# Supplementary Materials for Structurally Disentangled Feature Fields Distillation for 3D Understanding and Editing

## 1. Project Page

Our project page is available at: https://structurallydisentangled.github.io/. Our project page contains all scenes in the main text, including corresponding segmentation, removal, and editing results from various novel views.

### 1.1. Preliminaries

For a self-contained exposition, we provide a brief overview of NerF preliminaries. NeRF [4] represents a 3D scene using a multi-layer perceptron network (MLP) that parameterizes a continuous 5D radiance field. This field $f : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$, maps a 3D coordinate $\mathbf{x} \in \mathbb{R}^3$ and a viewing direction $\mathbf{d} \in \mathbb{R}^2$ to an emitted color $\mathbf{c} \in \mathbb{R}^3$ and volume density $\sigma \in \mathbb{R}$. To render 2D views, NeRF employs volumetric rendering. In particular, rays are cast through the scene, with $f(\mathbf{x}, \mathbf{d})$ queried along each ray. The volume rendering equation is then used to composite color and opacity:

$$C(r) = \int T(t)\mathbf{c}(\mathbf{r}(t))\sigma(\mathbf{r}(t))dt, \tag{1}$$

where $r(t)$ parameterizes the ray, $T(t)$ is the accumulated transmittance, and $\mathbf{c}(r(t))$ and $\sigma(r(t))$ are the color and density at point t. NeRF is trained by minimizing a reconstruction loss between images rendered from the field and a set of ground-truth images with known camera poses. This differentiable process allows NeRF to implicitly learn scene geometry and appearance for photorealistic rendering from unseen viewpoints.

When considering feature distillation [2, 3, 7], one additionally learns an MLP that parameterizes a continuous 3D feature field $f_{feat}$. Unlike $f$, $f_{feat} : \mathbf{x} \rightarrow \mathbf{f}$ maps a 3D coordinate $\mathbf{x} \in \mathbb{R}^3$ to a feature value $\mathbf{f} \in \mathbb{R}^n$. 2D feature maps can then be rendered similarly to color values, using $\sigma$ predicted by $f$. That is:

$$F(r) = \int T(t)\mathbf{f}(\mathbf{r}(t))\sigma(\mathbf{r}(t))dt, \tag{2}$$

where $r(t), T(t)$ and $\sigma(r(t))$ are as above and $\mathbf{f}(r(t))$ represents the predicted feature value at point $t$ (identical for every view direction). In addition to image-based reconstruction loss, one also minimizes a reconstruction loss between rendered features and ground-truth feature *maps*, obtained by applying a pre-trained 2D feature extractor (such as DINOv2 [5]) to ground-truth RGB views.

## 2. Implementation Details

We implement our method using Pytorch, based on the public implementation of RefNeRF [6]. In the iNGP hash grid hierarchy, we use 15 levels (from 32 to 4096) with 4 channels for each level, where each level has 4 channels. Two rounds of proposal sampling are used as in MipNeRF-360 [1]. We also penalize the mean of the sum of squared grid-hash values with a loss multiplier of 0.1. Our models are trained on a single A100 GPU for around 2 hours per scene. The code will also be made publicly available.

## 3. Loss Function

When training the color, loss function (Eq.13 in the main paper), we set $\lambda_c = 5$, $\lambda_e = 0.1$, $\lambda_o = 1 \times 10^{-6}$, and $\lambda_p = 3 \times 10^{-8}$. When training features (Eq.14 in the main paper), we freeze all color-dependent parts and set $\lambda_f = 0.05$.

## 4. Training Details

For training, we use the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.99$, and $\epsilon = 10^{-15}$. We warm up the learning rate in the first 500 steps and then decay it exponentially.

## 5. Per Scene IoU Results

In Tab. 1, we provide per-scene and per-object results accompanying Tab. 1 of the main text. As can be seen, our method results in a superior performance, particularly in reflective objects such as the car windshield.

## 6. Appearance Results

In Tab 2, Tab 3, and Tab 4, we also evaluate our explicit model on appearance (SSIM/PSNR/LPIPS) to Ours-implicit, an approach with implicit formulation, and Ours-optt, our approach, optimized for feature and color together,

| Scene - Object | Ours | DFF | DFF+ Ref | Feature 3DGS |
|---|---|---|---|---|
| Bicycle - Bench | **0.750** | 0.659 | 0.627 | 0.213 |
| Bicycle - Wheels | **0.415** | 0.378 | 0.380 | 0.343 |
| Counter - Mittens | 0.869 | 0.787 | **0.907** | 0.651 |
| Counter - Plant | 0.755 | **0.791** | 0.256 | – |
| Counter - Tray | **0.847** | 0.561 | 0.722 | 0.484 |
| Garden - Ball | 0.770 | **0.838** | 0.732 | 0.416 |
| Garden - Plant | **0.805** | 0.795 | 0.737 | 0.772 |
| Garden - Tabletop | **0.946** | 0.840 | 0.631 | 0.329 |
| Kitchen - Lego | **0.871** | 0.856 | 0.850 | 0.582 |
| Gardenspheres - Cone | **0.845** | 0.833 | 0.817 | 0.793 |
| Gardenspheres - Head | **0.764** | 0.739 | 0.749 | 0.686 |
| Gardenspheres - Spheres | **0.866** | 0.757 | 0.742 | 0.757 |
| Sedan - Bonnet-top | 0.471 | 0.452 | **0.497** | 0.290 |
| Sedan - Windshield | **0.659** | 0.569 | 0.342 | 0.176 |
| Sedan - Hubcaps | 0.746 | **0.764** | 0.653 | 0.646 |
| Sedan - Wheel | 0.697 | **0.714** | 0.557 | – |
| Toycar - Body | 0.775 | 0.693 | **0.801** | 0.457 |
| Toycar - Wheel | 0.644 | **0.686** | 0.507 | 0.295 |
| Teapot - Cover | **0.762** | 0.529 | 0.399 | 0.669 |
| Car - Windshield | **0.838** | 0.357 | 0.260 | 0.196 |
| Car - Wheels | **0.760** | 0.688 | 0.630 | 0.203 |
| Toaster - Body | 0.938 | **0.967** | 0.764 | 0.712 |
| Toaster - Toasts | 0.873 | **0.913** | 0.811 | 0.583 |
| Helmet - Body | **0.929** | 0.900 | 0.860 | 0.499 |
| Helmet - Windshield | **0.796** | 0.562 | 0.084 | 0.678 |

Table 1. Per Scene results accompanying Tab. 1 of the main text. We consider mean IoU for segmentation of objects from the Shiny Blender synthetic dataset [6] (bottom) and real-world scenes (top). The first four scenes are taken from the real-world RefNeRF [6] dataset while the other four real-world scenes are from the Mip-NeRF-360 dataset. We compare our approach to DFF [3], a baseline where DFF is optimized for features while RefNeRF is optimized for appearance, and Feature 3DGS [8]. For each scene, we indicate the segmented objects.

and DFF, accompanying Tab. 1 of the main text. The explicit model offers a superior appearance on average.

## 7. Computational requirements

We require around 30 seconds to render a 500x400 resolution view at inference. This scales linearly with resolution. To perform segmentations and edits as detailed, we require about 60 seconds per frame on average. As for memory, we require an average of 22GB per scene when considering high-resolution scenes with full-resolution 2D ground views and corresponding semantics. Full per-scene memory requirements will be added in the next revision.


Figure 1. An ablation study on 3D semantic segmentation from four novel views for the real world garden scene, comparing the segmentation obtained by clicking on each object and using (i). our view-independent feature component (used as default), or (ii) using the combination of both the view-independent and view-dependent components.

| Objects | Ours | Ours optt | Ours implicit | Dff |
|---|---|---|---|---|
| Gardenspheres | 0.742 | 0.738 | **0.758** | 0.663 |
| Sedan | 0.648 | 0.678 | **0.696** | 0.664 |
| Toycar | **0.780** | 0.720 | 0.748 | 0.723 |
| Bicycle | 0.592 | 0.570 | **0.610** | 0.609 |
| Garden | **0.782** | 0.560 | - | 0.747 |
| Counter | **0.817** | 0.778 | 0.750 | 0.755 |
| Kitchen | **0.885** | 0.822 | 0.823 | 0.802 |
| Car | **0.928** | 0.921 | 0.904 | 0.893 |
| Helmet | **0.963** | 0.951 | 0.943 | 0.909 |
| Teapot | **0.995** | 0.993 | 0.993 | 0.984 |
| Toaster | 0.874 | **0.886** | - | - |

Table 2. Per Scene results accompanying Tab. 1 of the main text. We calculated SSIM for scenes from the Shiny Blender synthetic dataset [6] (bottom) and real-world scenes (top). The first three scenes are taken from the real-world RefNeRF [6] dataset while the other four real-world scenes are from the Mip-NeRF-360 dataset. We compare our approach to DFF [3] and an ablation study of our approach with implicit representation of physical properties and our approach where we optimized the color and semantic features together.

| Objects | Ours | Ours optt | Ours implicit | Dff |
|---|---|---|---|---|
| Gardenspheres | **28.92** | 26.26 | 28.27 | 24.46 |
| Sedan | 25.99 | **26.12** | 25.75 | 25.44 |
| Toycar | **26.96** | 25.85 | 25.56 | 25.54 |
| Bicycle | **23.81** | 23.53 | 23.27 | 23.67 |
| Garden | **26.02** | 23.39 | - | 25.83 |
| Counter | **27.60** | 25.89 | 23.50 | 24.22 |
| Kitchen | **29.78** | 27.84 | 26.72 | 25.96 |
| Car | **28.10** | 25.84 | 24.92 | 24.21 |
| Helmet | **33.89** | 31.40 | 29.08 | 25.28 |
| Teapot | **44.95** | 41.72 | 43.68 | 39.94 |
| Toaster | 24.93 | **25.02** | 23.48 | 21.94 |

Table 3. As for Tab. 2 but measuring PSNR

| Objects | Ours | Ours optt | Ours implicit | Dff |
|---|---|---|---|---|
| Gardenspheres | **0.119** | 0.140 | 0.162 | 0.216 |
| Sedan | **0.227** | 0.251 | 0.290 | 0.269 |
| Toycar | **0.119** | 0.149 | 0.138 | 0.138 |
| Bicycle | **0.253** | 0.287 | 0.280 | 0.252 |
| Garden | **0.107** | 0.400 | - | 0.154 |
| Counter | **0.102** | 0.109 | 0.104 | 0.107 |
| Kitchen | 0.072 | 0.061 | **0.043** | 0.076 |
| Car | 0.062 | 0.056 | **0.043** | 0.067 |
| Helmet | 0.038 | 0.046 | **0.028** | 0.062 |
| Teapot | 0.0032 | 0.0057 | **0.0020** | 0.016 |
| Toaster | **0.101** | 0.098 | 0.113 | 0.129 |

Table 4. As for Tab. 2 but measuring LPIPS

## References


[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5470–5479, 2022. 1

[2] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. LERF: Language Embedded Radiance Fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19729–19739, 2023. 1

[3] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing NeRF for Editing via Feature Field Distillation. *Advances in Neural Information Processing Systems*, 35: 23311–23330, 2022. 1, 2

[4] 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. *Communications of the ACM*, 65(1):99–106, 2021. 1

[5] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 1

[6] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5491–5500, 2022. 1, 2

[7] Jianglong Ye, Naiyan Wang, and Xiaolong Wang. FeatureNeRF: Learning Generalizable NeRFs by Distilling Foundation Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8962–8973, 2023. 1

[8] Shijie Zhou, Haoran Chang, Sicheng Jiang, Zhiwen Fan, Zehao Zhu, Dejia Xu, Pradyumna Chari, Suya You, Zhangyang Wang, and Achuta Kadambi. Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21676–21685, 2024. 2