H2G: Hierarchy-Aware Hyperbolic Grouping for 3D Scenes

Byungha Ko[1,2*] **Youngmin Lee**[1,2*] **Dong Hwan Kim**[1,2]

[1] Department of Computer Science and Engineering, Korea University
[2] Intelligence and Interaction Research Center, Korea Institute of Science and Technology
{rhqo123, aidi0724, gregorykim}@kist.re.kr

Abstract

Hierarchical 3D grouping aims to recover scene groups across multiple granularities, from fine object parts to complete objects, without relying on semantic labels or a fixed vocabulary. The main challenge is to transform 2D foundation-model cues into coherent hierarchy supervision and embed that hierarchy in a 3D representation. We propose H2G, a hyperbolic affinity field for hierarchical 3D grouping. Our method derives semantically organized tree supervision by interpreting foundation-model affinities through Dasgupta's objective for similarity-based hierarchical clustering. This supervision is distilled into a single Lorentz hyperbolic feature field, whose geometry is well suited for tree-like branching structures. A hierarchy-aware objective aligns the field with fine-level assignments, coarse object structure, compact feature clusters, and LCA (Lowest Common Ancestor) ordering. This formulation represents multiple grouping levels in one feature space, enabling semantic hierarchical grouping grounded in 2D foundation-model knowledge. Code is available at https://github.com/Rhqo/H2G.

1 Introduction

3D scene grouping is inherently hierarchical: the same point may belong to an object part, a whole object, or a larger object group depending on the desired granularity. Recent methods lift category-agnostic 2D masks, such as SAM[1] proposals, into NeRF-style[2] 3D fields for multi-view consistent grouping. The central challenge is not only to lift masks into 3D, but to organize overlapping and conflicting 2D proposals into a coherent hierarchy.

However, how these lifted masks should be organized into a hierarchy remains under-specified. GARField[3] resolves grouping ambiguity with a scale-conditioned affinity field, but this formulation ties hierarchy to depth-derived physical scale, which can be unreliable for large objects, partially observed objects, and amorphous background regions. It also requires repeated rendering over query scales. OmniSeg3D[4] and Ultrametric Feature Fields[5] avoid explicit scale queries through a unified feature-field representation, but the resulting hierarchies are still largely driven by mask-derived relations such as overlap, containment, or pixel-level correspondences among SAM-generated[1] masks. These relations provide useful boundary structure, but they do not necessarily align with semantic hierarchy.

Therefore, we propose H2G, a hyperbolic affinity field for hierarchical 3D grouping. Our method converts SAM[1] proposals into non-overlapping regions, computes DINO[6] descriptors for them, and constructs semantic affinity graphs over the resulting regions. Since their affinities are symmetric pairwise similarities rather than ordered hierarchy labels, we reconstruct them into tree supervision

*Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

using a top-down construction motivated by Dasgupta's objective[7] for similarity-based hierarchical clustering.

The resulting 2D hierarchy is distilled into a single 3D grouping field in Lorentz hyperbolic space. The negative curvature of hyperbolic space provides a natural bias for representing branching structures with many descendants, while our hierarchy-aware objective combines leaf separation, root separation, cluster compactness, and LCA (Lowest Common Ancestor) ordering. This enables scale-free hierarchical 3D grouping from semantic relations, without depth-based scale queries or repeated scale-conditioned rendering.

Our contributions are threefold:

- We introduce a semantic 2D hierarchy construction that converts SAM[1] proposals and DINO[6] region affinities into tree supervision using a top-down procedure motivated by Dasgupta's hierarchical clustering[7] objective.
- We formulate hierarchical 3D grouping as a scale-free Lorentz hyperbolic feature field, representing fine-to-coarse groups in a single embedding space without scale-conditioned rendering.
- We design a hierarchy-aware learning objective that combines leaf separation, root-level grouping, cluster compactness, and LCA-order preservation to align the 3D feature geometry with the constructed hierarchy.

The rest of this paper is organized as follows. Section 2 reviews neural feature fields, hierarchical 3D grouping, and hyperbolic representation learning. Section 3 describes the construction of 2D hierarchy supervision and its distillation into a Lorentz hyperbolic feature field. Section 4 presents quantitative and qualitative evaluations, followed by the conclusion.

2 Related work

Neural Feature Fields for 3D Scene Understanding Neural radiance fields (NeRF)[2] have been extended beyond RGB and density to store semantic, instance, language, and foundation-model features in 3D[8, 9, 10, 11, 12, 13, 14]. LERF[15], OpenScene[16], and OpenNeRF[17] align neural fields or 3D points with vision-language features for open-vocabulary scene understanding, while Contrastive Lift[18] shows that 2D segmentation cues can be lifted into 3D for instance segmentation. These works establish neural feature fields as a useful substrate for storing foundation-model signals in 3D. However, they typically support language queries, category retrieval, or flat instance labels, rather than class-agnostic hierarchical grouping. H2G instead uses feature fields to store hierarchy-aware grouping cues.

Hierarchical 3D Grouping from 2D Masks Closest to our setting are methods that lift multi-granular 2D masks, such as SAM[1] proposals, into view-consistent 3D representations[3, 4, 5, 19]. GARField[3] resolves grouping ambiguity with a scale-conditioned affinity field, Ultrametric Feature Fields[5] reveal hierarchy by thresholding feature distances, and OmniSeg3D[4] distills hierarchical 2D mask representations into a single 3D feature field. These methods show the importance of hierarchy, but differ in its source. Scale-conditioned inference requires repeated rendering across query scales, while mask overlap or containment may reflect proposal geometry more than semantic organization. H2G differs by reconstructing the 2D hierarchy from DINO[6] semantic affinities and embedding all grouping levels in a single Lorentz hyperbolic field.

Hyperbolic Representation Learning for Hierarchical Structure Hyperbolic representation learning[20, 21, 22, 23] is well suited to tree-like data because hyperbolic volume grows exponentially with radius, allowing branching structures to be embedded with lower distortion than in Euclidean space. This geometry has been used for symbolic hierarchies, hierarchical clustering[24, 25], and vision-language representations[26, 27]. MERU[26] models visual-semantic specificity in hyperbolic image-text space, HypHC[24] uses hyperbolic geometry for hierarchical clustering and LCA recovery, and OpenHype[28] introduces hyperbolic embeddings into open-vocabulary radiance fields. These works motivate hyperbolic geometry as an inductive bias for hierarchy, but do not directly address dense class-agnostic 3D grouping. H2G applies this inductive bias to dense 3D grouping by learning ray-level Lorentz features with angular and LCA-order objectives.

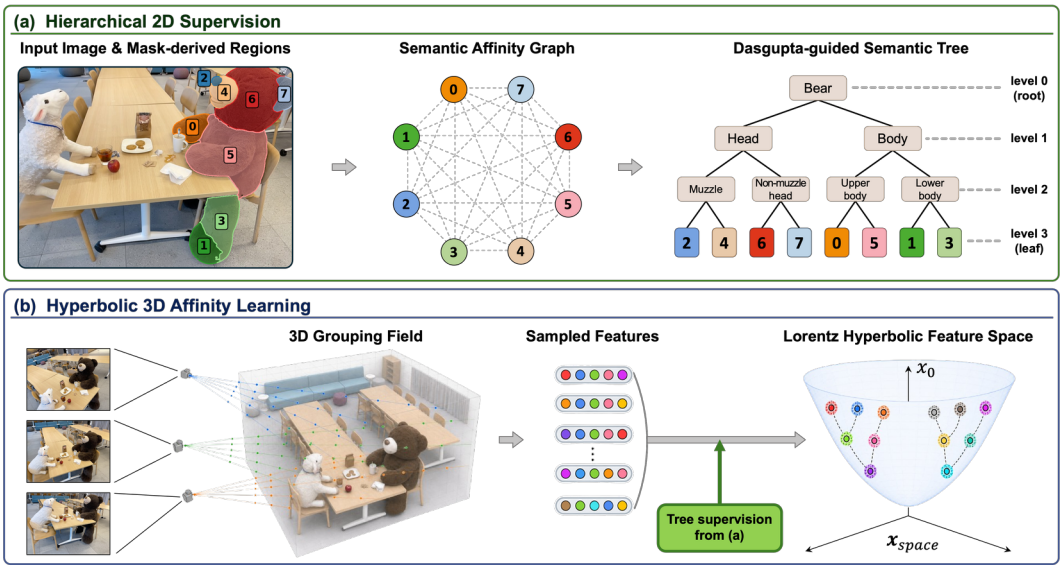

Figure 1: Overview of H2G. (a) SAM leaf masks are connected by DINO affinities and organized into a Dasgupta-guided semantic tree. (b) The tree supervises the rendered ray features of a NeRF-style grouping field in a Lorentz feature space. Text labels in the tree are illustrative; H2G does not use semantic class labels for supervision.

3 Method

Figure 1 summarizes the H2G pipeline. H2G first constructs image-wise hierarchy supervision from SAM[1] and DINO[6] cues, then distills it into a Lorentz hyperbolic feature field. The 2D stage converts overlapping SAM[1] proposals into non-overlapping leaf masks, organizes them into semantic trees using DINO[6] affinities, and provides leaf, root, and LCA relations for 3D learning. This design follows prior observations that SAM masks typically nest as whole, part, and subpart[1] and that the similarity of self-supervised features decreases from fine to coarse granularity[29].

3.1 Constructing 2D Hierarchy Supervision

Let $\{I^n\}_{n=1}^N$ denote the training images, where n is the image index. Given I^n, we first collect SAM[1] mask proposals and assign each proposal to its closest containing parent, defined as the smallest larger proposal whose containment ratio exceeds a threshold; proposals without parents become roots. Proposals sharing the same root are processed together, and their overlaps are resolved into fine non-overlapping leaf masks with unique ancestor paths. We denote one such leaf a by m_a^n.

Each leaf m_a^n receives a DINO[6] descriptor $\mathbf{z}_a^n$ by average-pooling the DINO feature map Φ^n over its covered patches and applying ℓ_2 normalization:

$$\bar{\mathbf{z}}_a^n = \frac{1}{|\Omega_a^n|} \sum_{\mathbf{u} \in \Omega_a^n} \Phi^n(\mathbf{u}), \quad \mathbf{z}_a^n = \frac{\bar{\mathbf{z}}_a^n}{\|\bar{\mathbf{z}}_a^n\|_2}.$$

where Ω_a^n is the set of covered DINO[6] patch indices and $\mathbf{u}$ indexes a DINO patch location. For each root-induced group indexed by i, we define an affinity graph $\mathcal{G}_i^n = (\mathcal{V}_{\text{leaf},i}^n, \mathcal{E}_i^n, \mathbf{W}_i^n)$, where $\mathcal{V}_{\text{leaf},i}^n$ contains the induced leaf vertices and $\mathcal{E}_i^n$ fully connects them. The affinity matrix entry for an edge $(u, v) \in \mathcal{E}_i^n$ for each group i is $(\mathbf{W}_i^n)_{uv} = W_{uv}^n = \max(0, (\mathbf{z}_u^n)^\top \mathbf{z}_v^n)$. The graph is symmetric, whereas the desired supervision is an ordered tree. Dasgupta's objective[7] for similarity-based hierarchical clustering provides the guiding principle for this conversion. For a tree $\mathcal{T}$ and pairwise similarities $\mathbf{W}$, the objective assigns cost

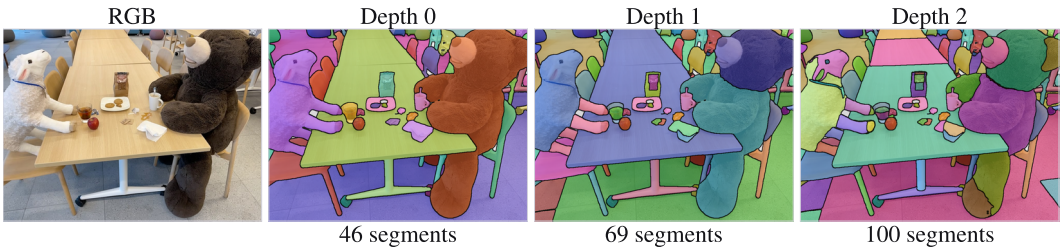

Figure 2: Example of 2D hierarchy supervision. Shallower leaves keep their own segments, and a segment keeps its color until it is split.

$$\mathcal{C}(\mathcal{T}; \mathbf{W}) = \sum_{a<b} W_{ab} \left|\mathrm{Leaf}\left(\mathrm{LCA}_{\mathcal{T}}(a, b)\right)\right|.$$

Here $\mathrm{LCA}_{\mathcal{T}}(a, b)$ is the lowest common ancestor of leaves a and b, and $\mathrm{Leaf}(v)$ is the set of leaf descendants under node v. This objective favors trees where high-affinity leaves meet in small subtrees. Dasgupta's top-down approximation recursively applies an α-approximation algorithm for sparsest cut[7], but solving such cuts on our dense DINO[6] graphs is expensive. We therefore use recursive spectral bisection[30] as a practical surrogate. This surrogate gives comparable costs with substantially lower preprocessing time; we report this comparison in Appendix G.

Starting from each affinity graph, we recursively split the current leaf set to build a tree. At each split, we compute the Fiedler vector of the normalized graph Laplacian of the induced DINO[6] affinity subgraph and divide leaves by its sign. Each split creates a virtual internal vertex whose children are the two resulting subtrees. Recursion continues until singleton leaves, with optional flattening of adjacent internal nodes for multi-way relations. Applying this process to all root-induced groups in image I^n yields a 2D hierarchy forest $\mathcal{T}^n = \{\mathcal{T}_i^n\}_{i=1}^{R_n}$, i.e., a collection of trees, one for each group.

Each tree $\mathcal{T}_i^n = (\mathcal{V}_{\mathcal{T},i}^n, \mathcal{E}_{\mathcal{T},i}^n)$ contains leaf vertices and virtual internal vertices, with $\mathcal{V}_{\mathcal{T},i}^n = \mathcal{V}_{\mathrm{leaf},i}^n \cup \mathcal{V}_{\mathrm{int},i}^n$. Thus, each labeled pixel or ray has a leaf label and a leaf-to-root ancestor chain $\mathcal{A}(y) = (y, \mathrm{par}(y), \ldots, \mathrm{root}(y))$.

Figure 2 shows an example of the resulting 2D hierarchy supervision, with each tree of $\mathcal{T}^n$ cut at depth d from its root. The forest encodes object–part relations, with the bear at depth 0 as the parent of its head and body at depth 1. The decomposition follows semantics rather than spatial connectivity, as the four disconnected legs of the sheep form one segment at depth 1.

The DINO[6] graph is used only to construct the tree; 3D learning uses the resulting leaf labels, root labels, ancestor relations, and LCA relations as hierarchy supervision.

3.2 Hyperbolic 3D Affinity Learning

Hyperbolic Affinity Field The 2D hierarchy provides supervision for pixels or rays in each training view, but the goal is not to learn independent 2D segmentations for each view. Instead, H2G learns a multi-view consistent 3D grouping field. To this end, we use a NeRF-style[2] 3D feature field. The RGB/density branch provides scene geometry and volume-rendering weights, while the grouping branch predicts features at the same 3D sample locations. These 3D sample features are rendered along each ray to obtain a ray-level grouping feature aligned with the 2D hierarchy.

We use the D-dimensional Lorentz model $\mathbb{H}_c^D \subset \mathbb{R}^{D+1}$ with curvature parameter $c > 0$. For $\mathbf{x} = (x_0, \mathbf{x}_{\mathrm{space}})$ and $\mathbf{y} = (y_0, \mathbf{y}_{\mathrm{space}})$, the Lorentz inner product is $\langle \mathbf{x}, \mathbf{y} \rangle_L = -x_0 y_0 + \mathbf{x}_{\mathrm{space}}^\top \mathbf{y}_{\mathrm{space}}$, and $d_{\mathbb{H}}$ denotes the corresponding geodesic distance[21]. A Euclidean network output $\mathbf{u} \in \mathbb{R}^D$ is projected to the Lorentz hyperboloid by

$$\Pi_c(\mathbf{u}) = \left(\sqrt{\frac{1}{c} + \|\mathbf{u}\|_2^2}, \mathbf{u} \right) \in \mathbb{H}_c^D.$$

Concretely, for sample points $\mathbf{x}_{r,k}$ on ray r, a hashgrid encoder[31] E_ψ predicts sample features, and a projection head G_ψ maps the rendered feature to a Euclidean tangent parameter before hyperbolic projection. We aggregate the sample features over a high-weight sample set $\mathcal{K}_r$. Let $\tilde{w}_{r,k}$ denote normalized rendering weights detached from the gradient path. The ray-level feature and its hyperbolic projection are

$$\bar{\mathbf{h}}_r = \sum_{k \in \mathcal{K}_r} \tilde{w}_{r,k} E_\psi(\mathbf{x}_{r,k}), \quad \mathbf{s}_r = \Pi_c\left(G_\psi(\bar{\mathbf{h}}_r)\right) \in \mathbb{H}^D_c.$$

Here $\mathbf{s}_r$ is the hyperbolic grouping feature for ray r, and it serves as the basic unit for the hierarchy-aware losses below.

Hierarchical Prototypes Direct pixel- or ray-wise contrastive learning over all pairs is costly and creates many redundant comparisons among rays from the same mask. Inspired by ProtoNCE [32], H2G treats each observed leaf mask as a prototype target: ray features assigned to the same leaf are aggregated into a single mask-level representative for contrastive supervision. This reduces the number of comparison targets and stabilizes learning by averaging noisy ray features within each leaf.

Let $\mathcal{B}$ be a mini-batch of rays sampled from image I^n. Each ray $r \in \mathcal{B}$ has a leaf label y_r from the non-overlapping leaf partition, and the observed leaf set is $\mathcal{V}_{\mathcal{B}}^{\text{leaf}} = \{y_r \mid r \in \mathcal{B}\}$. For batches spanning multiple images, the same definitions are applied per image. We instantiate prototypes only for leaves in $\mathcal{V}_{\mathcal{B}}^{\text{leaf}}$. Virtual internal vertices are not prototype targets; they provide root and LCA metadata through the corresponding forest. To respect the Lorentz geometry, we aggregate ray features using the Klein-coordinate-based Einstein midpoint[33]:

$$\kappa(\mathbf{x}) = \frac{\mathbf{x}_{\text{space}}}{x_0}, \quad \mathrm{E}_{\text{mid}}(\mathcal{X}) = \kappa^{-1}\left(\frac{\sum_{\mathbf{x}\in\mathcal{X}} \gamma(\mathbf{x})\kappa(\mathbf{x})}{\sum_{\mathbf{x}\in\mathcal{X}} \gamma(\mathbf{x})}\right).$$

Here $\gamma(\mathbf{x}) = (1 - \|\kappa(\mathbf{x})\|_2^2)^{-1/2}$, and κ^{-1} denotes the inverse map from Klein coordinates to the Lorentz hyperboloid[33].

Using this operator, the prototype for each observed leaf label $\ell \in \mathcal{V}_{\mathcal{B}}^{\text{leaf}}$ is computed from its assigned ray features: $\mathbf{p}_\ell = \mathrm{E}_{\text{mid}}\left(\{\mathbf{s}_r \mid y_r = \ell,\ r \in \mathcal{B}\}\right)$. The resulting prototype set is $\mathcal{P}_{\mathcal{B}}^{\text{leaf}} = \{\mathbf{p}_\ell \mid \ell \in \mathcal{V}_{\mathcal{B}}^{\text{leaf}}\}$.

Angular Separation Loss Following the hyperbolic angular formulation of ATMG [27], H2G uses angular classification at two levels. The leaf-level loss aligns each ray feature with its assigned leaf prototype for fine discrimination, while the root-level loss aligns each leaf prototype with its root centroid for coarse group separation. Both losses use the same hyperbolic exterior angle. For the hyperbolic triangle formed by the origin O, a reference point $\mathbf{p}$, and a query point $\mathbf{s}$, we use the angle at $\mathbf{p}$, computed in the Lorentz model as

$$\theta(\mathbf{p}, \mathbf{s}) = \cos^{-1}\left(\frac{s_0 + p_0 c\langle \mathbf{p}, \mathbf{s}\rangle_L}{\|\mathbf{p}_{\text{space}}\|_2 \sqrt{(c\langle \mathbf{p}, \mathbf{s}\rangle_L)^2 - 1}}\right).$$

A smaller $\theta(\mathbf{p}, \mathbf{s})$ indicates stronger directional alignment. We use angle rather than hyperbolic distance so that semantic identity is organized mainly by direction, while radial position remains available for compactness and hierarchy ordering.

For leaf-level supervision, each ray feature $\mathbf{s}_r$ is classified against the leaf prototypes observed in the current batch:

$$\mathcal{L}_{\text{leaf}} = -\frac{1}{|\mathcal{B}|} \sum_{r\in\mathcal{B}} \log \frac{\exp\left(-\theta(\mathbf{p}_{y_r}, \mathbf{s}_r)/\tau_{\text{leaf}}\right)}{\sum_{v\in\mathcal{V}_{\mathcal{B}}^{\text{leaf}}} \exp\left(-\theta(\mathbf{p}_v, \mathbf{s}_r)/\tau_{\text{leaf}}\right)}.$$

This loss separates leaves in angular space while aligning rays with their assigned leaf prototype.

Leaf-level separation alone does not guarantee that different root-level groups are well separated. The purpose of the root-level loss is not to classify individual rays again, but to make leaf-level representatives under the same root share a coarse group direction. Therefore, the classified queries are leaf prototypes rather than ray features. Let $\mathcal{R}_{\mathcal{B}} = \{\mathrm{root}(\ell) \mid \ell \in \mathcal{V}_{\mathcal{B}}^{\mathrm{leaf}}\}$ be the set of roots observed in the current batch. For each root $\rho \in \mathcal{R}_{\mathcal{B}}$, we compute a root centroid as the Einstein midpoint of all ray features in the batch whose leaves descend from ρ:

$$\mathbf{q}_\rho = \mathrm{E}_{\mathrm{mid}}\left(\{\mathbf{s}_r \mid \mathrm{root}(y_r) = \rho,\ r \in \mathcal{B}\}\right).$$

Each leaf prototype $\mathbf{p}_\ell$ then uses its root $\mathrm{root}(\ell)$ as the target class and is classified against the root centroids $\{\mathbf{q}_\rho\}$:

$$\mathcal{L}_{\mathrm{root}} = -\frac{1}{|\mathcal{V}_{\mathcal{B}}^{\mathrm{leaf}}|} \sum_{\ell \in \mathcal{V}_{\mathcal{B}}^{\mathrm{leaf}}} \log \frac{\exp\left(-\theta(\mathbf{q}_{\mathrm{root}(\ell)}, \mathbf{p}_\ell)/\tau_{\mathrm{root}}\right)}{\sum_{\rho \in \mathcal{R}_{\mathcal{B}}} \exp\left(-\theta(\mathbf{q}_\rho, \mathbf{p}_\ell)/\tau_{\mathrm{root}}\right)}.$$

This term aligns leaf prototypes with their corresponding root centroids, encouraging leaves under the same root to share a coarse direction.

Compactness Loss Angle-based supervision separates semantic directions, but it does not control within-cluster spread. The compactness loss contracts only ray features that lie outside a margin around their assigned prototype.

For each ray feature $\mathbf{s}_r$, we penalize the excess hyperbolic distance to its assigned leaf prototype $\mathbf{p}_{y_r}$:

$$\mathcal{L}_{\mathrm{comp}} = \frac{1}{|\mathcal{B}|} \sum_{r \in \mathcal{B}} \left(ReLU(d_{\mathbb{H}}\left(\mathbf{s}_r, \mathrm{sg}(\mathbf{p}_{y_r})\right) - \epsilon)\right)^2.$$

Here $\mathrm{sg}(\cdot)$ denotes the stop-gradient operation, and ϵ is the compactness margin. Thus, rays within the margin receive no compactness gradient, while rays outside the margin are pulled toward the current prototype.

LCA Ordering Loss Angular losses separate leaves and roots, but they do not encode the merging order inside each tree. We adopt the LCA-order surrogate from HypHC [24] and apply it to tree-derived leaf prototypes in our 3D grouping field. LCA order therefore ranks prototype pairs by tree-derived ancestor depth. Let $\mathbf{k}_i = \kappa(\mathbf{p}_i)$ and $\mathbf{k}_j = \kappa(\mathbf{p}_j)$ be the Klein coordinates of two leaf prototypes. Since Klein geodesics are straight segments, the continuous LCA surrogate is approximated by the point on the segment closest to the origin:

$$t_{ij} = \mathrm{clip}_{[0,1]}\left(-\frac{\mathbf{k}_i^\top(\mathbf{k}_j - \mathbf{k}_i)}{\|\mathbf{k}_j - \mathbf{k}_i\|_2^2}\right), \quad \widehat{\mathbf{k}}_{ij} = \mathbf{k}_i + t_{ij}(\mathbf{k}_j - \mathbf{k}_i).$$

We use the hyperbolic distance from the origin to this point as a pairwise LCA-depth surrogate:

$$d_o(i,j) = \frac{1}{\sqrt{c}} \cosh^{-1}\left(\frac{1}{\sqrt{1 - \|\widehat{\mathbf{k}}_{ij}\|_2^2}}\right).$$

Triplets $(i,j,k) \in \mathcal{P}_{\mathrm{LCA}}$ are sampled so that i and j come from the same parent, while k comes from a different sibling branch under the same grandparent. The desired relation is

$$d_o(i,j) > d_o(i,k), \quad d_o(i,j) > d_o(j,k).$$

This relation is optimized with a softmax ranking objective:

$$\mathcal{L}_{\mathrm{LCA}} = -\frac{1}{|\mathcal{P}_{\mathrm{LCA}}|} \sum_{(i,j,k) \in \mathcal{P}_{\mathrm{LCA}}} \log \frac{\exp(d_o(i,j)/\tau_{\mathrm{LCA}})}{\exp(d_o(i,j)/\tau_{\mathrm{LCA}}) + \exp(d_o(i,k)/\tau_{\mathrm{LCA}}) + \exp(d_o(j,k)/\tau_{\mathrm{LCA}})}.$$

This loss enforces only relative LCA-depth ordering, preserving the merge order of the 2D hierarchy without regressing absolute pairwise distances.

Overall Objective The final training objective is the weighted sum of the four hierarchy losses and a max-norm regularizer:

$$\mathcal{L} = \lambda_{\text{leaf}}\mathcal{L}_{\text{leaf}} + \lambda_{\text{root}}\mathcal{L}_{\text{root}} + \lambda_{\text{comp}}\mathcal{L}_{\text{comp}} + \lambda_{\text{LCA}}\mathcal{L}_{\text{LCA}} + \lambda_{\text{norm}}\mathcal{L}_{\text{norm}}.$$

Rather than using a single contrastive objective for all hierarchy relations, H2G decomposes supervision into four complementary constraints: leaf-level angular separation, root-level angular grouping, within-leaf compactness, and LCA-order preservation. The max-norm regularizer only prevents Lorentz features from drifting too far from the origin and does not define hierarchy relations.

Through this decomposition, H2G distills the image-wise hierarchy obtained from SAM[1] and DINO[6] into a 3D hyperbolic affinity field. Each ray therefore receives a hyperbolic feature that reflects not only a leaf-level grouping label, but also root-level group structure and tree ordering.

H2G uses a NeRF[2] framework and training setup that closely follows GARField [3], while replacing scale-conditioned Euclidean grouping features with Lorentz features trained by our hierarchy-aware objective. Additional architecture and training details are provided in Appendix A.

4 Experiments

We evaluate whether H2G can identify and recover 3D groups across fine-to-coarse levels of detail within a single learned feature space.

4.1 Dataset

We evaluate on the GARField[3] scenes with hierarchical annotations. For 3D Completeness, each scene provides three evaluation views with query points corresponding to the same 3D location and ground-truth masks at two or three hierarchy levels. For group recall, each scene provides ground-truth hierarchical groups for best-match proposal evaluation.

Benchmark protocol Following the GARField 3D completeness protocol[3], we use query-conditioned mask retrieval across hierarchy levels. H2G renders each view once and obtains candidate masks by thresholding a tangent-space affinity map computed from the query feature, without scale input or per-level rendering. Additional evaluation details are provided in Appendix B.

For fair comparison, we compare H2G against baselines reproduced under the same evaluation protocol, denoted by *. Published SAM[1] and GARField[3] results are included only as references. We report OmniSeg3D[4] in Appendix B.3 because reproduced results are unavailable for one evaluation scene; the main table focuses on baselines with complete scene coverage.

4.2 3D Completeness

We report 3D completeness under two thresholding settings: view-wise selection, which chooses the best threshold for each view and level, and level-wise selection, which shares one threshold per hierarchy level across views. Each level is averaged over the scenes that annotate it, and the Mean averages the resulting level values. All averages over scenes and levels in this paper follow this convention, detailed in Appendix B.1.

Table 1 shows that H2G achieves the best reproduced mean in both settings, improving over GARField* from 0.749 to 0.835 with view-wise thresholds and from 0.731 to 0.826 with level-wise thresholds. The gains are largest at the medium level, where semantic grouping is less directly tied to physical scale, supporting the role of semantic hierarchy supervision. The full scene-wise breakdown is reported in Appendix B.3; on `living_room`, GARField* remains slightly better in the view-wise mean.

We also introduce Cross-view Threshold Retention (CTR), the ratio between level-wise and view-wise 3D Completeness averaged over scenes. CTR measures how much performance is retained when one threshold is shared across views for each hierarchy level.

Table 1: Average 3D Completeness under view-wise (V) and level-wise (L) threshold selection. * denotes methods evaluated under our reproduced setting.

	Fine		Medium		Coarse		Mean	
Model	V	L	V	L	V	L	V	L
SAM[1]	0.594	–	0.741	–	0.887	–	0.741	–
GARField[3]	0.846	–	0.843	–	0.933	–	0.874	–
SAM*	0.557	–	0.742	–	**0.901**	–	0.733	–
GARField*	0.824	0.823	0.607	0.587	0.815	0.784	0.749	0.731
Ours	**0.848**	**0.840**	**0.813**	**0.809**	0.844	**0.828**	**0.835**	**0.826**

Table 2: Cross-view Threshold Retention (CTR). Each value is computed as level-wise 3D Completeness divided by view-wise 3D Completeness for each scene, then averaged across scenes.

Method	Fine	Medium	Coarse	Mean
GARField*	**0.998**	0.967	0.959	0.975
Ours	0.990	**0.995**	**0.978**	**0.988**

Table 2 shows that both methods retain most of their view-wise performance, while H2G has higher overall retention. Since both methods use a shared 3D representation, CTR should not be interpreted as multi-view consistency alone; rather, it reflects how consistently hierarchy levels are placed along each method's inference variable. The higher mean CTR suggests that H2G calibrates medium and coarse levels more consistently in feature affinity, without relying on an explicit scale query.

4.3 Group Recall

We follow the GARField group-recall protocol[3] to evaluate whether each method generates useful group candidates. GARField[3] accumulates candidates from scale-conditioned renderings, while H2G generates candidates from a single rendered feature field. We report native recall, fixed-budget recall, and budget-curve AUC; details are in Appendix B.

Table 3 shows that H2G produces a substantially smaller proposal pool while achieving higher native-pool mIoU and recall. H2G also outperforms GARField* at every evaluated proposal budget across mIoU, $R@0.50$, and $R@0.75$, and obtains higher budget-curve AUC. These results indicate that the learned hierarchy does not merely recover groups through dense proposal enumeration; instead, it concentrates useful hierarchical groups into a compact candidate pool.

4.4 Effect of Hyperbolic Geometry

To isolate the effect of the geometry, we train a Euclidean counterpart of H2G whose grouping features lie on the unit hypersphere, the standard Euclidean embedding in contrastive learning [34, 35], with each hyperbolic operation replaced by its Euclidean counterpart. We compare both geometries at dimensions $D = 256$ and $D = 16$ under the same 2D supervision, as detailed in Appendix C.

Table 4 shows that Lorentz features achieve a higher mean at both dimensions, but the gain depends on the scene and the dimension. At $D = 256$, the two geometries are nearly identical on `bouquet` and `teatime`, while `ramen` and `living_room` clearly favor the Lorentz field, so with enough dimensions a Euclidean field can also represent the hierarchy on some scenes. At $D = 16$, the Lorentz field leads on every scene. On three of the four scenes, it loses little when D is reduced from 256 to 16, whereas the Euclidean field drops noticeably, consistent with prior work showing that hyperbolic space can represent hierarchies with few dimensions [20].

4.5 Ablation study

Table 5 shows that angular separation is the most important component for fine-level grouping, while the root-level term controls the trade-off between local and coarse grouping. Removing the coarse term improves Fine and Medium scores, but sharply reduces Coarse completeness, leading to a lower overall mean. This suggests that coarse supervision intentionally sacrifices some local

Table 3: Group recall. Native uses all generated candidates. Budgeted rows sample at most K candidates without ground-truth information and average over 100 seeds. AUC values are normalized to the range of each metric.

	GARField*			Ours		
Budget	mIoU	R@0.50	R@0.75	mIoU	R@0.50	R@0.75
Candidates		2010.8			314.8	
Native	0.758	0.892	0.619	**0.780**	**0.896**	**0.746**
$K = 150$	0.368	0.373	0.210	**0.541**	**0.580**	**0.413**
$K = 500$	0.613	0.692	0.439	**0.780**	**0.896**	**0.746**
$K = 1000$	0.710	0.831	0.563	**0.780**	**0.896**	**0.746**
$K = 2000$	0.747	0.884	0.612	**0.780**	**0.896**	**0.746**
AUC_{500}	0.427	0.457	0.274	**0.596**	**0.662**	**0.521**
AUC_{2000}	0.641	0.741	0.494	**0.734**	**0.837**	**0.689**

Table 4: Scene-wise 3D Completeness (view-wise) of unit-norm Euclidean and Lorentz features with identical supervision and objective. Scene rows report per-scene means, and Δ denotes the difference of Lorentz relative to Euclidean.

	$D = 16$			$D = 256$		
Scene	Euclidean	Lorentz	Δ	Euclidean	Lorentz	Δ
bouquet	0.676	**0.742**	+0.066	0.749	**0.750**	+0.001
teatime	0.916	**0.953**	+0.037	0.951	**0.952**	+0.001
ramen	0.662	**0.807**	+0.145	0.631	**0.859**	+0.228
living_room	0.729	**0.815**	+0.086	0.776	**0.824**	+0.048
Mean	0.734	**0.817**	+0.083	0.761	**0.835**	+0.074



Figure 3: PCA (Principal component analysis) visualization of rendered grouping features. From left to right, the scenes are `teatime`, `ramen`, `living_room`, and `bouquet`. GARField is shown at scale 0 for comparison.

thresholdability to preserve larger root-level groups. Compactness also improves the thresholdability of groups; LCA-order has a smaller effect on this metric but regularizes the relative merge order of leaf prototypes.

4.6 Qualitative results

Figures 3 and 4 visualize properties that are not fully captured by IoU-based scores. Compared with GARField[3], H2G produces clearer feature regions on large surfaces and more structured fine-grained clusters around object parts in cluttered or visually ambiguous regions. These qualitative differences suggest that part- and object-level groups form coherent neighborhoods in a single feature space, even when the final oracle IoU does not increase for every annotated group.

Table 5: Ablation study on the proposed loss components.

Variant	Fine	Medium	Coarse	Mean
Full	0.848	0.813	**0.844**	**0.835**
w/o angle	0.631	0.774	0.830	0.745
w/o coarse	**0.862**	**0.878**	0.585	0.775
w/o compact	0.843	0.771	0.782	0.799
w/o LCA	0.847	0.806	0.839	0.831



Figure 4: HDBSCAN clustering of rendered grouping features. From left to right, the scenes are `teatime`, `ramen`, `living_room`, and `bouquet`. GARField clusters are visualized from its recursive scale-sweep refinement, while H2G clusters are obtained from a single rendered feature field.

5 Limitations

H2G depends on the quality of its 2D supervision. Noisy SAM[1] proposals or unreliable DINO[6] affinities can produce imperfect hierarchy trees, especially for small, occluded, or visually ambiguous regions. Since trees are constructed per image, some cross-view inconsistencies may remain after 3D distillation. Moreover, recursive spectral bisection is a practical approximation to hierarchy construction, and final mask extraction still requires thresholding or clustering in the learned feature space. Finally, our evaluation is limited to the four scenes of the GARField benchmark[3], which to our knowledge is the only public 3D dataset with both cross-view hierarchical query masks and hierarchical group annotations.

6 Conclusion

We presented H2G, a hierarchy-aware hyperbolic grouping field for 3D scenes. H2G reconstructs tree supervision from SAM[1] proposals and DINO[6] affinities, and distills it into a single Lorentz feature field through leaf, root, compactness, and LCA-order objectives. On the GARField 3D Completeness benchmark[3], H2G improves over reproduced baselines under both view-wise and level-wise threshold selection, especially at medium granularity. These results show that semantic hierarchy supervision and hyperbolic geometry provide an effective basis for hierarchical 3D grouping without scale-conditioned rendering.

Acknowledgments

This work was supported by the KIST Institutional Program (Project No. 26E0210, 26E0213).

References

[1] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023.

[2] 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.

[3] Chung Min Kim, Mingxuan Wu, Justin Kerr, Ken Goldberg, Matthew Tancik, and Angjoo Kanazawa. Garfield: Group anything with radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21530–21539, 2024.

[4] Haiyang Ying, Yixuan Yin, Jinzhi Zhang, Fan Wang, Tao Yu, Ruqi Huang, and Lu Fang. Omniseg3d: Omniversal 3d segmentation via hierarchical contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20612–20622, 2024.

[5] Haodi He, Colton Stearns, Adam W Harley, and Leonidas J Guibas. View-consistent hierarchical 3d segmentation using ultrametric feature fields. In *European Conference on Computer Vision*, pages 268–286. Springer, 2024.

[6] Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

[7] Sanjoy Dasgupta. A cost function for similarity-based hierarchical clustering. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 118–127, 2016.

[8] Suhani Vora*, Noha Radwan*, Klaus Greff, Henning Meyer, Kyle Genova, Mehdi S. M. Sajjadi, Etienne Pot, Andrea Tagliasacchi, and Daniel Duckworth. Nesf: Neural semantic fields for generalizable semantic segmentation of 3d scenes. *Transactions on Machine Learning Research*, 2022. https://openreview.net/forum?id=ggPhsYCsm9.

[9] 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.

[10] Kunhao Liu, Fangneng Zhan, Jiahui Zhang, Muyu Xu, Yingchen Yu, Abdulmotaleb El Saddik, Christian Theobalt, Eric Xing, and Shijian Lu. Weakly supervised 3d open-vocabulary segmentation. *Advances in Neural Information Processing Systems*, 36:53433–53456, 2023.

[11] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J. Davison. In-place scene labelling and understanding with implicit scene representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15838–15847, October 2021.

[12] Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Bulò, Norman Müller, Matthias Nießner, Angela Dai, and Peter Kontschieder. Panoptic lifting for 3d scene understanding with neural fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9043–9052, June 2023.

[13] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas J. Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic neural fields: A semantic object-aware neural scene representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12871–12881, June 2022.

[14] William Shen, Ge Yang, Alan Yu, Jansen Wong, Leslie Pack Kaelbling, and Phillip Isola. Distilled feature fields enable few-shot language-guided manipulation. In *7th Annual Conference on Robot Learning*, 2023.

[15] 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*, pages 19729–19739, 2023.

[16] Songyou Peng, Kyle Genova, Chiyu Jiang, Andrea Tagliasacchi, Marc Pollefeys, Thomas Funkhouser, et al. Openscene: 3d scene understanding with open vocabularies. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 815–824, 2023.

[17] Francis Engelmann, Fabian Manhardt, Michael Niemeyer, Keisuke Tateno, Marc Pollefeys, and Federico Tombari. OpenNeRF: Open Set 3D Neural Scene Segmentation with Pixel-Wise Features and Rendered Novel Views. In *International Conference on Learning Representations*, 2024.

[18] Yash Bhalgat, Iro Laina, João F Henriques, Andrew Zisserman, and Andrea Vedaldi. Contrastive lift: 3d object instance segmentation by slow-fast contrastive fusion. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[19] Jiazhong Cen, Zanwei Zhou, Jiemin Fang, Chen Yang, Wei Shen, Lingxi Xie, Xiaopeng Zhang, and Qi Tian. Segment anything in 3d with nerfs. In *NeurIPS*, 2023.

[20] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30, 2017.

[21] Maximillian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International conference on machine learning*, pages 3779–3788. PMLR, 2018.

[22] Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. Hyperbolic image embeddings. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6418–6428, 2020.

[23] Wei Peng, Tuomas Varanka, Abdelrahman Mostafa, Henglin Shi, and Guoying Zhao. Hyperbolic deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):10023–10044, 2022.

[24] Ines Chami, Albert Gu, Vaggos Chatziafratis, and Christopher Ré. From trees to continuous embeddings and back: Hyperbolic hierarchical clustering. *Advances in neural information processing systems*, 33:15065–15076, 2020.

[25] Teng Long and Nanne van Noord. Cross-modal scalable hyperbolic hierarchical clustering. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16655–16664, 2023.

[26] Karan Desai, Maximilian Nickel, Tanmay Rajpurohit, Justin Johnson, and Shanmukha Ramakrishna Vedantam. Hyperbolic image-text representations. In *International Conference on Machine Learning*, pages 7694–7731. PMLR, 2023.

[27] Sameera Ramasinghe, Violetta Shevchenko, Gil Avraham, and Ajanthan Thalaiyasingam. Accept the modality gap: An exploration in the hyperbolic space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27263–27272, 2024.

[28] Lisa Weijler, Sebastian Koch, Fabio Poiesi, Timo Ropinski, and Pedro Hermosilla. Openhype: Hyperbolic embeddings for hierarchical open-vocabulary radiance fields. *NeurIPS*, 2025.

[29] Simone Rossetti and Fiora Pirri. Hierarchy-agnostic unsupervised segmentation: Parsing semantic image structure. *Advances in Neural Information Processing Systems*, 37:98898–98935, 2024.

[30] Miroslav Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak mathematical journal*, 25(4):619–633, 1975.

[31] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022.

[32] Junnan Li, Pan Zhou, Caiming Xiong, and Steven C.H. Hoi. Prototypical contrastive learning of unsupervised representations. In *ICLR*, 2021.

[33] Abraham Albert Ungar. *Analytic hyperbolic geometry: Mathematical foundations and applications*. World Scientific, 2005.

[34] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR, 2020.

[35] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International conference on machine learning*, pages 9929–9939. PMLR, 2020.

[36] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[37] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. Density-based clustering based on hierarchical density estimates. In Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu, editors, *Advances in Knowledge Discovery and Data Mining*, pages 160–172, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[38] Yunhui Guo, Haoran Guo, and Stella X Yu. Co-sne: Dimensionality reduction and visualization for hyperbolic data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21–30, 2022.

A Implementation details

We train H2G in two stages. First, we optimize the RGB radiance field for 30k iterations. We then freeze the RGB field, proposal networks, and camera optimizer, and train only the semantic grouping field for 20k iterations with hierarchy supervision enabled from the beginning.

For 2D hierarchy preprocessing, SAM[1] masks are generated in automatic mode and sorted by area. We form top-down proposal groups using a containment threshold of 0.8, convert overlapping masks into non-overlapping leaf masks by assigning pixels to the smallest covering mask, and apply morphological closing and connected-component filtering. DINO[6] descriptors are mean-pooled over the resulting leaf masks, and neighboring internal nodes with a small height gap can be flattened to represent multi-way relations.

The grouping feature dimension is 256. During semantic training, we sample 8192 rays from one image per iteration and optimize the semantic branch with Adam. The initial learning rate is 10^{-3}, the final learning rate is 10^{-4}, warmup lasts 1000 steps, weight decay is 10^{-6}, and gradients are clipped with maximum norm 1.0. Unless otherwise stated, we use $\lambda_{\text{leaf}} = 1.0$, $\lambda_{\text{root}} = 2.0$, $\lambda_{\text{comp}} = 1.0$, $\lambda_{\text{LCA}} = 0.05$, and $\lambda_{\text{norm}} = 1.0$, with temperatures $\tau_{\text{leaf}} = 0.22$, $\tau_{\text{root}} = 0.20$, and $\tau_{\text{LCA}} = 0.5$, compactness margin $\epsilon = 0.1$, and curvature $c = 1.0$. All experiments were run on a local workstation equipped with NVIDIA RTX 3090 GPUs, each with 24GB VRAM, except the inference timings in Appendix F, which were measured on an NVIDIA RTX 4090. Generating the 2D hierarchy supervision takes approximately 30 minutes per scene, and hierarchy-supervised grouping-field training takes approximately 20 minutes per scene.

A.1 Reproduced Baselines

For the reproduced baselines, we used the same GARField[3] evaluation scenes and query annotations, and matched the available SAM ViT-H checkpoint and mask-generation settings as closely as possible. We observed differences from the published results, which can arise from checkpoint availability, implementation details, environment differences, and the sensitivity of scale-conditioned grouping to ambiguous same-scale groups. The GARField* rows in Tables 1 and 8 come from a single run at the default seed (42), and we did not search over seeds. Below, we re-evaluate both GARField and H2G over five runs with seeds 0–4.

Seed sensitivity Neither the GARField paper [3] nor its official implementation specifies a seed, and the evaluation protocol document distributed with the code notes that grouping conflicts within a single scale can resolve either way and that, in this case, "perhaps changing the random seed could be helpful." We reran GARField five times, varying the seed of the official implementation. The spread differed sharply across cells (scene $\times$ level): `teatime` Coarse ranged over 0.752–0.964 and `ramen` Medium over 0.542–0.904, while some cells stayed within ± 0.007.

Matched-seed comparison Running H2G over the same seeds, under the same SAM masks and the same protocol, gives the 3D Completeness in Table 6. H2G still leads in mean, although by a smaller margin than in the single-seed Table 1. The two methods differ in how evenly they cover the levels. GARField is strongest at Coarse but weakest at Medium, where its run-to-run standard deviation is also an order of magnitude larger than that of H2G. H2G's three levels stay close to one another, so even its weakest level remains well above GARField's Medium.

Table 6: 3D Completeness (view-wise) over seeds 0–4, aggregated as in Table 1. Each entry is the mean ± standard deviation over the five seeds.

Method	Fine	Medium	Coarse	Mean
GARField*	0.826 ± 0.009	0.669 ± 0.075	$\mathbf{0.900} \pm 0.020$	0.798 ± 0.029
Ours	$\mathbf{0.852} \pm 0.007$	$\mathbf{0.801} \pm 0.004$	0.832 ± 0.014	$\mathbf{0.829} \pm 0.004$

Dependence on SAM Both methods take SAM[1] masks as their input, so what SAM produces carries directly into their 3D results. For these runs, we used the checkpoint GARField specifies (`sam_vit_h_4b8939.pth`) with the implementation that checkpoint belongs to. The pipeline nevertheless turned out to be sensitive to how SAM is run. The original release (fb) and the HuggingFace port (hf) of that same checkpoint are indistinguishable in 2D under our evaluation, yet they lead GARField to different 3D results (Table 7). This does not suggest that we used a different SAM from the authors; it indicates how far GARField's 3D output can move on details that leave 2D quality untouched.

Table 7: Sensitivity to the SAM implementation. Masks come from the original release (fb) or the HuggingFace port (hf) of the same SAM ViT-H checkpoint. 2D evaluates the SAM masks alone, as for SAM* in Table 1; 3D is the view-wise 3D Completeness of GARField* averaged over seeds 0–4. Scene rows are scene means, and the Mean row follows Appendix B.1.

Scene	SAM masks, 2D (fb / hf)	GARField*, 3D (fb / hf)
teatime	0.845 / 0.844	0.902 / 0.911
bouquet	0.514 / 0.514	0.748 / 0.673
ramen	0.799 / 0.801	0.789 / 0.721
living_room	0.809 / 0.809	0.832 / 0.813
Mean	0.733 / 0.733	0.798 / 0.754

Like the seed, other settings that neither the GARField paper nor its code specifies may have influenced the published results. We therefore mark every reproduced result with *, and our claims are made against these reproduced baselines rather than against the reference results.

B Evaluation Details

B.1 3D Completeness Protocol

For each scene, we evaluate H2G using the annotated camera poses, query pixels, and COCO-format[36] hierarchy masks provided by the GARField 3D Completeness benchmark[3]. Each evaluation view contains a query pixel corresponding to the same 3D point, and each hierarchy level provides a ground-truth mask that contains the query. H2G renders the RGB image and grouping feature field once for each evaluation camera; unlike scale-conditioned methods, the renderer does not receive a separate scale or hierarchy-level input.

For the reported H2G scores, we use tangent-space query affinity. Let $f(p)$ be the rendered Lorentz feature at pixel p, and let q be the query pixel. Features are mapped to the tangent space at the origin with the Lorentz logarithmic map, and the query-conditioned score is computed as

$$z(p) = \log_0(f(p); c), \quad \text{score}_q(p) = \exp\left(-\|z(p) - z(q)\|_2^2\right).$$

A candidate mask for threshold t is $M_q(t) = \{p \in I \mid \text{score}_q(p) \geq t\}$. In the view-wise setting, we sweep 1000 linearly spaced thresholds in $[0.0, 0.99]$ independently for each evaluation view and annotated hierarchy level, then select the threshold that maximizes IoU with the corresponding ground-truth mask. The per-level score is the mean IoU over the three evaluation views. In the level-wise setting, we use the same tangent affinity maps but select one threshold per hierarchy level by maximizing the total IoU across all evaluation views, then evaluate every view at that level with the shared threshold. The reported mean averages the available Fine, Medium, and Coarse levels for each scene.

Table 8: Scene-wise 3D Completeness under view-wise (V) and level-wise (L) threshold selection. * denotes methods evaluated under our reproduced setting. OmniSeg3D* is not included in the mean rows because its `living_room` result is unavailable.

		Fine		Medium		Coarse		Mean	
Scene	Model	V	L	V	L	V	L	V	L
teatime	SAM[1]	0.816	–	–	–	0.973	–	0.895	–
	GARField[3]	0.927	–	–	–	0.979	–	0.953	–
	SAM*	0.714	–	–	–	**0.976**	–	0.845	–
	GARField*	0.893	0.893	–	–	0.754	0.633	0.824	0.763
	OmniSeg3D*	0.306	0.281	–	–	0.249	0.192	0.278	0.237
	Ours	**0.931**	**0.928**	–	–	0.973	**0.972**	**0.952**	**0.950**
bouquet	SAM[1]	0.174	–	0.735	–	0.761	–	0.557	–
	GARField[3]	0.760	–	0.816	–	0.854	–	0.810	–
	SAM*	0.126	–	0.584	–	**0.831**	–	0.514	–
	GARField*	**0.785**	**0.785**	0.617	0.563	0.632	**0.632**	0.678	0.660
	OmniSeg3D*	0.388	0.388	0.702	0.697	0.267	0.254	0.453	0.447
	Ours	0.753	0.748	**0.862**	**0.858**	0.635	0.601	**0.750**	**0.736**
ramen	SAM[1]	0.533	–	0.747	–	0.926	–	0.735	–
	GARField[3]	0.792	–	0.907	–	0.955	–	0.885	–
	SAM*	0.538	–	0.916	–	**0.944**	–	0.799	–
	GARField*	0.754	0.749	0.507	0.507	0.937	**0.936**	0.733	0.731
	OmniSeg3D*	0.484	0.483	0.813	0.800	0.190	0.161	0.496	0.481
	Ours	**0.823**	**0.806**	**0.916**	**0.913**	0.838	0.811	**0.859**	**0.843**
living_room	SAM[1]	0.853	–	0.742	–	0.886	–	0.827	–
	GARField[3]	0.905	–	0.807	–	0.944	–	0.885	–
	SAM*	0.850	–	**0.727**	–	0.851	–	0.809	–
	GARField*	0.863	0.863	0.698	**0.690**	**0.936**	**0.935**	**0.832**	**0.829**
	OmniSeg3D*	–	–	–	–	–	–	–	–
	Ours	**0.883**	**0.876**	0.660	0.656	0.929	0.928	0.824	0.820
mean	SAM[1]	0.594	–	0.741	–	0.887	–	0.741	–
	GARField[3]	0.846	–	0.843	–	0.933	–	0.874	–
	SAM*	0.557	–	0.742	–	**0.901**	–	0.733	–
	GARField*	0.824	0.823	0.607	0.587	0.815	0.784	0.749	0.731
	Ours	**0.848**	**0.840**	**0.813**	**0.809**	0.844	**0.828**	**0.835**	**0.826**

Aggregation over scenes and levels The evaluation scenes do not annotate the same set of hierarchy levels. `bouquet`, `ramen`, and `living_room` provide Fine, Medium, and Coarse masks, whereas `teatime` provides two levels, which we assign to Fine and Coarse. Each level is averaged over the scenes that annotate it, i.e., Fine and Coarse over four scenes and Medium over three, and the Mean averages the resulting level values. The Mean is therefore not the average of the scene-wise Means. This applies to both view-wise and level-wise scores in Tables 1 and 8, while a scene-wise Mean in Table 8 averages the levels annotated in that scene. CTR in Table 2 follows the same order: the ratio is computed per scene and level, averaged per level over the scenes that annotate it, and the Mean averages the level values. For the published SAM[1] and GARField[3] results, we take the scene-wise values and recompute every average under this convention. Group recall has no level annotations, so the aggregate values in Table 3 are averages over the four scenes.

B.2 Group Recall Protocol

For GARField*, we keep its official evaluation loop, which renders scale-conditioned features at 41 scales decreasing from 2.0 to 0.0 and, at each scale, splits every current leaf with HDBSCAN at a fixed cluster-selection epsilon of 0.05, with the other settings as described below. For H2G group recall, we render the evaluation view once and project the rendered Lorentz features to the tangent space at the origin. Since H2G has a single feature field, re-clustering it at a fixed epsilon would regenerate the same groups, so we decrease the cluster-selection epsilon instead of the scale. Candidate groups are then generated by recursively splitting the current leaf regions with HDBSCAN[37] in tangent space. We sweep the HDBSCAN cluster-selection epsilon from 0.50 down to 0.002 using a coarse-to-fine schedule: 0.05 steps from 0.50 to 0.20, 0.025 steps from 0.20 to 0.10, 0.01–0.015 steps from 0.10 to 0.04, 0.005 steps from 0.04 to 0.02, and smaller steps down to 0.002. At each epsilon, every current leaf with at least 30 pixels is clustered with HDBSCAN using `min_samples=30` and

Table 9: Scene-wise group recall on the GARField group annotation benchmark. Budgeted rows use random proposal sampling averaged over 100 seeds. AUC is computed from the random budget curve with an origin point at zero budget.

		GARField*			Ours		
Scene	Budget	mIoU	$R@0.50$	$R@0.75$	mIoU	$R@0.50$	$R@0.75$
bouquet	Candidates		2220			310	
	Native	0.717	0.929	0.429	0.719	0.786	0.643
	$K = 50$	0.217	0.185	0.069	0.253	0.214	0.111
	$K = 100$	0.352	0.341	0.139	0.419	0.391	0.231
	$K = 150$	0.430	0.440	0.176	0.533	0.533	0.349
	$K = 180$	0.463	0.483	0.202	0.558	0.556	0.366
	$K = 300$	0.588	0.683	0.280	0.710	0.772	0.626
	$K = 500$	0.650	0.792	0.347	0.719	0.786	0.643
	$K = 750$	0.688	0.878	0.402	0.719	0.786	0.643
	$K = 1000$	0.699	0.902	0.411	0.719	0.786	0.643
	$K = 1500$	0.712	0.927	0.424	0.719	0.786	0.643
	$K = 2000$	0.716	0.929	0.429	0.719	0.786	0.643
	AUC_{500}	0.479	0.537	0.224	0.564	0.591	0.446
	AUC_{2000}	0.645	0.811	0.365	0.680	0.737	0.594
living_room	Candidates		3256			342	
	Native	0.724	0.778	0.667	0.722	0.889	0.667
	$K = 50$	0.067	0.057	0.038	0.186	0.186	0.118
	$K = 100$	0.125	0.117	0.077	0.329	0.361	0.234
	$K = 150$	0.190	0.172	0.120	0.419	0.481	0.311
	$K = 180$	0.211	0.197	0.130	0.487	0.571	0.383
	$K = 300$	0.323	0.309	0.224	0.662	0.800	0.596
	$K = 500$	0.425	0.417	0.320	0.722	0.889	0.667
	$K = 750$	0.521	0.538	0.449	0.722	0.889	0.667
	$K = 1000$	0.582	0.621	0.509	0.722	0.889	0.667
	$K = 1500$	0.652	0.706	0.606	0.722	0.889	0.667
	$K = 2000$	0.679	0.743	0.638	0.722	0.889	0.667
	AUC_{500}	0.254	0.243	0.176	0.514	0.613	0.442
	AUC_{2000}	0.512	0.540	0.447	0.670	0.820	0.610
ramen	Candidates		1438			366	
	Native	0.773	0.864	0.682	0.813	0.909	0.773
	$K = 50$	0.255	0.223	0.145	0.270	0.237	0.160
	$K = 100$	0.358	0.329	0.222	0.430	0.405	0.269
	$K = 150$	0.433	0.400	0.286	0.538	0.537	0.376
	$K = 180$	0.478	0.448	0.330	0.605	0.616	0.447
	$K = 300$	0.549	0.543	0.399	0.757	0.830	0.666
	$K = 500$	0.638	0.651	0.505	0.813	0.909	0.773
	$K = 750$	0.707	0.750	0.587	0.813	0.909	0.773
	$K = 1000$	0.745	0.802	0.631	0.813	0.909	0.773
	$K = 1500$	0.773	0.864	0.682	0.813	0.909	0.773
	$K = 2000$	0.773	0.864	0.682	0.813	0.909	0.773
	AUC_{500}	0.471	0.458	0.338	0.609	0.647	0.508
	AUC_{2000}	0.675	0.723	0.563	0.762	0.844	0.706
teatime	Candidates		1129			241	
	Native	0.819	1.000	0.700	0.869	1.000	0.900
	$K = 50$	0.203	0.194	0.105	0.315	0.328	0.242
	$K = 100$	0.357	0.377	0.212	0.519	0.576	0.434
	$K = 150$	0.419	0.481	0.258	0.673	0.769	0.614
	$K = 180$	0.489	0.560	0.319	0.733	0.836	0.698
	$K = 300$	0.610	0.720	0.445	0.869	1.000	0.900
	$K = 500$	0.738	0.907	0.582	0.869	1.000	0.900
	$K = 750$	0.794	0.977	0.662	0.869	1.000	0.900
	$K = 1000$	0.816	0.998	0.699	0.869	1.000	0.900
	$K = 1500$	0.819	1.000	0.700	0.869	1.000	0.900
	$K = 2000$	0.819	1.000	0.700	0.869	1.000	0.900
	AUC_{500}	0.506	0.591	0.359	0.699	0.797	0.689
	AUC_{2000}	0.732	0.889	0.602	0.826	0.949	0.847

`min_cluster_size`=30. Noise pixels are assigned to the nearest non-noise cluster, and clusters with fewer than 30 pixels are discarded. Each accepted cluster is added as a candidate group and becomes a new leaf for subsequent, smaller-epsilon splits. Candidate generation is capped at 10,000 nodes, although this cap is not reached in our reported results.

For native group recall, all generated candidates are used. For budgeted recall, we evaluate proposal budgets $K \in \{50, 100, 150, 180, 300, 500, 750, 1000, 1500, 2000\}$. When a scene has fewer than K H2G candidates, all available candidates are used. Random budget sampling is repeated for 100 seeds with seed 0 as the base seed. For each ground-truth group, we report the best IoU over the selected candidate set, and compute mIoU, $R@0.50$, and $R@0.75$. The reported AUC values are normalized areas under the corresponding budget curves.

B.3 Full 3D Completeness Results

Table 8 reports the scene-wise 3D Completeness results summarized in Table 1, together with OmniSeg3D*.

B.4 Full Group Recall Results

Table 9 reports the scene-wise group recall results summarized in Table 3, including the budgets omitted there.

C Hyperbolic versus Euclidean geometry

Euclidean counterpart The output of the grouping head is mapped onto the unit hypersphere $\mathbb{S}^D \subset \mathbb{R}^{D+1}$ by the exponential map at a fixed pole. The leaf and root angular terms use the cosine distance, prototypes and root centroids use the arithmetic mean re-projected onto the sphere, compactness uses the Euclidean distance, and the LCA surrogate uses the Euclidean norm of the point on the prototype segment closest to the origin. The loss weights, temperatures, RGB checkpoint, learning-rate schedule (10^{-3} to 10^{-4} over 20k iterations), and seed are those of the main runs. Evaluation follows the same protocols, with the logarithmic map replaced by the identity.

Level-wise 3D Completeness Table 10 reports the same comparison per level under view-wise (V) and level-wise (L) threshold selection. Lorentz features score higher than Euclidean features at every level for both dimensions and both threshold settings, and the 16-dimensional Lorentz field exceeds the 256-dimensional Euclidean field in mean under both settings.

CTR and group recall 3D Completeness selects the best of 1000 thresholds for each view, which can hide level boundaries that fall at different affinities across views. Tables 11 and 12 report measurements at $D = 256$ that involve no per-view selection. With one threshold per level shared across views, Euclidean features retain less of their view-wise score than Lorentz features, most clearly at the coarse level, so Euclidean level boundaries are not consistent across views. For group recall, both geometries use the recursive HDBSCAN of Appendix B.2 with identical hyperparameters. The Euclidean field yields a more compact candidate pool, but Lorentz gives higher native mIoU, $R@0.50$, and $R@0.75$ on every scene, including `bouquet` and `teatime`, where 3D Completeness shows no difference. A smaller pool with lower recall suggests that Euclidean features merge groups that should be separated, so the recursive clustering stops splitting early.

Table 10: 3D Completeness of unit-norm Euclidean and Lorentz features under view-wise (V) and level-wise (L) threshold selection.

		Fine		Medium		Coarse		Mean	
Geometry	D	V	L	V	L	V	L	V	L
Euclidean	16	0.726	0.712	0.774	0.729	0.703	0.613	0.734	0.685
Lorentz	16	**0.854**	**0.783**	**0.806**	**0.801**	**0.791**	**0.758**	**0.817**	**0.780**
Euclidean	256	0.752	0.742	0.755	0.739	0.775	0.721	0.761	0.734
Lorentz	256	**0.848**	**0.840**	**0.813**	**0.809**	**0.844**	**0.828**	**0.835**	**0.826**

Table 11: CTR at $D = 256$.

Geometry	Fine	Medium	Coarse	Mean
Euclidean	0.988	0.979	0.918	0.962
Lorentz	**0.990**	**0.995**	**0.978**	**0.988**

Table 12: Native group recall at $D = 256$, Euclidean / Lorentz. Lorentz values come from a separate run of the recursive clustering and differ slightly from Table 9.

Scene	Candidates	mIoU	$R@0.50$	$R@0.75$
bouquet	95 / 305	0.631 / **0.719**	0.714 / **0.786**	0.571 / **0.643**
teatime	73 / 248	0.595 / **0.868**	0.600 / **1.000**	0.600 / **0.900**
ramen	84 / 367	0.724 / **0.813**	0.727 / **0.909**	0.545 / **0.773**
living_room	209 / 339	0.518 / **0.722**	0.556 / **0.889**	0.222 / **0.667**
Mean	115.2 / 314.8	0.617 / **0.781**	0.649 / **0.896**	0.485 / **0.746**

D LCA Ordering Accuracy

In the ablation of Table 5, removing the LCA ordering loss changes 3D Completeness the least among the four components. 3D Completeness, however, does not measure ordering, so we evaluate the merge order of the learned field directly.

Depth of the supervision trees LCA ordering constrains the relative merge order among leaf prototypes and is most informative when many distinct ancestor depths must be ordered. With the flattening threshold used in our experiments, every supervision tree has depth at most two (Appendix H.1). The lowest common ancestor of two distinct leaves can therefore lie at only two depths, 1 for siblings and 0 for cousins, so exactly one type of ordering relation exists. The evaluation annotations likewise provide only two or three levels per scene, which leaves few orderings to get right or wrong.

Triplet accuracy We measure whether the learned field preserves merge order as a triplet accuracy. If i and j join at a deeper ancestor than i and k in the supervision tree, the learned hierarchy score should satisfy $s(i, j) > s(i, k)$:

$$\text{LCA-Acc} = \frac{1}{|\mathcal{Q}|} \sum_{(i,j,k)\in\mathcal{Q}} \mathbb{I}\left[s(i, j) > s(i, k)\right],$$

where $\mathbb{I}[\cdot]$ is the indicator function and $\mathcal{Q}$ is the set of evaluation triplets. Here $s(a, b)$ is the LCA-depth surrogate d_o of Section 3, computed between rendered features of pixels sampled from leaves a and b; the Euclidean field uses the Euclidean norm instead, as in Appendix C. Triplets are drawn from the supervision tree of each view and pooled over the views of a scene; an uninformative predictor scores 0.5.

Table 13: LCA-Acc of the learned grouping fields. Euclidean is the unit-norm counterpart of Appendix C trained with all four losses. Chance is 0.5.

Model	teatime	bouquet	ramen	living_room	Mean
Euclidean	**0.9737**	0.8171	0.9377	0.9345	0.9158
Lorentz w/o $\mathcal{L}_{\text{LCA}}$	0.9700	0.8325	0.9452	0.9395	0.9218
Lorentz (full)	**0.9737**	**0.8545**	**0.9526**	**0.9505**	**0.9328**

Effect of geometry and the LCA loss Table 13 shows that the learned field preserves the merge order of the hierarchy far above chance on every scene. Much of this preservation comes from the curvature. With the losses fixed, changing only the geometry raises the mean more than removing $\mathcal{L}_{\text{LCA}}$ lowers it with the geometry fixed, and the Lorentz field without $\mathcal{L}_{\text{LCA}}$ still scores above the Euclidean field with it in mean. The contribution of $\mathcal{L}_{\text{LCA}}$ is small but consistent. It reduces ordering errors, $1 - \text{LCA-Acc}$, by 14% on average and improves every scene; the absolute gain varies across scenes, but the relative error reduction stays within 12–18%.

Fidelity to the supervision trees LCA-Acc measures fidelity to our own supervision trees, not whether that order is correct, so a field that follows a systematically biased tree would also score well. Measuring correctness requires an external, human-annotated hierarchy deeper than the two or three levels available to us, which remains open. The comparison itself is nonetheless sound, since all three models are evaluated against the same trees. Because per-view trees do not agree with one another, the ceiling of this metric is below 1.0. A 3D field can express only one view-independent ordering and cannot follow the idiosyncrasies of individual views, so high average agreement indicates an ordering close to the consensus of the views rather than one memorized from any of them.

E Hyperparameter Sensitivity

We vary the loss weights and the curvature around the main configuration. Each run retrains only the grouping field from the same RGB checkpoint, with everything else (other weights, geometry, dimension, temperatures, margins, and seed) held at the main run.

Loss weights Each of the four hierarchy weights and λ_{norm} is scaled to $0.5\times$ and $2\times$ its default (Table 14). Across this fourfold range, the mean changes only slightly for every hierarchy weight, and the default is the best of these nine settings. λ_{norm} moves the mean even less, and both of its settings stay within the run-to-run variation of Appendix A.1. The field is therefore robust to all five weights.

Table 14: 3D Completeness (view-wise) when each loss weight is scaled to $0.5\times$ or $2\times$ its default.

Weight	Factor	Fine	Medium	Coarse	Mean
—	$1.0\times$ (main)	0.848	0.813	0.844	0.835
λ_{leaf}	$0.5\times$	0.837	0.802	0.822	0.820
	$2\times$	0.863	0.792	0.786	0.814
λ_{root}	$0.5\times$	0.860	0.813	0.810	0.828
	$2\times$	0.836	0.795	0.817	0.816
λ_{comp}	$0.5\times$	0.849	0.795	0.819	0.821
	$2\times$	0.846	0.808	0.825	0.826
λ_{LCA}	$0.5\times$	0.848	0.804	0.806	0.819
	$2\times$	0.849	0.806	0.796	0.817
λ_{norm}	$0.5\times$	0.848	0.827	0.842	0.839
	$2\times$	0.846	0.825	0.838	0.836

Curvature Table 15 varies the curvature c over two orders of magnitude. $c = 1.0$ is best in mean but not on every scene: `teatime` prefers $c = 2$ and `ramen` prefers $c = 10$, while `bouquet` and `living_room` are best at $c = 1.0$. Curvature is thus worth tuning, and we have not optimized it. All loss weights and the curvature are fixed across the four scenes; none is tuned per scene.

Table 15: 3D Completeness (view-wise) for different curvatures c. Scene columns are scene means, level columns average each level over the scenes that annotate it, and Mean follows Appendix B.1.

	Scene				Level			
c	teatime	bouquet	ramen	living_room	Fine	Medium	Coarse	Mean
0.1	0.915	0.662	0.802	0.734	0.802	0.740	0.749	0.764
1.0 (main)	0.952	**0.750**	0.859	**0.824**	**0.848**	**0.813**	**0.844**	**0.835**
2	**0.955**	0.709	0.837	0.786	0.843	0.772	0.805	0.807
10	0.939	0.619	**0.884**	0.798	0.847	0.759	0.779	0.795

F Inference Cost

We time inference under the 3D Completeness protocol for the four scenes and their three evaluation views, with 369×497 rays per view. Only the grouping branch is timed, and the RGB head is excluded for both methods. All timings are measured on a single NVIDIA RTX 4090 with CUDA events, and each entry is the median of five timed repetitions after two warm-up runs (Table 16).

GARField cannot know the right scale for a query in advance, so its evaluation sweeps 41 scales from 2.0 to 0.0 and renders the scale-conditioned affinity field at each. Its released script repeats this sweep

Table 16: Inference time per evaluation view in seconds, measured on an RTX 4090. *Renders reused* shares the 41 scale-conditioned renders across hierarchy levels, and *as scripted* follows the released GARField evaluation script.

			GARField*	
Scene	Levels	H2G	Renders reused	As scripted
teatime	2	0.456	14.34	28.67
bouquet	3	0.489	14.12	42.35
ramen	3	0.488	14.18	42.54
living_room	3	0.486	14.17	42.52
Mean		0.480	14.20	39.02

at every hierarchy level, which amounts to 123 renders per view for a three-level scene. We also time a variant modified in GARField's favor that renders the 41 scales once and shares them across levels. H2G has no scale axis: it renders the grouping field once and computes the tangent-space affinity map and the per-level threshold sweep from that single render. H2G is therefore more than an order of magnitude faster than either GARField variant. Its cost does not depend on the number of query scales and barely changes with the number of levels, whereas the cost of the released GARField script grows in proportion to it.

G Spectral Bisection Analysis

Dasgupta's objective motivates a recursive top-down construction in which each branch is split by a sparsest-cut approximation. In our setting, however, each proposal group induces a dense DINO affinity graph, making exact recursive sparsest cut expensive as the number of leaves grows. We therefore use recursive spectral bisection in the main pipeline and compare it with exact sparsest cut on the same DINO graphs.

Figure 5 shows that spectral bisection closely tracks exact sparsest cut in normalized Dasgupta cost across max-leaf settings. The relative cost gap remains small: the median gap is below 0.15%, and the mean gap stays below about 0.22% even at the largest setting. This indicates that the surrogate preserves the tree quality measured by Dasgupta's objective.

The computational difference is much larger. Exact recursive sparsest cut grows rapidly with the max-leaf setting, reaching over 10^2 seconds in total preprocessing time, whereas spectral bisection remains around one second. The same trend appears in mean per-image time, where exact sparsest cut increases by orders of magnitude while spectral bisection stays nearly flat. These results support using spectral bisection as a practical surrogate: it introduces only a minor Dasgupta-cost increase but removes the main preprocessing bottleneck for dense DINO affinity graphs.

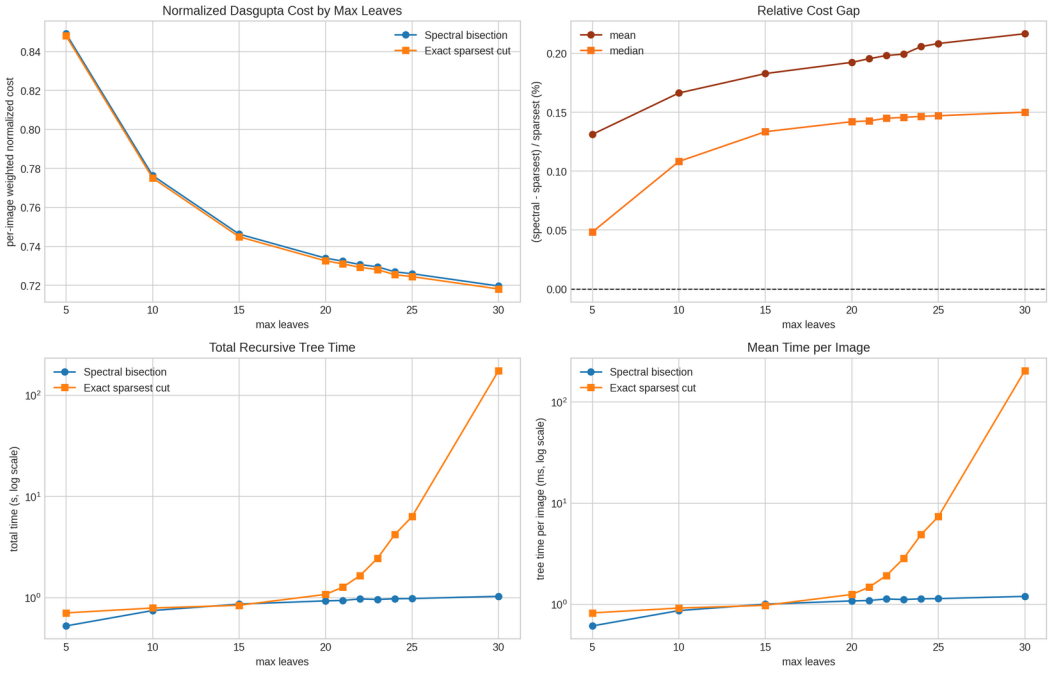

Figure 5: Comparison between recursive spectral bisection and exact recursive sparsest cut for 2D hierarchy construction. Spectral bisection closely matches the normalized Dasgupta cost of exact sparsest cut while keeping preprocessing time nearly constant. Time plots use a log scale.

H Additional Qualitative Results

H.1 2D Hierarchy Supervision

As in Figure 2, Figure 6 visualizes the 2D hierarchy supervision for `bouquet`, `ramen`, and `living_room`, with each tree of $\mathcal{T}^n$ cut at depth d from its root. For example, the bowl contents in `ramen` are split into individual toppings at deeper levels. The tree depth depends on the flattening threshold. With the threshold of 0.18 used in our experiments, every tree has depth at most two, so depth 2 shows the leaf masks.

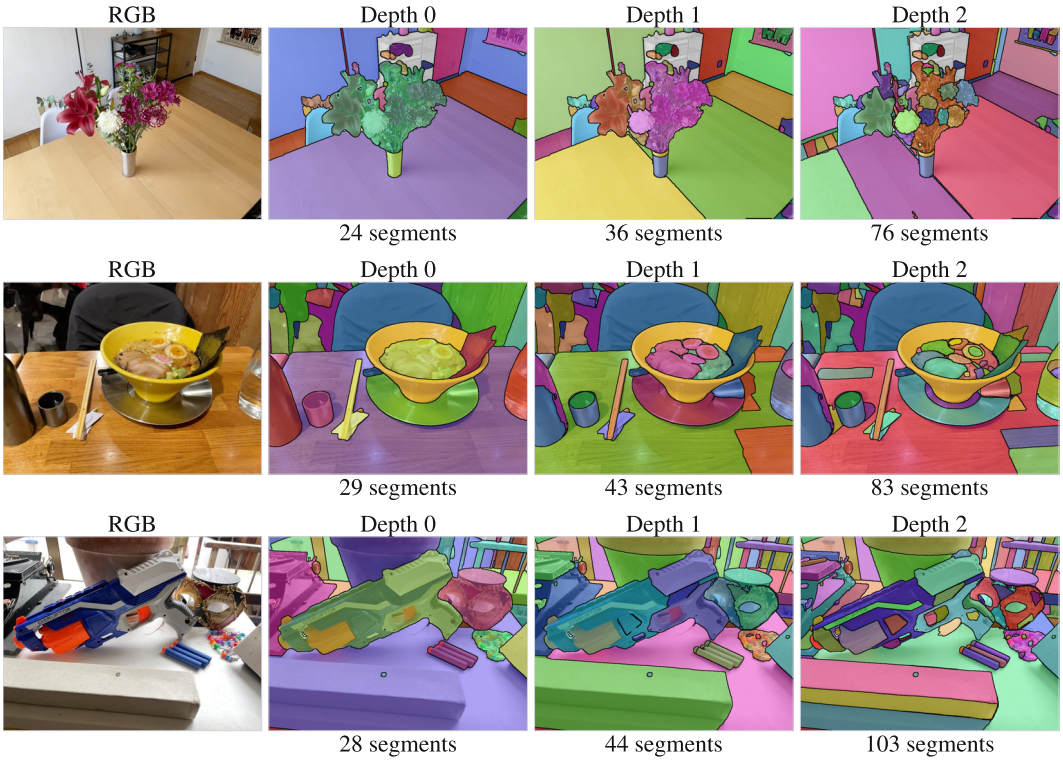

Figure 6: Additional examples of 2D hierarchy supervision, visualized as in Figure 2.

H.2 Hyperbolic Embedding of Ground-Truth Groups

Figure 7 projects rendered features of the H2G model in Table 1 onto the Poincaré disk with CO-SNE[38], which preserves distances from the origin. For each scene, we sample 200 pixels per ground-truth group in the first evaluation view, together with as many context pixels outside the groups. The groups of each scene gather in one angular direction, apart from the context. Pixels lie at similar distances from the origin whichever group they belong to, but group midpoints move toward the origin from fine to coarse in every scene, as a coarser group spreads over a wider angle.

H.3 K-means on the Grouping Field

Figures 8–11 apply K-means to the same model for K from 2 to 48, rendering each scene cropped to a box from a novel viewpoint. As in our evaluation, the features are mapped to the tangent space at the origin, and colors are not matched across K. As K grows, objects first separate from the background and then split into parts. Since K-means is not hierarchical, clusters at different K need not be nested, and large uniform surfaces can split arbitrarily.

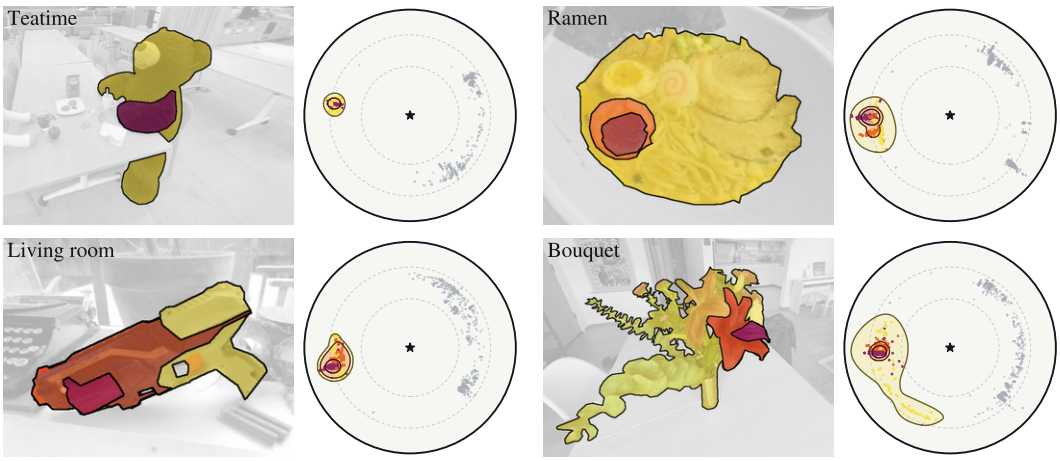

Figure 7: CO-SNE projections of H2G features. Ground-truth groups are colored from fine (purple) to coarse (yellow), gray points are context pixels, and the star marks the origin.

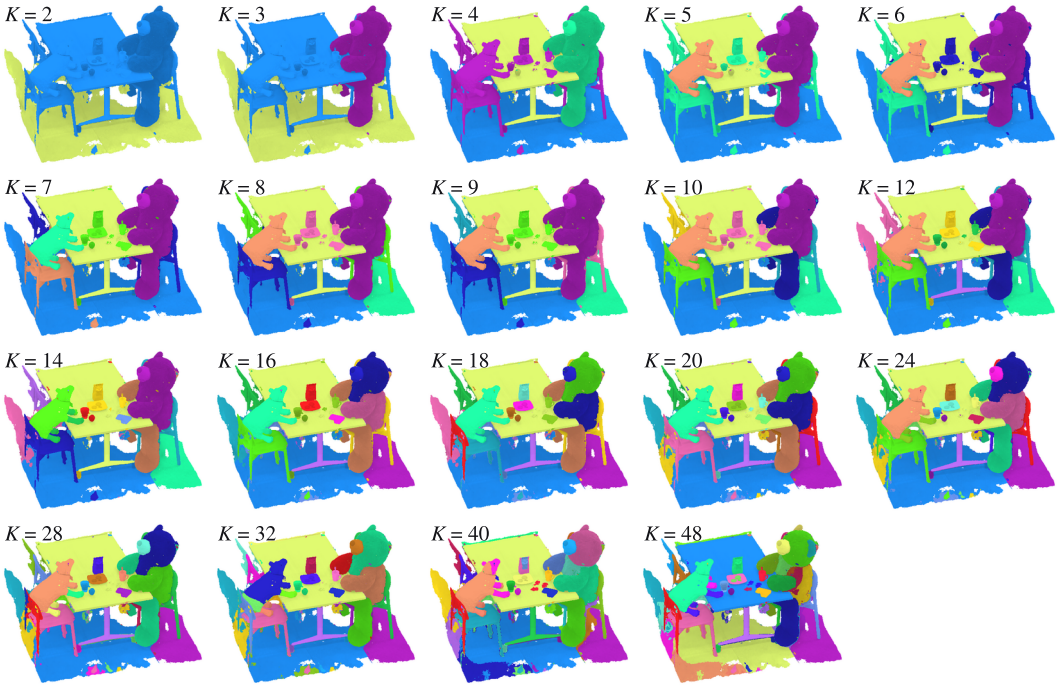

Figure 8: K-means clusters on `teatime` for K from 2 to 48.

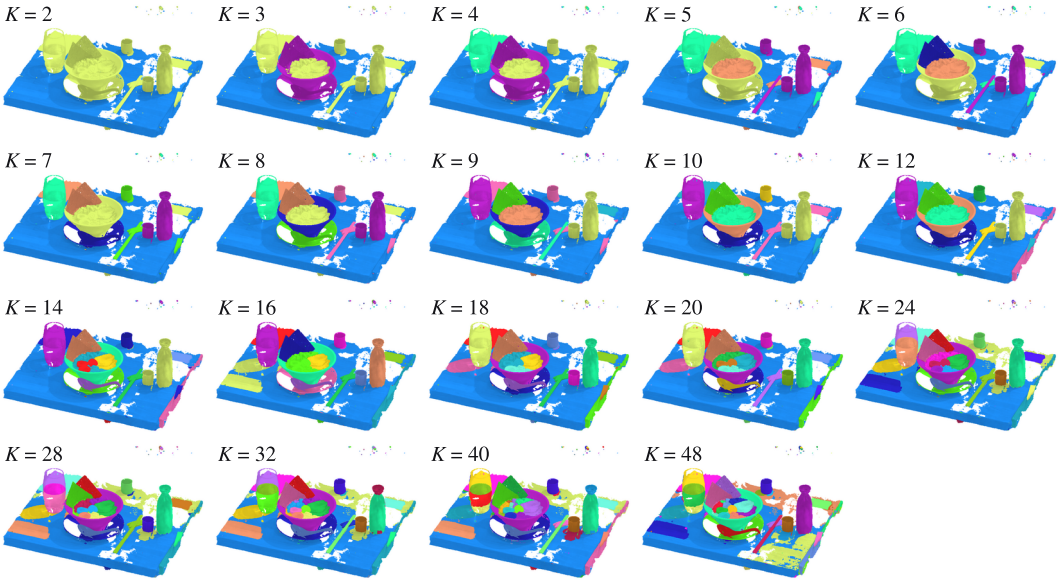

Figure 9: K-means clusters on `ramen` for K from 2 to 48.

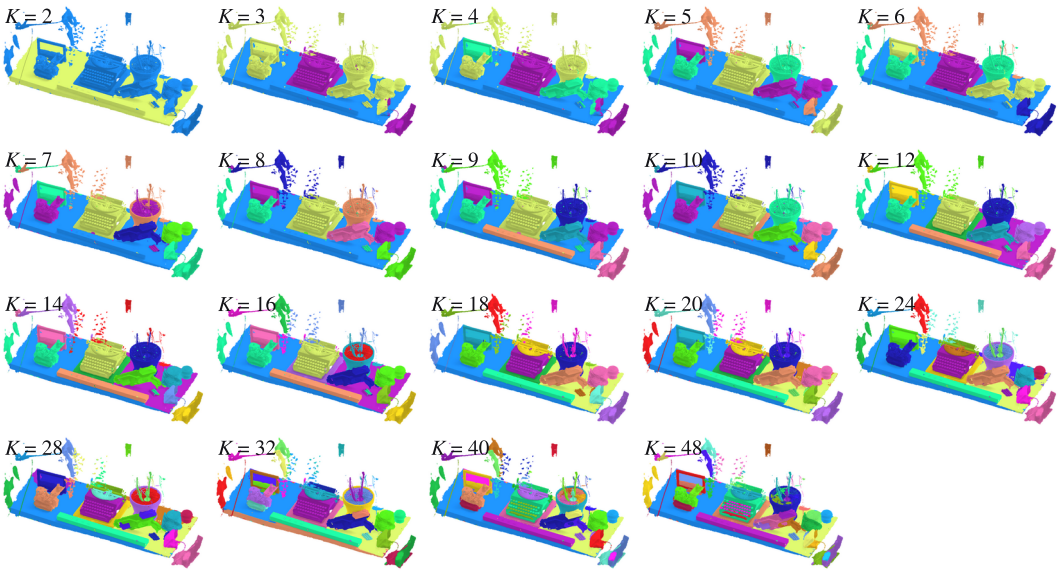

Figure 10: K-means clusters on `living_room` for K from 2 to 48.

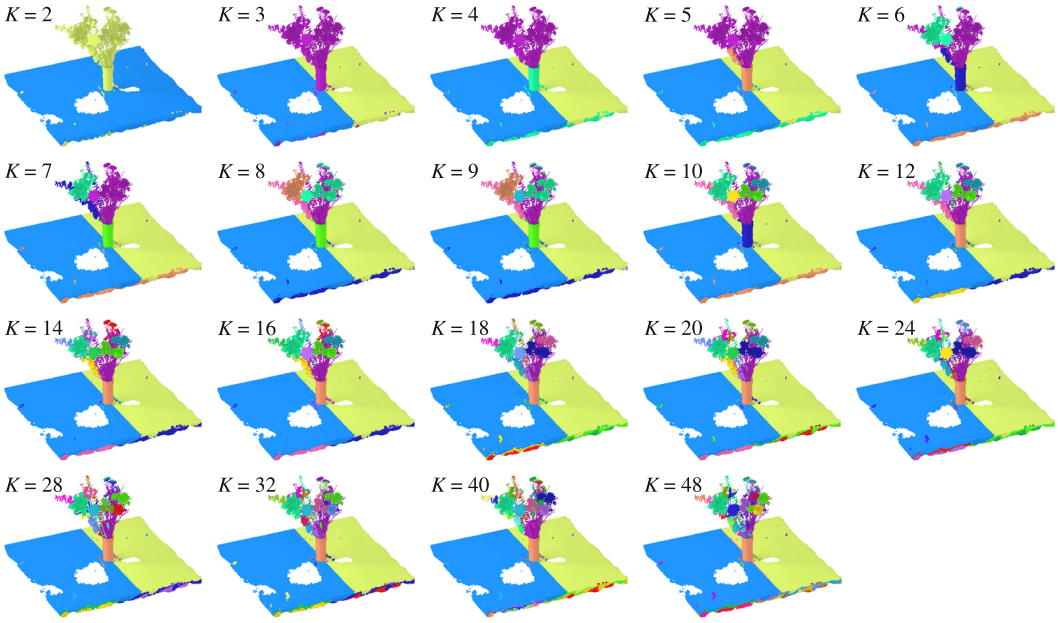

Figure 11: K-means clusters on `bouquet` for K from 2 to 48.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state that H2G learns a hierarchy-aware hyperbolic grouping field from SAM/DINO-derived tree supervision and evaluate it on GARField-style hierarchical grouping benchmarks. The claims are supported by the 3D Completeness, group recall, qualitative, and ablation results

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a dedicated Limitations section discussing dependence on 2D supervision quality, possible cross-view inconsistencies, approximate hierarchy construction, and the remaining need for thresholding or clustering at inference time.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not present formal theoretical results, theorems, or proofs. The mathematical content defines the model objectives and evaluation quantities used in the method and experiments.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper describes the datasets, reproduced baselines, training setup, model hyperparameters, runtime, and evaluation protocols in Section 4 and Appendices A–B. The appendix specifies the group-recall epsilon sweep, HDBSCAN parameters, proposal budgets, and sampling procedure.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is publicly available at `https://github.com/Rhqo/H2G`, as linked in the abstract. Implementation details, hyperparameters, and evaluation protocols needed to reproduce the main results are provided in Appendices A–B.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Training details, optimizer settings, feature dimensions, loss weights, evaluation scenes, thresholding settings, HDBSCAN parameters, and proposal-budget evaluation settings are specified in Section 4 and Appendices A–B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Appendix A.1 reports 3D Completeness as the mean $\pm$ standard deviation over five training runs with seeds 0–4, for both GARField* and H2G under the same SAM masks and evaluation protocol (Table 6). The factor of variability is the training seed, with all other settings fixed, and the reported value is the standard deviation across runs, not the standard error. The budgeted group-recall results in Table 3 are averaged over 100 random sampling seeds.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The appendix reports the compute setup and approximate per-scene runtime. Experiments were run on a local workstation with NVIDIA RTX 3090 GPUs, each with 24GB VRAM. Generating 2D hierarchy supervision takes approximately 30 minutes per scene, and hierarchy-supervised grouping-field training takes approximately 20 minutes per scene.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses public research datasets/models and standard 3D vision evaluation protocols, and we are not aware of deviations from the NeurIPS Code of Ethics. The paper preserves anonymity and does not involve human-subject experiments or sensitive personal data collection.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The manuscript does not include a dedicated broader-impact discussion because the work is a foundational 3D scene grouping method evaluated on research benchmarks, rather than an application-specific or deployed system. We do not claim that the method has no possible societal impact; rather, the current paper focuses on the technical method, limitations, and benchmark evaluation.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk generative models, language models, scraped datasets, or deployed systems requiring special safeguards. The method is a 3D scene grouping approach evaluated on existing research benchmarks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the original works for all existing research assets used in the method and experiments. GARField benchmark/code assets are used for evaluation under the MIT License, SAM is used for 2D mask proposal generation under the Apache License 2.0, and DINOv3 is used for semantic feature extraction under the DINOv3 License. We use these assets under their released licenses or terms and do not claim ownership of them.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce a new dataset or benchmark asset. It introduces a method and evaluation results built on existing GARField scenes and publicly described model components.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

Justification: The research does not involve crowdsourcing experiments or human-subject studies. No participants were recruited or compensated.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The research does not involve crowdsourcing or human-subject experiments, so IRB approval or equivalent review is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs are not used as an important, original, or non-standard component of the method or experiments. Any use of writing or formatting assistance, if applicable, does not affect the core methodology, scientific rigor, or originality of the research.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.