

- [Bernardo Taveira](#)
- [Carl Lindström](#)
- [Maryam Fatemi](#)
- [Lars Hammarstrand](#)
- [Fredrik Kahl](#)

Scalable GPU Construction of 3D Voronoi and Power Diagrams

Published in: [ACM SIGGRAPH 2026 Conference Papers](#)

Presented by: [Ikram El Ouardy](#)

Groupe de Lecture 23/09/2026

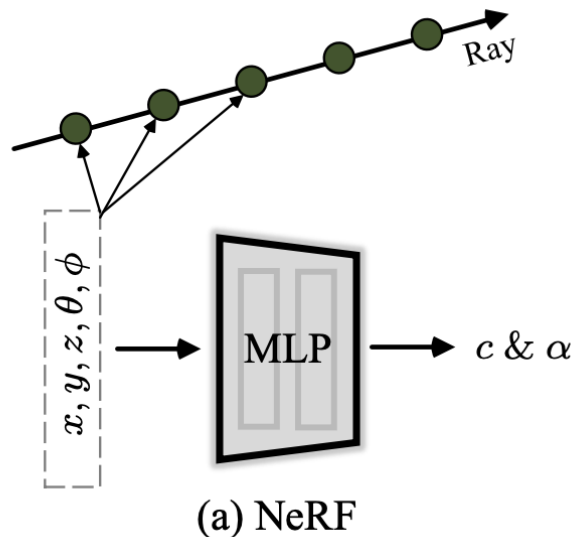


Evolution of Neural Scene Representations

NeRF: Implicit Continuous Representation

- Continuous volumetric representation
- High-quality novel-view synthesis

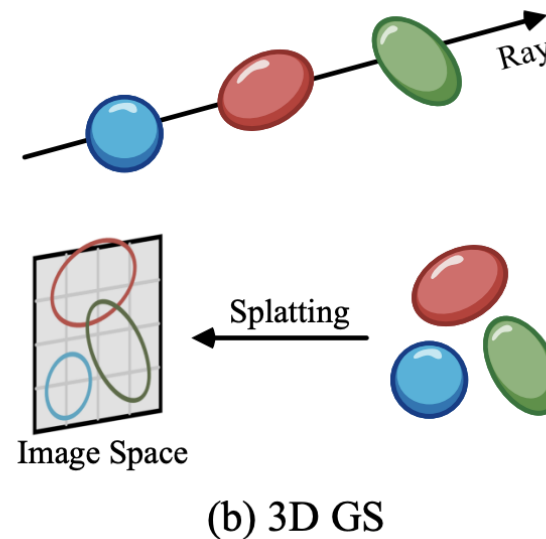
- Expensive ray sampling + MLP evaluations
- No explicit geometry/connectivity



3DGS: Explicit Point-Based Representation

- Fast differentiable rasterization
- Real-time rendering

- Rasterization-related limitations
- No explicit volumetric mesh



Can we retain efficient rendering while explicitly partitioning 3D space?

Mesh-Based
Neural
Rendering

Radiant Foam: Voronoi-Based Neural Rendering

Scene Representation:

- 3D space is **partitioned into Voronoi cells**
- Each cell stores learned **density and appearance**
- Rays **traverse cells** and accumulate their contributions

Why Voronoi?

- Explicit volumetric partition
- Explicit cell adjacency
- Adaptive spatial discretization



Voronoi

Volumetric Render

But during training...

Sites move

Voronoi connectivity
changes

Mesh must be updated

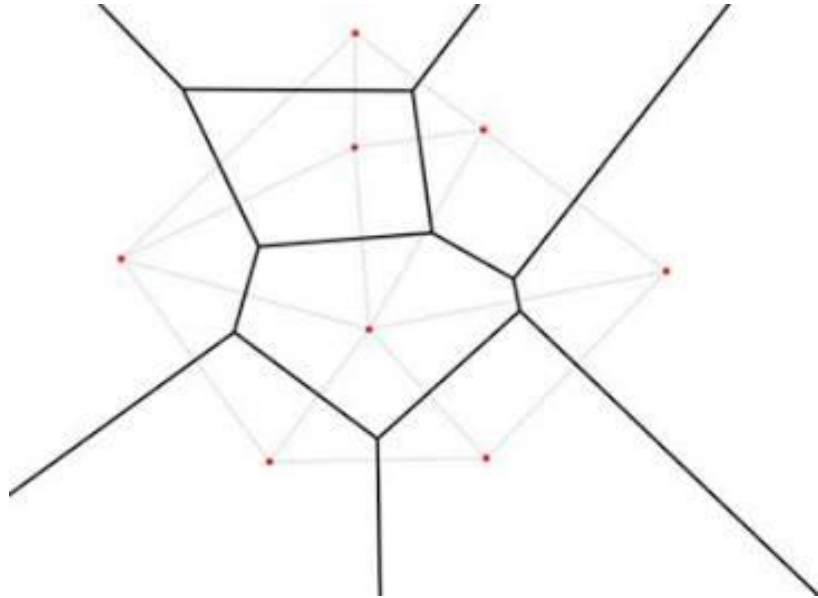
**Bottleneck: frequent Voronoi/Delaunay reconstruction
becomes expensive at millions of sites**

Voronoi, Power Diagram and Delaunay Duality

Voronoi cells

$$C_i = \{x, : \|x - p_i\|^2 \leq \|x - p_j\|^2 \forall j\}.$$

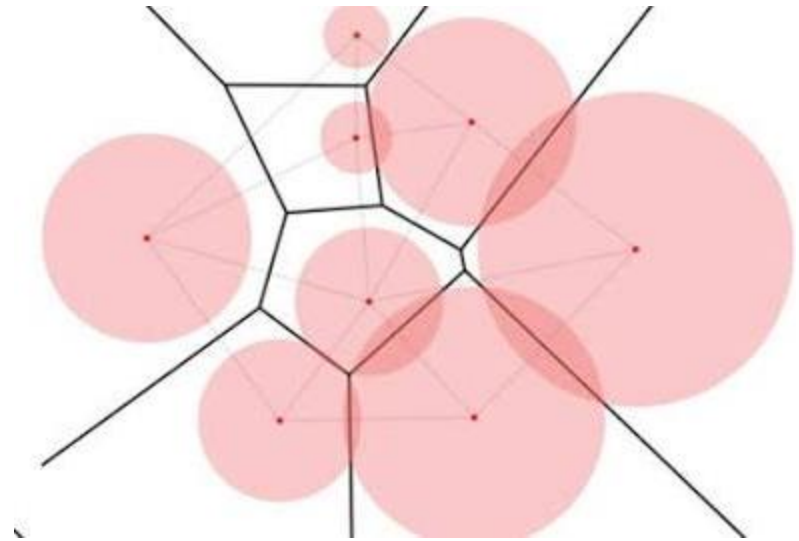
- Cells are defined by seeds
- boundary = perpendicular bisector between two seeds $s_i s_j$



Power Diagram

$$C_i = \{x, : \|x - p_i\|^2 - w_i \leq \|x - p_j\|^2 - w_j \forall j\}.$$

- Cells are defined by seeds + weights
- hyperplane defined by equality of power distances (hyperplane orthogonal to $s_i s_j$, shifted by $w_i - w_j$)



Voronoi = special case of Power Diagram when all weights are equal.

Existing Construction Methods and Their Limits

Incremental insertion

Principle:

Build the Delaunay tetrahedralization by inserting sites and locally retriangulating affected regions.

Examples:

- Bowyer-Watson
- CGAL / Geogram
- gDel3D on GPU

Convex-Hull lifting

Principle:

Transform the geometric problem into a convex-hull problem in one higher dimension:

Examples

- QHull / SciPy

Cell-oriented clipping

Principle:

Construct each cell independently: by progressively clipping a convex volume with neighboring bisecting half-spaces.

Examples:

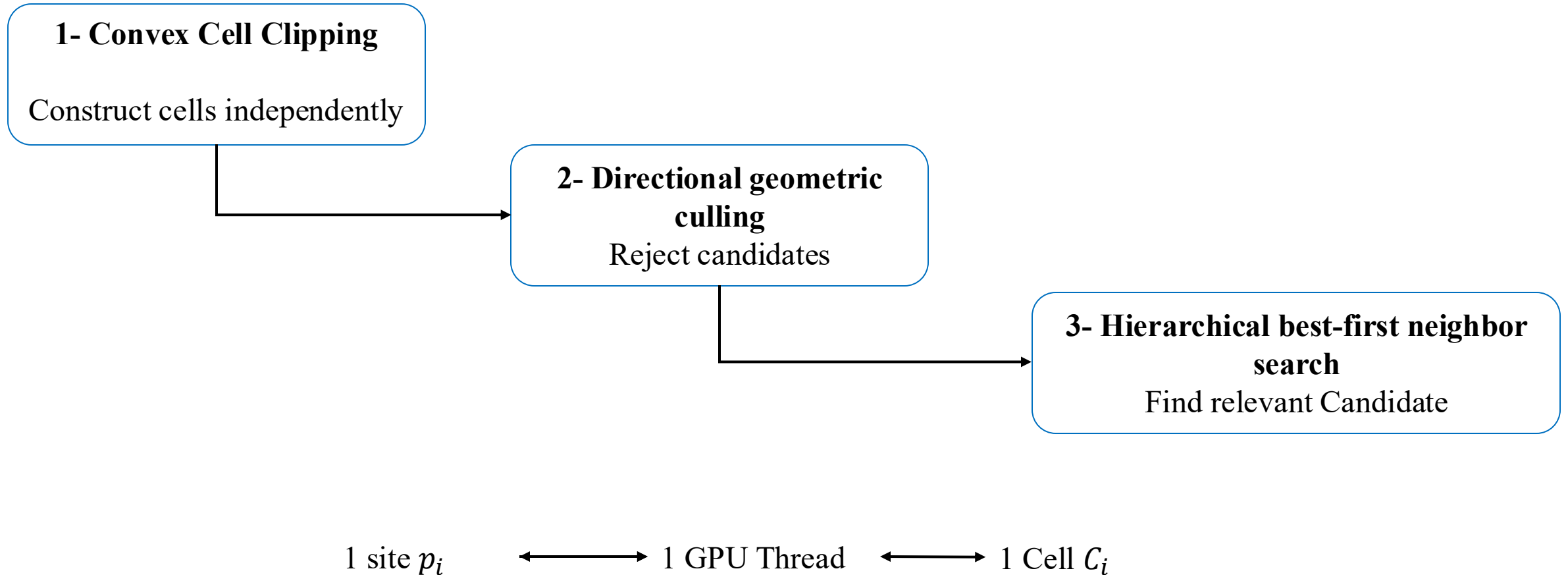
- Voro++
- Ray et al.
- Basselin et al.

- Large-scale computational cost
- GPU scalability / resource limitations
- Assumptions on point distributions
- Weighted case less broadly supported

Key advantage:

Each cell is constructed independently
=> **Massive GPU parallelism**

Method Overview: GPU-Parallel Cell Construction



Step 1: Convex Cell Clipping¹

Construct each cell as an intersection of half-spaces

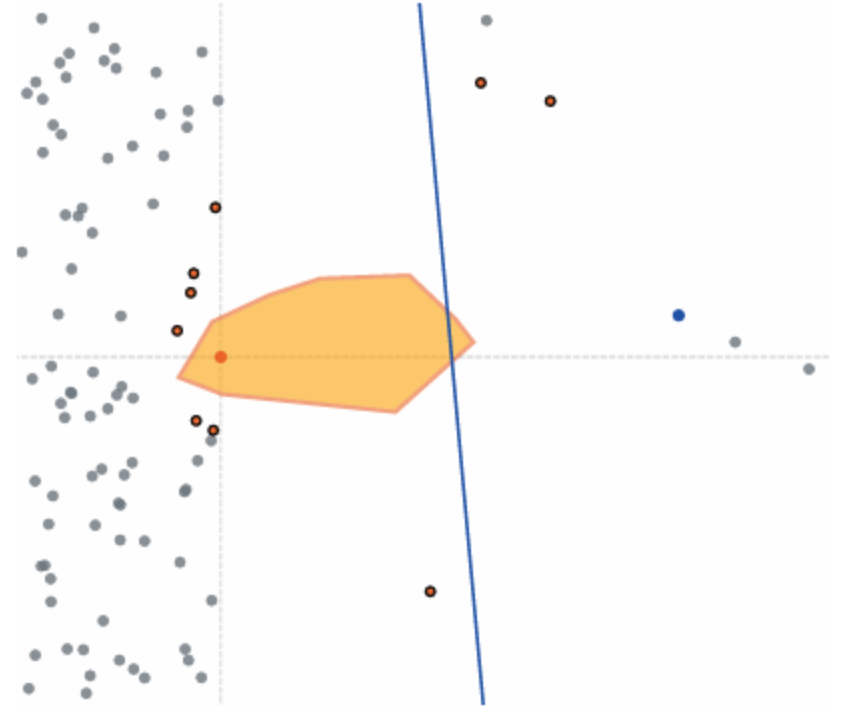
For each site p_i :

1. initialize C_i with a big convex volume;
2. candidate p_j defines a bisecting half-space H_{ij} ;
3. Clip the current cell;

$$C_i \leftarrow C_i \cap H_{ij}$$

- Remove vertices outside H_{ij}
- Create the new intersection face
- Update the cell

4. repeat.



The Problem:

Naively: N cells $\times$ $(N - 1)$ candidate sites

➡ Testing every possible candidate is prohibitively expensive.

Step 2: Geometric
Culling

Step 2: Candidate Culling

Previous approach: Isotropic Bound

- **Isotropic Cell Bound**

Previous work uses one radius:

$$r_i = \max_{v \in C_i} \|v - p_i\|$$

- **Distance to the bisecting plane**

For a site p_i and a candidate p_j :

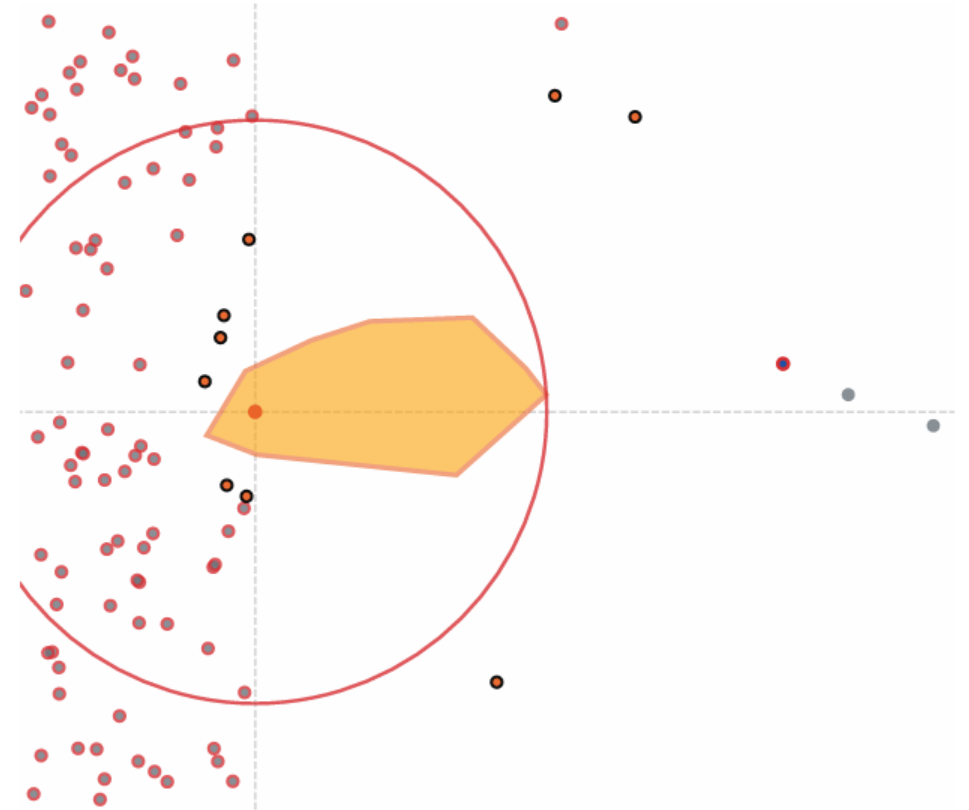
$$d_{ij} = \frac{|p_i - p_j|^2 + w_i - w_j}{2|p_i - p_j|}$$

d_{ij} : the distance from p_i to the bisecting plane induced by p_j

- **Culling Test:**

- If $d_{ij} > r_i \Rightarrow$ Cull p_j
- Else: $d_{ij} \leq r_i \Rightarrow$ Keep as candidate

One radius is used for all directions.



For an anisotropic cell more candidate can be culled!

Step 2: Directional Culling

Replace one global radius with a direction-dependent bound

1. Bound the current cell C_i

$$\text{AABB}(C_i)$$

2. Locate the candidate direction

Determine the octant of p_j relative to p_i

$$p_j - p_i \rightarrow \text{sign}(x, y, z) \rightarrow \text{octant}$$

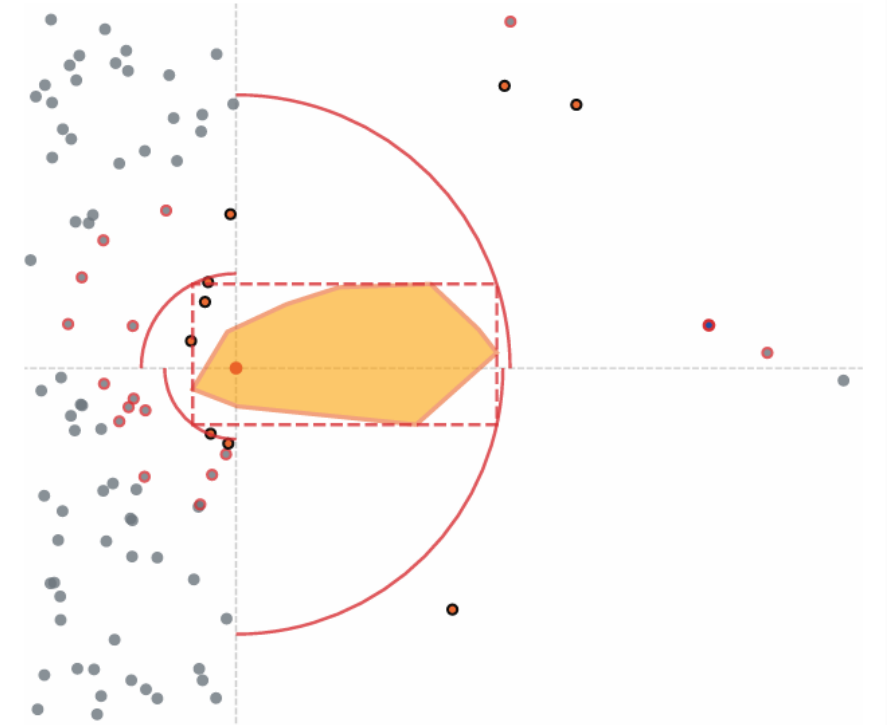
3. Compute a directional bound

$$r_i(p_j) = \max_{c \in C_{\text{Octant}}} \|c - p_i\|$$

Then:

$$d_{ij} > r_i(p_j) \Rightarrow \text{Cull } p_j$$

Same bisector-distance test, but with a much tighter direction-dependent bound.



Cell shrinks $\Rightarrow$ AABB shrinks $\Rightarrow$ tighter bounds $\Rightarrow$ More culling

Step 3: Hierarchical Neighbor Search

Cull groups of candidates and process the most promising regions first

Bounding-volume culling

Can this p_j clip C_i ? $\longrightarrow$

Can Any site inside this BVH node clip C_i ?

Node culling score:

$$\delta_n = d_n - r_n$$

where conceptually:

- d_n : lower bound on the **closest possible bisector** from the BVH node n
- r_n : directional upper bound of the current cell toward that node n

Then:

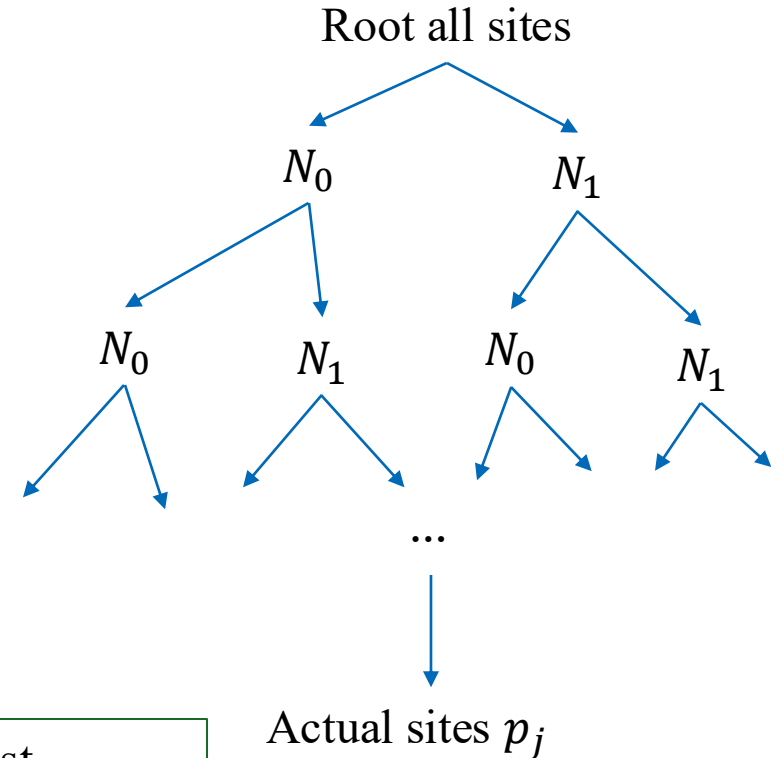
$\delta > 0 \Rightarrow$ Prune entire BVH node

$\delta \leq 0 \Rightarrow$ node remains potentially relevant

PS: Prioritize the branch giving the smallest δ the most likely to induce a restrictive cut

Organize sites using a BVH:

- Each node represents a spatial group of sites
- Node stores: bounding box + maximum weight



Results

Synthetic datasets

- Uniform / white noise
- Density gradient
- Gaussian clusters, $K = 5, 10$
- $0.1M \rightarrow 15M$ sites

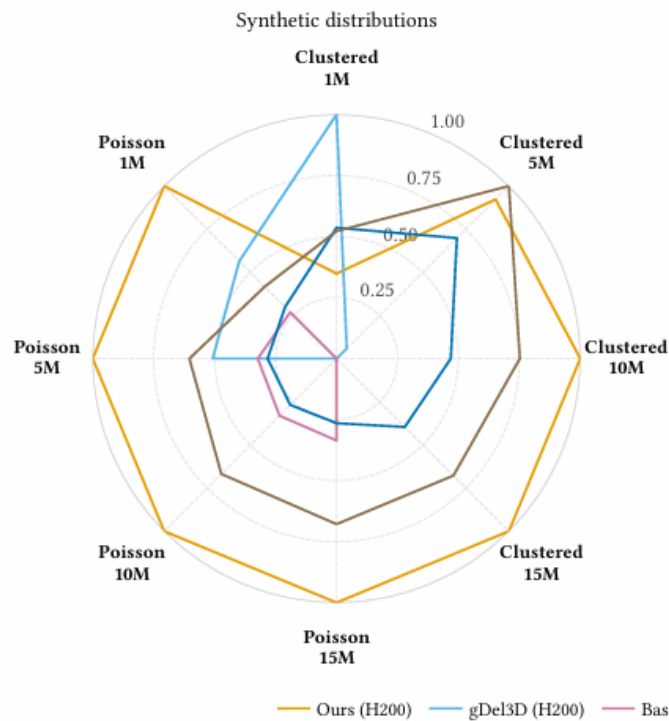
Real World scenes



Fig. 7. Images from the real-world test cases (described in Sec. 5.1), with point sets projected to illustrate the density and spatial spread of the data. The scenes span a range of indoor and outdoor environments with complex spatial distributions and point counts ranging from 2M to 4.2M.

Results

Synthetic datasets



Real World scenes

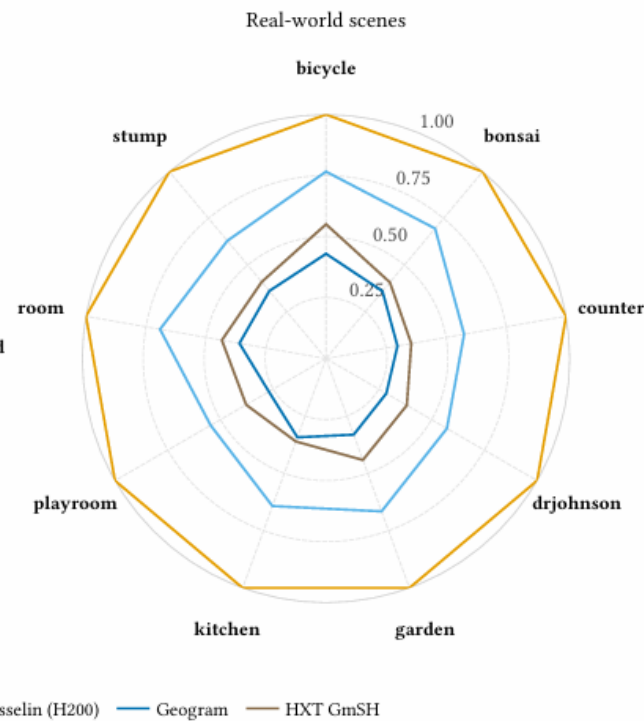


Fig. 5. Runtime comparison on synthetic and real-world point sets. The outer ring corresponds to the fastest method on that dataset (score = 1). A score of zero indicates that the method failed to complete a valid result. *Poisson* represents the average of the Poisson-distributed test cases (white noise and density gradient). *Clustered* represents the average of the clustered test cases. Our method consistently achieves scores near the outer ring across all distributions, while competitors degrade at higher point counts and non-uniform distributions.

Results

Real World scenes

Table 1. **Voronoi diagram creation/Delaunay triangulation.** Runtime results (in seconds) on point sets obtained from Radiant Foam checkpoints trained on the Mip-NeRF 360 dataset. Color coding denotes fastest, second fastest, and third fastest.

		bicycle 4.2M	bonsai 2.0M	counter 2.0M	drjohnson 2.8M	garden 4.1M	kitchen 2.0M	playroom 3.2M	room 1.9M	stump 4.2M
CPU	SciPy	279.710	121.480	127.273	187.908	260.904	133.174	207.022	119.729	268.345
	Voro++	-	-	-	-	-	-	-	-	-
	CGAL	20.445	9.591	9.366	13.522	19.205	9.781	15.016	8.877	19.559
	CGAL Parallel	26.986	18.900	97.873	21.249	20.066	57.639	49.721	40.528	40.466
	Geogram	1.593	0.826	0.859	1.244	1.732	0.865	1.370	0.812	1.560
	HXT GmSH	1.243	0.730	0.720	0.928	1.299	0.821	0.980	0.675	1.375
5090	Basselin*	-	-	-	-	-	-	-	-	-
	RF Del	124.929	25.299	24.980	24.201	40.413	24.762	93.342	16.889	44.777
	gDel3D	1.204	0.570	0.592	0.826	1.168	0.604	0.931	0.542	1.197
	Ours	0.570	0.245	0.248	0.331	0.496	0.235	0.359	0.240	0.491
H200	Basselin*	-	-	-	-	-	-	-	-	-
	RF Del	250.688	22.501	24.800	25.142	39.551	25.851	122.038	18.020	43.680
	gDel3D	0.893	0.428	0.444	0.620	0.862	0.462	0.678	0.424	0.896
	Ours	0.684	0.298	0.256	0.355	0.574	0.297	0.371	0.293	0.564

Table 2. **Power diagram creation/regular Delaunay triangulation.** Runtime results (in seconds) on point sets obtained from Radiant Foam checkpoints trained on the Mip-NeRF 360 dataset. Color coding denotes fastest, second fastest, and third fastest.

		bicycle 4.2M	bonsai 2.0M	counter 2.0M	drjohnson 2.8M	garden 4.1M	kitchen 2.0M	playroom 3.2M	room 1.9M	stump 4.2M
CPU	SciPy	212.538	94.919	117.140	174.597	230.981	124.567	188.740	110.533	214.884
	CGAL	16.000	7.754	8.606	12.016	17.151	8.931	13.742	8.033	16.669
	CGAL Parallel	-	110.926	141.335	49.209	269.222	118.085	174.701	63.181	154.698
	Geogram	2.939	1.368	1.444	1.776	2.670	1.379	2.666	1.321	3.206
5090	Basselin*	-	-	-	-	-	-	-	-	-
	Ours	1.134	0.319	0.266	0.346	0.613	0.248	0.385	0.264	0.836
H200	Basselin*	-	-	-	-	-	-	-	-	-
	Ours	1.697	0.827	0.414	0.419	1.038	0.465	0.462	0.344	1.280

Et c'est tout!

Merci pour votre attention