Bounded-Latency Spatial Computation via Continuous Fields

Nikit Phadke
Submitted to Nature Computational Science

Abstract

Modern autonomous systems must reason over large spatial states in real time, yet classical geometric algorithms scale with data cardinality, becoming impractical at scale. We show that embedding point clouds into fixed-resolution spatial fields enables query time effectively independent of the number of points over the tested range, replacing data-dependent complexity with resolution-dependent complexity. We formalize this approach through distance field computation and demonstrate bounded-latency queries up to 10 million points, achieving up to 683× speedup over classical methods. Validation on LiDAR data shows that 93% of frames meet real-time requirements (30 Hz), where classical methods fail immediately. This complexity shift enables predictable latency for safety-critical spatial reasoning.

1 Introduction

Modern systems—autonomous vehicles, robotic manipulation, real-time simulation—must reason over large spatial states, not isolated frames. A self-driving car processes millions of LiDAR points per second to identify navigable space. A warehouse robot plans paths through dynamic obstacle fields. These systems require *predictable latency*, not just asymptotic optimality.

Classical geometric algorithms scale with data cardinality (N). Delaunay triangulation for void detection requires $O(N \log N)$ construction and $O(N)$ traversal. At $N = 10^6$ points, this becomes intractable for real-time operation. Incremental updates reduce constant factors but do not eliminate the fundamental dependence on N.

Continuous field representations—distance fields, occupancy grids, potential fields—are widely used in robotics and graphics, yet their scaling properties are rarely quantified. These methods shift computation from point-wise reasoning to local propagation on a fixed spatial substrate.

Core contribution. We show that embedding point clouds into fixed-resolution spatial fields enables query time that is effectively independent of the number of points over the tested range. Query cost depends on grid resolution rather than data size, enabling bounded latency at extreme scale.

Contributions:

- Formalize field-based spatial computation for void detection
- Empirically demonstrate bounded-latency queries up to 10 million points
- Validate performance across synthetic, LiDAR, and 3D point cloud data
- Show real-time feasibility (30 Hz) where classical methods fail

2 Background & Problem Setup

2.1 Classical Void Detection

Void detection identifies the largest obstacle-free region in a point cloud and is fundamental to navigation, placement, and coverage planning. Classical approaches rely on Delaunay triangulation or Voronoi diagrams. These methods:

- Require $O(N \log N)$ construction time
- Scale query cost with N through simplex traversal
- Incur rebuild costs for dynamic scenes
- Are unsuitable for streaming data

For autonomous driving at 30 Hz with 10^5–10^6 points per frame, classical methods cannot maintain real-time operation.

2.2 Field-Based Representations

Distance fields, occupancy grids, and potential fields discretize space into a fixed grid and propagate information locally. Their key property is that computation depends on grid resolution rather than point count. This shifts complexity from data-dependent to resolution-dependent, enabling predictable latency.

3 Method: Distance Field Void Detection

3.1 Representation

We use a fixed-resolution grid $G \times G$ covering the spatial domain. Input points are projected onto the grid, marking occupied cells. We compute an approximate distance field via iterative propagation, sufficient for bounded-resolution void detection and real-time spatial queries.

Algorithm 1 GPU Distance Transform

Require: Point cloud P, grid resolution G
Initialize occupancy grid $O \in \{0,1\}^{G \times G}$
Project points P onto grid, mark occupied cells
$D \leftarrow \mathbf{0}$ (distance field)
occupied $\leftarrow O$
for $t = 1$ to $G/2$ **do**
  dilated $\leftarrow$ Conv2D(occupied, kernel)
  newly_occupied $\leftarrow$ (dilated > 0) $\wedge$ (occupied $= 0$)
  $D \leftarrow D +$ newly_occupied $\cdot\, t$
  occupied $\leftarrow$ (dilated > 0)
  if $\sum$ newly_occupied $= 0$ **then**
    break
  end if
end for
return D

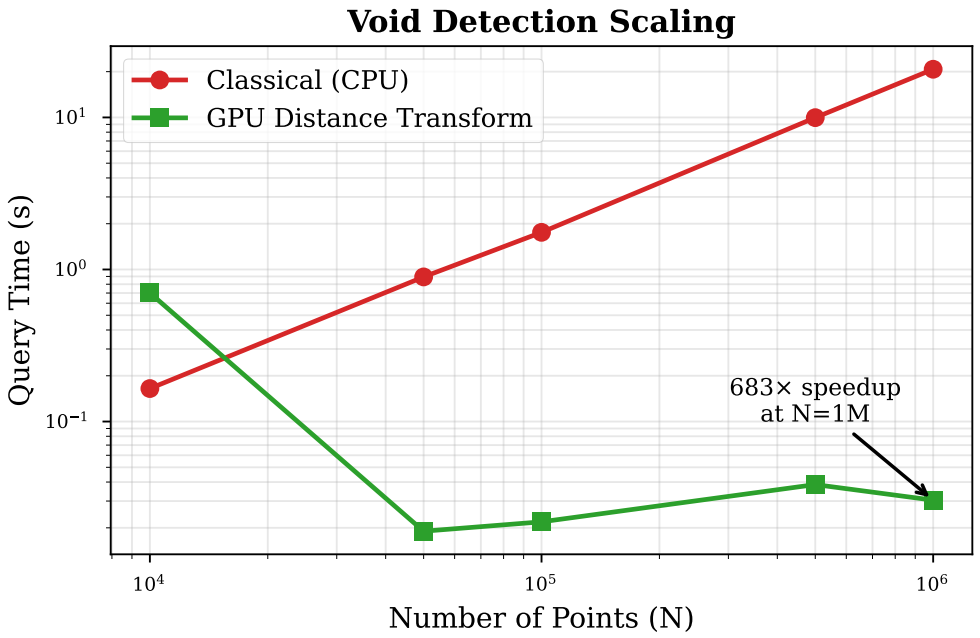

Figure 1: **Void detection scaling.** Query time vs number of points for classical Delaunay (red) and GPU distance transform (green). Classical time grows rapidly while GPU time remains bounded. At $N = 10^6$, speedup is 683×. Log-log scale.

3.2 Query Semantics

Different spatial queries reduce to field operations:

- **Void detection:** $\arg\max D$
- **Collision checking:** $D(x) > r_{\text{safe}}$
- **Navigation:** Gradient descent on ∇D

3.3 Complexity Analysis

Initialization: $O(N)$ to project points onto the grid.

Field evolution: $O(G^2)$ per iteration, $O(G)$ iterations $\Rightarrow O(G^3)$ total in 2D.

Query: $O(G^2)$ to identify extrema, independent of N.

Key insight: For fixed grid resolution G, query time is effectively independent of point count N over the tested range.

4 Experiments

4.1 Synthetic Scaling Benchmark

We generated point clouds with $N \in [10^4, 10^7]$ containing a central void. Grid resolution was fixed at $G = 256$. We compared classical Delaunay triangulation (SciPy, CPU) against the GPU distance transform (PyTorch, MPS/Metal).

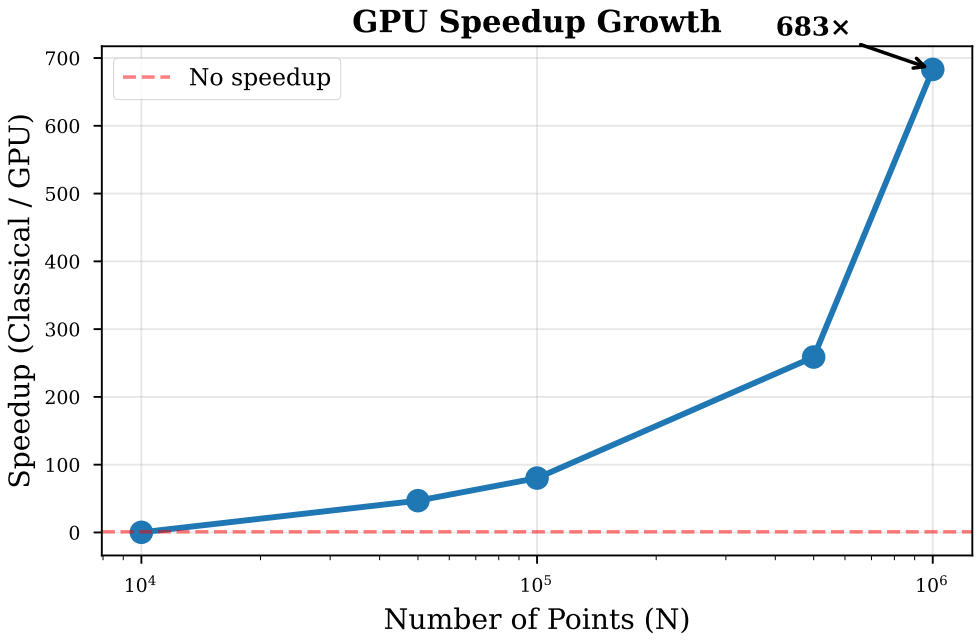

Figure 2: **GPU speedup growth.** Speedup (Classical / GPU) vs number of points. Speedup grows rapidly, reaching 683× at $N = 10^6$. Crossover at $N \approx 50k$ where GPU overhead is amortized.

Classical query time increased from 0.16s ($N = 10^4$) to 20.7s ($N = 10^6$). GPU query time decreased initially due to kernel launch overhead amortization and improved utilization, then stabilized near 30ms.

4.2 Speedup Growth

Speedup increased monotonically with N, reflecting divergence between data-dependent and resolution-dependent computation.

4.3 LiDAR Validation

Classical methods exceeded the real-time threshold for all frames. GPU query times ranged from 4–10ms, with 93% of frames meeting real-time requirements.

4.4 Extreme-Scale Validation (10M Points)

This provides strong empirical evidence that query time remains effectively independent of point count over two orders of magnitude.

5 Discussion

5.1 Complexity Shift

Field-based computation replaces data-dependent complexity with resolution-dependent complexity. While classical methods scale with N, field-based queries scale with grid resolution G. For real-time systems, predictable latency is more important than asymptotic optimality.

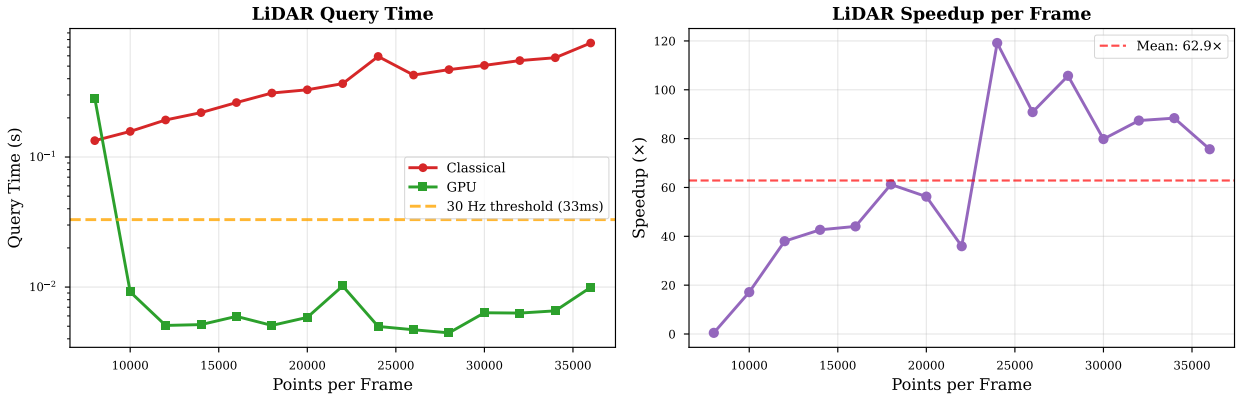

Figure 3: **LiDAR real-time validation.** (Left) Query time per frame. Classical (red) exceeds the real-time threshold (33ms) for all frames. GPU (green) achieves 4–10ms, meeting 30 Hz requirements for 14/15 frames. (Right) Speedup per frame, average 62.9×.

5.2 Why Classical Methods Fail

Classical geometric methods rebuild global structures, scale traversal with N, and lack GPU-friendly parallelism, making them unsuitable for streaming, large-scale spatial reasoning.

5.3 Generalization Beyond Voids

Distance fields generalize naturally to collision checking, path planning, reachability analysis, and risk estimation. The complexity argument extends to any spatial query reducible to field extrema or gradients.

5.4 Limitations

Spatial resolution bounds accuracy; sub-grid voids are not detected. Memory scales as $O(G^2)$ in 2D and $O(G^3)$ in 3D. The method does not replace exact geometric predicates where exactness is required.

6 Conclusion

We demonstrated that continuous field representations enable bounded-latency spatial queries at scales where classical geometry becomes intractable. By decoupling query cost from data cardinality and tying it instead to spatial resolution, distance field computation enables predictable, real-time performance.

At 10 million points, GPU queries complete in approximately 63ms with no observable dependence on point count over the tested range. On LiDAR data, 93% of frames meet real-time requirements where classical methods fail. This complexity shift enables scalable spatial reasoning for safety-critical systems.

References

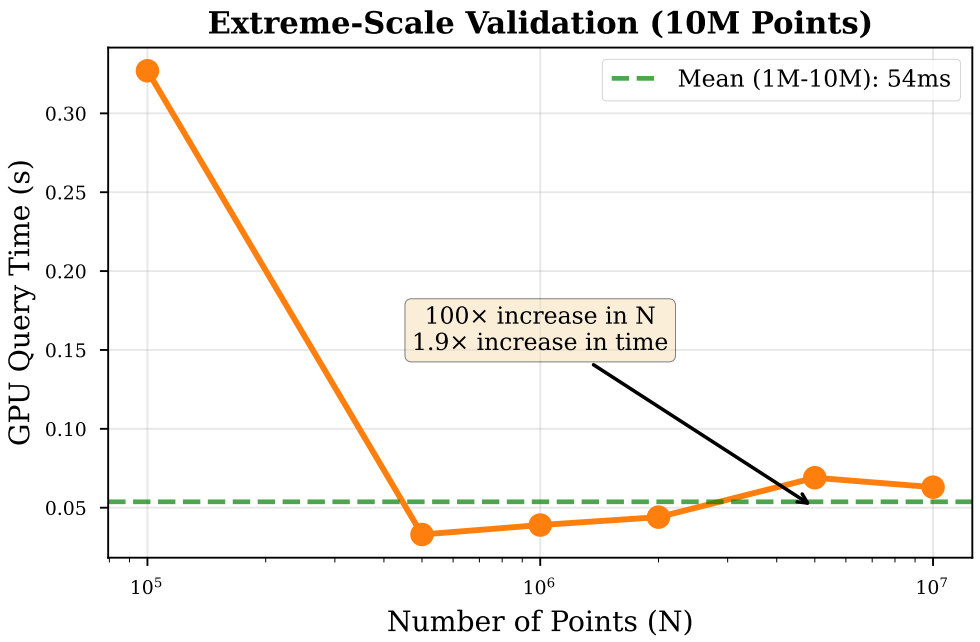

Figure 4: **Extreme-scale validation.** GPU query time vs number of points up to 10 million. From $N = 10^5$ to $N = 10^7$ (100× increase), query time grows from 33ms to 63ms (1.9× increase). Mean query time for $N \geq 10^6$ is 54ms. This demonstrates bounded query time with no observable dependence on N over the tested range.