h-NNE v2: Fast Hierarchical Nearest Neighbor Graph Embedding for Efficient Dimensionality Reduction

M. Saquib Sarfraz

September 2025

Abstract

Hierarchical Nearest-Neighbor Embedding (h-NNE [1]) reduces dimensionality by building a cluster hierarchy and, level by level, mapping each point into the 1-NN basin of its cluster anchor. This simple pipeline is fast and scalable, but the direct point→anchor pull can *over-contract* structure: large clusters get squeezed, small ones become "dots," and accidental anchor overlaps create artifacts and wasted space when many anchors coexist.

We introduce *h-NNE v2*, a packing layer that replaces free-form anchor motion at coarse levels with a geometry-aware, weighted circle/sphere packing. In 2D, the layer combines: (i) automatic k-NN edge scaling from anchor geometry; (ii) a vectorized, grid-broadphase overlap resolver with capped Jacobi/Gauss–Seidel impulses; (iii) relaxed/capped k-NN target lengths to prevent blow-up; and (iv) span normalization that shrinks center distances to a robust multiple of the median touching distance for compact, overlap-free layouts. Children are then placed *inside* each parent via an isotropic center+radius fit followed by a few "balloon" inflation steps under hard wall/pair caps (centers fixed) to fill available area. For $D > 2$ we provide a light ND fallback with the same separation principles.

The layer integrates into `h-NNE` through a vectorized point-to-anchor mapping that uses exact or batched k-NN radii with robust capping. Empirically, v2 reduces dotification and overlap while improving space utilization, with sub-second to few-second runtimes for tens–thousands of anchors.

1 Background and motivation

Hierarchical Nearest-Neighbor Embedding (h-NNE [1]) is a method for visualizing high-dimensional data in 2D/3D, akin in spirit to t-SNE [2] and UMAP [3]. t-SNE optimizes a Kullback–Leibler divergence between *pairwise similarity* distributions constructed from a sparse k-NN graph of the data, and UMAP optimizes a cross-entropy between fuzzy simplicial set representations of that k-NN graph in data and embedding spaces. Both objectives aggregate a large number of pairwise terms (on the order of Nk per iteration), which makes their optimization comparatively expensive at large scale, even with neighbor pruning and negative sampling.

In contrast, h-NNE first constructs a sequence of clusterings via FINCH [4] and then embeds *cluster anchors* level by level, propagating the layout down to the points. Concretely, FINCH yields a parameter-free hierarchy $\mathcal{L} = \{\Gamma_0, \Gamma_1, \ldots, \Gamma_{L-1}\}$; at each level ℓ, every cluster $c \in \Gamma_\ell$ has an anchor $a_c^{(\ell)}$ (e.g., the mean of its members). The original h-NNE projection proceeds from coarse to fine by repeatedly mapping points into 1-NN–defined basins of their current anchors, using a per-anchor scale chosen to be at most a fixed fraction of the nearest-anchor distance—e.g., $r_c \leq \alpha \min_{c' \neq c} \|a_c^{(\ell)} - a_{c'}^{(\ell)}\|$ with $\alpha < \frac{1}{2}$—which prevents cross-cluster mixing.

By operating on a 1-NN graph and propagating through the clustering tree—rather than optimizing a global pairwise objective—h-NNE achieves substantially lower computational cost while preserving local neighborhood structure. For more details, see the corresponding paper [1].

This h-NNE anchor-centric, hierarchical strategy affords several advantages:

- **Speed and scalability.** Working with *few* anchors at coarse levels and only performing simple, vectorized point-to-anchor updates avoids large global optimizations; the method scales to very large N with modest memory.
- **Local-neighborhood faithfulness.** Because the FINCH hierarchy is built from first-neighbor relations, nearby points in the original high dimensional space tend to coalesce early and share anchors; the per-anchor, radius-capped mapping preserves these local neighborhoods as one descends the hierarchy, reducing the chance of local distortions.
- **Global-structure stability.** The clustering hierarchy is built fine$\rightarrow$coarse from first-neighbor links, while the embedding proceeds coarse$\rightarrow$fine. Placing a *small* set of high-mass anchors first (top levels) creates a stable global scaffold. As children are inserted, this scaffold is preserved, so the relative arrangement of major groups (macro-structure) remains stable while local detail is revealed. Empirically, inter-cluster relations visible at the coarse-cluster levels persist as the hierarchy expands.
- **Structure-aware zoom.** The hierarchy provides a natural coarse$\rightarrow$fine control: users can start at a parent level (few anchors) for global context and then reveal children for local detail.
- **Label availability.** Cluster labels are produced as part of FINCH; in unlabeled settings these labels are directly usable for downstream analysis and for structured visualization (coloring, grouping).
- **Parameter-light.** FINCH is parameter-free and the projection stage uses a small number of robust defaults; this reduces tuning compared to global nonconvex objectives.

However, the original point-to-anchor mapping also has characteristic failure modes at coarse scales:

- **Dotifying / under-filled clusters.** Points are radially scaled into small anchor-centered balls; at coarse levels this can visually collapse clusters to dots, obscuring inter-cluster relations.
- **Squeezed large clusters.** Because motion is dominated by nearest-anchor geometry, large clusters may be forced into overly small extents when neighboring anchors are close, even if a more balanced placement exists.
- **Overlap artifacts.** Accidental proximity or overlap of coarse anchors can propagate artifacts as one descends the hierarchy.

To address these issues, a better approach is to first lay out *cluster anchors* of few coarse levels into non-overlapping discs/balls that (i) preserve neighborhood relations, (ii) remain compact, and (iii) respect the hierarchy (children inside parent). The v2 packing layer provides exactly this: a fast, geometry-aware layout of anchors, later used to guide the h-NNE refinement. Empirically it removes the dotifying artifact at coarse scales, prevents large clusters from being squeezed, and reduces the amount of point-level work needed downstream.

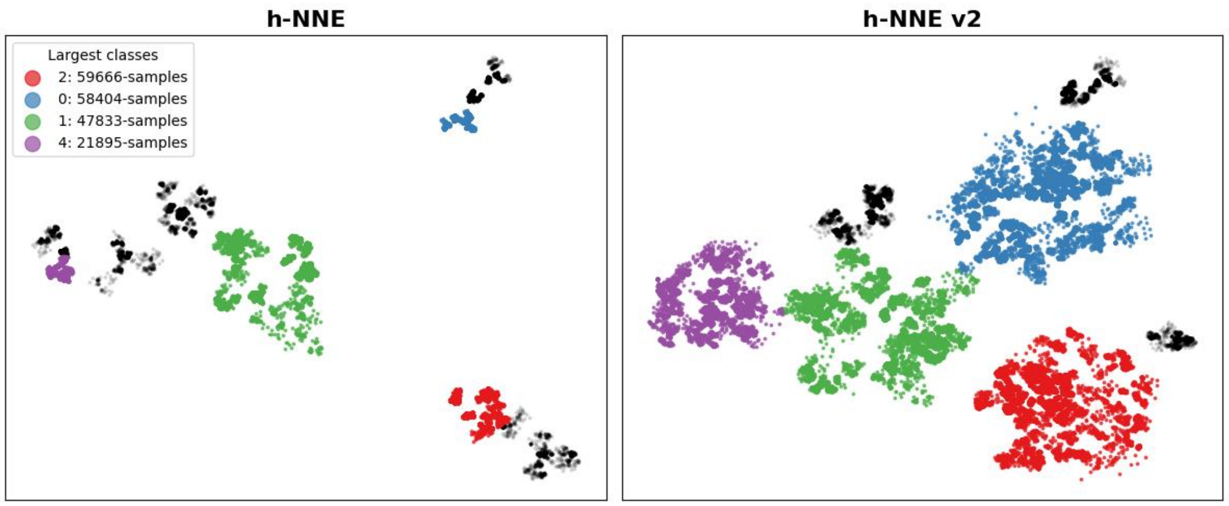


Figure 1: **h-NNE vs. h-NNE v2 on 200k points (2048 D).** Four largest classes are colored—red (59,666), blue (58,404), green (47,833), purple (21,895)—others in black. *Left:* baseline h-NNE shows squeezed large classes, dotified interiors, and unused white space. *Right:* h-NNE v2 first packs the top two parent levels (8 and 39 clusters) via a fast weighted circle packing, then applies the usual hierarchical refinement; the result is a compact, space-filling embedding that better reflects relative cluster mass while preserving separations.

Illustrative comparison. Figure 1 contrasts the original h-NNE projection with h-NNE v2 on a dataset of $\sim$ 200,000 points in 2048 D. For visual clarity, we color only the four largest ground-truth classes, the remainder is shown in black. The baseline h-NNE (left) exhibits the failure modes discussed above: the two largest classes (red/blue) are visibly *squeezed* relative to the smaller green/purple clusters, there is substantial unused white space, and interiors appear "dotified" due to point-to-anchor contraction. In contrast, h-NNE v2 (right)—which first packs the top two parent levels (8-cluster and 39-cluster) with the proposed geometric layer and then runs the standard refinement—produces a space-filling layout that better respects relative cluster mass while maintaining separations and local neighborhood structure.

Taken together, this example motivates our design: a fast geometric packing of anchors at the top of the hierarchy yields a compact, mass-respecting scaffold that the standard h-NNE refinement can safely inherit. In the remainder of this paper, we formalize the notation and constraints, detail the h-NNE v2 packing layer and show how the layer integrates into h-NNE, along with notes on complexity, defaults, and limitations.

2 Notation and constraints

Let $X = \{x_i\}_{i=1}^N \subset \mathbb{R}^D$ be the preliminary embedding (e.g., PCA). Following FINCH, let the clustering hierarchy be

$$\mathcal{L} = \{\Gamma_0, \Gamma_1, \ldots, \Gamma_{L-1}\}, \qquad \Gamma_\ell = \{\, C_1^{(\ell)}, \ldots, C_{K_\ell}^{(\ell)} \,\},$$

where Γ_ℓ is a valid partition of $\{1, \ldots, N\}$ into K_ℓ clusters. We also use a label matrix $\mathbf{Z} \in \mathbb{Z}^{N \times L}$ with entries $z_{i\ell} \in \{1, \ldots, K_\ell\}$, meaning $i \in C_{z_{i\ell}}^{(\ell)}$. Each child cluster has a unique parent at the next level; we denote this parent map by $\pi : \{1, \ldots, K_\ell\} \to \{1, \ldots, K_{\ell+1}\}$ so that $C_c^{(\ell)} \subseteq C_{\pi(c)}^{(\ell+1)}$.

For cluster c at level ℓ, define its anchor and a base radius by

$$a_c^{(\ell)} \;=\; \frac{1}{|\{\, i:\; z_{i\ell}=c \,\}|} \sum_{i:\, z_{i\ell}=c} x_i, \qquad r_{c,\mathrm{base}}^{(\ell)} \;\propto\; \sqrt{|\{\, i:\; z_{i\ell}=c \,\}|}, \tag{1}$$

with a median-normalization to be scale-free. At level ℓ we seek centers $p_c^{(\ell)} \in \mathbb{R}^d$ (d=2 in the fast path; d=D in ND) and radii $r_c^{(\ell)} > 0$ such that

$$\textbf{(pair)} \quad \big\|p_c^{(\ell)} - p_{c'}^{(\ell)}\big\|_2 \;\geq\; r_c^{(\ell)} + r_{c'}^{(\ell)} + \delta \quad \forall\, c \neq c', \tag{2}$$

$$\textbf{(parent)} \quad \big\|p_c^{(\ell)} - p_{\pi(c)}^{(\ell+1)}\big\|_2 \;\leq\; r_{\pi(c)}^{(\ell+1)} - r_c^{(\ell)} - \delta, \tag{3}$$

for a small safety gap $\delta > 0$ (we use $\delta \in [0.003R,\, 0.008R]$ relative to the parent radius R).[1]

3 2D fast packing: design and rationale

The 2D layer must satisfy three goals: *disjointness* (no overlaps), *local faithfulness* (neighbors remain neighbors), and *compactness* (the frame does not explode). Global stress or spring models are expensive and fragile with heavy-tailed radii; pure collision resolution preserves disjointness but discards geometry. We therefore adopt a *projection–separation* loop with strong safeguards.

3.1 Auto kNN touch scaling

Anchors from PCA can be globally mis-scaled relative to radii. Starting at a bad scale either causes deep interpenetrations (many separation sweeps) or an overextended layout. We rescale anchors by

$$s_0 \;=\; \mathrm{clip}\!\left(\frac{\mathrm{median}_{(i,j)\in E}(r_i + r_j)}{\mathrm{median}_{(i,j)\in E}\,\|a_i - a_j\|},\; s_{\min}, s_{\max}\right), \qquad a_i^{\mathrm{sc}} = s_0 a_i,$$

where E are mutual-kNN edges on anchors (typically k=8). Clamping s_0 avoids collapse/explosion and reduces the number of separation sweeps.

3.2 Vectorized overlap separation via capped Jacobi

At coarse levels many anchor discs coexist; even after an initial anchor-respecting placement, small residual overlaps remain. A naive all-pairs repulsion is *quadratic* and numerically fragile: large discs can drag the entire configuration, and simultaneous updates can over-shoot, amplifying scale. Rather than solving a linear system (where classical Jacobi/GS apply directly [5, 6]), our problem is a set of *geometric inequality constraints*. We therefore adopt a *constraint–projection* view in the spirit of contact resolution / Position-Based Dynamics (PBD) and projected Gauss–Seidel (PGS) [7, 8, 9, 10, 11]: each violated pairwise constraint generates a small position correction; corrections are mass-weighted, under-relaxed, and *capped* to avoid global dilation. We apply them in a *Jacobi* fashion (all at once for vectorization) with a brief all-pairs "polish" that behaves like a GS refinement.

Our goal at a given level is *feasibility*: find centers $\{p_i\}$ such that for every pair (i, j), Each pair (i, j) induces

$$\|p_i - p_j\| \;\geq\; r_i + r_j + \gamma, \tag{4}$$

with small safety gap $\gamma \geq 0$. We seek updates that resolve violations with *minimum translation* of centers while keeping the overall span compact.

[1] All inequalities are enforced numerically with small tolerances (e.g., 10^{-9} absolute plus relative components).

Broadphase: grid-based candidate generation. Testing all $\binom{n}{2}$ pairs is wasteful when n is large. We avoid $O(n^2)$ pair enumeration by a uniform grid of cell size h (quantile-based, see below) to enumerate candidate pairs in near-linear time per sweep. Points share a cell key $c(p) = \lfloor p/h \rfloor$. Candidate pairs are formed only within a cell or a small neighbor stencil ("5-neighborhood"). To avoid missing widely interacting pairs when radii are heavy-tailed, we introduce two extensions: (i) a fixed 5-offset stencil $\{(0,0),(1,0),(0,1),(1,1),(-1,1)\}$ that covers typical discs; (ii) for the top 5% largest radii (r_{big}), a rectangular halo whose half-width scales with $(r_{\text{big}} + r_{q90})/h$. This *grid broadphase* (a form of spatial hashing [12]) yields $O(n)$ candidates in typical layouts and is vectorizable end-to-end.

Quantile cell size. Let r be the radii and r_{50}, r_{90} their median and 90th percentile. We set

$$h \;=\; \max\{\, 2\, r_{90},\; 4\, r_{50},\; 10^{-6}\,\}, \tag{5}$$

which keeps cells tight for the majority (hence few pairs) while still capturing big-disc neighborhoods.

Gauss–Seidel vs. Jacobi (and "capped Jacobi"). For each candidate we compute the violation $\Delta_{ij} = (r_i + r_j + \gamma) - \|p_i - p_j\|$. Classical Gauss–Seidel (GS) updates constraints sequentially and propagates information quickly but is hard to vectorize; pure Jacobi updates all constraints in parallel but tends to *dilate* layouts due to simultaneous outward pushes. We reconcile them by accumulating pair impulses into node-wise increments (Jacobi) and then *capping* each node's step by a fraction of its worst incident violation: $\|\Delta p_i\| \;\le\; \max_{j\sim i} \Delta_{ij}$. This per-node trust region, together with under-relaxation, yields GS-like compactness with Jacobi's vectorization [5, 6]. We re-center each sweep to remove drift. Empirically a few coarse-to-fine sweeps suffice.

Capped Jacobi update (vectorized). Let (i, j) index a candidate pair. The *penetration need* (positive if violating) is defined as

$$\Delta_{ij} \;=\; (r_i + r_j + \delta) - \|p_i - p_j\|.$$

Pairs with $\Delta_{ij} \le \varepsilon$ are ignored. For the rest (*violations*), we compute a damped (*under-relaxed*) impulse of size

$$s_{ij} \;=\; \underbrace{\eta}_{\text{UR}}\,(\Delta_{ij} + \varepsilon), \qquad \eta \in (0,1),\; \varepsilon > 0. \tag{6}$$

We resolve violations by applying equal-and-opposite *mass-weighted minimum translation* impulses along the unit direction $u_{ij} = p_i - p_j / \|p_i - p_j\|$, or a fixed axis if coincident.

$$\Delta p_i = +\, t_i\, s_{ij}\, u_{ij}, \quad \Delta p_j = -\,(1 - t_i)\, s_{ij}\, u_{ij}.$$

with weights t_i chosen from inverse masses $(1/m_i)$ via $t_i = \frac{m_i^{-1}}{m_i^{-1} + m_j^{-1}}$ with $m_i \propto r_i^2$ in 2D ("area mass"; $m_i \propto r_i^D$ in D dims is analogous). This impulse-like rule is standard in rigid-body contact [7, 8] and is the PBD/PGS counterpart of a Gauss–Seidel step on a linearized complementarity system [9, 10].

All incident contributions to node i are accumulated $\Delta p_i = \sum_j \Delta p_i^{(ij)}$, then *capped* by a node-wise trust region:

$$\|\Delta p_i\| \;\leftarrow\; \min\!\big(\|\Delta p_i\|,\; \alpha\, \max_j s_{ij}\big),$$

Algorithm 1 2D Fast Packing (high level)

```
Require: Anchors $\{a_i\}$, radii $\{r_i\}$, edges $E$ (mutual-$k$NN on anchors)
1: $a^{\mathrm{sc}} \leftarrow$ touch_scale$(a, r)$
2: $p \leftarrow a^{\mathrm{sc}}$;    $p \leftarrow$ separate_overlaps$(p, r, \delta)$        ▷ capped Jacobi on grid pairs
3: for $t = 1, \ldots, T$ do                                                    ▷ usually $T = 1$
4:     $L \leftarrow$ relaxed_capped_targets$(a^{\mathrm{sc}}, r, E)$
5:     $p \leftarrow$ kNN_edge_projection$(p, r, E, L)$
6:     $p \leftarrow$ separate_overlaps$(p, r, \delta)$;    $p \leftarrow$ rigid_align$(p, a^{\mathrm{sc}})$
7: $p \leftarrow$ span_normalize$(p, r, E, \beta)$;    $p \leftarrow$ separate_overlaps$(p, r, 0.01\,\mathrm{med}(r))$
8: return $p$
```

with $0 < \alpha < 1$ (we use $\alpha \approx 0.85$). This prevents large correlated motions that can "inflate" the configuration. Finally $p_i \leftarrow p_i + \Delta p_i$. We recenter by subtracting the mean to avoid drift.

In matrix form, one sweep is a *Jacobi* update on the nonlinear feasibility system, followed by a node-wise projection onto balls of radius $\alpha \max_j s_{ij}$ (trust region)—a PBD-style damping [9, 11]. We set $\eta \in [0.6, 0.8]$ (under-relaxation) and terminate the sweep when the fraction of violating pairs or the maximum s_{ij} drops below relative thresholds.

Two-stage schedule and polish. To reduce grid rebuilds, we use a short *coarse* stage (larger cell size $1.6h$) with a few inner Jacobi micro-iterations that reuse the same candidate list, then a *fine* stage (smaller h). To address any remaining tiny residual overlaps, we run a short *all-pairs* capped pass (Gauss–Seidel–like polish) with a visible gap proportional to median(r). This combination yields compact, overlap-free layouts with sub-second to few-second runtimes on tens to thousands of discs. Intutively, Each under-relaxed, capped Jacobi step is a non-expansive projection toward the half-space defined by (4); the per-node cap makes the composite map an averaged operator in the fixed-point sense, preventing diameter blow-up. Mass-splitting damps large discs, avoiding global drift. Coarse-to-fine staging and inner reuse of pairs mimic GS's fast local convergence while preserving vectorization [5, 6, 9, 11].

Remark on relation to PGS/constraint solvers. Our scheme is closest to PBD/PGS-style contact solvers [9, 10], augmented with a spatial broadphase and a *nodewise* trust-region cap. Unlike strict PGS (sequential), we vectorize updates (Jacobi) and then apply capping to control global scale; unlike naive Jacobi, we stage coarse→fine and reuse pairs to reduce overhead. See [5, 6, 7, 8] for classical relaxation and contact fundamentals, and [11] for stabilized PBD variants.

Complexity. The broadphase and updates are $O(n)$ in memory and nearly linear time for typical radii distributions; worst-case cost is proportional to the number of grid candidate pairs (empirically $\ll n^2$). Empirically, 3–8 outer sweeps with 2–3 inner micro-iterations suffice even for $n \sim 10^3$–10^4 discs.

3.3 Relaxed/capped kNN edge targets)

Anchors come from PCA (or another preliminary embedding) and often exhibit anisotropic *stretch* relative to the "touching" metric induced by radii: the distance $a_{ij} = \|A_i - A_j\|$ along an anchor edge can be far larger than the sum of radii $t_{ij} = r_i + r_j$. If we naively force p_i, p_j to reproduce a_{ij}, the layout *blows up*; if we force them to t_{ij}, we risk over-compression and topological tangles. We therefore define *relaxed* and *capped* edge lengths L_{ij} that interpolate between *touching* and *anchor*, but never exceed a robust multiple of touching.

Mutual-kNN edge set. We build E by mutual kNN on the anchors A (with k small, e.g. 1–3). This yields a sparse, locally symmetric graph that captures neighborhood relations while avoiding hub-induced long edges.

Target lengths. For $(i, j) \in E$, let $a_{ij} = \|A_i - A_j\|$, $t_{ij} = r_i + r_j$. Define the *edgewise relax* and *cap* via

$$L_{ij}^{\mathrm{rel}} \;=\; (1-\alpha)\, t_{ij} + \alpha\, a_{ij}, \qquad L_{ij}^{\mathrm{cap}} \;=\; \beta\, t_{ij}, \tag{7}$$

with $\alpha \in [0, 1]$ and $\beta > 1$. The final target is

$$L_{ij} \;=\; \max\{\, t_{ij},\, \min\{\, L_{ij}^{\mathrm{rel}},\, L_{ij}^{\mathrm{cap}}\,\}\,\}. \tag{8}$$

We choose α, β *adaptively* from the empirical stretch ratios $\rho_{ij} = a_{ij}/\max(t_{ij}, \epsilon)$: when the median/p90 of ρ is large, we *decrease* α (relax more toward touching) and *decrease* β (tighter cap), preventing blow-up; when ρ is near 1, we let $L_{ij} \approx a_{ij}$, preserving local geometry.

Edge projection step. With L_{ij} fixed, one projection pass updates all $(i, j) \in E$ by moving along the edge direction to reduce the signed error

$$e_{ij} \;=\; \|p_i - p_j\| - L_{ij},$$

again with mass-splitting and under-relaxation:

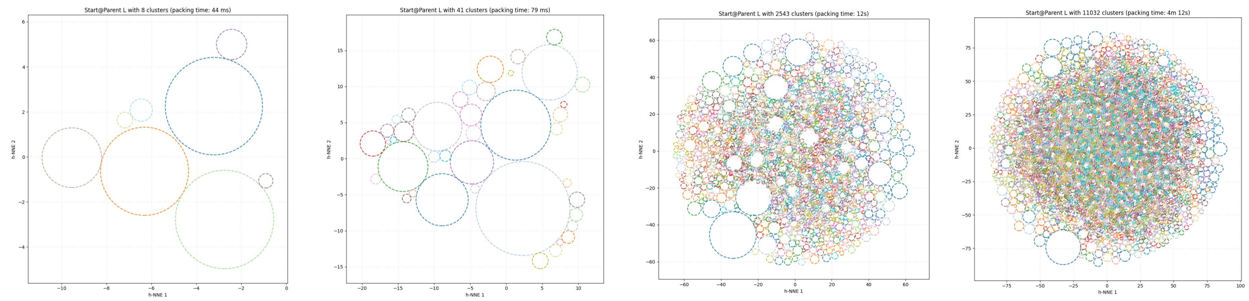
$$\Delta p_i \;=\; -\, t_i\, e_{ij}\, u_{ij}, \qquad \Delta p_j \;=\; +\,(1-t_i)\, e_{ij}\, u_{ij}, \tag{9}$$

where u_{ij} is the unit vector along $p_i - p_j$ (axis fallback if coincident) and t_i is the same inverse-mass proportion as above. We then run a short overlap-separation pass to re-establish (4) and rigidly realign to the anchors by a best-fit rotation (no scaling) so the frame orientation follows anchors A without re-introducing stretch.

Why (8) is robust. (i) The lower bound $L_{ij} \geq t_{ij}$ guarantees *touching or larger* along edges, preventing immediate pair violations. (ii) The cap $L_{ij} \leq \beta\, t_{ij}$ bounds the global span growth along long anchor edges and accelerates convergence (edges cannot demand more than a constant-factor expansion). (iii) The relaxation L_{ij}^{rel} preserves local *relative* distances when anchor geometry is not overly stretched, improving neighbor faithfulness.

Interplay with separation. Edge projection may create small local inconsistencies w.r.t. (4); the following separation pass removes them while keeping the span compact (Sec. 3.2). Alternating these two maps drives the system toward a feasible set where edge errors are small and pair constraints hold. In practice, one or two outer passes suffice at coarse levels.

Span normalization. Even with relaxed targets, different datasets produce different absolute spans. We therefore optionally apply a final *span normalization* that uniformly rescales centers (about the centroid) so that the *median* kNN edge length equals β_0 times the *median touching length* (e.g. $\beta_0 \approx 1.25$), followed by a short separation to restore the tiny gap. This makes the output scale stable across runs and avoids frames that are either too sparse or too cramped.



(a) Start at 8 clusters Level (44 ms) **(b)** Start at 41 clusters Level (79 ms) **(c)** Start at 2,543 clusters Level (12 s) **(d)** Start at 11,032 clusters Level (4m 12s)

Figure 2: **Single packed parent level.** Dataset: 200k samples (2048D). FINCH clustering hierarchy, 6 levels with sizes: [49316, 11032, 2543, 421, 41, 8]. Each panel shows the effect of v2 packing applied to just one parent level, used as the seed for subsequent steps.

Complexity and defaults. Edge construction is $O(nk)$ on anchors; per-pass projection is $O(|E|)$ and fully vectorized; separation is near-linear in n thanks to the grid broadphase. We use small k (1–3), adaptive (α, β) as above, one outer pass (`outer_sweeps`=1) at the top level, and the same mass model as in Sec. 3.2.

Relation to stress/force methods. Eq. (9) is a local, under-relaxed step for the quadratic stress $\sum_{(i,j)\in E}(\|p_i - p_j\| - L_{ij})^2$, but—crucially—our targets (8) are *geometry-aware* (touching-aware and capped) and we interleave with hard contact separation, which classical MDS/force-directed models do not; this combination yields compact, overlap-free layouts suitable for subsequent hierarchical mapping.

3.4 Visualization of 2D packing layout

We illustrate how the selected *start level* in the FINCH hierarchy shapes the packed layout. On a dataset of 200,000 points in 2048 dimensions, FINCH produces six levels with cluster counts [49,316, 11,032, 2,543, 421, 41, 8] (finest $\rightarrow$ coarsest). In Figure 2 we pack a single parent level at four different starts: 8, 41, 2,543, and 11,032 clusters. At each choice, anchors are placed by respecting cluster mass ((radii $\propto \sqrt{|C|}$) and preserving local k-NN relations, while fast grid-broadphase solver removes overlaps and span normalization keeps the frame compact. Qualitatively, shallow starts (e.g., 8 clusters) emphasize global structure and produce a compact view of the major groups. As the start level moves deeper (dozens $\rightarrow$ hundreds $\rightarrow$ $\sim$10k clusters), the layout gains degrees of freedom: anchors spread more uniformly, leaving less unused space and reducing the risk of crowding when many children will be mapped downstream. In practice this "start level" behaves like a zoom control for the subsequent h-NNE refinement: users can optionally select a coarse, high-level view (e.g., 3 - 100 clusters) or a finer, more detailed initialization (e.g., 2K or 10K clusters) depending on downstream visualization goals. The subcaptions report wall-clock times for each panel, showing that packing scales from sub-second for few anchors to seconds/minutes for $\mathcal{O}(10^4)$ anchors on a single CPU core, while maintaining visually coherent, overlap-free layouts.

4 Children-inside-parent (2D)

Given a parent disc (P, R) and children $\{(a_i, r_i^0)\}_{i=1}^m$, we produce $\{(p_i, r_i)\}$ inside the parent, with high fill and no overlaps.

1. **Local pack (geometry preservation).** Children of a given parent are first packed *locally* using the 2D fast layer of Sec. 3 on their own anchors and base radii, yielding provisional centers $\{p_i^{\text{loc}}\}_{i=1}^m$ that approximately preserve mutual-kNN relations among siblings.

2. **Isotropic fit into the parent (centers and radii).** Let $P \in \mathbb{R}^2$ and $R > 0$ be the parent center and radius. Set $c = \frac{1}{m} \sum_i p_i^{\text{loc}}$ and

$$\text{ext} \;=\; \max_{i=1,\dots,m} \Big(\|p_i^{\text{loc}} - c\| + r_i^0 \Big), \qquad s \;=\; \frac{\text{fill} \cdot R}{\text{ext}},$$

with fill $\in [0.95, 0.99]$ (we use fill $= 0.99$). Apply a single *isotropic* scale to both centers and radii:

$$p_i^{\text{fit}} \;=\; P + s\,(p_i^{\text{loc}} - c), \qquad \tilde{r}_i \;=\; s\,r_i^0, \tag{10}$$

and work henceforth with *relative* centers $q_i = p_i^{\text{fit}} - P$ (so the parent is centered at the origin). *Motivation.* Compared to a center-only fit, this isotropic step caps all children radii proportionally. Empirically, it fits all children reliably but may over-shrink the smallest ones because s is tied to the maximal extent ext.

3. **Balloon inflation under hard caps (a few small steps).** The above robust fit may cause smaller radii children discs to over-shrink due to the use of the largest sibling radii and norm for scaling. A small inflation step helps enlarge small discs and fill the parent. Starting from $\tilde{r} = \{\tilde{r}_i\}$, we perform T very small vectorized inflation steps with rate $\eta \in (0, 1)$ (we use $\eta = 0.25$, $T = 5$). At each step $t = 0, \dots, T-1$:

$$r_i^{t+\frac{1}{2}} \;\leftarrow\; (1+\eta)\, r_i^t, \tag{11}$$

$$r_i^{t+1} \;\leftarrow\; \min\Big\{\, r_i^{t+\frac{1}{2}},\; \underbrace{R - \|q_i\|}_{\text{wall cap}},\; \underbrace{\min_{j \neq i} \big(\|q_i - q_j\| - r_j^t \,\big)}_{\text{pair cap (current step)}} \Big\}_+, \tag{12}$$

initialized with $r_i^0 = \tilde{r}_i$. The $(\cdot)_+$ denotes clamping at zero. Equation (12) guarantees feasibility *relative to the current step*: every r_i^{t+1} respects the parent wall and the current radii of its neighbors (hence no new overlaps are introduced by growth). Because the inflation is incremental and capped every step, the sequence $\{r^t\}$ monotonically converges to a locally maximal non-overlapping assignment at the given centers.
Motivation. Iterating a small multiplicative growth with analytic caps is a fast surrogate for dense circle packing (exact joint optimization would be NP-hard). It keeps computations vectorized and avoids any center motion, which is important for preserving the sibling geometry obtained in step 1.

4. **Span normalization (compactness) with light safety.** Finally, we normalize the *center span* inside the parent so that the median kNN edge length among $\{q_i\}$ is a fixed multiple of the median touching length among the final radii $\{r_i^T\}$:

$$\text{median}_{(i,j)\in E} \|q_i - q_j\| \;\approx\; \beta \cdot \text{median}_{(i,j)\in E} (r_i^T + r_j^T), \qquad \beta \simeq 1.25,$$

by shrinking q_i about the origin (the parent center) and applying a short overlap separation with a tiny gap (about 1% of the median radius) *inside the parent.* This brings siblings into a compact, visually balanced configuration without changing their radii. We then output $p_i = P + q_i$ with radii $r_i = r_i^T$.

Algorithm 2 Children-inside-parent (2D): isotropic fit + balloon + span norm

```
Require: Parent $(P, R)$; local children $\{(a_i, r_i^0)\}_{i=1}^m$
1: $\{p_i^{\mathrm{loc}}\} \leftarrow$ 2D fast packing on $(a_i, r_i^0)$                              ▷ Sec. 3
2: $c \leftarrow \frac{1}{m}\sum_i p_i^{\mathrm{loc}}$;  ext $\leftarrow \max_i(\|p_i^{\mathrm{loc}} - c\| + r_i^0)$;  $s \leftarrow$ fill $\cdot R/$ext
3: $q_i \leftarrow s\,(p_i^{\mathrm{loc}} - c)$;  $r_i^0 \leftarrow s\, r_i^0$                                   ▷ relative to $P$
4: for $t = 0, \ldots, T-1$ do                                        ▷ balloon steps, $\eta \in (0,1)$
5:     $r_i^{t+\frac{1}{2}} \leftarrow (1+\eta)\, r_i^t$                                          (vectorized)
6:     $r_i^{t+1} \leftarrow \min\Big\{r_i^{t+\frac{1}{2}},\ R - \|q_i\|,\ \min_{j\neq i} \|q_i - q_j\| - r_j^t\Big\}_+$
7: $q \leftarrow$ span_normalize_inside_parent$(q, r^T; \beta)$        ▷ kNN median target + tiny-gap separation
8: return $p_i \leftarrow P + q_i$,  $r_i \leftarrow r_i^T$
```

Remarks and trade-offs.

- **Bias toward small children.** The isotropic fit multiplies all radii by the same s. Because s is chosen from the maximal extent, smaller children often have more slack to grow during the balloon phase; this produces fuller parents while keeping the largest children near their geometric caps.
- **Parameters.** We use fill=0.99, η=0.25, T=5, and β=1.25. Fewer balloon steps suffice when siblings are already dense; increasing T beyond 5 gives diminishing returns.
- **Complexity.** Each balloon step is fully vectorized: wall caps are $O(m)$ and pair caps are computed from the $m \times m$ distance matrix of $\{q_i\}$ once per step (acceptable because m is the number of children *of a single parent*, typically small relative to the global n). The span normalization uses the same kNN-edge statistics as Sec. 3.

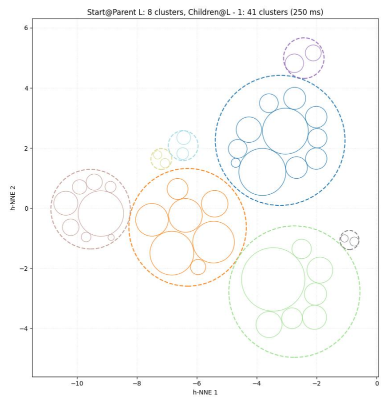
4.1 Visualization of children placement across levels

Figure 3 illustrates how the proposed children-inside-parent procedure composes across successive levels of the FINCH hierarchy. To relate, we use the same dataset as in Figure 2 i.e., 200K samples in 2048D; FINCH levels with cluster counts [49,316, 11,032, 2,543, 421, 41, 8] (finest $\rightarrow$ coarsest). In each panel, parent circles are first packed to preserve local k-NN relations while respecting cluster weights ($r_c \propto \sqrt{|C|}$). Given a fixed parent, its children are (i) isotropically fit into the parent by a joint center/radius scaling, (ii) "ballooned" via a one-shot radius expansion under analytic wall and pairwise caps (no center motion), and (iii) lightly polished by a capped, center-only separation to re-establish a small safety gap.

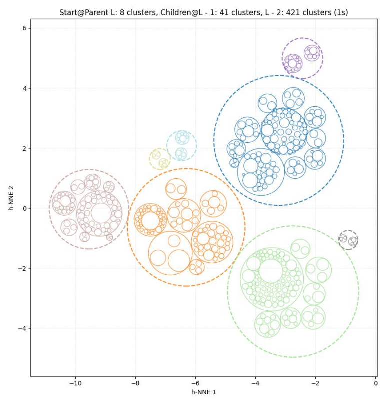
Starting at coarser levels (top row, left) yields a compact high-level arrangement that remains stable as deeper levels are inserted. Beginning at a finer parent level (bottom row) produces a more uniform spread at the expense of slightly less global regularity—useful as an optional user-controlled "zoom" on very large datasets. Across all starts, the method fills available area inside each parent while avoiding overlaps and preserving neighborhood structure inherited from the parent packing.

5 ND fallback ($D > 2$): fast, memory-safe packing in $\mathbb{R}^D$

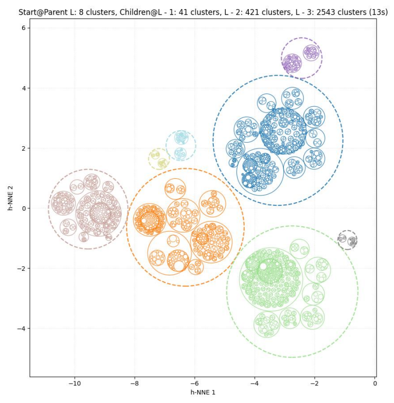
When the ambient dimension $D > 2$, we retain the same geometric objective: place cluster anchors as non-overlapping D-balls while preserving local neighborhood relations up to mild relaxation and



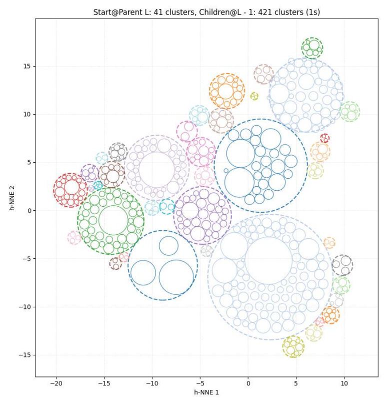
(a) Start at 8 parents; add 41 children ($8 \to 41$) (250 ms)



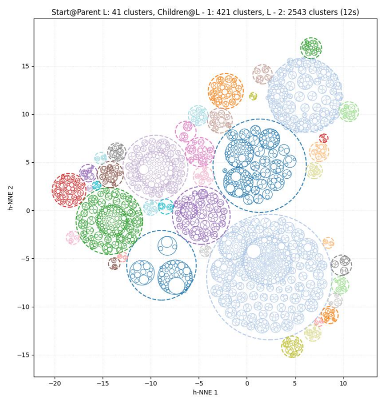
(b) Continue: add 421 children ($8 \to 41 \to 421$) (1 s)



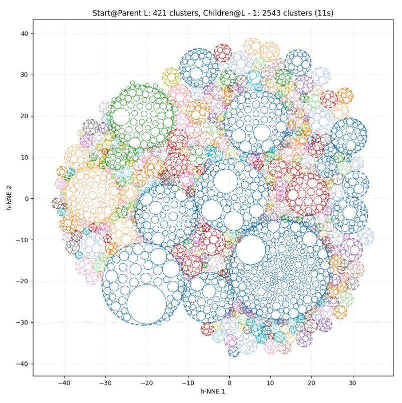
(c) Continue: add 2,543 children ($8 \to 41 \to 421 \to 2543$) (13 s)



(d) Start at 41 parents; add 421 ($41 \to 421$) (1 s)



(e) Continue: add 2,543 ($41 \to 421 \to 2543$) (12 s)



(f) Start at 421; add 2,543 ($421 \to 2543$) (11 s)

Figure 3: Nested placement in 2D: parent circles are packed at the starting level, and children are placed *inside* each parent by isotropic center/radius fit followed by balloon inflation under hard wall/pairwise caps and a light center-only polish. Rows show different starting levels; columns add one deeper level left$\to$right.

keeping the layout compact. Let $\{(p_i, r_i)\}_{i=1}^{n}$ be centers and radii in $\mathbb{R}^D$ with hard constraints

$$\|p_i - p_j\| \;\geq\; r_i + r_j + \gamma \quad \forall\, i \neq j, \qquad r_i > 0, \tag{13}$$

where $\gamma \geq 0$ is a small safety gap (Section 3.2). Our ND fallback keeps the algorithmic structure of the 2D layer but adapts the broadphase and the solver so that both time and memory scale well in higher dimensions.

Why a separate ND path? The 2D layer uses a uniform grid broadphase and several compaction/edge-target steps tuned to planar layouts. In $\mathbb{R}^D$, naively lifting those choices either causes a combinatorial explosion (e.g., 3^D neighbor stencils), or complicates the geometry (Procrustes alignment in 2D has a closed form, but not all normalization steps extend cleanly). Our ND path therefore (i) keeps only the *overlap resolver* and a *light inside-parent fit*, (ii) avoids any memory-heavy constructs (no meshgrids over D axes), and (iii) uses simple, dimension-aware mass models to bias motion, which is standard in physically-based solvers [7, 8].

5.1 Broadphase in $\mathbb{R}^D$: uniform grid hashing

We detect candidate pairs for Eq. (13) using a uniform grid with cell size h chosen from robust radius statistics:

$$h = \max\big(2 \max r,\ 4 \operatorname{median} r,\ 10^{-6}\big).$$

Each center p_i is mapped to an integer cell index $c_i = \lfloor p_i/h \rfloor \in \mathbb{Z}^D$ and inserted into a hash table keyed by c_i (a standard spatial hashing scheme [12]). For each particle i we only test buckets within a fixed local stencil $\{-1, 0, 1\}^D$ around c_i.

Memory safety. We *do not* materialize the full D-dimensional offset mesh (which would be 3^D entries); instead, offsets are generated on the fly by a small iterator (cartesian product) and evaluated per queried cell. This keeps memory $\mathcal{O}(n)$ even when $D \le 6$.

Cutover for high D. When $D > 6$, the size of the stencil becomes large enough that the broadphase yields diminishing returns. In that regime we switch to a vectorized *all-pairs* pass: compute (i, j) for $i < j$ once, then reuse it across sweeps. Empirically this is faster and simpler beyond the cutover dimension.

5.2 Overlap resolution: minimum-translation impulses with dimension-aware masses

Given a candidate pair (i, j) with separation $d_{ij} = \|p_i - p_j\|$ and unit direction $u_{ij} = (p_i - p_j)/\max(d_{ij}, \varepsilon)$, the penetration (positive if violating) is

$$\Delta_{ij} = (r_i + r_j + g) - d_{ij}.$$

We resolve violations by applying equal-and-opposite *minimum-translation* impulses along u_{ij},

$$\Delta p_i = \tau_i\,(\Delta_{ij} + \varepsilon)\,u_{ij}, \qquad \Delta p_j = -\tau_j\,(\Delta_{ij} + \varepsilon)\,u_{ij}, \tag{14}$$

with weights τ_i, τ_j chosen from inverse masses $w_i = 1/m_i$ via

$$\tau_i = \frac{w_i}{w_i + w_j}, \qquad \tau_j = \frac{w_j}{w_i + w_j}.$$

We use two mass models:

$$\begin{aligned} \textsc{powerD}: &\quad m_i \propto r_i^D, &&\text{(volume-weighted)},\\ \textsc{radius}: &\quad m_i \propto r_i, &&\text{(lighter bias)}. \end{aligned}$$

The POWERD model is the default, damping the motion of large spheres and preventing them from "steamrolling" small ones during separation, a standard trick from rigid-body simulation [7, 8]. The small ε (e.g., 10^{-9} times a median radius) avoids division-by-zero when $p_i = p_j$.

Solver style. We employ our Gauss–Seidel/Jacobi hybrid (*Capped Jacobi*) introduced earlier: candidate pairs are formed once per sweep, impulses for all pairs are computed, then *accumulated* per node and capped before being applied. This combines the stability of Gauss–Seidel with the vectorization of Jacobi (see Sec. 3.2 for the 2D variant. Two early exits are used: (i) if no pair violates Eq. (13) beyond tolerance, and (ii) if the maximum penetration falls below a relative threshold. We also recenter the configuration each sweep to avoid drift.

5.3 Complexity

Let ρ be the average number of points per occupied cell. For $D \leq 6$, each particle tests at most 3^D neighbor buckets; if the distribution is not adversarial, the total checked pairs scale as $\mathcal{O}(n\,\rho\,3^D)$ per sweep, typically close to linear. For $D > 6$ we switch to vectorized all-pairs with $\mathcal{O}(n^2)$ arithmetic per sweep, but entirely in dense BLAS-friendly operations. In both cases we observe a small, data-independent number of sweeps (tens to a few hundreds) to reach feasibility in practice.

5.4 ND inside-parent packing

The 2D layer fills parents aggressively (Section 4) with area-aware inflation and center-only polishing. In $\mathbb{R}^D$ we use a lighter routine that is fast and stable:

1. **Local pack.** Children anchors within a parent are first packed in $\mathbb{R}^D$ by calling the ND separator on their anchor positions with their base radii (§5 above).

2. **Isotropic fit (centers and radii).** Let c be the centroid of the packed children, and define the robust extent $\mathrm{ext} = \max_k \|P_k - c\| + r_k$. Scale both centers and radii by $s = \mathrm{fill\_frac} \cdot R/\mathrm{ext}$ and translate to the parent center p_{par}:

$$P_k \leftarrow p_{\mathrm{par}} + s(P_k - c), \qquad r_k \leftarrow s\,r_k,$$

   with fill_frac ≈ 0.98. This preserves relative sizes and ensures everything starts strictly inside the parent ball.

3. **Safe tighten (centers only).** Run a few sweeps of the ND separator *inside the parent* with pair gap $g = 0$ to remove incidental child–child grazes introduced by the fit. Because radii are already feasible w.r.t. the parent wall, only small center corrections are needed.

This variant avoids radius optimization in $D > 2$ (which would require repeated dense $n \times n$ distance updates and quickly becomes costly), yet fills most of the available volume because radii were uniformly scaled during the fit.

5.5 Relation to 2D layer and limitations

The ND fallback intentionally omits the 2D-specific span normalization and edge-target projection. In high dimensions, those steps do not directly translate to visually meaningful compaction, while the uniform-grid + impulse solver already provides good stereo-structure preservation with minimal code and memory. Limitations are (i) the $\mathcal{O}(n^2)$ all-pairs regime for very high D or highly clustered data, and (ii) the lack of radius optimization inside parents in ND (a design choice for speed and simplicity). In practice, the ND module is used at the top few coarse levels to initialize structure; fine levels with more dense clusters then follows the standard h-NNE scheme downstream.

6 Integration with h-NNE

Let the FINCH hierarchy be $\mathcal{L} = \{\Gamma_\ell\}_{\ell=0}^{L-1}$ with label matrix $\mathbf{Z} \in \mathbb{Z}^{N\times L}$, where $z_{i\ell} \in \{1, \ldots, K_\ell\}$ denotes the cluster of point x_i at level ℓ. At a chosen set of upper levels $\ell \in \{\ell_{\min}, \ldots, L-1\}$ we run the hierarchical packing layer (Sec. 3–5) to obtain packed anchors $\{(p_c^{(\ell)}, r_c^{(\ell)})\}$ (2D fast path or ND fallback). The topmost packed level $\ell_\star$ serves as the *seed* for h-NNE's `multi_step_projection`.

Seed anchors and per-anchor scale. Let $\mathcal{C}_{\ell_\star} = \{1, \dots, K_{\ell_\star}\}$ index seed anchors $\{p_c^{(\ell_\star)}\}_{c \in \mathcal{C}_{\ell_\star}}$. We define a robust *anchor scale* $\widehat{r}_c$ for each seed anchor from local anchor–anchor spacings with a symmetric cap that prevents cross-cluster mixing at dense levels. Construct a k-NN graph on the seed anchors (typically $k{=}1$):

$$\mathcal{N}_k(c) \;\subseteq\; \mathcal{C}_{\ell_\star} \setminus \{c\}, \qquad d_{cc'} \;=\; \big\|p_c^{(\ell_\star)} - p_{c'}^{(\ell_\star)}\big\|_2.$$

Define the one-sided k-radius $d_c^{(k)} = \mathrm{median}\{\, d_{cc'} : c' \in \mathcal{N}_k(c) \,\}$. To guarantee that two neighboring anchors cannot encroach on each other during the first mapping step, we apply a *symmetric half-1NN* cap (for $k{=}1$ this is the pairwise half distance):

$$\widehat{r}_c \;=\; \rho \cdot \min\Big\{ d_c^{(k)}, \min_{c' \in \mathcal{N}_k(c)} d_{c'}^{(k)} \Big\} \Big/ 2, \qquad \rho \in (0,1).$$

We use $\rho \approx 0.45$–0.50 in practice; larger values may be safe when seed anchors are already strictly separated by the packer. Distances $d_{cc'}$ are computed *exactly* with a balanced KD-tree [13] when $K_{\ell_\star}$ is moderate; for very large sets we employ NN-Descent [14] or batched KD-tree queries to bound memory.

Vectorized point-to-anchor update. For the descent $\ell \to \ell - 1$, define the index map of point i to its level-ℓ anchor by

$$c(i) \;=\; z_{i\ell}, \qquad S_c \;=\; \{\, i : z_{i\ell} = c \,\}.$$

Compute per-anchor statistics over its children (points or lower-level anchors) in the preliminary embedding space (PCA):

$$\mu_c \;=\; \frac{1}{|S_c|} \sum_{i \in S_c} x_i, \qquad R_c \;=\; \max_{i \in S_c} \big\|x_i - \mu_c\big\|_2,$$

(with the convention $R_c \leftarrow \max\{R_c, \varepsilon\}$, $\varepsilon{=}1$, to avoid division by zero). The v2 seed step maps all points in one vectorized operation:

$$y_i \;=\; p_{c(i)}^{(\ell)} \;+\; \widehat{r}_{c(i)} \, \frac{x_i - \mu_{c(i)}}{R_{c(i)}}.$$

This places each child *inside* a radius-$\widehat{r}_c$ neighborhood of its parent anchor, preserving the shape of S_c up to an isotropic rescaling. The symmetric cap on $\widehat{r}_c$ prevents cross-cluster spill even when anchors are dense.

Subsequent h-NNE steps. After the seed step at $\ell_\star$, we proceed with the standard h-NNE iterations down the hierarchy $(\ell_\star - 1, \ell_\star - 2, \dots)$, using the same vectorized formula but *without* externally supplied anchor radii (i.e., the routine's internal per-step scale is disabled or set to `None` so that the seed geometry governs global scale). In practice, we pass the packed $\{p_c^{(\ell_\star)}\}$ and $\{\widehat{r}_c\}$ as *seed anchors* and *seed radii* into `multi_step_projection`; thereafter, anchor radii are not re-estimated, maintaining the v2 layout's compactness and separation while the original h-NNE update refines local detail.

Implementation notes. All quantities are computed in a fully vectorized manner over anchors and points. For very large $K_{\ell_\star}$, we use either (i) batched KD-tree queries to compute exact 1-NN distances without forming dense distance matrices, or (ii) NN-Descent to obtain high-recall neighbors and then compute exact distances only for those candidates (memory-safe hybrid). Algorithm 3. provides a summary of complete h-NNE v2 flow.

Algorithm 3 h-NNE v2 Algorithmic Summary: hierarchical packing (top $\rightarrow$ bottom)

```
Require: $X \in \mathbb{R}^{N\times D}$ (prelim. embedding), FINCH labels $\mathbf{Z} \in \mathbb{Z}^{N\times L}$ (top at $\ell = L-1$); subset of
    levels to pack $\mathcal{S} = \{\ell_{\min}, \dots, L-1\}$
1: for $\ell = L-1$ down to $\ell_{\min}$ do
2:     compute anchors $\{a_c^{(\ell)}\}$ and base radii $\{r_{c,\text{base}}^{(\ell)}\}$
3:     if $\ell = L-1$ then                                                ▷ top level
4:         if $D = 2$ then $\{p_c^{(\ell)}\} \leftarrow$ 2D fast packing on $(a_c^{(\ell)}, r_{c,\text{base}}^{(\ell)})$
5:         else $\{p_c^{(\ell)}\} \leftarrow$ ND overlap separation on $(a_c^{(\ell)}, r_{c,\text{base}}^{(\ell)})$
6:     else                                                                 ▷ children inside parent
7:         for each parent $P$ at level $\ell+1$ do
8:             collect its children $\mathcal{C}_P \subset \mathcal{C}_\ell$
9:             run 2D fast packing locally on $(a_c^{(\ell)}, r_{c,\text{base}}^{(\ell)})$, $c \in \mathcal{C}_P$      ▷ if $D > 2 \rightarrow$ ND fallback
10:            Isotropic fit into parent; one-shot balloon inflation; Span normalization
11:            set $p_c^{(\ell)} \leftarrow p_P^{(\ell+1)} + p_c^{\text{rel}}$
12: return packed layout $\{(p_c^{(\ell)}, r_c^{(\ell)})\}$ for $\ell \in \mathcal{S}$          ▷ seed levels for h-NNE
```

7 Complexity, defaults, and limitations

Complexity. In 2D, one separation sweep with grid broadphase is near-linear in the number of discs n; a few sweeps suffice. Mutual-kNN on anchors is $O(n^2)$ exact but with very small constants at coarse levels (k=1 by default). Children-inside-parent operates on groups of size m_p; total cost $\sum_p \text{cost}(m_p)$ with $\sum_p m_p = n_\ell$. The anchor-to-point mapping is $O(N)$ given precomputed per-anchor statistics; KD-tree queries are $O(n \log n)$ in batches.

Defaults. We use k=1, one outer projection pass, span factor β=1.25; safety gaps $\delta \in [0.003R,\ 0.008R]$; balloon floors $\phi \in [0.75, 0.85]$ with gentle growth caps; anchor radius factor $\rho \approx 0.9$ for the point mapping. These produced compact, overlap-free layouts with faithful local relations.

Limitations and scope. The v2 layer is designed for the *upper* levels of the FINCH hierarchy—typically tens to a few thousand anchors—where packing quality materially improves the downstream mapping. It is not intended for the finest levels with extremely large anchor counts, which h-NNE treats by direct point-to-anchor projection. Within this scope, several trade-offs remain. (i) *Compactness vs. fidelity:* relaxed/capped kNN targets and span normalization favor visually compact frames over exact edge-length preservation; the knob β and the edge-relaxation schedule (Sec. 3.3) control this balance. (ii) *Heavy-tailed sizes:* when radii vary by orders of magnitude, volume-based masses can make large anchors relatively inert; switching to radius masses or lowering the growth cap $\gamma_{\max}$ restores motion, while a small uniform blend in the area initialization prevents tiny anchors from vanishing. (iii) *Pathological constellations:* highly anisotropic or filamentary anchor layouts may require a slightly larger k for stable local geometry. (iv) *Runtime scaling:* the 2D grid+*capped* Jacobi separation is near-linear per sweep but still requires multiple passes; single-level packs beyond ∼10k–15k anchors can take minutes on a single CPU core. Using the hierarchical (top-down) pack or light parallelization mitigates this. (v) *ND fallback:* for $D > 2$ we omit radius optimization inside parents to keep memory and time modest; this can leave a bit of unused volume but provides robust initialization before the final 2D visualization layer. Finally, as with most iterative relaxations, the method does not compute a global optimum; results are nonetheless stable under small perturbations and parameter defaults, and failure modes (e.g., residual overlaps or excessive spread) can be addressed by modestly increasing the final polish sweeps or tightening the gap settings.

References

[1] M. S. Sarfraz, M. Koulakis, C. Seibold, and R. Stiefelhagen, "Hierarchical nearest neighbor graph embedding for efficient dimensionality reduction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[2] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[3] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[4] M. S. Sarfraz, V. Sharma, and R. Stiefelhagen, "Efficient parameter-free clustering using first neighbor relations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[5] Y. Saad, *Iterative Methods for Sparse Linear Systems.* SIAM, 2 ed., 2003.

[6] G. H. Golub and C. F. V. Loan, *Matrix Computations.* Johns Hopkins University Press, 4 ed., 2013.

[7] D. Baraff, "Fast contact force computation for nonpenetrating rigid bodies," in *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '94)*, ACM, 1994.

[8] D. Baraff, "Physically based modeling: Rigid body simulation." SIGGRAPH Course Notes, 2001.

[9] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff, "Position based dynamics," in *Proceedings of Virtual Reality Interactions and Physical Simulations (VRIPHYS)*, 2007.

[10] E. Catto, "Soft constraints: Reinventing the spring," in *Game Developers Conference*, 2009. Tech. notes on sequential impulses / PGS.

[11] M. Macklin, M. Müller, N. Chentanez, and S. Jeschke, "Xpbd: Position-based simulation of compliant constrained dynamics," in *Motion in Games (MIG)*, 2016.

[12] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino, "Collision detection for deformable objects," *Computer Graphics Forum*, vol. 24, no. 1, 2005.

[13] J. L. Bentley, “Multidimensional binary search trees used for associative searching,” *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[14] W. Dong, C. Moses, and K. Li, “Efficient k-nearest neighbor graph construction for generic similarity measures,” in *Proceedings of the 20th International Conference on World Wide Web (WWW)*, 2011.