ and the naive object rendering integral would be:

$$C_{\text{obj}}(r) = \int_{t_n}^{t_f} T_{\text{obj}}(t)\sigma\big(r(t)\big)\mathbb{1}\Big(p\big(r(t)\big) \geq 0.5\Big)c\big(r(t), d\big)\,\mathrm{d}t, \tag{5}$$

where the object transmittance T_{obj} is defined as:

$$T_{\text{obj}}(t) = \exp\bigg(-\int_{t_n}^{t} \sigma\big(r(s)\big)\mathbb{1}\Big(p\big(r(s)\big) \geq 0.5\Big)\,\mathrm{d}s\bigg). \tag{6}$$

This approach leads to a hard-partitioning of the scene into two categories: *1*) The extracted object, *2*) background and foreground. Although this approach works in practice and gives an overview of the object, if we use numerical quadrature on Eq. 5 to evaluate the pixel values for the object, the gradients of the resulting pixel colors with respect to the weights of the MLP become zero in most of the domain due to the use of the indicator function. As a result, it is not possible to define a loss function to enforce properties on the rendered images of the object (more on this loss function later).

We fix the aforementioned issue with a minor tweak. The transmittance (density) $\sigma(x)$ can be interpreted as the differential probability of an arbitrary ray passing through x being terminated at the particle at point x in the space. Let T_x be the event of termination of a ray at point x, i.e. we have $\sigma(x) = \mathbb{P}(T_x)$. In addition, denote the event of the location x containing a particle belonging to the OOI as O_x ($p(x) = \mathbb{P}(O_x)$). With this notion, the object can be rendered with a new density function $\sigma_{\text{obj}}(x)$ defined as the joint probability of T_x and O_x as $\sigma_{\text{obj}}(x) = \mathbb{P}(T_x, O_x)$. With the assumption of T_x and O_x being independent, the object density function is:

$$\sigma_{\text{obj}}(x) = \mathbb{P}(T_x, O_x) = \mathbb{P}(T_x)\mathbb{P}(O_x) = \sigma(x)p(x). \tag{7}$$

Having the above equation, the object rendering integral in Eq. 5 can be changed to:

$$C_{\text{obj}}(r) = \int_{t_n}^{t_f} T_{\text{obj}}(t)\sigma\big(r(t)\big)p\big(r(t)\big)c\big(r(t), d\big)\,\mathrm{d}t, \tag{8}$$

and the object transmittance T_{obj} in Eq. 6 becomes:

$$T_{\text{obj}}(t) = \exp\Big(- \int_{t_n}^{t} \sigma\big(r(s)\big) p\big(r(s)\big)\, \mathrm{d}s \Big). \tag{9}$$

Now, similar to the previous quadrature, the integral in Eq. 8 is approximated as follows:

$$\hat{C}_{\text{obj}}(r) = \sum_{i=1}^{N} T_i^{\text{obj}} \big(1 - \exp(-\sigma_i p_i \delta_i)\big) c_i, \tag{10}$$

where the discretized object transmittance is $T_i^{\text{obj}} = \exp\big(- \sum_{j=1}^{i-1} \sigma_j p_j \delta_j \big)$ and $p_i = p\big(r(t_i)\big)$. Note that with this new object rendering method, gradients are smoother and it is possible to use iterative optimization to minimize common loss functions defined over the rendered views of the object. Compared to the previous rendering equation with the indicator function, the new equation can be interpreted as a soft-partitioning of the space where every point belongs to each of the two classes (the object and the foreground/background) with a probability and the partitioning is stochastic.

4.3 Object Classification using Pixel Annotations

The first source of clues obtained from a human annotator to extract the object is the pixel-level information. The user selects a few of the 2D input images of the scene and, for each of them, chooses a few pixels on the OOI and a few pixels corresponding to the background or the foreground. After collecting these data, for a ray r corresponding to one of the annotated pixels, the label $t(r)$ is defined as 1 if the pixel is labeled as the object, and $t(r) = 0$ if it is labeled as either foreground or background. The objectness score $S(r)$ is calculated using the classical volume rendering principles:

$$S(r) = \int_{t_n}^{t_f} T(t) \sigma\big(r(t)\big) s\big(r(t)\big)\, \mathrm{d}t. \tag{11}$$

The objectness probability for a ray r is obtained as $P(r) = \text{Sigmoid}\big(S(r)\big)$. In the implementation, the numerical estimation of the above integral that has been used is:

$$\hat{S}(r) = \sum_{i=1}^{N} T_i \big(1 - exp(-\sigma_i \delta_i)\big) s_i. \tag{12}$$

where $s_i = s\big(r(t_i)\big)$. Consequently, the approximated objectness probability is $\hat{P}(r) = \text{Sigmoid}\big(\hat{S}(r)\big)$ and following [54], the classification loss $\mathcal{L}_{\text{clf}}$ is defined as:

$$\mathcal{L}_{\text{clf}} = -t(r) \log \hat{P}(r) - \big(1 - t(r)\big) \log\big(1 - \hat{P}(r)\big). \tag{13}$$

This loss function guides the model to tune $s(x)$ to be high for points that belong to the object, and decrease it for the rest of the points. The MLP

is able to propagate the sparse pixel-level data over the OOI and the background/foreground [54] resulting in a binary classification of points in the space. However, this loss function alone is not able to reason about the occluded parts of the object, which results in an extracted object with possible holes and artifacts around the parts that are not visible in any of the input images, or the areas visible in only a small number of the inputs.

4.4 CLIP Loss

To mitigate the shortcomings of the classification loss in inpainting the unseen parts of the object, we propose the use of a phrase describing the object as the additional signal to help train the MLP and extract the OOI from the scene with the potential holes being filled. The user is asked to describe the object in a few words and point out its semantic and appearance. Let t_{text} represent this user input. Subsequently, the CLIP loss is used to make sure that the rendered object from random views is similar to the textual clue.

The contrastive language-image pre-training (CLIP) [31] is a multi-modal feature extractor trained on a large dataset of image and caption pairs collected from the Internet. It includes two encoders with a shared normalized latent (output) space. For an image I and text t (typically a sentence or phrase), the similarity of their features is proportional to the probability of the text t being associated with I. According to CLIP's massive and diverse train data, it has been shown to be useful in zero-shot transfer applications for visual and textual data, e.g. object recognition and image synthesis. Recent works suggest that it is applicable and beneficial to novel view synthesis tasks [14,13,46].

The object can be rendered from a random view v pixel by pixel using the differentiable object rendering scheme in Eq. 10 resulting in an image $I_{\text{obj}}(v)$ showing the current estimation of the object by the model. In order to maximize the similarity between the rendered object $I_{\text{obj}}(v)$ and the clue phrase t_{text}, the CLIP loss function $\mathcal{L}_{\text{CLIP}}$ is defined as follows:

$$\mathcal{L}_{\text{CLIP}} = -\text{Sim}_{\text{CLIP}}(I_{\text{obj}}(v), t_{\text{text}}), \tag{14}$$

where $\text{Sim}_{\text{CLIP}}(I, t)$ is the similarity of the features of the image I and the text t extracted by the pre-trained CLIP model. This loss is influenced by the recent work [13] that utilizes the CLIP model to generate 3D shapes from text descriptions. Nevertheless, our proposed method starts with a scene as a starting point and already has a template of the object, and uses the CLIP loss to fix the obscured volumes and surfaces and reason about the shape and color of the missing parts.

4.5 Training Details

In summary, the reconstruction loss $\mathcal{L}_{\text{rec.}}$ is to make sure that the final object's shape and appearance are consistent with the training images of the scene, while the classification loss $\mathcal{L}_{\text{clf}}$ is to guide the model to find the OOI in the scene and

resolve potential ambiguities and object redundancies that can occur with only using the text query. Meanwhile, the CLIP loss $\mathcal{L}_{\text{CLIP}}$ is utilized to inpaint potential holes and occluded parts of the object using a high-level semantic and appearance description which is a piece of text describing the OOI. The final loss function $\mathcal{L}$ used to train the model is:

$$\mathcal{L} = \mathcal{L}_{\text{rec.}} + \lambda_{\text{clf}}\mathcal{L}_{\text{clf}} + \lambda_{\text{CLIP}}\mathcal{L}_{\text{CLIP}}, \tag{15}$$

where the constants λ_{clf} and λ_{CLIP} are hyperparameters of the model. Having both λ_{clf} and λ_{CLIP} set to zero (or close to zero) results in reproducing the whole scene without any object extraction.

5 Experiments

Training an object radiance field using LaTeRF can be done with either 360° inward-facing posed 2D images or forward-facing posed views of the scene, in addition to the visual and textual cues for OOI selection. Besides qualitative results, we study the consequences of different loss terms, the number of visual cues, and the underlying NeRF's representation quality on the quality of the resulting object asset on our synthetic dataset of challenging scenes with object occlusions.

Datasets In our experiments, we use a subset of real-world scenes borrowed from NeRD [4], forward-facing scenes that we collected from real-world objects, and a collection of synthetic scenes full of occlusions that we specifically designed to conduct our quantitative results. Our synthetic data is rendered using Blender, and consists of views of the scene, ground-truth object masks, and the ideal extracted object as illustrated in Figure 2. The lighting between different views of most of the scenes provided by NeRD [4] which are partly from the British Museum's photogrammetry dataset [30] are inconsistent because they are designed for the task of decomposition of a scene into its underlying shape, illumination, and reflectance. As a result, we manually select a subset of the images in these datasets that are roughly similar in the sense of their lighting and train LaTeRF on them.

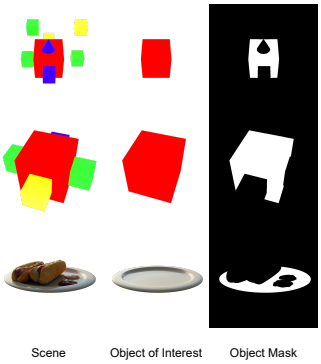

Fig. 2: An overview of 3 of the scenes in our synthetic dataset used in the quantitative evaluations.

Baseline To the best of our knowledge, LaTeRF is the only method able to pull out and inpaint a 3D object from 2D views of a scene, without having a huge category-specific multi-view or video data prior to extraction. We use a NeRF [23,48] model trained on the images of each scene after applying object masks as a baseline which we call it *Mask+NeRF*. In Mask+NeRF, prior to the training, we substitute every pixel marked as non-object in the ground-truth mask with a black pixel and train a NeRF on the new masked images.

Fig. 3: Real-world objects extracted from 360° degree inward-facing and forward-facing 2D input images.

Fig. 4: The pipeline used to filter background noises from rendered objects.

Metrics For the quantitative comparisons of the synthetic dataset, we report the peak signal-to-noise ratio (PSNR) and the structural similarity index measure (SSIM) and for both of them, the higher is better.

5.1 Real-World Scenes

Qualitative results of LaTeRF in extracting objects from real-world scenes are shown in Figure 3. There are both forward-facing datasets and 360° inward-facing scenes [4,30] used in this experiment to show the flexibility of LaTeRF in extracting objects from either scenario.

In the real-world scenes with textured and complex backgrounds, we observed that small particles (points with small densities) emerge throughout the scene which are almost invisible from the training views. These particles blend in the background or the OOI and disappear, but when filtering the object, they become visible. This leads to object renderings full of noise in the background (See soft threshold results in Figure 4). In order to remove the noise, we first make the sampled densities on each ray smoother by substituting the density of each point with the average of its density and the density of its direct neighbors and we do this for 5 steps. After smoothing the densities, most of the small particles which are abrupt peaks in the value of density become close to zero. As a result, they can be filtered via applying a hard threshold based on the densities and filtering all the points with a density below the threshold. Later, the object mask is rendered via substituting the objectness score instead of color in Eq. 10 and applying the sigmoid function to the results while assuming that on each ray and in an infinitely far point, there is a particle with high density and low objectness score. As a result of this process, the denoised object renderings can be obtained via applying object masks on the noisy images (the result of each step of this denoising pipeline is shown in Fig 4). Please refer to the supplementary material for more details on the denoising process.

5.2 Synthetic Scene Evaluations

In order to be able to quantitatively evaluate our proposed method in different scenarios, we generate synthetic scenes using Blender, each containing the following information: 1) Multi-view 400×400 pixel images of a scene that contains

different objects including an OOI, 2) the ground-truth images of the OOI so that we can evaluate the reconstruction quality of our object selection method, 3) the mask for the OOI in each of the 100 training images of the scene to use instead of manual pixel-level annotations for evaluations, 4) and the ground-truth camera intrinsic and extrinsic parameters for each of the input images.

Having ground-truth object masks in the dataset makes it easy to automatically sample a certain number of pixels from the object and the foreground/background to study the effects of pixel labels in the results. Sample data from some of the challenging cases of our synthetic dataset can be found in Figure 2. As shown in the figure, the synthetic scenes have been designed to have parts of the OOI obscured with other objects to make the extraction more challenging.

Pixel Label Count We ablate along the number of pixel labels used in the training to demonstrate its effect on the reconstruction quality of our proposed method. Having the ground-truth object masks in the synthetic dataset, at each step, we simply sample a different number of pixel labels from the mask based on uniform distribution while making sure that the number of positive labels (labels on the OOI) and the negative ones (non-objects) are the same. Using the sampled subset of the pixel-labels, we extract the OOI from the scene. In order to calculate the reconstruction probability, we compare the results of our method to the ground-truth renderings of the OOI in the dataset from 30 different viewpoints and report the average PSNR and SSIM. Table 1 shows the results of the experiment with different numbers of pixel labels used. Since each of the 100 training images has 400×400 pixels and the ground-truth object mask includes labels for each of these pixels showing whether each of them belongs to the OOI or the foreground/background, there are overall $16,000,000$ labels for each of the scenes. As evident in Table 1, the reconstruction quality of the OOI stays within the same range when reducing the number of labels from $16,000,000$ to only 160 labels (an overall of 80 positive labels and 80 negative labels which are uniformly distributed across 100 training images). Note that a total of 160 labels can be easily obtained within a few seconds or minutes by a human annotator and is much easier than having someone mask out the object for every single training image.

Pixel Labels #	PSNR↑	SSIM↑
16,000,000	**26.93**	**0.95**
1,600,000	26.90	0.94
160,000	26.48	0.90
16,000	26.36	0.90
1,600	26.10	0.94
160	26.00	0.85
16	20.81	0.86

Table 1: The effect of the label count on the reconstruction quality of the OOI.

Importance of Boundary Pixel Labels Intuitively, it is clear that not all pixel labels have the same importance. Pixels close to the intersection of the object and non-object parts of the views can help the model to extract the OOI with sharper and more accurate edges. In this experiment, we show that

selecting the labels close to the boundaries of the object helps to improve the reconstruction quality. This way, the human annotator can be asked to spend most of their time labeling pixels around the edges. Moreover, we can increase the annotator's brush size (allow the user to label an area at a time instead of a single pixel) to let them quickly annotate the points further from the boundaries and then focus on the more important boundary labels.

In this experiment, the boundary labels are computed by applying minimum and maximum filters with kernel size 3 on the object masks, reducing the number of labels from $16,000,000$ to less than $500,000$ for our scenes. The pixel labels for this experiment are then uniformly sampled among these boundary pixels. Table 2 contains the results with different number of boundary labels. As shown in the table, the reconstruction quality of the model is less affected than the experiment with uniform labels all around the images when reducing the boundary labels. The results show up to 4.81 improvement in PSNR compared to the previous experiment which happens when using only 16 pixel labels. It is worth mentioning that this case has an overall of 16 labels across all of the input images and not 16 per image.

Boundary Labels #	PSNR↑	SSIM↑
16,000	**26.78**	**0.93**
1,600	26.50	0.91
160	26.36	0.92
16	24.82	0.90

Table 2: The effect of the number of boundary labels on the reconstruction quality of the model for the OOI on the synthetic dataset.

Effects of Different Loss Functions As shown in Eq. 15, the overall loss function used to train the main model contains three different parts: 1) The reconstruction loss $\mathcal{L}_{\text{rec.}}$ which is the same loss function used in NeRF [23] to let the model learn the whole scene, 2) the classification loss $\mathcal{L}_{\text{clf}}$ which is the loss defined over the pixel-level annotations to detect the OOI, 3) and the CLIP loss $\mathcal{L}_{\text{CLIP}}$ guiding the model to fill the covered chunks of the OOI. In this section, the effect of each of these functions on the final reconstruction quality of the model is studied.

Model	PSNR↑	SSIM↑
Mask+NeRF	15.74	0.32
$\mathcal{L}_{\text{rec.}}$	12.91	0.79
$\mathcal{L}_{\text{rec.}} + \lambda_{\text{clf}}\mathcal{L}_{\text{clf}}$	14.10	0.82
$\mathcal{L}_{\text{rec.}} + \lambda_{\text{CLIP}}\mathcal{L}_{\text{CLIP}}$	21.95	0.84
LaTeRF (ours)	**26.93**	**0.95**

Table 3: The consequence of using each of the loss functions compared to the baseline.

The reconstruction loss is the main loss enforcing the 3D consistency of the learned scene and OOI. Thus, it can not be removed from the loss function and has to be present in all of the baselines. Our scenarios include 4 cases where for each of them, some of the classification loss and the CLIP loss are used for the training on the synthetic dataset. These scenarios include: 1) $\mathcal{L}_{\text{rec.}}$ where non of the classification loss and the CLIP loss are used. This is similar to the original NeRF [23] and there is no clue guiding the model to find

Fig. 5: A qualitative representation of the effectiveness of the CLIP loss in filling the parts that are invisible in the input images. The OOI is the empty plate without its contents. The text prompt used to calculate the CLIP loss is *"A plain empty plate"*.

the OOI. 2) $\mathcal{L}_{\text{rec.}} + \lambda_{\text{clf}}\mathcal{L}_{\text{clf}}$ where the CLIP loss is ignored and the object is only being extracted using the pixel labels. The model has no clue to reason about the missing parts with this baseline. 3) $\mathcal{L}_{\text{rec.}} + \lambda_{\text{CLIP}}\mathcal{L}_{\text{CLIP}}$ where only the text clue is used to lead the model towards finding the OOI in the scene. 4) $\mathcal{L}_{\text{rec.}} + \lambda_{\text{clf}}\mathcal{L}_{\text{clf}} + \lambda_{\text{CLIP}}\mathcal{L}_{\text{CLIP}}$ which is our complete proposed method. We compare the results with the baseline which is Mask+NeRF.

Table 3 shows the result of this experiment with the mentioned baselines. Where only the reconstruction loss is being used, the semantic head of the MLP can not be trained and the results are unsatisfactory as expected. The model with only the classification loss is unable to produce on par results with the proposed method as the synthetic dataset is designed to include objects of interest that are partly covered by other objects, and the CLIP loss is the only semantic clue that is used in our method to fill the blank spots. The best result is obtained when using all of the loss functions together and each of them contributes to extracting high-quality objects.

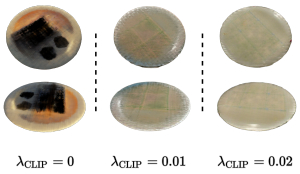

Fig. 6: The effect of the weight of the CLIP loss on the extracted plate from the hotdog scene.

Figure 5 visually compares the effect of the presence or absence of the CLIP loss in the rendered object. The scene used in this example is a challenging scenario where the goal is to extract the empty plate and remove the contents of the plate. There are various parts of the plate's surface for which no visual information is available in the 2D input images of the scene. This example shows that the CLIP loss, using a text phrase like *"A plain empty plate"*, is able to reason about the missing particles on the surface of the plate and render a plausible empty plate. In addition, the CLIP loss was able to remove the reflectance of the hotdogs on the plate to some extent. However, there still are mild artifacts on the plain plate after using the CLIP loss due to the challenging nature of this task since a large portion of the plate's surface is covered in the inputs. As another example, the model has difficulty with removing the hotdog's shadow from the original plate and has tried to complete the shadow instead of erasing it. This issue can be mitigated by increasing the CLIP loss, but it will result in

a plate less consistent with the one in the original scene as shown in Figure 6. Notice that not only the depth of the plate is reduced compared to the one in the original scene when increasing the CLIP weight λ_{CLIP} to 0.02, the light flash on the plate is also getting rotated and differs from the original radiance. Note that for the case without the CLIP loss (the middle row in Fig. 5), the background has been set to black to better visualize the holes in the plate.

Importance of the Number of Input Views The neural volumetric rendering method that is used in LaTeRF is not limited to a certain implementation of NeRF. We argue that with better quality and faster neural rendering approaches, the reconstruction quality and the speed of LaTeRF can increase. In this experiment, the effect of the quality of the base neural renderer on the final object reconstruction quality is studied. In order to mimic the behavior of a lower quality neural rendering model, we limit the number of training input views fed to the model for the calculation of the reconstruction loss $\mathcal{L}_{\text{rec.}}$. The results in Table 4 depict that the details and quality of the rendered object via LaTeRF is tightly related to the reconstruction quality of the base NeRF model used for the consistency of the extracted object with the one present in the scene. Consequently, as better volumetric rendering approaches for 3D reconstruction are introduced, LaTeRF can take advantage of their higher qualities enabling the creation of better object extractors.

Input Views #	PSNR↑	SSIM↑
100	**26.93**	**0.96**
80	26.62	0.95
60	26.50	0.92
40	26.20	0.95
20	25.35	0.94

Table 4: The effect of the number of 2D input images as training data for the calculation of the reconstruction loss $\mathcal{L}_{\text{rec.}}$ on the extracted object.

6 Conclusion

We present LaTeRF, a method to extract digital objects via neural fields from 2D input images of a scene and a minimal set of visual and natural language clues guiding the model to the object of interest. The geometry and color of different points are primarily captured via a NeRF model, while the objectness probability of points is mainly trained via pixel labels provided by the user which contains binary labels of some of the input pixels to show that each of them belongs to the object of interest or not. Additionally, a CLIP loss is defined to ensure high similarity between the rendered images of the object and a text prompt expressing the appearance and semantics of the object to reason about the missing parts of the object. The effectiveness of each of the components of the training scheme is shown in our experiments. However, we observed a need for per-scene fine-tuning of the hyper-parameters λ_{clf} and λ_{CLIP}. An additional challenge for LaTeRF is the extraction of transparent objects or objects with shiny surfaces having reflectances of other objects. Moreover, LaTeRF naturally comes with the general limitations associated with NeRFs including the need

for per-scene training and data hungriness. Additionally, the rendered images used for the calculation of the CLIP loss had to be downsized to avoid memory shortage. Applying the CLIP loss to higher resolution images will result in better inpaintings.

References

1. Armeni, I., He, Z.Y., Gwak, J., Zamir, A.R., Fischer, M., Malik, J., Savarese, S.: 3d scene graph: A structure for unified semantics, 3d space, and camera. In: ICCV (2019) 3
2. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. arXiv (2021) 3
3. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. In: IEEE transactions on pattern analysis and machine intelligence (2013) 4
4. Boss, M., Braun, R., Jampani, V., Barron, J.T., Liu, C., Lensch, H.: Nerd: Neural reflectance decomposition from image collections. In: ICCV (2021) 9, 10, 19
5. Candès, E.J.: Harmonic analysis of neural networks. Applied and Computational Harmonic Analysis (1999) 3
6. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: ICML (2020) 4
7. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N.: Convolutional sequence to sequence learning. In: ICML (2017) 3
8. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: CVPR (2020) 4
9. Henzler, P., Mitra, N.J., Ritschel, T.: Escaping plato's cave: 3d shape from adversarial rendering. In: ICCV (2019) 3
10. Henzler, P., Reizenstein, J., Labatut, P., Shapovalov, R., Ritschel, T., Vedaldi, A., Novotny, D.: Unsupervised learning of 3d object categories from videos in the wild. In: CVPR (2021) 3
11. Hermans, A., Floros, G., Leibe, B.: Dense 3d semantic mapping of indoor scenes from rgb-d images. In: ICRA (2014) 3
12. Hénaff, O.J., Srinivas, A., Fauw, J.D., Razavi, A., Doersch, C., Eslami, S.M.A., van den Oord, A.: Data-efficient image recognition with contrastive predictive coding. In: ICML (2020) 4
13. Jain, A., Mildenhall, B., Barron, J.T., Abbeel, P., Poole, B.: Zero-shot text-guided object generation with dream fields. arXiv (2021) 4, 8
14. Jain, A., Tancik, M., Abbeel, P.: Putting nerf on a diet: Semantically consistent few-shot view synthesis. In: ICCV (2021) 1, 4, 8
15. Jiang, G., Kainz, B.: Deep radiance caching: Convolutional autoencoders deeper in ray tracing. Computers & Graphics (2021) 3
16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015) 18
17. Kuang, Z., Olszewski, K., Chai, M., Huang, Z., Achlioptas, P., Tulyakov, S.: NeROIC: Neural object capture and rendering from online image collections. arXiv (2022) 3
18. Lin, C.H., Ma, W.C., Torralba, A., Lucey, S.: Barf: Bundle-adjusting neural radiance fields. In: ICCV (2021) 1

19. Liu, L., Gu, J., Lin, K.Z., Chua, T.S., Theobalt, C.: Neural sparse voxel fields. In: NeurIPS (2020) 3
20. Ma, L., Stückler, J., Kerl, C., Cremers, D.: Multi-view deep learning for consistent semantic mapping with rgb-d cameras. In: IROS (2017) 3
21. Mascaro, R., Teixeira, L., Chli, M.: Diffuser: Multi-view 2d-to-3d label diffusion for semantic scene segmentation. In: ICRA (2021) 3
22. McCormac, J., Handa, A., Davison, A., Leutenegger, S.: Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In: ICRA (2017) 3
23. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV (2020) 1, 3, 4, 5, 9, 12, 18, 19, 21
24. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. arXiv (2022) 3
25. Niemeyer, M., Geiger, A.: Giraffe: Representing scenes as compositional generative neural feature fields. In: CVPR (2021) 3
26. van den Oord, A., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv (2019) 4
27. Ost, J., Mannan, F., Thuerey, N., Knodt, J., Heide, F.: Neural scene graphs for dynamic scenes. In: CVPR (2021) 1, 3
28. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable neural radiance fields. In: ICCV (2021) 3
29. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) NeurIPS (2019) 18
30. Pett, D.: BritishMuseumDH/moldGoldCape: First release of the Cape in 3D (2017). https://doi.org/10.5281/zenodo.344914 9, 10, 20
31. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. In: Meila, M., Zhang, T. (eds.) ICML (2021) 2, 4, 8
32. Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., Sutskever, I.: Zero-shot text-to-image generation. arXiv (2021) 4
33. Rebain, D., Jiang, W., Yazdani, S., Li, K., Yi, K.M., Tagliasacchi, A.: Derf: Decomposed radiance fields. In: CVPR (2020) 3
34. Rubinstein, M., Joulin, A., Kopf, J., Liu, C.: Unsupervised joint object discovery and segmentation in internet images. In: CVPR (2013) 1
35. Sara Fridovich-Keil and Alex Yu, Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: CVPR (2022) 1
36. Sitzmann, V., Martel, J.N., Bergman, A.W., Lindell, D.B., Wetzstein, G.: Implicit neural representations with periodic activation functions. In: NeurIPS (2020) 3
37. Sonoda, S., Murata, N.: Neural network with unbounded activation functions is universal approximator. Applied and Computational Harmonic Analysis (2017) 3
38. Stelzner, K., Kersting, K., Kosiorek, A.R.: Decomposing 3d scenes into objects via unsupervised volume segmentation. arXiv (2021) 2, 3
39. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3d shape recognition. In: ICCV (2015) 3

40. Takikawa, T., Litalien, J., Yin, K., Kreis, K., Loop, C., Nowrouzezahrai, D., Jacobson, A., McGuire, M., Fidler, S.: Neural geometric level of detail: Real-time rendering with implicit 3D shapes. In: CVPR (2021) 3
41. Tewari, A., Thies, J., Mildenhall, B., Srinivasan, P., Tretschk, E., Wang, Y., Lassner, C., Sitzmann, V., Martin-Brualla, R., Lombardi, S., Simon, T., Theobalt, C., Niessner, M., Barron, J.T., Wetzstein, G., Zollhoefer, M., Golyanik, V.: Advances in neural rendering. In: SIGGRAPH (2021) 3, 5
42. Tulsiani, S., Zhou, T., Efros, A.A., Malik, J.: Multi-view supervision for single-view reconstruction via differentiable ray consistency. In: CVPR (2017) 3
43. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017) 3
44. Vineet, V., Miksik, O., Lidegaard, M., Nießner, M., Golodetz, S., Prisacariu, V.A., Kähler, O., Murray, D.W., Izadi, S., Pérez, P., et al.: Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction. In: ICRA (2015) 3
45. Vora, S., Radwan, N., Greff, K., Meyer, H., Genova, K., Sajjadi, M.S.M., Pot, E., Tagliasacchi, A., Duckworth, D.: Nesf: Neural semantic fields for generalizable semantic segmentation of 3d scenes. arXiv (2021) 3
46. Wang, C., Chai, M., He, M., Chen, D., Liao, J.: Clip-nerf: Text-and-image driven manipulation of neural radiance fields. arXiv (2021) 1, 4, 8
47. Wu, S., Jakab, T., Rupprecht, C., Vedaldi, A.: Dove: Learning deformable 3d objects by watching videos. arXiv (2021) 2, 3
48. Yen-Chen, L.: Nerf-pytorch. https://github.com/yenchenlin/nerf-pytorch/ (2020) 3, 9, 18
49. Yen-Chen, L., Florence, P., Barron, J.T., Rodriguez, A., Isola, P., Lin, T.Y.: iNeRF: Inverting neural radiance fields for pose estimation. In: IROS (2021) 3
50. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelnerf: Neural radiance fields from one or few images. In: CVPR (2021) 3
51. Yu, H.X., Guibas, L.J., Wu, J.: Unsupervised discovery of object radiance fields. In: ICLR (2022) 2, 3
52. Zhang, C., Liu, Z., Liu, G., Huang, D.: Large-scale 3d semantic mapping using monocular vision. In: ICIVC (2019) 3
53. Zhang, K., Riegler, G., Snavely, N., Koltun, V.: Nerf++: Analyzing and improving neural radiance fields. arXiv (2020) 3
54. Zhi, S., Laidlow, T., Leutenegger, S., Davison, A.: In-place scene labelling and understanding with implicit scene representation. In: ICCV (2021) 3, 5, 7, 8, 18
55. Zhi, S., Sucar, E., Mouton, A., Haughton, I., Laidlow, T., Davison, A.J.: Ilabel: Interactive neural scene labelling (2021) 1, 2, 3
56. Zhu, J.Y., Wu, J., Xu, Y., Chang, E., Tu, Z.: Unsupervised object class discovery via saliency-guided multiple class learning. IEEE Transactions on Pattern Analysis and Machine Intelligence (2015) 1

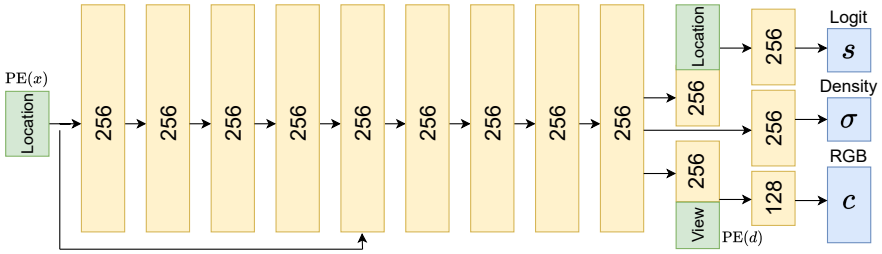

Fig. 7: The architecture used for LaTeRF is a multi layer perceptron which inspired by [54], extends the original NeRF model [23] to contain an additional output s to reason about the probability of different points in the space being part of the OOI. As evident in the figure, s is independent of the view d and only depends on the location x. Following the literature, both location x and view direction d are passed through a positional encoding function (PE).

A Additional Implementation Details

Our proposed method is implemented in Pytorch [29], and the network is optimized using the Adam optimizer [16] to learn the view-dependent radiance (color), and view-independent densities and objectness probabilities for points of the space. The network is randomly initialized and trained from scratch for each scene individually. Hyper-parameters related to the base NeRF model and the optimizer are set based on the Pytorch implementation [48] of the original NeRF paper [23]. The training is done on one Nvidia GeForce RTX 3090 GPU. Due to the high GPU memory usage of the calculations of the CLIP loss, the object renders used for computing $\mathcal{L}_{\text{CLIP}}$ are of $\frac{1}{3}$ of the size of the original input images, e.g., for 400×400 pixels synthetic views, object is rendered as 133×133 pixels images and the CLIP loss is defined over the downsized object renderings. Moreover, in order to speedup the training, the CLIP loss is only calculated every 10 steps. For the test time object renderings, instead of using the soft-partitioning approach, a hard-thresholding method is used to denoise the background. In the next section we introduce the details of this denoising mechanism. An overview of the architecture of the MLP with the additional objectness score s as output is shown in Figure 7.

For the real-world scenes, a minimal user-interface is designed to get the pixel annotations from an end-user. In this procedure, instead of limiting the user to give visual cues pixel-by-pixel, we allow brush-size changes to let users select areas corresponding to either the OOI or the non-objects. This allows to quickly annotate points far from the object boundaries resulting in a better non-object removal and object discovery in the scene. Using dynamic brush sizes, we were able to collect millions of pixel-level annotations in just a few minutes. Being able to change the brush size, the end-user can coarsely label the points that are not close to the boundaries, and then reduce the annotation area as they get closer

to the boundaries to finely label the more important pixel-level data around the intersections of the object and the non-objects (as shown in the experiments).

B Denoising the Views

As mentioned in section 5.1, we use a post-processing approach to denoise the rendered images of the objects. The noises are mostly caused by the small particles that emerge in the training of NeRF that blend in with the background from the training views, but become visible as the non-objects including the background are removed from the scene. In addition, for the points in the space that are inside a dense object or behind the background the objectness score is not trained well since the densities of the boundaries of this object are high and block the rays passing through these points. All these reasons contribute to have noisy renderings of the OOI when using LaTeRF without additional denoising (see soft threshold results in Fig 8). Our first step to mitigate this issue is to smooth the densities. Because the noisy particles are mostly steep bumps in density compared to their neighbouring points, substituting the value of every density with the average of their neighbours including themselves will make these bumps smoother. We repeat this averaging for 5 steps. After that, we filter the points with densities lower than a small threshold (which was set to 0.2 in the example in Fig 8). We call this approach hard thresholding and it is evident in Fig 8 that it has helped to reduce the noise, but still there are some unpleasant gray areas in the renderings. However, we don't directly use hard thresholding to render the RGB images of the object; We only use it to render the mask of the object to mask it out from the soft threshold results. After applying the hard threshold, we render the object mask by substituting the color with the objectness scores in Eq. 10 and applying the sigmoid function:

$$\hat{P}_{\text{obj}}(r) = \text{Sigmoid}\left(\sum_{i=1}^{N} T_i^{\text{obj}}\big(1 - \exp(-\sigma_i p_i \delta_i)\big) s_i\right). \tag{16}$$

Note that we assume a background with 0 objectness probablity when rendering the object so that only object points with high densities can dominate this background and we get the object mask as output. The rendered mask is then applied on the soft threshold results to give us rendered images of the object without background artifacts (final results are shown in Figure 8).

C Additional Real-world Results

Novel-view renderings of additional objects borrowed from [23,4] are shown in Fig. 9. These examples include a fortress model with lots of details (a forward-facing scene), and a metallic head statue with a shiny surface (a 360° inward-facing scene).

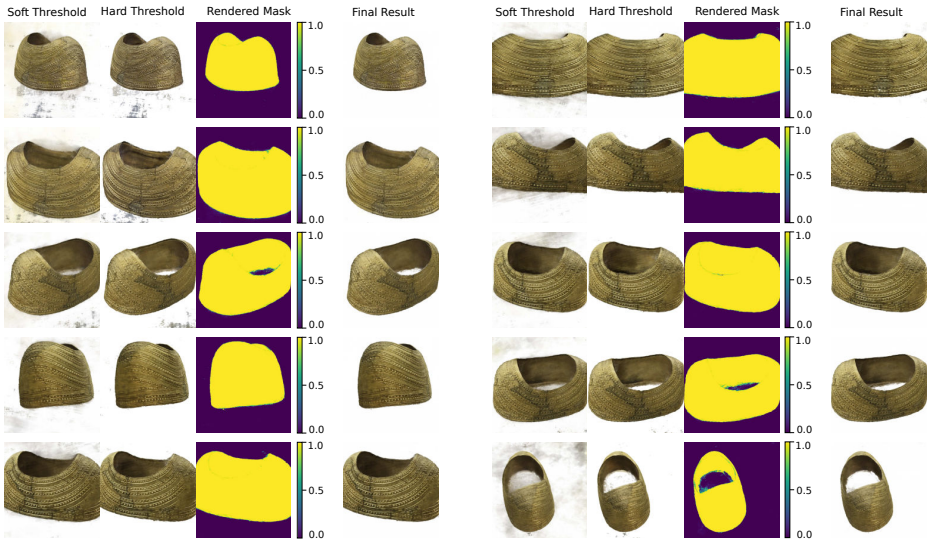

Fig. 8: More examples of the effectiveness of our denoising approach for removing the background artifacts using rendered objectness probabilities which act as object masks on the goldcape scene [30].

C.1 Real Examples with Occlusion

Figure 10 shows examples of using LaTeRF for extracting objects from 3 challenging scenes. The text queries used in these examples are "A dark green book", "A mug", and "A wooden ukulele". In the first example, a 3D model of the book is recovered, and in the next two examples, the pixels corresponding to the foreground objects are selected as negative samples to be removed, while keeping the object of interest and the background.

Fig. 9: Additional object extraction results.

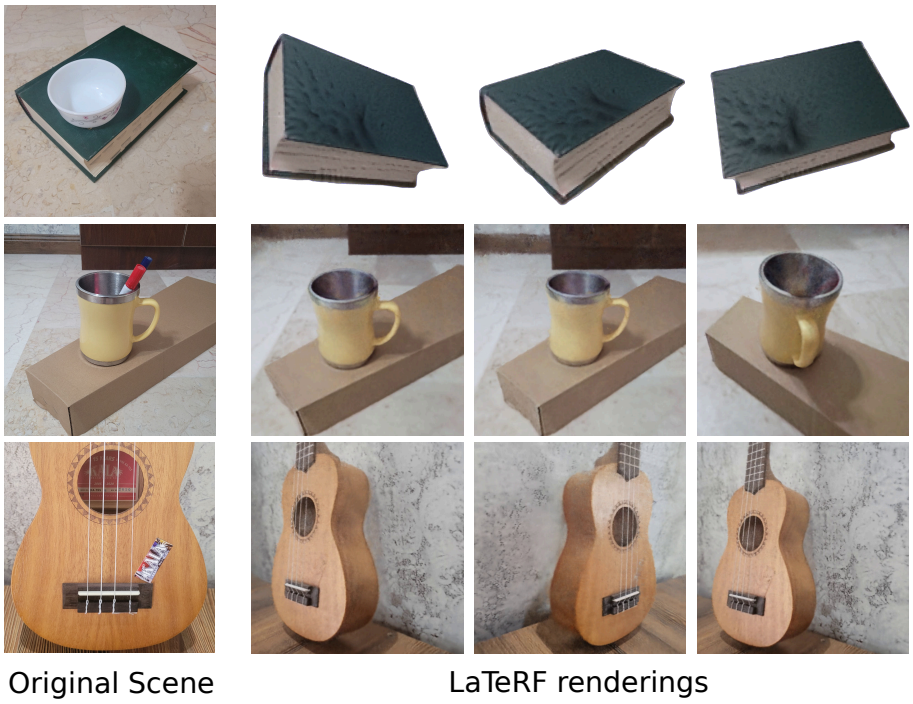

Fig. 10: Extracting occluded objects.

D Relighting the Object

It is possible to leverage the dependence of color of a point to the view direction and trick the learned object radiance field in order to render the OOI under novel lighting settings [23]. The view direction fed to the network to find the radiance (color) of points in the space can be manually rotated, while keeping the camera fixed. The effect caused by this approach is similar to changing the lighting of the scene, and it is possible to fit the novel lighting to be consistent with certain lighting conditions. Figure 11 shows some of the real-world objects under 3 different illumination settings from a fixed view.

D.1 Blending the Object in Novel Scenes

The lighting setting of the extracted objects can later be optimized with respect to the lighting in a novel scene, making the object look consistent in new scenes. Figure 12 shows an example of putting a 3D asset extracted by LaTeRF in a scene under two different illumination settings.

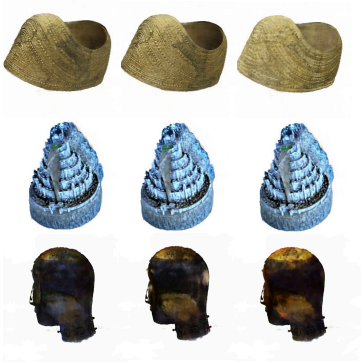

Fig. 11: Relighting objects under 3 different illumination conditions.

Fig. 12: An example of placing an extracted object in a scene with two different lighting conditions.