A Detailed Explanation of Non-Differentiable Feature Driving (NDFD)

Algorithm 1 General Feature Fusion and Densification Framework

$\mathbf{F}_{gt} \leftarrow$ GETFEATUREFROMSAM($\mathbf{I}_{gt}$) ▷ Features
$(p, \mathcal{H}) \leftarrow$ GETPOINTFROMSFM($\mathbf{I}_{gt}$) ▷ Positions & Hash
$\mathbf{f} \leftarrow$ POINTSFEATUREFUSION($p, \mathbf{F}, \mathcal{H}$) ▷ Point Features
$(\mathbf{R}, \mathbf{S}, c, op) \leftarrow$ INITATTRIBUTES() ▷ Rotations, Scales, Color, Opacity
$i \leftarrow 0$ ▷ Iteration Counter
while not converged **do**
  $(m, \mathbf{I}^m_{gt}, \mathbf{F}^m_{gt}) \leftarrow$ SAMPLETRAININGVIEW() ▷ Camera, Image, Feature
  $\mathbf{I}^m_{re} \leftarrow$ DRF($p, \mathbf{R}, \mathbf{S}, op, c, m$) ▷ Rasterization
  $L_{rgb} \leftarrow$ LOSS($\mathbf{I}^m_{gt}, \mathbf{I}^m_{re}$) ▷ Photometric Loss
  $(L_{gt}, L_{cf}) \leftarrow$ NDFD($p, \mathbf{R}, \mathbf{S}, op, \mathbf{f}, \mathbf{F}^m_{gt}, m$) ▷ Semantic Loss
  $L \leftarrow L_{rgb} + L_{gt} + L_{cf}$ ▷ Total Loss
  $(p, \mathbf{R}, \mathbf{S}, op, c) \leftarrow$ ADAM(∇L) ▷ Update
  if ISREFINEMENTITERATION(i) **then**
    DENSIFICATION($p, \mathbf{R}, \mathbf{S}, op, c, \mathbf{f}$) ▷ Adaptive Density
  end if
  $i \leftarrow i + 1$
end while

Details of the Rasterizater: Our implementation builds directly on the GPU rasterizer proposed in 3D Gaussian Splatting. Following that design, the image plane of size $w \times h$ is partitioned into 16×16 px tiles. Each gaussian primitive that overlaps a tile is *duplicated* for that tile and assigned a 64-bit key whose lower 32 bits encode depth and upper bits encode the tile index. A single parallel radix sort on these keys resolves global depth order and produces a compact, per-tile, depth-sorted list of instances; a second pass identifies the start–end range for each tile (see CULLGAUSSIAN, DUPLICATEWITHKEYS, and SORTBYKEYS in 3DGS). This eliminates sequential primitive traversal and maximizes GPU utilization.

Algorithm 2 Non-Differentiable Feature Driving Mechanism

function NDFD($p, \mathbf{R}, \mathbf{S}, op, \mathbf{f}, \mathbf{F}^m_{gt}, m$)
  $\mathbf{x} \leftarrow$ HOMOGENIZATION(m) ▷ Camera Homogenization
  $g \leftarrow \mathcal{G}_i(\mathbf{u}(\mathbf{x})) \leftarrow$ 2DSCREENGAUSSIANS($p, \mathbf{R}, \mathbf{S}, \mathbf{x}$) ▷ Screen-space Gaussians
  $T \leftarrow$ CREATETILES(m) ▷ Tile Grid
  $(I, K) \leftarrow$ DUPLICATEWITHKEYS(g, T) ▷ Indices & Keys
  SORTBYKEYS(K, I) ▷ Global Sort
  $T_r \leftarrow$ IDENTIFYTILERANGES(T, K) ▷ Tile Ranges
  $L_{gt} \leftarrow 0,\ L_{cf} \leftarrow 0$ ▷ Initilize Loss Buffers
  for all $t \in T$ **do**
    for all $i \in t$ **do**
      $r \leftarrow$ GETTILERANGE(T_r, t) ▷ Index Range in K
      for all $j \in r$ **do**
        $\sigma_j \leftarrow$ SIGMOID($\mathbf{F}^m_{gt}(i), f_j$) ▷ Polarity Response
        $w_j \leftarrow$ WEIGHTCALC(g_j, op_j) ▷ Opacity-weighted Area
        $L_{gt}[i]$ += EPFC(w_j, f_j, σ_j) ▷ External Potential
        $L_{cf}[i]$ += ICD(w_j, f_j, σ_j) ▷ Internal Clustering
      end for
    end for
  end for
  return (L_{gt}, L_{cf})
end function

Non-Differentiable Feature Driving (NDNF): Alg. 2 augments the aforementioned rasterizer with a feature-centric branch that runs entirely on the sorted gaussian primitives lists and never invokes

α blending. Given the current view m, camera homogenization first projects gaussian primitives means into screen space, after which key generation and radix sorting produce per-tile ranges. For every pixel i in a tile t, we then traverse the corresponding range r in front-to-back order. A sigmoid activation $\sigma_j = \textsc{Sigmoid}(\mathbf{F}_{gt}^m(i), f_j)$ converts the cosine similarity between the frozen feature f_j of the $j\text{-}th$ gaussian primitives and the ground-truth embedding $\mathbf{F}_{gt}^m(i)$ into a charge-like polarity. The raster weight w_j, which combines projected area and opacity exactly as in α blending, is accumulated only by this feature branch. Then, two loss terms are computed: the external-potential loss L_{gt} attracts σ_j-weighted features toward $\mathbf{F}_{gt}^m(i)$, whereas the internal-clustering loss L_{cf} applies the cumulative-feature rule to penalize incoherent neighbors. These loss buffers are initialized once per frame and updated atomically in the innermost loop, so no intermediate feature image is rendered. During back-propagation, gradients propagate solely through the weights w_j, which reuse the same cumulative prefix employed for α-blending in the forward traversal, thereby retaining the $\mathcal{O}(N)$ complexity of the original rasterizer.

A.1 Derivation of the feature similarity

Internal clustering loss L_{cf}: For a given pixel p, let $\{(w_i, f_i)\}_{i=1}^N$ be the set of gaussian primitive whose screen-space footprints cover that pixel, where w_i is the weight and $f_i \in \mathbb{R}^d$ is the frozen semantic feature of the $i-th$ primitive. The internal-clustering loss:

$$L_{cf} = \sum_{i=1}^{N} \sum_{j=1}^{N} w_i \, w_j \, \left(1 - \cos\langle f_i \cdot f_j \rangle\right) \tag{1}$$

computes the entropy of the local feature distribution by accumulating the weighted cosine dissimilarity between every pair of primitives. The process of minimizing L_{cf} pushes feature vectors of neighboring primitive to align, suppresses noisy outliers, and tightens semantic coherence within the pixel neighborhood. And simultaneously, this process allows primitives belonging to different objects to repel each other through their low cosine similarity.

We further convert it to an $\mathcal{O}(N)$ backward traversal by noting that feature vectors are normalized, therefore $\cos\langle f_i, f_j \rangle = f_i \cdot f_j$. We can rearranged the representation of L_{cf} as:

$$\begin{aligned} L_{cf} &= \sum_{i=1}^{N} \sum_{j=1}^{i-1} \sigma_i \, w_i \, w_j \, (1 - \cos\langle f_i \cdot f_j \rangle) \\ &= \sum_{i=1}^{N} \sigma_i \, w_i \left(\sum_{j=1}^{i-1} w_j - \sum_{j=1}^{i-1} w_j \, f_j \cdot f_i \right) \\ &= \sum_{i=1}^{N} \sigma_i w_i \left(\sum_{j=1}^{i-1} w_j - \sum_{j=1}^{i-1} w_j f_j \cdot f_i \right) \\ &= \sum_{i=1}^{N} \sigma_i \, w_i \, (W_{i-1} - F_{i-1} \cdot f_i) \end{aligned} \tag{2}$$

where the cumulative weight W_{i-1} and cumulative feature F_{i-1} are updated one time in each step during the front-to-back blend. The final form of L_{cf} retains the physical meaning of pairwise semantic attraction–repulsion so evaluates in $\mathcal{O}(N)$ time. The cumulative values of W_{N-1} and F_{N-1} are recorded.

A.2 Calculation process of the gradients

As derived in the main text, we obtain the partial derivatives:

$$\frac{\partial L_{cf}}{\partial w_k} = \sigma_k \, (W_{k-1} - \mathbf{F}_{k-1} \cdot \mathbf{f}_k) + \sum_{i=k+1}^{N} \sigma_i w_i \, (1 - \mathbf{f}_i \cdot \mathbf{f}_k) \tag{3}$$

In the forward pass the gaussian primitive is processed in descending depth order from the farthest to the nearest with respect to the camera. The backward pass visits the same primitive in the reverse

order. Because the index k is defined with respect to the forward ordering, we re-index the backward traversal by a new counter $q = 1, \ldots, N$. Exploiting this forward–backward symmetry, the gradient of the internal-clustering loss with respect to the weight of the current primitive can be rewritten as:

$$\frac{\partial L_{cf}}{\partial w_q} = \sigma_q\big((W_N - W_q) - (\mathbf{F}_N - \mathbf{F}_q)\cdot \mathbf{f}_q\big) + \sum_{i=1}^{q-1} \sigma_i w_i \big(1 - \mathbf{f}_i \cdot \mathbf{f}_q\big) \tag{4}$$

Based on the $w_i = \alpha_i T_i$, $\alpha_i = op_i \mathcal{G}_i(\mathbf{u}(\mathbf{x}))$, $T_i = \prod_{j=1}^{i-1}(1-\alpha_j)$, we can obtain:

$$\frac{\partial L_{gt}}{\partial \alpha_k} = \frac{\partial L_{gt}}{\partial w_k} \cdot \frac{\partial w_k}{\partial \alpha_k} = T_k \cdot \frac{\partial L_{gt}}{\partial w_k} - \frac{1}{1-\alpha_k} \sum_{i=k+1}^{N} \frac{\partial L_{gt}}{\partial w_i} \cdot w_i \tag{5}$$

Analogously to the equation above, we can also obtain:

$$\frac{\partial L_{gt}}{\partial \alpha_q} = \frac{\partial L_{gt}}{\partial w_q} \cdot \frac{\partial w_q}{\partial \alpha_q} = T_q \cdot \frac{\partial L_{gt}}{\partial w_q} - \frac{1}{1-\alpha_q} \sum_{i=0}^{q-1} \frac{\partial L_{gt}}{\partial w_i} \cdot w_i \tag{6}$$

B Comprehensive Results of Experiments

We conduct a detailed comparison between our method and Feature3DGS on the DTU indoor dataset. As shown in the Fig. 1, our method yields more uniform feature distributions and sharper boundaries. Moreover, it effectively suppresses background clutter, which remains prominent in Feature3DGS. The enhanced clarity and selectivity of our features also benefit downstream tasks such as segmentation and reconstruction. These observations highlight the strength of our feature driving mechanism in promoting structural coherence and semantic focus.

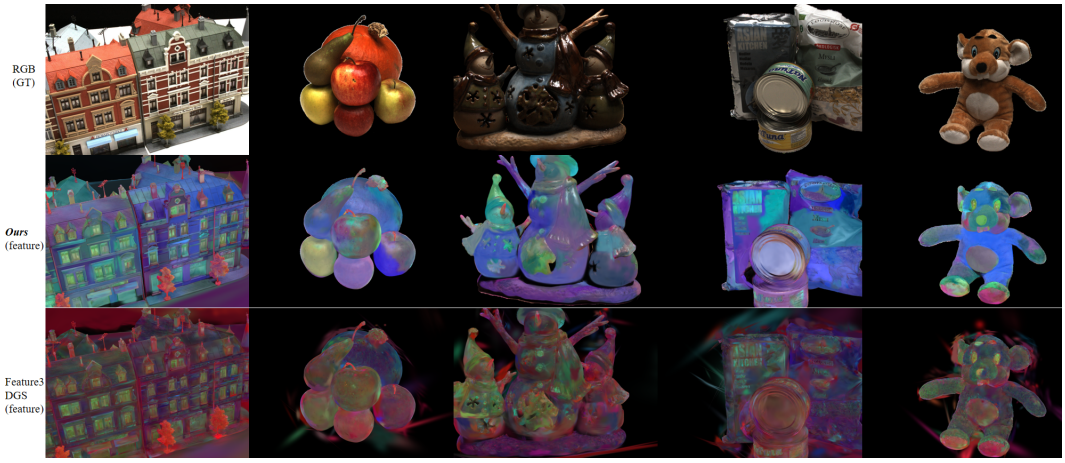

Figure 1: Qualitative results to compare our *FHGS* with Feature3DGS in feature field. The results shown that *FHGS* achieves better feature extraction with more uniform feature distributions, shaper boundaries and cleaner background.

We further evaluate our method against Feature3DGS on challenging outdoor scenes from the TnT and MipNeRF360 datasets. As shown in Fig. 2, our method consistently delivers more coherent and spatially uniform feature fields, with significantly clearer object boundaries and effective suppression of background noise. In addition to semantic features, we also visualize the surface normal maps extracted from our reconstruction, which exhibit plausible geometric structures and fine-grained surface details. These results demonstrate the robustness of our method under natural lighting, large-scale geometry, and high-frequency textures, confirming its generalization to diverse outdoor environments.

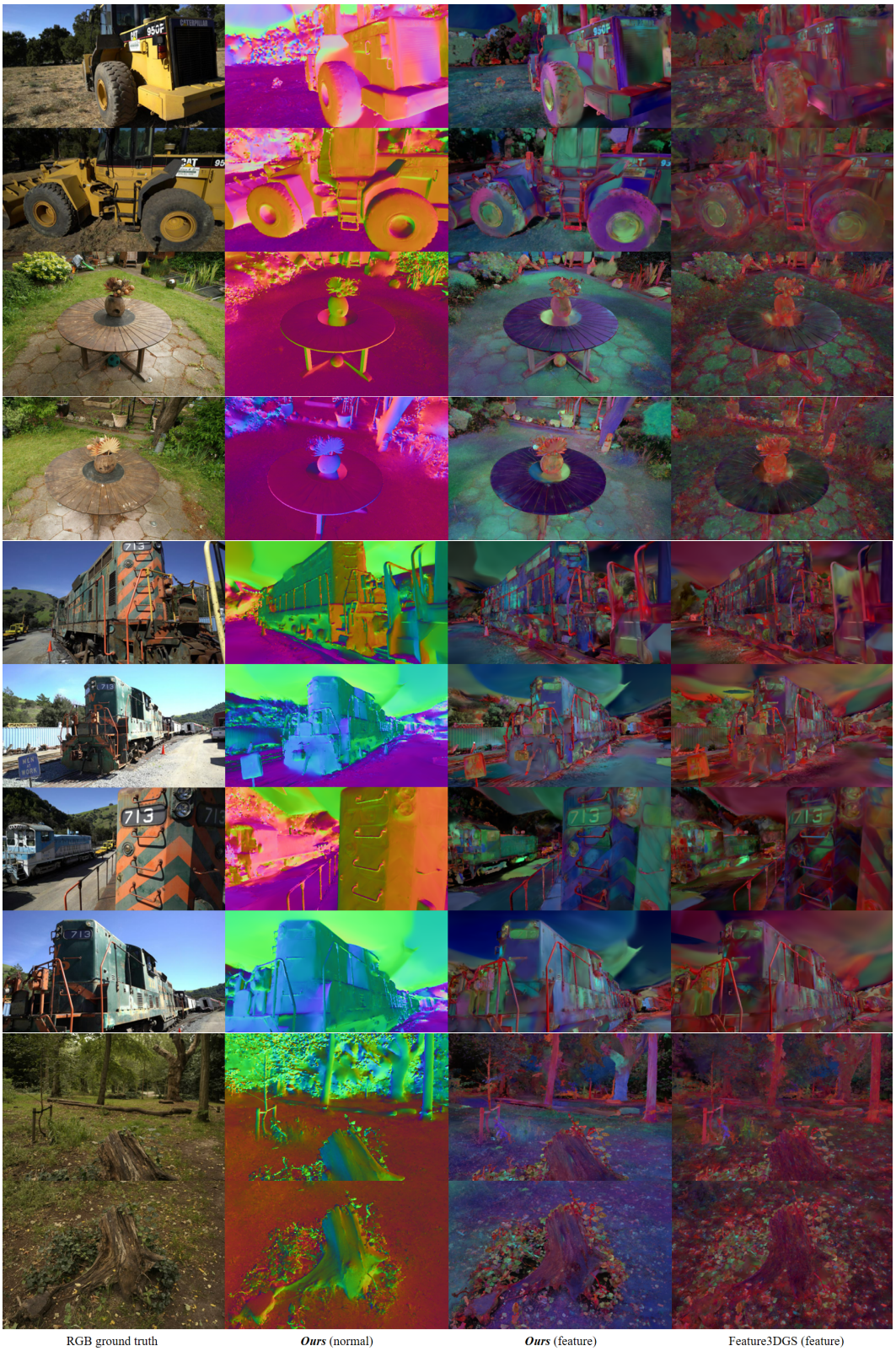

Figure 2: Qualitative comparison on outdoor scenes from TnT and MipNeRF360