8 Supplementary Material

We provide additional implementation details in Sec. A. We present additional qualitative results on Blender-HS dataset, PartNet dataset, and LLFF dataset in Sec. B. Additionally, we include a video attachment with visualizations of the view-consistent hierarchical segmentation results.

A. Additional Implementation Details

A.1. Hyperparameters On Blender-HS and PartNet, we train our model for $20,000$ iterations with a batch size of 4096 and use the same optimization parameters as DFF. In contrastive learning (see Eq. (3)), we set the temperature τ to 0.1 and sample 64 positive and negative pairs from each mask. We set the loss weight α of the Euclidean loss in Eq. (4) to 1. For depth continuity loss, we sample 16 patches per mask, and begin using the depth continuity loss after 5000 iterations. During 3D inference, we extract a point cloud from training-view depth maps, apply voxel downsampling with voxel size 2×10^{-3}, and run outlier removal with distance threshold 4×10^{-3} and number threshold of 1. We build the graph of points using $k_{graph} = 16$ nearest neighbors, and we transfer point segmentation labels into a novel view using the mode of $k_{query} = 5$ nearest neighbors. We retain $N = 200$ graph components and set the distance threshold d to be 5×10^{-3}. Please refer to Sec. 4 for definitions of the above hyperparameters.

A.2. Hierarchical Sampling We first organize the segmentation masks into a hierarchical structure determined by the inclusion ratio between them. One mask A is designated as a *child* of another mask B when $\frac{|A \cap B|}{|A|} > p_{in}$ and $\frac{|A \cap B|}{|A \cup B|} < p_{IoU}$. We empirically set $p_{in} = 0.95$ and $p_{IoU} = 0.85$. We present the hierarchical sampling algorithm we introduced in Sec. 4.1 in Algorithm 1.

We sample same number of positive pairs and negative pairs for training for training efficiency. Implementing a 4-1 ratio ($4\times$ more negatives than positives) instead of 1-1, the normalized covering (NC) score increases from 0.709 to **0.720**. However, this slow down our training by 43%, primarily due to the time-consuming computation of ultrametric distances and the associated minimum spanning tree.

A.3. Additional Details on Evaluation Metrics

Normalized Covering Score As discribed in Sec. 5.1, we measure the *quality* of hierarchical segmentation with the Normalized Covering (NC) score [19]. This metric averages the Intersection over Union (IoU) between each ground truth mask and the *best-matching* (*i.e.* most-overlapping) predicted mask. The metric is defined as

$$\mathrm{NC}(S' \to S) = \frac{1}{|S|} \sum_{A \in S} \max_{A' \in S'} \frac{|A \cap A'|}{|A \cup A'|} \tag{6}$$

Algorithm 1 Data Sampling

```
pos_samples ← []
neg_samples ← []
for all $A \in$ leaf_masks do
    # Sampling positive pairs from the leaf node
    sample ← (Random($A$), Random($A$))
    while $A$ has Parent do
        $B \leftarrow A$.Parent
        pos_samples += sample
        # Sampling negative pairs for the current level
        sample ← (Random($A$), Random($\bar{A} \cap B$))
        neg_samples += sample
        $A \leftarrow B$
    end while
end for
return pos_samples, neg_samples
```

Where S denotes all segmentation masks and S' denotes all predicted masks. For LSeg, DFF, and LeRF, which only output feature fields without segmentation results (S'), we adopt a similar approach as our method, and we extract segmentations by thresholding feature distances.

Segmentation Injectivity Score We propose the Segmentation Injectivity (SI) score to measure if each pixel belongs to only one mask for each level of granularity. As described in Sec. 5.1, given a ground truth mask, we first randomly sample p_1 and p_2 from that mask, and then query the model at these points and granularity for a new mask prediction. Then, we measure the IoU between the two resulting masks. We iterate this process $N = 100$ for each ground truth mask, calculating scores for each run. The final SI score is obtained by averaging the scores across all ground truth masks and viewpoints.

We represent the segmentation model as $F(v, p, t) \to A'$ where v denotes the viewpoint, p represents the pixel query, t corresponds to the granularity level, and A' is the resulting segmentation mask. The SI score is defined as

$$\mathrm{SI}(S' \to S) = \frac{1}{|N||S|} \sum_{A \in S} \sum_{i=1}^{N} \frac{|F(v, p_1^i, t) \cap F(v, p_2^i, t)|}{|(F(v, p_1^i, t) \cup F(v, p_2^i, t)|}$$
$$\text{where} \quad t = \arg\max_t \frac{|A \cap |F(v, p_1^i, t)|}{|A \cup |F(v, p_1^i, t)|}$$

where v represents the view corresponding to the ground truth mask A.

View Consistency Score We use the View Consistency (VC) score to measure the 3D consistency of image segmentations. Starting with the source view, we rotate the camera by 10 degrees, rendering both a new image and the corresponding ground truth visibility mask in the shifted view – Fig. 6 provides an example of two viewpoints and their visibility mask on the Blender Hotdog scene.

For a given point query p_1 and a granularity t and its mask prediction $A_1 = F(v, p_1, t)$ in the source view, we leverage the ground truth camera parameters to warp the point to $p_2 = T(p_1)$ and the mask prediction to $T(A_1)$ in the shifted view where T denotes the pixel transformation. Following this, we query the model in the shifted view with p_2 using the same threshold t, resulting in $A_2 = F(v', p_2, t)$.

Utilizing the visibility mask V, we eliminate pixels that are occluded in either view from A_2 and $T(A_1)$. The Intersection over Union (IoU) between the remaining masks is computed as the VC score for this sample. We reduce the noise induced by random sampling by computing this score for $N = 100$ times per ground truth mask.

Taken together, the VC score is defined as

$$\begin{aligned}
\mathrm{VC}(S' \to S) &= \frac{1}{|N||S|} \sum_{A \in S} \sum_{i=1}^{N} \frac{|T(A_1^i) \cap A_2^i \cap V|}{|(T(A_1^i) \cup A_2^i) \cap V|} \\
\text{where} \quad A_1^i &= F(v, p_1^i, \arg\max_t \frac{|A \cap |F(v, p_1^i, t)|}{|A \cup |F(v, p_1^i, t)|}) \\
A_2^i &= F(v', p_2^i, \arg\max_t \frac{|A \cap |F(v, p_1^i, t)|}{|A \cup |F(v, p_1^i, t)|})
\end{aligned}$$

For additional details, please refer to Sec. 5.1.

We also evaluate the View Consistency across multiple angles, in Tab. 5. The ranking of the methods is the same.

Table 5: View Consistency score with different view angles.

Method	$\mathrm{VC}_{10^\circ} \uparrow$	$\mathrm{VC}_{45^\circ} \uparrow$	$\mathrm{VC}_{90^\circ} \uparrow$	$\mathrm{VC}_{135^\circ} \uparrow$
LSeg [26]	0.536	0.522	0.510	0.498
LSeg + DFF [24]	**0.827**	**0.813**	**0.810**	**0.808**
SAM3D [6]	0.724	0.601	0.578	0.539
Ours	0.789	0.763	0.742	0.712

Depth Error We leverage the ground truth depth map rendered in blender to compute the depth error of our method. The scale of the depth error adheres to the normalized NeRF scene.

A.4. Additional Details on Baselines

DFF We configure DFF to use a white background and employ uniform ray sampling on the BlenderHS dataset. All other hyperparameters directly adhere

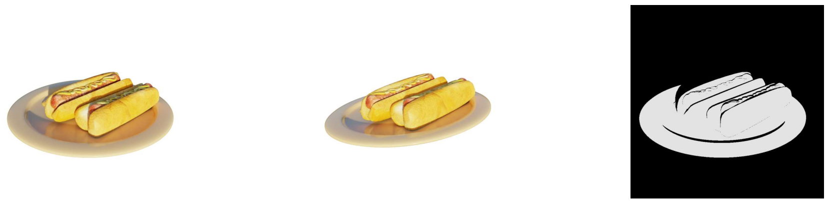

(a) Image **(b)** Shifted Image **(c)** Visibility Mask

Fig. 6: View Consistency: We evaluate the view consistency between the source viewpoint (a) and another one shifted 10 degrees (b). We render the ground truth visibility mask (c) with ray casting to avoid the occlusion/disocclusion between views.

Table 6: Distilled Feature Fields: We present the NC score of DFF with volume rendering (VR) and the NC score of the official codebase on our BlenderHS dataset.

Method	$\text{NC}_{\text{obj}} \uparrow$	$\text{NC}_{\text{coll}} \uparrow$	$\text{NC}_{\text{scene}} \uparrow$	$\text{NC}_{\text{mean}} \uparrow$
DFF (VR)	**0.164**	**0.427**	**0.839**	**0.477**
DFF (Official)	0.082	0.286	0.666	0.345

to the official implementation. Furthermore, DFF's official code[2] does not apply volume rendering to the feature branch. Instead, it generates a feature map by directly querying the volumetric features at the 3D locations of the predicted surface points. We extended their code to perform volume rendering, and we show that using volume rendering leads to improved performance on the BlenderHS dataset (see Tab. 6).

LeRF We use LeRF's reported NSVF hyperparameters for the Blender synthetic dataset. This includes configuring the background to white, selecting uniform sampling as the ray sampling strategy, disabling space distortion, and setting average appearance embedding to off. We train the model for 20000 steps. For the Normalized Covering Score, we report the highest result among all 30 semantic scales available in the LeRF feature field for each ground truth granularity. For the Segmentation Injectivity score and View Consistency scores, we evaluate LeRF at the semantic scale corresponding to the ground truth granularity which yields the highest NC score.

SAM3D Given a pretrained NeRF and a segmentation mask from a single view, SAM3D optimizes a binary voxel grid using mask inverse rendering and cross-view self-prompting to propagate the mask into 3D. In our experiments, we propagate the SAM masks from the segment-everything mode using 20 training views (while still using all 100 training images to pretrain the NeRF). We observed saturation in SAM3D's NC score after 20 views, and, on an A6000 GPU, it takes approximately a day per scene to propagate the segmentation maps from 20 views. In contrast, our method takes around 2 hours.

[2] `https://github.com/pfnet-research/distilled-feature-fields`

SAM We employ the ViT-H model from the official SAM GitHub repository[3] to generate mask predictions. To generate the training data of our model, we use the segment-everything mode to generate our supervision.

In the evaluation process, when querying segmentation models with a randomly sampled point, we employ the point as a prompt for SAM to generate the segmentation prediction. This approach, compared to evaluating based on the output of the segment-everything mode, yields a higher NC score and provides a clearer granularity level.

A.5. Training and Inference Time We train and perform inference on a Titan RTX GPU. Training typically takes ~70 minutes, while inference takes 5 seconds per granularity for 10 views. The main expense in inference is the watershed algorithm running on 3D point clouds, which is executed once per granularity and is view-independent.

B. Additional Qualitative Results

B.1. BlenderHS Dataset We first visualize the ground truth segmentations for the Drums scene in the BlenderHS Dataset [33] in Fig. 8. We then present qualitative results on the BlenderHS dataset [33] in Fig. 9. Our segmentations exhibit a hierarchical structure and maintain consistency across different views. We also visualize our ultrametric feature field on the Lego scene in Fig. 11, showing sharper features than DFF.

B.2. PartNet Dataset We present qualitative results on the PartNet dataset [36] in Fig. 10. Our method is able to generate hierarchical segmentation results of different objects. Leveraging the 2D masks predicted with SAM as guidance, our method proficiently segments various surfaces of sub-parts within the object, while those are not included in the PartNet ground truth annotations.

B.3. LLFF Dataset We present qualitative results on the the LLFF dataset [33] in Fig. 12. Our approach is able to generate view-consistent hierarchical segmentation results for real-world scenes.

[3] `https://github.com/facebookresearch/segment-anything`

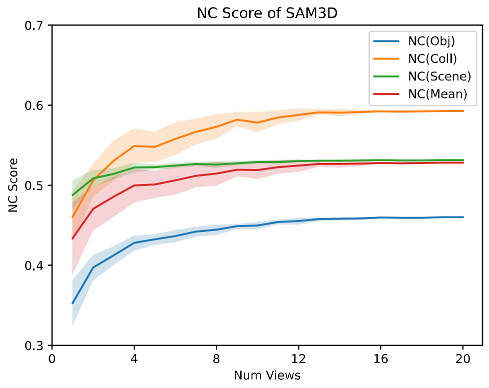


Fig. 7: NC Score of SAM3D: We present the Normalized Covering (NC) score (y-axis) of SAM3D, correlating it with the number of views (x-axis) from which we propagate the SAM segmentation masks.

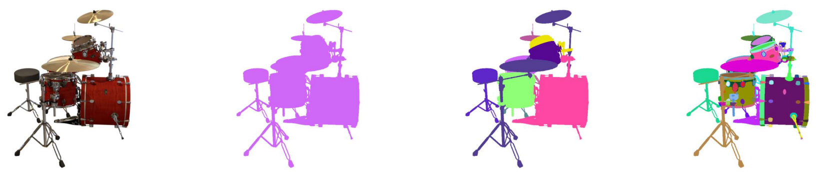
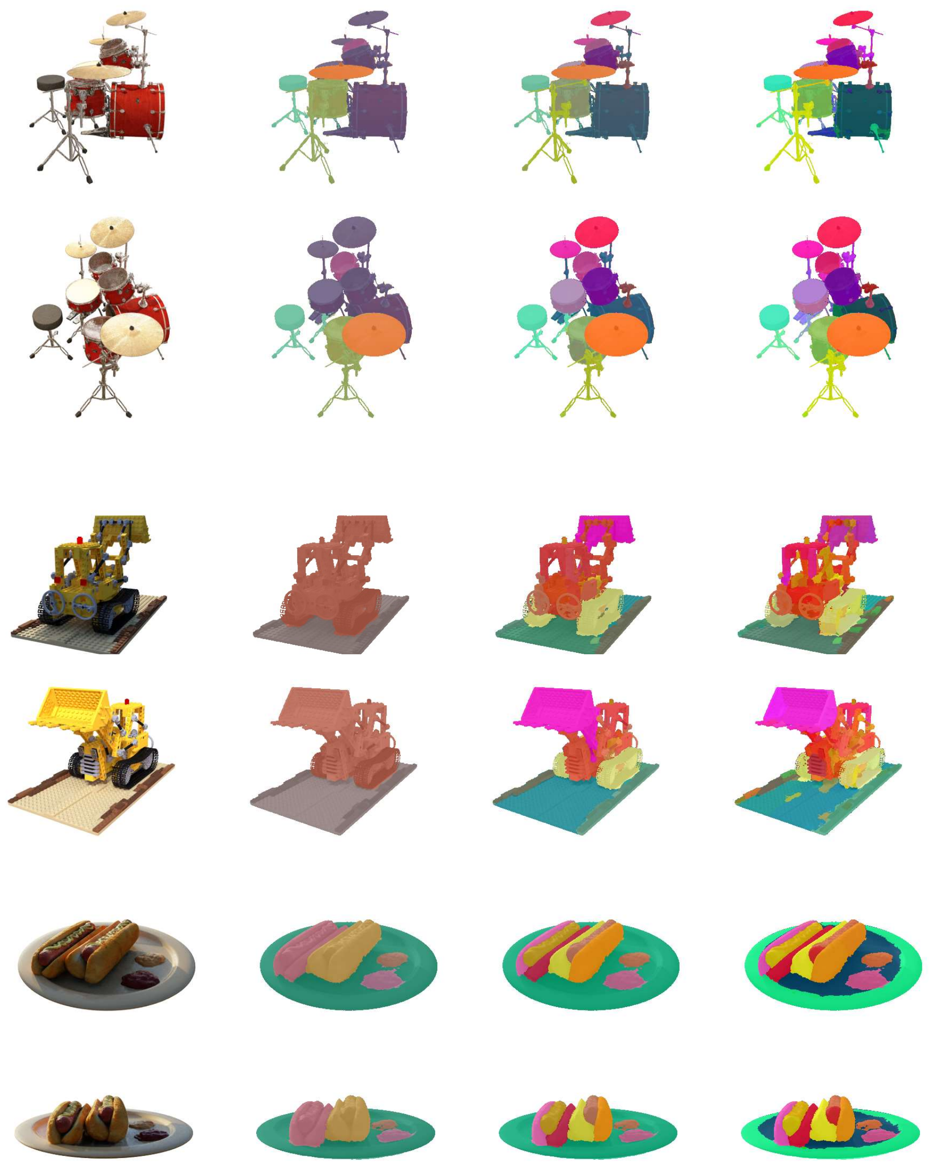


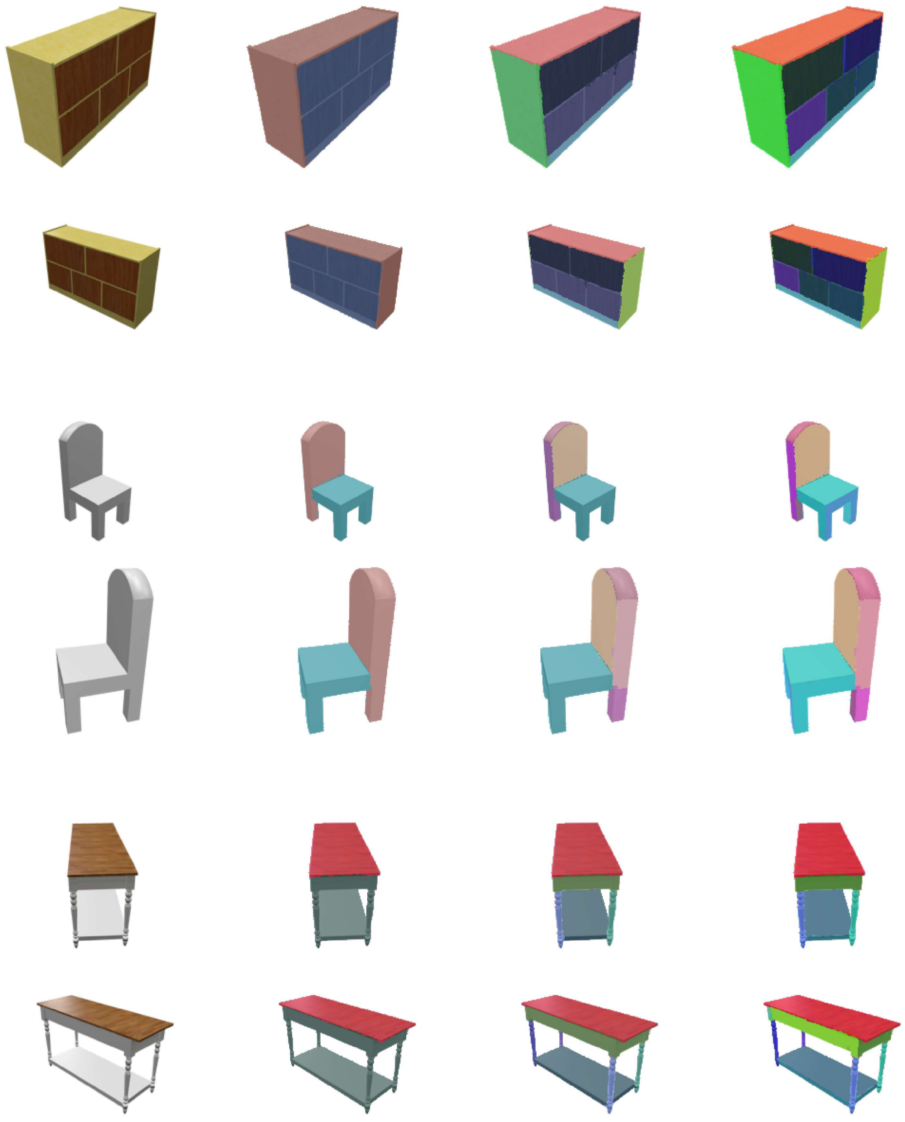
Fig. 8: Blender with Hierarchical Segmentation (Blender-HS): We render hierarchical segmentation maps at three levels of granularity, namely *Scene*, *Collection*, and *Object*, using information saved into the blender file by the scene artist.



(a) Image

(b) Hierarchical Segmentation

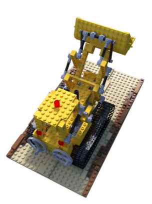
Fig. 9: BlenderHS Dataset: We present the qualitative results obtained from our BlenderHS dataset. The segmentation results demonstrate both view consistency and hierarchical structure.

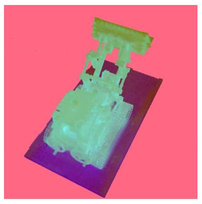


(a) Image

(b) Hierarchical Segmentation

Fig. 10: PartNet Dataset: We showcase the qualitative results on the PartNet dataset.


(a) Image


(b) DFF [26]

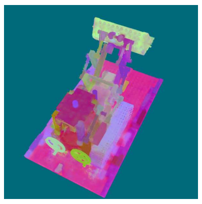

(c) Ours

Fig. 11: Feature Visualization: We visualize rendered feature maps using PCA. The feature map generated by DFF [26] fails to distinguish between different parts of the Lego. In contrast, our method learns features that can distinguish various Lego bricks.

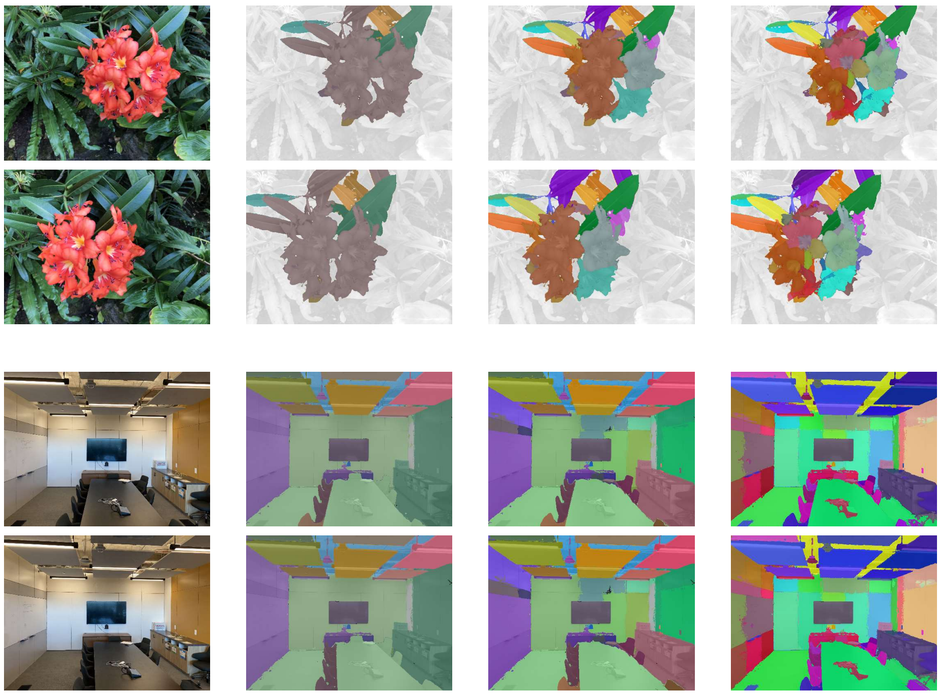


Fig. 12: LLFF Dataset: We showcase the qualitative results on the LLFF dataset.