Baking in the Feature: Accelerating Volumetric Segmentation by Rendering Feature Maps

Kenneth Blomqvist[1], Lionel Ott[1], Jen Jen Chung[1,2] and Roland Siegwart[1]

***Abstract*— Methods have recently been proposed that densely segment 3D volumes into classes using only color images and expert supervision in the form of sparse semantically annotated pixels. While impressive, these methods still require a relatively large amount of supervision and segmenting an object can take several minutes in practice. Such systems typically only optimize the representation on the scene they are fitting, without leveraging prior information from previously seen images.**

In this paper, we propose to use features extracted with models pre-trained on large existing datasets to improve segmentation performance on novel scenes. We bake this feature representation into a Neural Radiance Field (NeRF) by volumetrically rendering feature maps and supervising on features extracted from each input image. We show that by baking this representation into the NeRF, we make the subsequent classification task much easier.

Our experiments show that our method achieves higher segmentation accuracy with fewer semantic annotations than existing methods over a wide range of scenes.

I. INTRODUCTION

Neural Radiance Fields (NeRFs) [1] and other neural implicit representations have recently emerged as a popular representation for 3D scenes due to their many favourable properties. They can accurately infer the geometry of a scene by making use of strong geometric structure and rich supervision from captured calibrated images. They have few hyperparameters to tune and are able to handle a wide range of scales, geometries and materials.

As NeRFs use an MLP to map spatial coordinates to color values, they can easily be modified to predict other observable spatial properties. This led to NeRFs quickly being applied to semantic volumetric segmentation through SemanticNeRF [2], which is able to propagate semantic pixel labels from one frame of a scene to another and across image pixels through generalization. This was subsequently demonstrated within an interactive semantic segmentation system, iLabel [3], motivating the use of such a system to generate ground-truth data for downstream embedded, real-time computer vision algorithms.

While such systems achieve increasingly high accuracy as the amount of annotated pixels grows, they still require a lot of human supervision. SemanticNeRF used one labeled *randomly selected* pixel per class, per image. With over 900 training images in a scene, this is a lot of annotated pixels from a diverse set of viewpoints. If used to annotate

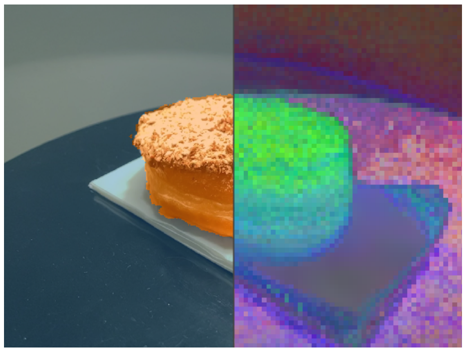

Fig. 1. The left side shows an example image from our dataset, overlayed with a segmentation mask produced by our system. The right side shows DINO features, which we reconstruct in our algorithm, mapped to RGB values using PCA.

data, providing such supervision can take an infeasibly long amount of time. An inherent limitation of current semantic NeRF approaches is that they blindly map 3D scene coordinates to radiance, density, and semantic class by optimizing a randomly initialized neural network per scene. Structure priors or previously learned information is not leveraged.

On the other hand, deep learning has fueled a whole body of work on representation learning, the goal often being to learn a feature representation that can be transferred to another task. Using pre-learned features in the context of NeRFs is not straightforward, as a NeRF maps scene-specific point coordinates to output values, which makes inducing desirable bias or structure into the model challenging. An example of this is *object* bias. For example, if a user of a semantic annotation system clicks on a cookie in a kitchen scene, the user might ideally want the system to, by default, automatically segment the entire cookie from other cookies and the background, without having to select each pixel.

The goal of our work is to infer a dense 3D semantic segmentation of the scene, and infer dense 2D semantic segmentation maps for each input image, while spending as little expert annotation time as possible. To this end, we develop a user interface which allows a user to quickly segment RGB-D video sequences, by drawing sparse annotations on the images. While the user is using the system, the program infers a segmentation given the current user inputs and shows it to the user. The user can refine the segmentation, until they are happy with the result.

To infer the best possible segmentation using a sparse set of labels, we propose a novel algorithm that models the scene using an implicit NeRF representation and leverages semantic image features obtained using a neural network feature extractor. We supervise our implicit scene model on these features, effectively *baking* a feature representation into

[1]Autonomous Systems Lab, Swiss Federal Institute of Technology in Zürich, Switzerland. kblomqvist@mavt.ethz.ch

[2]School of ITEE, The University of Queensland, Australia.

This project has received funding from EU Horizon 2020 program, project PILOTING H2020-ICT-2019-2 871542.

the learned MLP. We volumetrically render feature maps from the hidden layer activations of the semantic branch of the MLP, and minimize discrepancy between the rendered feature maps and extracted image feature vectors during training. This forces the MLP to encode additional structure, inducing desirable object, shape, and appearance bias into the learned representation, making the subsequent semantic classification much easier from sparse supervision. Our hypothesis is that such features encapsulate relevant spatial, object and semantic properties which are hard to learn by purely regressing color and radiance from position. Figure 1 illustrates how extracted features can encode information that can help distinguish between objects in a scene.

To summarize, our contributions are:

- A semantic NeRF algorithm that uses extracted image features to improve segmentation performance
- A hybrid feature encoding that is better suited for the semantic segmentation task
- A volumetric segmentation system, including a graphical user interface, that can be used to quickly generate dense segmentation masks from sparse pixel annotations

In experiments we validate our pipeline on a diverse set of real-world scenes. We perform a larger scale evaluation on scenes from the Replica [4] dataset, which contain many more objects and semantic classes. On these datasets, we compare the performance against baseline methods, as well as using different feature extractors, namely Fully Convolutional Neural Networks [5] and DINO [6]. Our results show that our DINO supervised semantic NeRF formulation outperforms previously proposed semantic NeRFs on both accuracy and learning speed across all scenes, and can do with much less human supervision.

We make visual results, data and code available through the accompanying web page[1].

II. RELATED WORK

A. Automated Labeling

Our goal is to infer a 3D segmentation of a scene and 2D semantic segmentation maps as quickly as possible. Several works have tackled a similar problem of inferring scene properties in an automated manner.

Object-based methods use knowledge of object geometry or category to speed up the workflow of annotating scenes. LabelFusion [7] introduced a tool to align an object model with a 3D scene with ICP refinement. While differentiable rendering was used to register the shape and pose of objects using shape priors in [8] and [9]. EasyLabel [10] introduced a semi-automatic method for obtaining instance segmentations of 3D scenes by incrementally building up the scene. RapidPoseLabels [11] presented a way to compute object pose and segmentation masks from sparse 2D keypoints, combining several pointclouds of an object.

Other approaches have explored speeding up RGB-D data annotation by leveraging structure in the data. SALT [12] introduced a GrabCut [13] based approach to speed up

[1]keke.dev/baking-in-the-feature

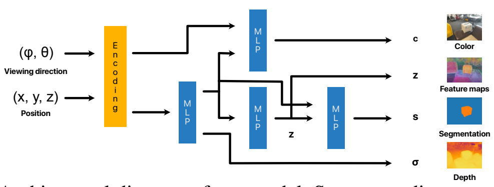

Fig. 2. Architectural diagram of our model. Scene coordinates and viewing direction are mapped to RGB, density, depth, feature maps and segmentation maps.

labeling of RGB-D data. DeepExtremeCut [14] computes dense object segmentation masks from extreme points of an object in an image.

B. Neural Radiance Fields

Mildenhall et al. [1] introduced NeRFs for novel view synthesis using volumetric rendering and have been extended in various ways. DS-NeRF [15] used depth measurements to speed up training, which we also levarage. SemanticNeRF [2] extended NeRF to also infer a semantic field from sparse pixel annotations and was extended with iLabel [3], a system for densely annotating scenes. We extend [2] to leverage representations which capture structure present in the input images to learn a better segmentation from fewer labels. While NeRF methods operate on single scenes, [16] introduced a representation that learns across multiple scenes. They disregard semantic labels, but their approach could be extended to segmentation. NeRF-Supervision [17] learns view-invariant dense object descriptors, generating image correspondences from a radiance field.

Other semantic fields include PanopticNeRF [18] which learns from noisy 2D predictions and 3D bounding box annotations. Panoptic Neural Field [19] learns several 4D neural fields that can reconstruct and track moving objects. NeSF [20] similarly uses known calibrated images and NeRF to learn a semantic field of a scene. Instead of adding a semantic output to the field, they learn a 3D UNet mapping density to semantics. They learn the semantic field across many scenes, not targeting single scene label propagation.

Concurrent work, D3F [21] and N3F, [22] use neural rendering via [23] and feature maps to supervise a NeRF model. N3F tackle one-shot object recognition, also showing some results for object segmentation using a thresholding algorithm, as opposed to propagating sparse labels. In contrast to [21], [23], our method runs at interactive rates, thanks to hybrid feature encoding and lower dimensional, autoencoded, feature rendering.

III. METHOD

We design a NeRF-style algorithm, that maps each point in the scene to color, density, depth, semantic class, and semantic feature vector. Figure 2 shows a high-level diagram of our model, which consists of five main components: a feature encoder, a geometry MLP, a color MLP, a feature MLP, and a semantic classifier MLP. In this section, we first describe how we encode scene positions into feature vectors that are fed to the neural network. We then describe how to use volumetric rendering to compute 2D image maps from the 3D representation and describe how we learn the parameters of the scene representation from provided data. Finally, we describe our graphical user interface, which allows a user to interact with the system.

A. Positional Encoding

NeRF [1] uses a frequency encoding based on sine and cosine functions:

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

where y corresponds to any of the 3D scene coordinates, or a viewing direction, normalized to the range $[-1,1]$. L defines the number of frequencies used. We use $L = 10$ for 3D coordinates and $L = 4$ for the viewing directions.

An alternative hash grid encoding was introduced in [24], which greatly speeds up the learning of NeRFs. They define a hierarchical voxel grid over the scene coordinates, with parameters at cell vertices for every level. To encode a point, parameters are looked up at each level of the grid, trilinearly interpolated and concatenated to produce an encoding. Parameters are learned jointly with the MLPs.

While the hash grid encoding produces great visual results for view synthesis and greatly speeds up training and rendering, it is not optimal for our application, as simply using the hash grid encoding causes the model to overfit to the sparse user annotations. The resulting segmentation maps fail to infer the spatial relation between annotated points, and large areas not belonging to objects are assigned the wrong semantic class. See Figure 3 for an illustration.

To solve this problem, we propose a hybrid approach combining both the hash grid encoding with low frequency positional encoding with $L = 2$, which we concatenate together. The low frequency encodings allow the model to reason about coarse spatial location in the volume, while the grid parameters allow encoding finer details in the scene. As we are targeting an interactive system, using only frequency encoding would not be an option, as it requires many more iterations to learn high frequency details.

B. Neural Radiance Fields

To integrate the scalar and vector outputs of our spatial field, we define a function R to volumetrically render vector or scalar values from a function h along a given ray $\mathbf{r}$:

$$R(\mathbf{r}, h) = \sum_{i=1}^{N} T_i (1 - \exp(-\sigma_i \delta_i)) h(\mathbf{x}_i), \quad (2)$$

$$T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j), \quad (3)$$

where T_i is the transmittance function measuring the amount light transmitted through the ray $\mathbf{r}$ to sample i, δ_j is the distance between samples and σ_i is the predicted density for encoded point samples $\mathbf{x}_i$ along the ray $\mathbf{r}$. We use this rendering function to render pixel color values, depth, semantic outputs and image features for a given ray $\mathbf{r}$:

$$\hat{\mathbf{c}}(\mathbf{r}) = R(\mathbf{r}, \mathbf{c}), \quad (4)$$

$$\hat{d}(\mathbf{r}) = R(\mathbf{r}, d), \quad (5)$$

$$\hat{\mathbf{s}}(\mathbf{r}) = R(\mathbf{r}, \mathbf{s}), \quad (6)$$

$$\hat{\mathbf{f}}(\mathbf{r}) = R(\mathbf{r}, \mathbf{f}), \quad (7)$$

using the following functions: $\mathbf{c}$ for color, d for depth, $\mathbf{s}$ for the semantic vector, and $\mathbf{f}$ for the intermediate feature vector of our model for encoded point samples along a ray $\mathbf{r}$.

We define the same photometric loss L_{rgb} as in NeRF [1] and a depth loss L_d similar to the one used by [15]:

$$L_{rgb}(\mathbf{r}) = \|\hat{\mathbf{c}}(\mathbf{r}) - \bar{\mathbf{c}}(\mathbf{r})\|_2^2, \quad (8)$$

$$L_d(\mathbf{r}) = \begin{cases} \left\|\hat{d}(\mathbf{r}) - \bar{d}(\mathbf{r})\right\|_1, & \text{if } \bar{d} \text{ is defined for } \mathbf{r} \\ 0, & \text{otherwise} \end{cases} \quad (9)$$

where $\bar{\mathbf{c}}(\mathbf{r})$ is the ground truth and $\hat{\mathbf{c}}(\mathbf{r})$ the predicted color for ray $\mathbf{r}$, $\bar{d}(\mathbf{r})$ is the ground truth depth (if available) and $\hat{d}(\mathbf{r})$ the integrated depth predictions along ray $\mathbf{r}$.

To learn the semantic class, we define a cross entropy loss:

$$L_s(\mathbf{r}) = \begin{cases} -\log \frac{\exp(\hat{s}(\mathbf{r})_{\bar{s}(\mathbf{r})})}{\sum_{c=1}^{C_n} \exp(\hat{s}(\mathbf{r}))_c)}, & \text{if } \bar{s} \text{ is defined for } \mathbf{r} \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

where $\bar{s}(\mathbf{r})$ is the ground truth class for ray $\mathbf{r}$, $\hat{\mathbf{s}}(\mathbf{r})$ is the integrated semantic MLP outputs and $\hat{s}(\mathbf{r})_c$ the output corresponding to class c.

C. Learning Feature Maps

We assume that we have a feature extractor, which maps images $\mathbf{I}^{H_i \times W_i \times 3}$ to feature maps $\bar{F}^{H_f \times W_f \times M}$ containing feature vectors $\bar{\mathbf{f}}$. $\mathbf{I}$ is an input image with height H_i and width W_i, $\bar{F}$ has height H_f and width W_f and M is the dimensionality of the feature vectors. The purpose of the feature extractor is to encode semantic and spatial information about a particular view, providing contextual information that cannot be inferred from a single scene, but can be learned by observing other data. Such functions include Vision Transformers [6] or Fully Convolutional Neural Networks [5] that come pretrained a priori on large datasets.

To distill information in the feature maps into our 3D scene representation and to inform a downstream classifier, we propose to volumetrically render feature outputs $\mathbf{f}$ along a ray using (2) to produce rendered feature maps $\hat{\mathbf{f}}$ and supervise on corresponding image features $\bar{\mathbf{f}}$.

The dimensionality of the image features $\bar{\mathbf{f}}$ will vary depending on the chosen pre-trained feature extractor and may be too large to render with reasonable batch sizes on regular GPUs. Therefore, to reduce memory use and the amount of floating point operations when using our system, we use a simple MLP autoencoder to reduce the dimensionality of the image features $\bar{\mathbf{f}}$. This step is completely optional and could be skipped. The autoencoder has an encoder *enc* and decoder *dec*. The encoder maps feature vectors with M dimensions to D dimensions and the decoder maps them back to M dimensions. We set D to 64 throughout our experiments. We

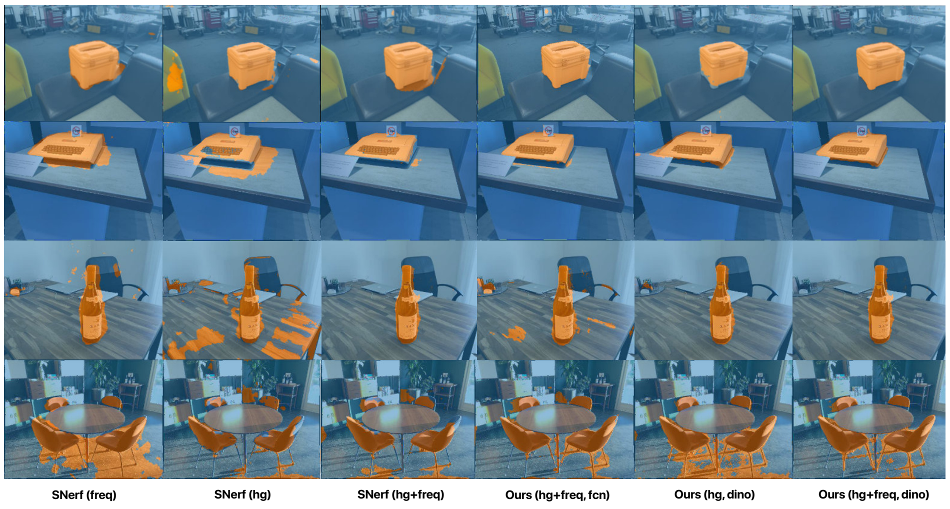

Fig. 3. Segmentation masks produced on the box scene using the different methods with the same set of annotations. The frequency encoding (freq) allows for spatial context while the hash grid encoding (hg) allows for a high level of detail locally. The feature supervision again helps in reasoning about object boundaries.

fit the autoencoder by minimizing a standard reconstruction loss:

$$L_{ae}(\bar{F}) = \|dec(enc(\bar{F}(\mathbf{p}))) - \bar{F}(\mathbf{p})\|_2 + \lambda_{ae}\, \|enc(\bar{F}(\mathbf{p})\|_1\,,$$

where $\bar{F}(\mathbf{p})$ is the feature corresponding to sampled pixel $\mathbf{p}$, and λ_{ae} is a weight for the sparsity term. We use an L_2 loss to minimize information loss. The sparsity term is to encourage a sparse feature representation that can easily be classified into different classes.

As image features only depend on raw input images and a given pre-trained feature extractor, we pre-compute the corresponding autoencoder offline before the user interacts with the system and keep it fixed while the scene representation is learned.

When fitting a scene in our volumetric segmentation pipeline, to bake the feature representation $\hat{\mathbf{f}}$ into the scene representation, we define an additional feature loss:

$$L_f(\mathbf{r}) = \left\|enc(\bar{\mathbf{f}}(\mathbf{r})) - \hat{\mathbf{f}}(\mathbf{r})\right\|_1, \tag{11}$$

where $\mathbf{r}$ is an image ray, $\bar{\mathbf{f}}(\mathbf{r})$ is the image feature corresponding to ray $\mathbf{r}$, $\hat{\mathbf{f}}(\mathbf{r})$ is the rendered feature for ray $\mathbf{r}$. Should H_i and H_f differ, during training, we use nearest neighbor interpolation to lookup the corresponding image feature.

D. Optimization and Sampling

We optimize the combined loss using stochastic gradient descent using Adam over rays $\mathbf{r}$ sampled from the images:

$$L(\mathbf{r}) = L_{rgb}(\mathbf{r}) + \lambda_d L_d(\mathbf{r}) + \lambda_s L_s(\mathbf{r}) + \lambda_f L_f(\mathbf{r}), \tag{12}$$

to jointly fit the positional encoding and MLP parameters.

As most pixels do not have a semantic class associated with them, we use a sampling scheme to balance the task of predicting a semantic class with the other objectives. When sampling examples for optimization, we select half the samples uniformly from all pixels. For the remaining half, to not bias the resulting function towards any class, we first select a class uniformly from the available classes. A pixel is then sampled from all pixels which are annotated with the sampled class.

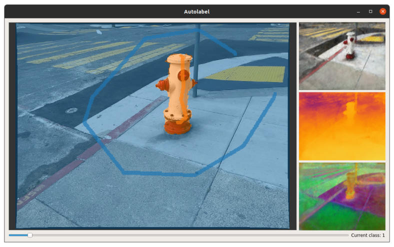

Fig. 4. The graphical user interface. The user provides sparse annotations by drawing on images with the appropriate class. The system infers a volumetric segmentation given the current sparse annotations from all views and renders a dense segmentation map which can be corrected with further annotations.

E. Graphical User Interface

Figure 4 shows the user interface of our system. The user can move through frames in a captured RGB-D video sequence and draw annotations, shown as opaque lines, while the system fits a model to the scene and annotations. The translucent mask shows the current segmentation, which the user can correct. The right side shows rendered color, depth and features mapped to RGB using PCA.

IV. Experimental Results

In our experiments we investigate whether our method improves segmentation accuracy when given the same amount of supervision and whether we improve labeling efficiency as part of an interactive labeling system.

A. Baselines

We compare our algorithm with SemanticNeRF[2]. We also compare the effect of using different positional encodings with both our algorithm and the baseline. For the positional encoding, **freq** refers to the frequency positional encoding in which case we use $L = 10$ frequencies for the position, **hg** refers to the hash grid encoding and **hg+freq** denotes hybrid encoding.

As our algorithm can make use of any features, we experiment with features from a Fully Convolutional Network [5] trained on COCO [25] on the semantic segmentation task,

Scene	SNeRF (freq)	SNeRF (hg)	SNerf (hg+freq)	Ours (hg+freq, fcn)	Ours (hg, dino)	Ours (hg+freq, dino)
apple 2	0.744	0.565	0.853	0.903	0.878	**0.942**
bench	0.803	0.817	0.816	0.818	0.873	**0.912**
box	0.901	0.686	0.868	0.951	0.922	**0.962**
chairs	0.531	0.635	0.705	0.686	0.696	**0.761**
cup	0.837	0.514	0.586	0.565	0.854	**0.934**
doughnut	0.653	0.576	0.502	0.673	0.879	**0.910**
fire hydrant 1	0.838	0.561	0.792	0.805	0.751	**0.887**
fire hydrant 2	0.757	0.216	0.853	0.598	0.848	**0.890**
hat	0.911	0.937	**0.960**	0.950	0.919	0.959
keyboard	0.895	0.699	0.900	0.926	0.943	**0.947**
shoe	0.791	0.814	0.833	0.894	0.970	**0.979**
valve	0.588	0.250	0.691	0.721	0.692	**0.730**
wine bottle red	0.523	0.362	0.521	0.794	0.690	**0.856**
wine bottle white	0.569	0.233	0.323	0.573	0.830	**0.884**
Average	0.739	0.560	0.732	0.778	0.703	**0.897**

TABLE I

INTERSECTION-OVER-UNION AGREEMENT OF PRODUCED SEGMENTATION MAPS ON OUR CAPTURED SCENES AGAINST MANUALLY ANNOTATED FRAMES.

denoted **fcn** and DINO ViT-S/8 vision transformer features [26] trained on ImageNet, denoted **dino**.

To make the comparison fair, all baselines and our method use the same sampling and training pipelines. They simply differ in the scene model and loss functions used.

B. Label Propagation Quality

To investigate the first question, we scan a number of scenes using an RGB-D camera and run them through [27] to obtain camera poses. We annotate the scenes using the GUI by drawing squiggles on pixels belonging to each desired semantic class on 2 to 10 individual images, depending on the scene. What the annotations look like from a single view can be seen in Figure 4. From these annotations, the algorithms are tasked to segment the scene and infer what the user considers as belonging to each class. All algorithms receive the exact same set of annotations. We run each algorithm on each scene and produce semantic segmentation maps for all images. We compare the produced segmentation maps against ground truth masks, obtained by labeling a reference subset of the images with a polygon mask for each object. We then compute the Intersection-over-Union agreement between the inferred and reference masks.

It should be noted that results on this experiment are not indicative of the performance at the limit on dense segmentation maps, rather they measure the ability of the algorithms to generalize and figure out where object boundaries lie from a specific set of reasonable, sparse annotations.

Table I shows IoU agreement between manually labeled individual frames and the segmentations produced by the different methods. The best accuracy is achieved using hybrid encoding, supervised by DINO features on virtually all scenes.

As illustrated in Figure 3 and quantified in Table I, using only hash grid encoding causes the model to overfit to the sparse annotations. Large areas outside of objects are assigned to the object class. This is especially apparent on the white wine scene (Figure 3, image row 3, column 2). Using hash grid encoding with DINO supervision removes some errors, as the model can reason about visual and contextual properties in the scene. However, object boundaries are not captured as sharply when not using feature supervision. The DINO features perform better than FCN features, indicating that the choice of feature extractor is an important consideration.

C. Human-in-the-loop Simulation

As our system is designed to quickly create annotated data for downstream learning algorithms with little human input, we design a simulated human-in-the-loop experiment to test the algorithms in a fair, scalable, and reproducible way. As a dataset, we use the openly available Replica dataset [4] and more specifically, the rendered sequences published by [2], which include dense ground truth semantic annotations. This dataset contains room-scale scenes with multiple objects of multiple classes, presenting a challenging and realistic test environment.

We initialize each algorithm by training only on RGB, depth and features for 15k iterations. We then run an iterative process predicting semantic labels using the current model parameters and select five pixels for which semantic classes are falsely inferred and add their ground truth labels to the training set. We then run 250 optimization steps with all labels accumulated thus far, as would be possible between user actions, and repeat the process.

We record the intersection-over-union agreement with the ground truth segmentation masks at each iteration step to observe how quickly the label propagation algorithm is able to produce high quality semantic labels. The ideal algorithm would converge to perfect labels after one click per object class by the user.

Figure 5 shows results on different scenes of the Replica dataset for the human-in-the-loop simulation. Our method performs better at the limit than the baseline methods on all of the scenes. The results show that the biggest benefit of feature supervision is that the model learns from much fewer labels, yet still achieves higher accuracy as more pixels are annotated.

The baseline SNeRF (freq) method takes on average over 5x longer to reach 80% IoU accuracy compared to our method with DINO features and hybrid positional encoding. This means a user would spend less than a fifth of the time using the system to achieve the same accuracy.

V. CONCLUSIONS

We presented an algorithm for volumetric segmentation, which we showcased in a human-in-the-loop data annotation and label propagation application. Our algorithm significantly speeds up learning and improves performance over baseline methods across our experiments. We performed an ablation study showing how different parts of our algorithm

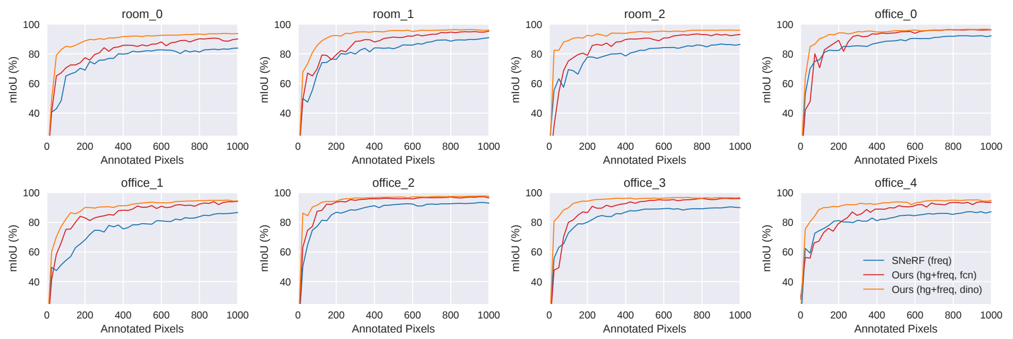

Fig. 5. Performance on different scenes of the Replica dataset. We outperform baseline methods on all scenes for both learning speed and final accuracy.

contribute to the overall performance. We demonstrated how using only hash grid encoding can cause a segmentation model to overfit to sparse user annotations and showed how this problem can be overcome by combining frequency encoding with hash grid encoding.

As shown by our experiments, the choice of feature is important. Learning a better feature representation that is viewpoint invariant, yet allows for efficient segmentation of objects presents a promising research direction. Furthermore, our system assumes a static scene, we intend to extend the work in the future to deal with dynamic scenes.

Our system produces high quality segmentation maps, but in many robotic applications computing other properties, such as object pose [7], shape [28] or keypoints [29] might be required. A framework which could with little input produce high quality segmentations in combination with object information, would be a breakthrough to fuel the learning-based perception algorithms proposed in recent years.

References

[1] 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," in *ECCV*, pp. 405–421, Springer, 2020.

[2] S. Zhi, T. Laidlow, S. Leutenegger, and A. J. Davison, "In-place scene labelling and understanding with implicit scene representation," in *IEEE/CVF CVPR*, pp. 15838–15847, 2021.

[3] S. Zhi, E. Sucar, A. Mouton, I. Haughton, T. Laidlow, and A. J. Davison, "ilabel: Interactive neural scene labelling," *arXiv preprint arXiv:2111.14637*, 2021.

[4] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, *et al.*, "The replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[5] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE/CVF CVPR*, pp. 3431–3440, 2015.

[6] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[7] P. Marion, P. R. Florence, L. Manuelli, and R. Tedrake, "LabelFusion: A pipeline for generating ground truth labels for real RGBD data of cluttered scenes," in *ICRA*, pp. 3235–3242, IEEE, 2018.

[8] S. Zakharov, W. Kehl, A. Bhargava, and A. Gaidon, "Autolabeling 3d objects with differentiable rendering of sdf shape priors," in *IEEE/CVF CVPR*, pp. 12224–12233, 2020.

[9] D. Beker, H. Kato, M. A. Morariu, T. Ando, T. Matsuoka, W. Kehl, and A. Gaidon, "Monocular differentiable rendering for self-supervised 3d object detection," in *ECCV*, pp. 514–529, Springer, 2020.

[10] M. Suchi, T. Patten, D. Fischinger, and M. Vincze, "EasyLabel: a semi-automatic pixel-wise object annotation tool for creating robotic RGB-D datasets," in *ICRA*, pp. 6678–6684, IEEE, 2019.

[11] R. P. Singh, M. Benallegue, Y. Yoshiyasu, and F. Kanehiro, "Rapid pose label generation through sparse representation of unknown objects," in *ICRA*, pp. 10287–10293, IEEE, 2021.

[12] D. Stumpf, S. Krauß, G. Reis, O. Wasenmüller, and D. Stricker, "Salt: A semi-automatic labeling tool for rgb-d video sequences," *arXiv preprint arXiv:2102.10820*, 2021.

[13] C. Rother, V. Kolmogorov, and A. Blake, ""GrabCut" interactive foreground extraction using iterated graph cuts," *ACM transactions on graphics (TOG)*, vol. 23, no. 3, pp. 309–314, 2004.

[14] K.-K. Maninis, S. Caelles, J. Pont-Tuset, and L. Van Gool, "Deep extreme cut: From extreme points to object segmentation," in *IEEE/CVF CVPR*, pp. 616–625, 2018.

[15] K. Deng, A. Liu, J.-Y. Zhu, and D. Ramanan, "Depth-supervised nerf: Fewer views and faster training for free," in *IEEE/CVF CVPR*, pp. 12882–12891, 2022.

[16] V. Lazova, V. Guzov, K. Olszewski, S. Tulyakov, and G. Pons-Moll, "Control-nerf: Editable feature volumes for scene rendering and manipulation," *arXiv preprint arXiv:2204.10850*, 2022.

[17] L. Yen-Chen, P. Florence, J. T. Barron, T.-Y. Lin, A. Rodriguez, and P. Isola, "Nerf-supervision: Learning dense object descriptors from neural radiance fields," *arXiv preprint arXiv:2203.01913*, 2022.

[18] X. Fu, S. Zhang, T. Chen, Y. Lu, L. Zhu, X. Zhou, A. Geiger, and Y. Liao, "Panoptic NeRF: 3D-to-2D label transfer for panoptic urban scene segmentation," *arXiv preprint arXiv:2203.15224*, 2022.

[19] A. Kundu, K. Genova, X. Yin, A. Fathi, C. Pantofaru, L. J. Guibas, A. Tagliasacchi, F. Dellaert, and T. Funkhouser, "Panoptic neural fields: A semantic object-aware neural scene representation," in *IEEE/CVF CVPR*, pp. 12871–12881, 2022.

[20] S. Vora, N. Radwan, K. Greff, H. Meyer, K. Genova, M. S. M. Sajjadi, E. Pot, A. Tagliasacchi, and D. Duckworth, "NeSF: Neural semantic fields for generalizable semantic segmentation of 3d scenes," *Transactions on Machine Learning Research*, 2022.

[21] S. Kobayashi, E. Matsumoto, and V. Sitzmann, "Decomposing nerf for editing via feature field distillation," in *NeurIPS*, vol. 35, 2022.

[22] V. Tschernezki, I. L. D. Larlus, and A. Vedaldi, "Neural feature fusion fields: 3d distillation of self-supervised 2d image representations," in *International Conference on 3D Vision, 3DV 2022, Prague, Czech Republic, September 12-15, 2022*, IEEE, 2022.

[23] V. Tschernezki, D. Larlus, and A. Vedaldi, "Neuraldiff: Segmenting 3d objects that move in egocentric videos," in *2021 International Conference on 3D Vision (3DV)*, pp. 910–919, IEEE, 2021.

[24] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Transactions on Graphics*, vol. 41, no. 4, pp. 1–15, 2022.

[25] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*, pp. 740–755, Springer, 2014.

[26] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *IEEE ICCV*, 2021.

[27] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4104–4113, 2016.

[28] W. Gao and R. Tedrake, "kpam-sc: Generalizable manipulation planning using keypoint affordance and shape completion," in *ICRA*, pp. 6527–6533, IEEE, 2021.

[29] K. Blomqvist, J. J. Chung, L. Ott, and R. Siegwart, "Semi-automatic 3D object keypoint annotation and detection for the masses," in *ICPR*, 2022.