Space Carving and Neural Radiance Fields for 3D Segmentation

Irmak Sivgin
EE, Stanford University

Abstract—We propose a practical and memory efficient method for obtaining 3D segmentation models of objects from a multi-view image dataset. Initially, 2D segmentation masks are generated by Meta AI's Segment Anything Model, which is a well established, open-source segmentation model. Then, space carving (SC) is performed using camera position matrices and boolean segmentation masks, yielding a reference voxel grid holding occupancy values corresponding to the camera-hull of the object. We then use these occupancy labels to supervise a neural radiance field (NeRF) model training. We compare our proposed method which combines SC and NeRF to SC, and NeRF supervised on 2D image space (i.e, loss function defined between the test view image and the predicted rendering). We find that our method achieves the same peak-signal-to-noise-ratio (PSNR) as SC, outperforming the conventional NeRF performance by more than 24 dB. Further, we are able to represent the 3D scene with only 0.5% of SC parameters, achieving memory efficiency and high data-fidelity at once. We also provide an analysis of robustness by introducing different levels of Gaussian blur to the original dataset, and show that our method can still replicate the PSNR value of SC, proving to be a robust method.

Index Terms—Computational imaging, space carving, neural radiance fields

1 Introduction

This project aims to provide a practical and memory efficient method for 3D segmentation of objects. For this aim, we utilize 3 main methods: Meta AI's Segment Anything Model (SAM), which is a well established, open-source segmentation model that can generate masks for different objects given an RGB image [1], space carving (SC), a traditional iterative algorithm for finding the photo hull of a scene given different views of the same scene [2], and neural radiance fields (NeRFs), providing an implicit representation of the 3D scene from multiple images at different camera positions [3]. With this project, we motivate the use of SAM, SC and NeRFs together to lift 2D segmentations that can easily be obtained without any training procedure to a 3D volumetric representation. Having the 3D segmentation mask of any object within a scene can be useful for many applications such as robotics, where the robot needs to model the object it is interacting with to manipulate it in a desired manner such as grasping [4], or for object collision avoidance [5].

2 Related Work

The task of inferring the 3D scene from a given set of 2D images acquired from different camera positions comes up in many fields such as computer graphics and vision. There have been 2 main methods for the representation of the underlying volumetric scene, explicit and implicit. Here, we consider space carving and neural radiance field methods for explicit and implicit representations, respectively.

2.1 Space carving

Space carving (SC) is a traditional method based on [2], which iteratively constructs the camera hull of an object from multiple views of the scene. Starting with a volume containing the scene, the algorithm removes non-photo-consistent voxels at each iteration. This method yields reliable and consistent representations of the scene, however scales with $\mathcal{O}(N^3)$ where N determines the grid resolution.

2.2 Neural Radiance Fields

Neural radiance fields (NeRFs) have emerged as the state-of-the-art methods for scene representation and novel view synthesis, being popularly applied in computer graphics [6]. Mildenhall et al. used the fully-connected network to represent scenes and optimized neural radiance fields to synthesize novel views of scenes with complicated geometry and appearance [3]. Compared to explicit volume grid representations, implicit representation of a scene through the parameters of a neural network can drastically reduce the memory requirement. However, the training procedure based on gradient descent is slow and challenging since the problem is ill posed.

2.3 3D segmentation

It is also possible to learn the object segmentations on the 3D volume representation directly, instead of mapping a 2D segmentation to a volumetric representation. In [7], authors present NeSF for obtaining 3D semantic fields from a given dataset of multiple camera poses and their associated images. They use implicit neural representations to parameterize the 3D scene, and carry out training of a 3D semantic segmentation model on this representation. Although the supervision of the 3D segmentation model is based on 2D space, this approach is much more challenging compared to starting with segmentations and mapping them to 3D.

3 Proposed Method

3.1 Problem Statement

We wish to get 3D segmentation maps of different objects given multiple photos of a single scene with known camera positions. We will denote the dataset as $\mathcal{D} = \{S_i, P_i, i = 1, \ldots, N_d\}$ where $S_i \in \mathbb{R}^{n \times n \times 3}$ is the i-th image of the scene, associated with camera pose matrix $P_i = \begin{bmatrix} R_i & t_i \\ 0 & 1 \end{bmatrix}$ with rotation matrix $R_i \in \mathbb{R}^{3\times3}$, translation vector $t_i \in \mathbb{R}^3$. We would like to lift object segmentations to 3D space, thus, we will use well established open source segmentation models on S_i to get boolean valued object masks M_i. For this project, the choice was to use Meta AI's Segment Anything Model (SAM) [1]. As the reader will recognise, this is an ill-posed problem with many consistent 3D segmentations matching the 2D data at hand. Specifically, any pixel with value 1 (indicating where the object lies), can indicate the presence of occlusion by the object at any coordinate value lying on the ray pointing towards that pixel. It will become clear how we address this challenge in section 3.3.

3.2 Compared Methods

3.2.1 NeRF

We can simplify the NeRF framework proposed in [3] as the images have been transformed into boolean masks. The network parametrizes the function $f_\theta(x) : \mathbb{R}^3 \mapsto \mathbb{R}$ which makes a scalar occupancy prediction for a given coordinate point x. We have also included a positional encoding step before querying the network on different points [3], [8]. This can be expressed by

$$\phi(x) = [\sin(2^0\pi x), \cos(2^0\pi x), \ldots, \sin(2^{L-1}\pi x), \cos(2^{L-1}\pi x)] \quad (1)$$

where L denotes the encoding dimension. The network is a fully-connected multi-layer-perceptron (MLP) with 3 layers using ReLU activation function:

$$f_\theta(x) = {W_3}^{\mathbf{T}}\text{ReLU}({W_2}^{\mathbf{T}}(\text{ReLU}({W_1}^{\mathbf{T}}x + b_1)) + b_2) + b_3 \quad (2)$$

Rendering and Image Loss: For a given camera, we have the ray $r(z) = o + zd$ sampled on depth values $z_{near} \leq z \leq z_{far}$, with vectors o, d denoting the origin and direction. o is independent of the pixel coordinate of a given image, while d varies for each pixel. Then, we can render the predicted image Im_{Nerf} at a certain camera angle as:

$$Im_{Nerf} = \frac{1}{n_z}\sum_z f_\theta(\phi(r(z))) \quad (3)$$

where n_z is the number of samples along each ray. We choose the loss function as mean squared error between the rendered image and the ground truth mask.

3.2.2 Space Carving

Space carving implementation for boolean masks M_i is achieved as follows: The initial reference volume grid $V \in \mathbb{R}^{N\times N\times N}$ has each voxel value set to 1 (i.e, all of the volume is assumed to be filled). We infer the range of coordinates corresponding to each voxel of the reference grid as $D = \frac{1}{N_d}\sum_i ||t_i||_2$. Then, for each M_i, P_i pair, ray sampling $r(z) = o + zd$ is done for 0-valued pixels. Points $r(z) = x$ are interpolated to their respective coordinates within V by $k = \lfloor \frac{x + \frac{D}{2}}{D} N \rfloor$. For those k that constitute a valid index of V, carving is performed by setting respective voxels to zero. After all images have been iterated over, carved volume $\hat{V}$ containing labels $\hat{V}_{i,j} \in \{0, 1\}$ of each voxel the scene has been divided into, is returned. This algorithm is outlined in algorithm 1.

Image Rendering: We can render the predicted image Im_{SC} at a certain camera angle, similar to the rendering procedure of NeRF, by:

$$k = \lfloor \frac{r(z) + \frac{D}{2}}{D} N \rfloor \quad (4)$$

$$Im_{SC} = \sum_k \hat{V}[k] \quad (5)$$

where we check the range of k before indexing $\hat{V}$. It is worth noting that, if all views of the scene as used, the SC reconstruction of any (known) view angle is exact (we can refer to this as the "ground truth" SC). We leave out some perspectives for testing (same as NeRF train-test split), and compare the SC rendering on a test angle to the ground truth SC of that angle.

Algorithm 1 Space carving

```
given D_train = {M_i, P_i, i = 1, ..., N_train}
V ← 1^{N×N×N}
for M_i, P_i ∈ D_train do
    for x ∈ [1, H], y ∈ [1, W] s.t. M_i[x, y] = 0 do
        ray sampling: r(z) = o + zd
        interpolation: k = ⌊ (r(z) + D/2) / D  N ⌋
        if 0 ≤ k ≤ N then
            V[k] ← 0
        end if
    end for
end for
return V
```

Projections of the SC volume $\hat{V}$ on different x, y, z slices are depicted on Fig. 1, where we can observe different parts of the object at varying depths.

3.3 Proposed Method

Recognising the strengths of both SC and NeRFs, we propose to merge these methods to get the best of both, obtaining a memory-efficient representation with improved data fidelity. For this end, we utilize SC labels $\hat{V}$ to supervise NeRF training, which introduces a 3D loss instead of mean squared error on 2D images. By the implementation of SC described in section 3.2.2, we ensure coordinate conventions of both methods to be compatible. The training takes the SC labels $y \in \hat{V}$ as well as the reference grid $ref_{grid} \in \mathbb{R}^{N\times N\times N\times 3}$ comprising of coordinate points x corresponding to labels to query the NeRF model. For a batch of points x, we calculate the encodings $\phi(x)$ and perform a forward pass on the model. The loss term to be backpropagated is the mean squared error between y and

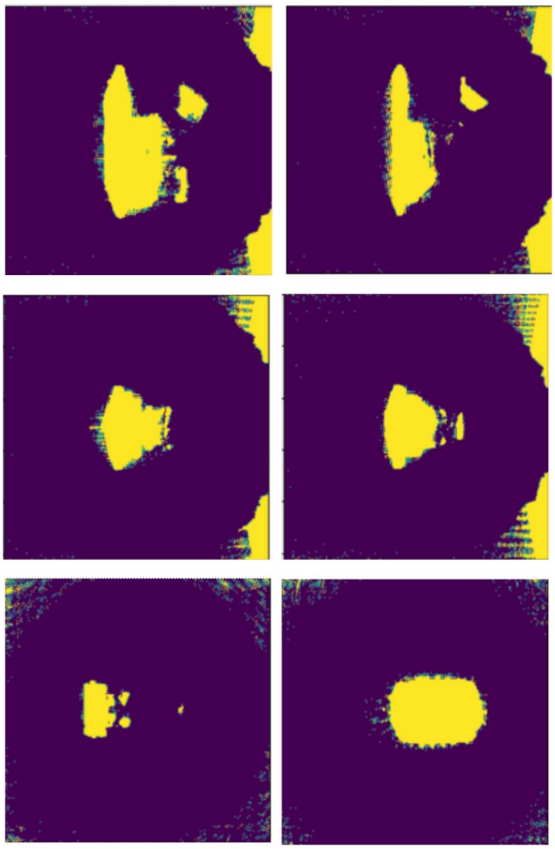

Fig. 1. Example two-column figure.

Algorithm 2 SC supervised NeRF training

given $\hat{V}$ **from SC**
$k \leftarrow [-\frac{D}{2}, -\frac{D}{2} + \frac{N-1}{D}, \ldots, \frac{D}{2}]$
$l \leftarrow [\frac{D}{2}, \frac{D}{2} - \frac{N-1}{D}, \ldots, -\frac{D}{2}]$
$m \leftarrow [-\frac{D}{2}, -\frac{D}{2} + \frac{N-1}{D}, \ldots, \frac{D}{2}]$
$ref_{grid} \leftarrow$ meshgrid(k, l, m)
for i, y **in enumerate**$(\hat{V})$ **do** ▷ y is the label
  coordinate: $x \leftarrow ref_{grid}[i, :]$
  encoding: $x \leftarrow \phi(x)$
  $\hat{y} \leftarrow f_\theta(x)$
  loss $\leftarrow ||y - \hat{y}||_2$
  backprop (loss)
end for

$f_\theta(\phi(x))$ over the batch. We outline the proposed procedure in algorithm 2.

This approach makes the problem less ill-posed by providing supervision on more points on the 3D space, thus, improving reliability. Since the neural network learns the SC representation with less parameters, it is a means of compressing the memory-heavy SC representation.

Image Rendering: Image rendering is achieved similar to that of SC. We can render the predicted image $Im_{SC-NeRF}$ at a certain camera angle by:

$$k = \lfloor \frac{r(z) + \frac{D}{2}}{D} N \rfloor \tag{6}$$

$$Im_{SC-NeRF} = \sum_k f_\theta(\phi(k)). \tag{7}$$

4 Analysis, Evaluation, Comparison to other methods

We conduct all our analyses using the TinyNeRF Dataset [1] with 100 training images and 6 test angles. We get boolean masks M_i by getting the "background" mask from SAM, and performing a logical not operation. The reason for this choice of masking is that SAM model parses the object itself into different parts such as the bulldozer bucket, the rectangular plate on the bottom. Merging these different masks is not straightforward since different view angles yield to different numbers of masks per image. Noting that there was only one object of interest in the scene, taking the background mask gives accurate masks for our pipeline.

4.1 NeRF Implementation Details

The vanilla NeRF relying on 2D image loss and SC supervised NeRF models share the same forward model as stated in Eq. 2. The hidden size is selected as $h = 256$. Positional encoding dimension is also kept the same ($L = 10$) for fair comparison.

4.2 Evaluation Metric

We use peak-signal-to-noise-ratio (PSNR) as the evaluation metric across different methods. PSNR is calculated by:

$$PSNR = 10\log_{10}(\frac{1}{\frac{1}{HW}(Im - GT)^2}) \tag{8}$$

where H, W are image dimensions, Im is the computed image, GT is the reference (ground truth) image, and both images have been scaled to $0 - 1$ pixel values.

4.3 SC Parameter Selection

There are two parameters for SC: resolution of the grid N and the scale D. We compare 4 different values of both, and record PSNR values compared to ground truth SC. Figures 2 and 3 display the PSNR trend while sweeping over N and D respectively.

We observe when the grid resolution is too coarse, the representation does not capture essential details. Thus, PSNR is improved by increasing N up to 256. However, beyond that, ray samples taken for each pixel interpolated to the reference grid coordinates leave too many voxels untouched, which cause the "cloudy" artifact seen in Fig. 4 when rendering at an unseen view angle.

Including the full scale yields the best volumetric representation, thus, D is kept at the value calculated from translation vectors. Qualitative results comparing SC rendered images at plotted values of N, D are provided in Figs. 4, 5.

1. http://cseweb.ucsd.edu/~viscomp/projects/LF/papers/ECCV20/nerf/tiny_nerf_data.npz

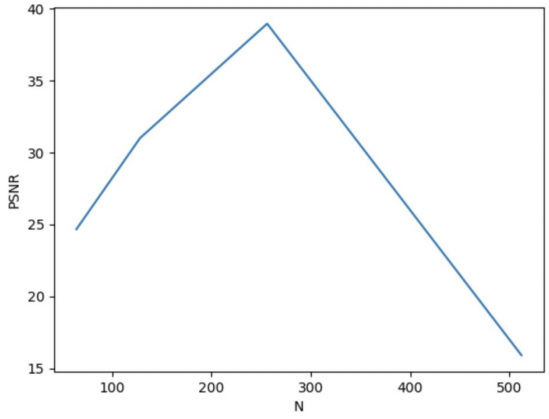

Fig. 2. PSNR change wrt. N, D is fixed.

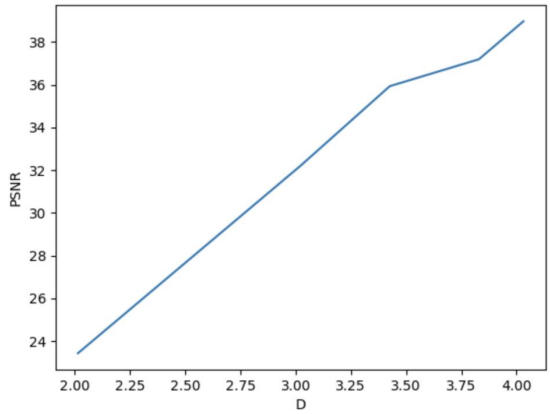

Fig. 3. PSNR change wrt. D, N is fixed at 256.

4.4 Robustness Analysis

We study the robustness of our pipeline by applying different levels of Gaussian blur (i.e Gaussian low-pass filter) on images S_i. Then, the workflow is illustrated by Fig. 6.

The resulting SC and SC & NeRF reconstructions are exemplified on Fig. 7. We also note that the edges of the SAM masks get smoother with increasing σ, which also limits the resolution of SC. One interesting effect that is observed consequent to Gaussian low-pass filtering is that the training of SC supervised NeRF is sped up. Specifically, the model is able to fit to $\hat{V}$ in 2.5% of the iterations needed to train vanilla NeRF.

5 Results

Here, we discuss the effectiveness of our proposed method by referring to PSNR, memory efficency and GPU runtime comparisons. We further eloborate on the robustness analysis with quantitative results.

5.1 Performance Comparison

Table 1 summarizes PSNR values, number of parameters and runtime of SC, NeRF, and SC supervised NeRF methods.

TABLE 1
Peak-signal-to-noise-ratio (PSNR), memory requirement and runtime of the compared methods.

	NeRF	SC	SC & NeRF
PSNR (dB)	14.49	38.97	39.38
Number of parameters	82433	16777216	82433
Runtime (s)	12000	3.8	3300

TABLE 2
PSNR comparison

Gaussian Blur	SC	SC & NeRF
No blur	38.97	39.38
$\sigma = 0.5$	28.13	28.37
$\sigma = 0.75$	27.630	27.999
$\sigma = 0.9$	27.627	27.997
$\sigma = 1$	27.54	27.96

We observe that SC supervised NeRF achieves slightly better PSNR compared to SC, although the difference is not major. The main advantage of this method becomes apparent on the comparison of the number of parameters: While SC has N^3 parameters, the parameters needed to represent f_θ is slightly less than 0.5% of that of SC, which enables a huge compression! Comparing the NeRF to both SC and SC supervised NeRF, we observe inferior data fidelity (significantly lower PSNR) and longer training time. Performing one iteration with 2D and 3D losses takes the same amount of time, however, using SC labels improves convergence time as less number of iterations are actually sufficient to fit the model to $\hat{V}$. Specifically, NeRF needed to be optimized for 10000 epochs while SC & NeRF was able to converge in 3000 epochs. The validation PSNR of SC & NeRF at different epochs, along with the end reconstruction and the ground truth, is displayed in Fig. 8.

5.2 Robustness Results

We evaluate the robustness of our proposed pipeline to different levels of Gaussian blur introduced to the image dataset. A drop in PSNR is observed as σ is increased, which is expected. In all test cases, SC supervised NeRF model is able to match the PSNR of the SC method, thus, we can conclude our proposed method works for blurred images as well. All results are summarised in table 2. Also, as we have noticed faster convergence in the absence of higher frequency details, if the exact accuracy of the edges is not too significant for the application, user may choose to filter the images to speed up the training process.

6 Discussion, Limitations, Future Works

As discussed in the analysis and results sections, the proposed method of supervising NeRF training with SC labels

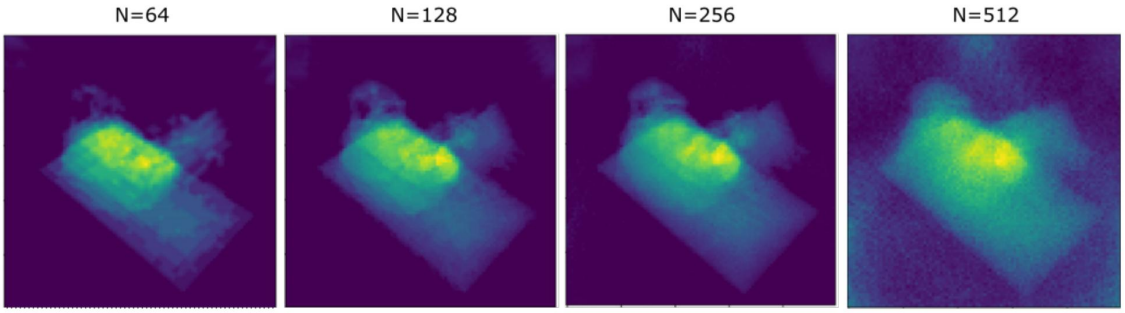

Fig. 4. Rendered images at different resolution levels N.

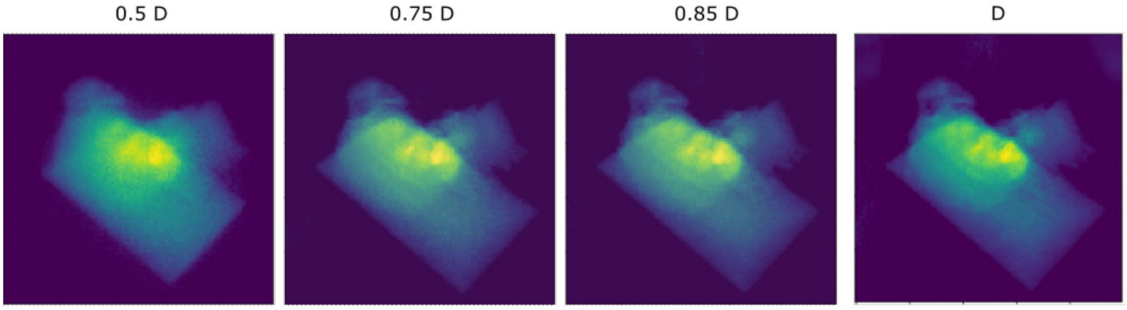

Fig. 5. Rendered images at different scaling levels D.

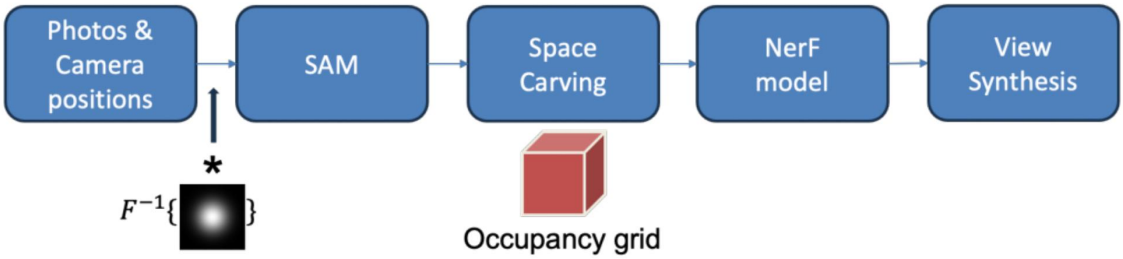

Fig. 6. For robustness analysis, a Gaussian LPF is applied to the images on the dataset before segmentation.

constitutes a high PSNR, low memory method for volumetric representation of 3D object segmentations.

One limitation of the current workflow is acquiring the initial masks with SAM. Although using the background masks was sufficient for this project, a better way of getting segmentation masks is necessary for more complex scenes. Methods such as using a bounding box, or specifying pixel coordinates to be included in the mask can be utilized. The challenge, however, remains to automatically get those masks. As each view has the object in a different orientation, finding correct pixel coordinates to give to SAM is not trivial. Similarly, bounding box approach would require the user to go thorugh the dataset manually. One future direction may be using promptable models such as Yolo World [9] to avoid using manually defined bounding boxes.

Although we also wanted to assess the effect of Gaussian additive noise, some of the SAM background masks were totally corrupted by that process. Not having reliable masks in the beginning would fail the whole pipeline, or we would need to manually assess each mask and leave the corrupted ones out, which corresponds to seeing less camera angles. Trying the bounding box could help SAM to generate accurate masks even in presence of additive Gaussian noise. This is another direction that can be explored.

We could also incorporate depth information from a depth estimating camera if we have access to that data. That would lead to a slight change in the SC algorithm, which can be outlined as follows:

- Start with filled voxels and iterate over all rays.
- For 0-rays, do the same carving procedure.
- For 1-rays, set the interpolated coordinates k that are closer to the camera than the measured depth value to 0. This step is additional to the SC algorithm outlined before, and incorporates more information from each 2D image when constructing $\hat{V}$.

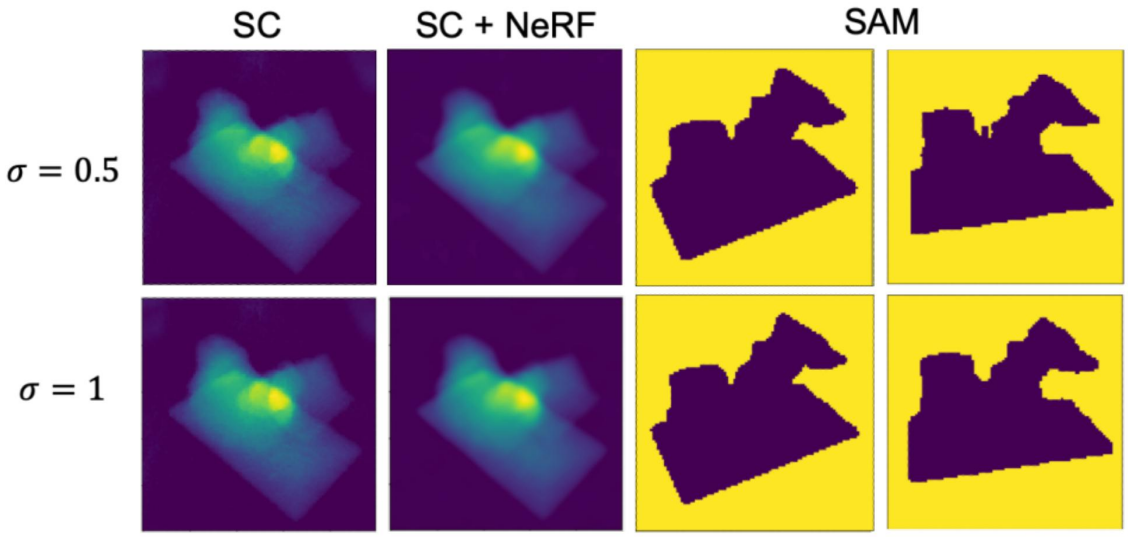

Fig. 7. Novel view rendering results for $\sigma = 0.5$ and $\sigma = 1$ for SC and SC & NeRF methods. 2 different SAM masks are also shown. Close inspection suggests the edges of the masks get smoother with higher σ.

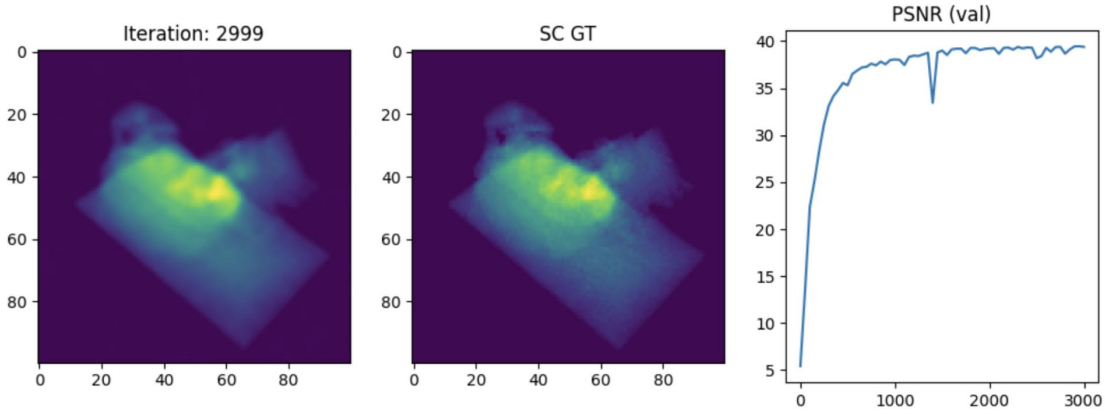

Fig. 8. Training outcome of SC & NeRF method. Left. Final test angle rendering. Middle. SC ground truth. Right. Validation PSNRs over epochs.

7 Conclusion

This project successfully outlined a novel framework that combines two well-known methods to achieve memory advantageous reliable 3D object segmentations. SC labels ensure better data fidelity compared to simple NeRF training procedure which relies on 2D image loss. Using implict volume representation provided by NeRF allows drastic compression compared to SC. Thus, SC & NeRF model combines the strengths of both to achieve superior performance.

References

[1] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," 2023.

[2] K. Kutulakos and S. Seitz, "A theory of shape by space carving," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 1, 1999, pp. 307–314 vol.1.

[3] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[4] J. Aleotti and S. Caselli, "A 3d shape segmentation approach for robot grasping by parts," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 358–366, 2012.

[5] M. Adamkiewicz, T. Chen, A. Caccavale, R. Gardner, P. Culbertson, J. Bohg, and M. Schwager, "Vision-only robot navigation in a neural radiance world," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4606–4613, 2022.

[6] A. Tewari, J. Thies, B. Mildenhall, P. Srinivasan, E. Tretschk, W. Yifan, C. Lassner, V. Sitzmann, R. Martin-Brualla, S. Lombardi *et al.*, "Advances in neural rendering," in *Computer Graphics Forum*, vol. 41, no. 2. Wiley Online Library, 2022, pp. 703–735.

[7] S. Vora, N. Radwan, K. Greff, H. Meyer, K. Genova, M. S. Sajjadi, E. Pot, A. Tagliasacchi, and D. Duckworth, "Nesf: Neural semantic fields for generalizable semantic segmentation of 3d scenes," *arXiv preprint arXiv:2111.13260*, 2021.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[9] T. Cheng, L. Song, Y. Ge, W. Liu, X. Wang, and Y. Shan, "Yolo-world: Real-time open-vocabulary object detection," *arXiv preprint*

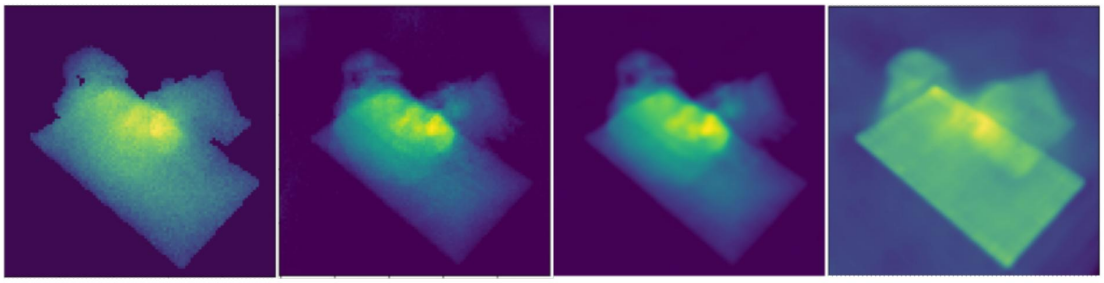

Fig. 9. Qualitative comparison of outlined methods SC, SC& NeRF, NeRF with respect to ground truth SC.

arXiv:2401.17270, 2024.