PROPOSAL

Devjyoti Chakraborty, Advait Sankhe, Shanwaz Waqar, Jaya Simha, Shanmukha Sai Jasti

Devjyoti.Chakraborty@uga.edu, Advait.Sankhe@uga.edu,

ABSTRACT

In recent years, Neural Radiance Fields (NeRF) have emerged as a pioneering paradigm for scene reconstruction and 3D modeling from 2D images, showcasing their remarkable capability to faithfully capture complex scenes. However, NERF models suffer from limited editability, posing challenges in interactive 3D content creation and manipulation. This project aims to address this issue by investigating recent neRF methods to enhance the editability of NeRF-based 3D representation spaces, with the following key objectives: Scene Editing for developing methods for users to intuitively modify scene attributes, including object placement, appearance, and lighting, within the NERF-derived 3D space, Semantic Understanding for creating techniques to infer and manipulate the underlying semantics of objects and elements in the volumetric representation, and Implementing responsive feedback mechanisms to enable users to observe the immediate impact of their edits, streamlining the creative workflow. We introduce a novel pipeline which leverages concepts from various papers and gives us a transparent accessible method to make edits to the 3D scene learned by the NeRF volumetric function, as well as individually siphon out the objects in the scene and edit them at will.

1. INTRODUCTION

In recent years, the field of computer vision and novel view synthesis has witnessed a transformative shift with the advent of Neural Radiance Fields (NeRF). NeRF-based models have revolutionized our ability to reconstruct complex 3D scenes from 2D images with unprecedented accuracy and fidelity. These models show remarkable aptitude in learning the volumetric representations of scenes, capturing intricate details of geometry and appearance.

While the success of NeRF in scene reconstruction is undeniable, an innate limitation has emerged: The editability of the 3D representation space learned by the volumetric function has become a central challenge in the field of computer vision and 3D modeling. NeRF models have proven to be adept at faithfully capturing the intricacies of real world scenes, enabling the creation of stunningly realistic digital representations. However, the research towards achieving a semantically reinforced 3D representation space which can allow us to make edits directly in the 3D space itself has been very limited.

Recently, there has been a significant dive into the study of interpretability and manipulation of the 3D representation space learned by neRF volumetric function. Our project aims to dive into the current frontrunners in this sphere, reproduce their results evaluating the feasibility within the constraints of our computational resources. Through this rigorous investigation, we aim to pinpoint any limitations or drawbacks in the existing methodologies. Furthermore, we seek to identify opportunities for enhancement, striving to propel the ongoing research in this field to new heights

Currently, we have identified "NeRF-Editing: Geometry Editing of Neural Radiance Fields" and "NaviNeRF: NeRF-based 3D Representation Disentanglement by Latent Semantic Navigation" as the most promising direction of our research. NeRF-Editing is a method for editing the geometry of 3D scenes represented by Neural Radiance Fields (NeRFs). It works by first establishing a correspondence between the explicit mesh representation and the implicit neural representation of the target scene. Users can then utilize well-developed mesh-based deformation methods to deform the mesh representation of the scene. NeRF-Editing then utilizes user edits from the mesh representation to bend the camera rays by introducing a tetrahedral mesh as a proxy, obtaining the rendering results of the edited scene. To our knowledge, this algorithm has a very transparent workflow which can go on to help in interpretability. The possible areas of improvement as noted by us are the failure when faced with complex scenes, or while dealing with dynamic scenes. We have proposed a possible solution for this problem in the hypothesis section.

NaviNeRF, or NeRF-based 3D Representation Disentanglement by Latent Semantic Navigation, is a method for disentangling the underlying factors of 3D data represented by Neural Radiance Fields (NeRFs). It does this by introducing a self-supervised Navigation branch that identifies interpretable semantic directions in the latent space of the NeRF. NaviNeRF is also equipped with an Inner Refinement Branch that refines the disentangled latent representations to produce high-quality renderings.

One thing to note is that none of the papers deal with the editing of objects separately. We introduce a novel pipeline that accomplishes the requirements of editing the 3D space as well as being able to isolate the objects in the 3D space

to edit them individually. We believe our work can help the world of 3D scene editing and make NeRF more transparent and accessible to researchers.

2. LITERATURE REVIEW AND BACKGROUND

2.1. NeRF

The base NeRF (Neural Radiance Fields)[1] paper introduced a novel way of reconstructing 3D scenes given a sparse set of input views (2D images). A fully connected (non-convolutional) neural network maps a continuous 5D coordinate input (spatial location (x, y, z) and viewing direction (θ, ϕ) to an output consisting of emitted radiance and volume density $f((x, y, z), (\theta, \phi)) \longrightarrow ((r, g, b), \rho)$ where ρ is the volume density at the point in question. The model requires data consisting of a set of RGB images of a scene along with corresponding camera poses, intrinsic parameters and scene bounds. A key takeaway of this paper is the method of representing scenes as radiance fields, allowing reconstruction of fine-grained details in 3D scenes. Conceptually, neRF can be imagined as a 3D mesh cuboidal mesh, where each side is a camera plane from where rays are shot into the 3D space of the cube. We sample points on those rays with their location and directional, leveraging the color and volumen density at those points. Improving interpretability of the representations is a clear next step for obtaining better results. Neural Radiance Fields acts as the basis of our research.

2.2. NeRF-Editing

NeRF-Editing [2] extends the capabilities of modifying the radiance fields from simple translation/rotation tasks to user-controlled shape deformations. This is done by establishing correspondence between the explicit mesh representation of the objects and their implicit neural representations, without needing to re-train the whole network. A triangular mesh representation that can be extracted from the radiance field is used to enable the user to introduce deformations using the ARAP (as-rigid-as-possible) method. Further, tetrahedral meshes are used to cover the extracted triangular meshes and are updated accordingly. Tetrahedral meshes allow for smoother, more continuous representations of the deformed objects. Within the new deformed space, views are synthesized by casting rays to the space containing this deformed tetrahedral mesh. Ray-bending is done using the displacement of vertices from the original tetrahedral mesh to the deformed mesh. Thus, the implicit representation can be 'deformed', allowing higher degrees of freedom for editing objects within a scene. This paper is currently going through a investigation by our team, as it is a forerunner as a choice for the model we will be choosing to continue our research with.

2.3. Seal-3D

Seal-3D [3] introduces pixel-level editing for 3D representations with interactive tools like the bounding box, brush, anchor and color tools that are commonly seen in 2D image editing software. Instead of relying on a mesh proxy, the model proposes the use of a proxy function to introduce correspondence between the explicit edits and the implicit neural representation. To enable local editing, a two stage pre-training =¿ fine-tuning process is proposed, which allows faster previews of edits and prevents global contamination. The pre-training stage updates positional embedding grids with local losses, while the fine-tuning stage takes into account the global photometric losses as well.

2.4. NaviNeRF

NaviNeRF [4], or NeRF-based 3D Representation Disentanglement by Latent Semantic Navigation, is a method for disentangling the underlying factors of 3D data represented by Neural Radiance Fields (NeRFs). It does this by introducing a self-supervised Navigation branch that identifies interpretable semantic directions in the latent space of the NeRF. NaviNeRF is also equipped with an Inner Refinement Branch that refines the disentangled latent representations to produce high-quality renderings. NaviNeRF works by first learning a NeRF representation of the input 3D data. Then, the Navigation branch is used to identify interpretable semantic directions in the latent space of the NeRF. This is done by training the Navigation branch to navigate through the latent space and identify different semantic factors, such as the shape, material, and color of objects in the scene. Once the Navigation branch has identified the interpretable semantic directions in the latent space, the Inner Refinement Branch is used to refine the disentangled latent representations. This is done by training the Inner Refinement Branch to produce high-quality renderings from the disentangled latent representations. This paper is conceptually the most intriguing paper we have come across. However, this is computationally very expensive.

3. BACKGROUND

Neural Radiance Fields (NeRF) can be defined as a continuous volumetric function that predicts the RGB color c and the volume density σ of a scene at a given 3D point X (spatial location x, y, z) and viewing direction d (θ, ϕ):

$$F : (X, d) \rightarrow (c, \sigma) \tag{1}$$

The NeRF pipeline consists of input images and corresponding camera poses. Directional rays are extended from the camera into the 3D canonical space, which intersect the image plane. The 3D point coordinates on the canonical space are obtained by sampling points on these directional rays.

Each ray r is defined as $r(t) = o + td$, where o is the origin and d is the directional vector of the ray. t represents the points that are sampled along $r(t)$. The color $c(r)$ of a ray is given by:

$$c(r) = \sum_{i=1}^{N} T_i \alpha_i c_i \tag{2}$$

Where T_i, α_i, and c_i are the transmittance, opacity, and color of the i_{th} respectively. The color of a point is weighted by the effect of transmittance T_i (Probability of light reaching the point) and α_i (probability of light being absorbed or scattered a the point). Both α and T are functions of volume density σ. The discrete weighted color values are summed up to obtain the color of a ray. The predicted colors are compared with the ground truth image pixel at which a ray intersects the image plane. The loss function becomes:

$$\sum_{r \in R} \|c(r) - c_{gt}(r)\|_2{}^2 \tag{3}$$

With this in mind, let's take a look at the methodology.

4. METHODOLOGY

Our first step will be to convert the implicitly learned 3D volume of the scene into an explicit representation. For that, we use the marching cubes algorithm (Figure 1). The Marching Cubes algorithm creates sub-surfaces based on the threshold criterion for vertices of voxels that lie in the 3-D space. Each vertex has a scalar value assigned to it, along with the coordinates in the 3-D space. The trained NeRF model is used to determine the volumetric density of sampled points in the scene at a fixed angle. These scalar volume density values are passed to the Marching Cubes Algorithm, implemented in the PyMCubes library, to obtain the representation of all objects in the scene as a single mesh. The threshold is set at a low value of 50 to account for minor inaccuracies in the volumetric function. The vertices of this mesh form the singular point cloud representation of the objects.

Our method will work for any index based 3D structures. For simplicity, the rest of the methodology will assume the extracted mesh is in point cloud format.

Let $X*$ be the collection of vertices in the point cloud. Since marching cubes does not return the RGB values, it is very difficult to distinguish the vertices belonging to the different objects in the scene. A key point part of complete transparency of the 3D space learned by the NeRF function is to be able to exactly map out where each object lies.

Let there be n ranges in which separate objects lie in the point cloud $X*$. The next part of our objective is to be able to isolate each object and apply our desired edits on them. To accomplish that, first we will take the help of the Segment-Anything-Model to isolate the objects from our training images. Meta's Segment Anything Model was

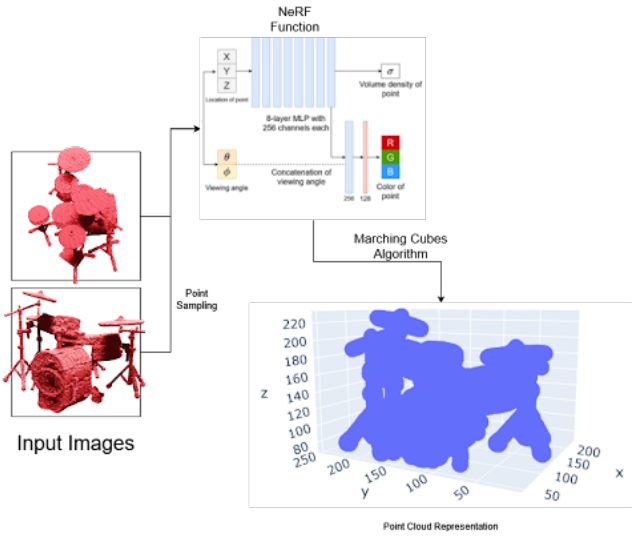

Fig. 1. First phase

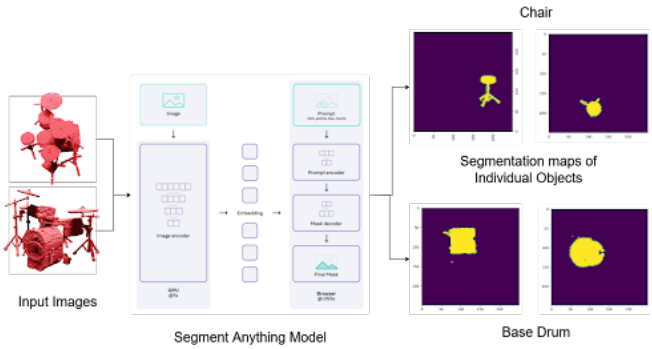

Fig. 2. Second phase

used to obtain segments of objects of interest. The Masked encoder pre-trained Vision Transformer structure allows zero-shot segmentation of images. As it is trained on a dataset of nearly 11 million images and 1.1 billion masks, it performs well on segmentation tasks of any type of images. SAM allows prompt-based segmentations which can include points, grids, masks, etc. 2-4 images were chosen from the training dataset that provided distinct, unobstructed views of the object. Image and point selection for the objects was done manually. Of the multiple masks obtained per image with high confidence scores, the most appropriate one, based on human judgment was selected. The publicly available ViT-H SAM model was used.

We take advantage of the fact that the training images for NeRF has views of the scene from multiple angles, enabling us to choose the images which are close to a $90degs$ horizontal or $90degs$ vertical views.

It is important to make sure that the extracted segmented maps have at least one clear view of the object that we wish to isolate in the 3D space. For example, we have chosen to

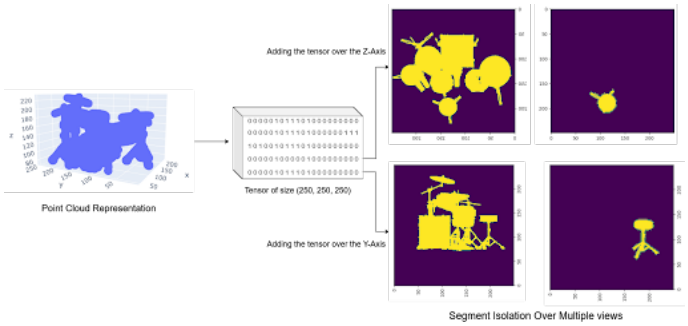

Fig. 3. Third phase

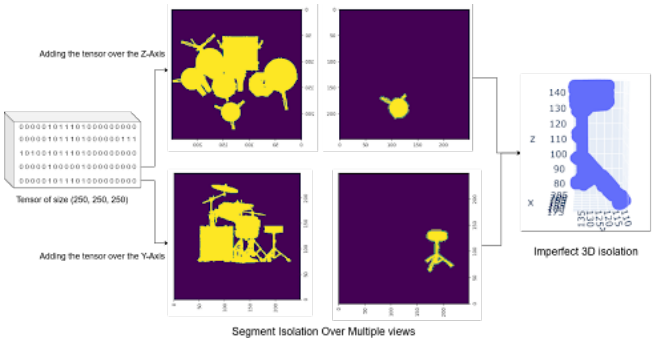

Fig. 4. Forth phase

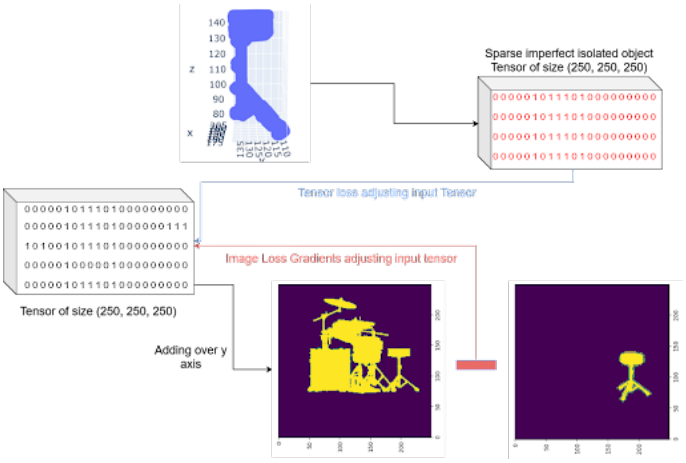

Fig. 5. Fifth phase

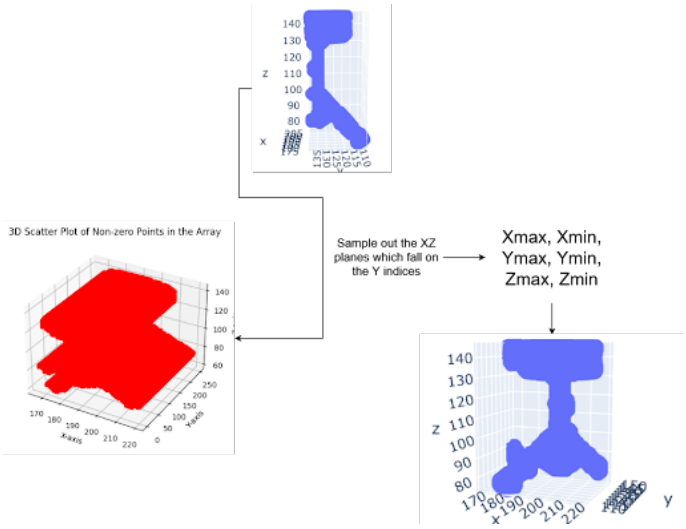

Fig. 6. Final phase

isolate the drum seat for demonstration purposes.

From the obtained segmentation maps, we will choose $>= 1$ number of maps each from the horizontal maps and the vertical maps.

We convert the point cloud $X*$ to a tensor X where the indices with vertices have 1 and and non index values are 0. We take a 2D segmentation map and pass it through the tensor, keeping the vertices with which the segmentation map intersects. Since this can also lead to inclusion of the vertices belonging to other objects. For this reason we repeat the process for both vertical and horizontal maps, keeping only the vertices which are common in both the operations.

However, as you can see in Figure 4, this yields an imperfect isolation. That is because realistically it is not possible to get perfectly horizontal and vertical segmentation maps which line up with the point clouds. We discuss this further in the Future Scope section.

To address this, we will include one additional step (Figure 5).

We introduce an optimization problem for perfecting the isolation. First, we convert the imperfect isolated object into a tensor of the same size as our input. Adding up the input tensor over the y axis will give us a horizontal view which will be compared to the horizontal segmentation map. The gradients from the loss will be used to alter the input tensor. However, this might lead to a situation where the gradients turn our input tensor into a stretched out version of the segmentation map itself. To counter it, we will be adding another component in the loss; the imperfect isolated tensor. We will adjust the effect of this by a factor of β. Given the input Tensor $X*$, the imperfect isolation tensor $z*$, segmentation map $y*$. the loss function becomes:

$$L_{total} = (X_y - y*) + \beta(\sum(X - z*))$$

where X_y is the input tensor added over the y axis. We keep the value of β at 0.25.

For a final sanity check, we filter the y axis vertices obtained from the optimization problem and keep only the ones that appear in the imperfect isolation tensor to obtain the bounding range of the object in 3D space. The edited point cloud was re-rendered in Blender. Nearly 2 million mesh primitives (icospheres) were attached to the points. The material was given color and made reflective by using

the principled BSDF shader with high metallic and specular properties. A light source (sun) was added to the scene at an angle of 11.4 degrees. This edited scene was then used to generate a new dataset, using the BlenderNeRF addon. This package helped capture images at a random camera angle while logging the respective camera transforms. The edited mesh is placed at the center of a sphere, while the camera faces the center from random points in the upper hemisphere. A NeRF model was then successfully trained using this synthetic dataset of the edited objects.

5. EXPERIMENTAL SETUP

We will be using the AI institute workstations with the NVIDIA RTX A5000 GPUs for our experiments. These machines should have sufficient computational capabilities to run our experiments efficiently. We will be using common ML/DL/image processing libraries like PyTorch, torchvision, scipy, tensorboard,etc along with 3D object editing software like Blender. We have used the NeRF synthetic drum set as our dataset of focus.

6. RESULTS

We show the effect of our pipeline by affecting the 3D space and editing specific objects like the drum seat and the bass drum (Figure 7 and Figure 8). Quantitative analysis shows that our obtained bounding box accuracies differ based on the object in question. Since the drum seat is not surrounded by many objects, the manually annoted bounding box label and the predicted box label has a similarity score of 0.87. On the other hand, since the bass drum is surrounded by many objects, our pipeline yields a similarity score of 0.73 based on manually annotated bounding box. This shows that our pipeline can be made better in terms of dealing with objects which is in a fairly crowded area of the 3D scene. Another thing to note is that our final renderings don't have the original RGB color. That is because Marching cubes doesn't give us the color, only the location of dense points.

7. FUTURE WORK

Obtaining the 3D scene makes the Implicitly learned volumetric information Explicit and transparent, which enables us to make easy and seamless real-time edits. However, isolating all objects present in the scene still needs some work to be done to obtain a 100 percent similarity score to the target labels. Also, tracing the edits back to the NeRF function can save us the hassle of retraining the editing 3D space to obtain rendering. This is currently being worked on and can be a groundbreaking contribution to NeRF and 3D scene editing.

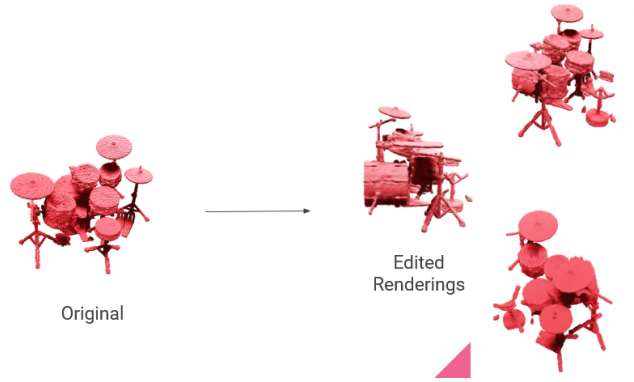

Fig. 7. The renderings of the drum seat turned upside down

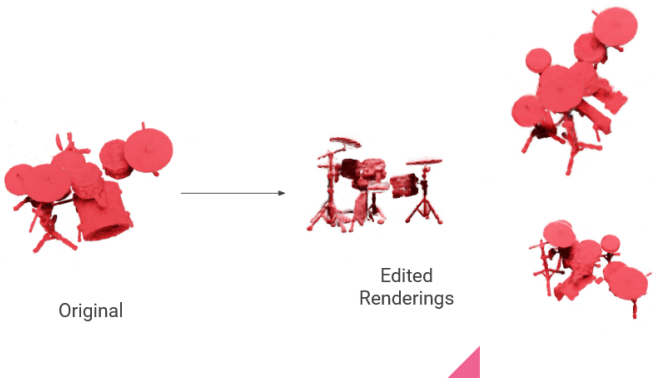

Fig. 8. The renderings of the bass drum removed from the 3D space

8. REFERENCES

[1] 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," .

[2] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao, "Nerf-editing: Geometry editing of neural radiance fields," .

[3] Xiangyu Wang, Jingsen Zhu, Qi Ye, Yuchi Huo, Yunlong Ran, Zhihua Zhong, and Jiming Chen, "Seal-3d: Interactive pixel-level editing for neural radiance fields," 2023.

[4] Baao Xie, Bohan Li, Zequn Zhang, Junting Dong, Xin Jin, Jingyu Yang, and Wenjun Zeng, "Navinerf: Nerf-based 3d representation disentanglement by latent semantic navigation," 2023.