IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

# Multi-Structure Mapping for Filtering Electric Arc Noise in Power Line Environments

Najlae Boulajoul, Alexis Lussier Desbiens, and François Ferland

***Abstract*—Electric arc noise around energized power lines corrupts drone LiDAR measurements, accumulating in occupancy grids and producing spurious obstacles that degrade navigation reliability. Existing filters designed for environmental clutter such as snow, dust, and rain fail to consistently reject these short-lived arc transients and remain difficult to deploy on resource-limited platforms. We propose a dual-structure filtering framework that dynamically separates transient arc noise from persistent environmental features. Instead of filtering scan-by-scan, the proposed filter leverages spatio-temporal neighborhood consistency across consecutive LiDAR frames to suppress short-duration particles. A transient k-d tree accelerates neighborhood queries and removes arc noise around valid structures, while a persistent octree integrates only enduring features into the global map. Experiments show up to 10 times faster filtering and mapping precision of 92.27% with F1-scores up to 95%. Real-world inspection flights over energized power lines confirm that the approach maintains accurate, up-to-date maps and robust performance in the presence of electric arc noise.**

***Index Terms*—LiDAR filtering, mapping, UAV perception, power line inspection, octree, K-D tree, electric arc noise.**

## I. Introduction

ACCURATE environmental perception is critical for robotic applications, and LiDAR sensors are widely employed to capture high-precision spatial data [1], [2], [3]. In particular, drone-based LiDAR mapping has proven effective for various inspection tasks [4], [5], [6]. However, environmental noise can compromise LiDAR data quality. Prior work has primarily addressed challenges posed by snow, dust, and rain by employing sensor fusion and advanced filtering techniques [7], [8], [9], [10]. In power line inspection tasks, electromagnetic interference near energized power lines introduces transient electric arc discharges [11], [12], which leads to false obstacle detections and navigation errors when such artifacts accumulate in maps. In our observations, arc noise exhibits several distinct characteristics in contrast to environmental noise such as dust, snow, and rain. Arc noise typically manifests as brief and sparse bursts, whereas environmental artifacts tend to be longer-lived, more frequent, and more persistent. Existing filtering methods that primarily rely on intensity thresholds [9] are not well-suited for the unpredictable intensity patterns exhibited by electric arc noise. Filtering based on spatial distribution characteristics such as Radius Outlier Removal (ROR) [8], [9] may be more effective. However, when applied directly to point clouds, these approaches can be computationally demanding for resource-constrained applications, especially given the large point clouds typical of power line inspection environments. We introduce a dual-structure filtering framework that detects transient arc noise by leveraging neighborhood consistency across successive point clouds instead of per-scan filtering. Locally, a k-d tree-based transient buffer captures short-lived, sparse arc particles in a small number of points, where k-d trees are most efficient, resulting in faster nearest-neighbor queries and quicker denoising. At the global level, we use OctoMap [13], an octree-based probabilistic occupancy map, to integrate only validated, repeatedly observed features, and its hierarchy scales well to large datasets for global mapping. We validate the approach on LineDrone-NADILE [14] data collected during inspections of 315 kV power lines, demonstrating reduced computational load and improved noise removal for accurate mapping in large-scale, noisy environments.

Our main contributions are:

- Quantification of electric arc noise sparsity, lifetime, and spatial footprint, motivating spatio-temporal and probabilistic filtering beyond single-scan heuristics.
- Dual-structure mapping with bounded-growth transient k-d tree and persistent global octree for selective integration of enduring structure.
- Field validation on custom datasets across multiple environments, showing accuracy and efficiency.

The remainder of the letter is organized as follows: Section II reviews related work; Sections III and IV detail the methodology; Section V presents experimental results; followed by conclusions in Section VI.

## II. Related Work

This section reviews robotic 3D mapping and LiDAR denoising methods, with an emphasis on techniques requiring high-accuracy noise filtering for reliable mapping.

### A. Mapping Approaches

Accurate 3D mapping is critical for robotics [2], [3]. Early techniques relied on LiDAR-generated point clouds and meshes [10], [15], [16], which are effective for visualization but lack volumetric occupancy for navigation. Probabilistic grid-based methods partition environments into cells with occupancy probabilities [17], yet fixed resolutions force a trade-off between detail and computational cost. Hierarchical structures like

Received 9 June 2025; accepted 17 December 2025. Date of publication 12 January 2026; date of current version 19 January 2026. This article was recommended for publication by Associate Editor B. Englot and Editor J. Civera upon evaluation of the reviewers' comments. This work was supported by Alliance and CRIAQ between University of Sherbrooke, Hydro-Quebec and DRONEVOLT under Grant 2601-2600-703. *(Corresponding author: Najlae Boulajoul.)*

The authors are with the Interdisciplinary Institute for Technological Innovation (3IT), Sherbrooke QC J1K 0A5, Canada (e-mail: najlae.boulajoul@usherbrooke.ca; alexis.lussier.desbiens@usherbrooke.ca; francois.ferland@usherbrooke.ca).

Digital Object Identifier 10.1109/LRA.2026.3653382

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

octrees offer an elegant solution; for example, OctoMap [13], [18] adaptively subdivides space, using coarse voxels in sparse regions and fine voxels where needed. Despite their efficiency, these methods are primarily designed for static or quasi-static environments and do not inherently address very dynamic and transient phenomena. Extensions like semantic mapping with annotated representations [19] exist but typically add computational complexity.

### B. Noise Filtering Approaches

Enhancing LiDAR reliability through noise filtering is essential for accurate mapping. Sensor fusion methods such as LiDAR-camera [20] and LiDAR-radar fusion [21] improve noise discrimination by leveraging complementary modalities. However, in high-voltage environments, the presence of thin, metallic, and often poorly reflective conductors, such as power lines, can challenge accurate depth estimation and feature extraction [22], [23]. These structures are difficult to detect consistently across different sensors, leading to sensor misalignment and calibration errors. Furthermore, electric arc discharges can introduce transient noise patterns that are not uniformly captured across modalities, making cross-sensor consistency harder to maintain. As such, the effectiveness of these fusion techniques in electric noise conditions remains to be thoroughly evaluated. Intensity-based approaches, like Low-Intensity Outlier Removal (LIOR), target low-reflectivity returns associated with dust and fog [24], but they may struggle with unpredictable intensity patterns observed with electric arc noise. Spatial filtering techniques, including Statistical Outlier Removal (SOR) and Radius Outlier Removal (ROR), analyze point distribution and density (e.g., via K-nearest neighbors) to isolate noise [8]. However, processing every point in large datasets can be computationally intensive. More recently, deep learning approaches, both supervised (e.g., voxel-wise classification and point-wise CNNs) and unsupervised clustering, have demonstrated high precision in differentiating noise [16], [25]. While these methods have been successfully tested on autonomous ground vehicles, their extensive computational requirements and reliance on large labeled datasets render them less effective on resource-limited autonomous aerial vehicles. These trade-offs between mapping accuracy, particularly arc-noise recognition, and computational efficiency, measured as per-frame point-cloud processing, motivate ongoing research into methods for dynamic, noisy environments.

## III. Preliminaries

### A. The LineDrone-NADILE Mission and Platform

The LineDrone-NADILE platform was developed for autonomous power line inspections as an enhanced version of the originally developed LineDrone [14]. The drone serves as a development platform for the algorithms developed at the authors' institution and includes four main software modules: perception, mapping, navigation, and planning, which run on a Jetson Xavier NX, as well as tracking control, which is handled by a Pixhawk CUAV X7 Pro. The drone is equipped with a dual-antenna GPS and a vertically mounted Velodyne VLP-16 LiDAR (set to a 10 Hz refresh rate during experiments) to enable the acquisition of precise, georeferenced point cloud data essential for robust obstacle detection and the creation of detailed three-dimensional maps. The LineDrone-NADILE

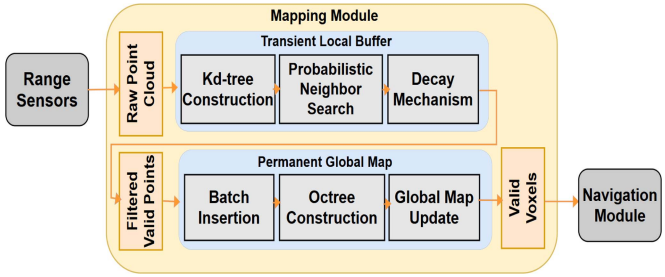



Fig. 1. The architecture of the overall proposed filtering and mapping approach.

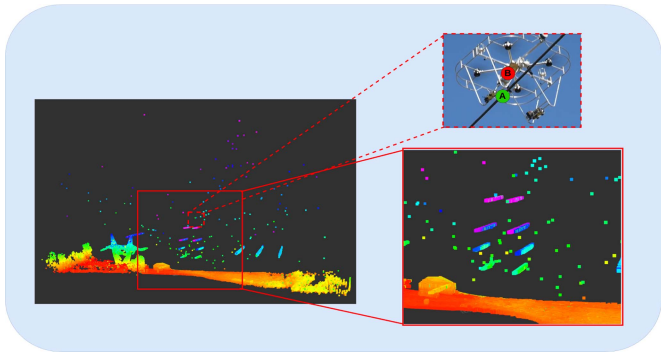



Fig. 2. Accumulated electric arc voxels during a LineDrone-NADILE test caused by the potential difference between the 315 kV power line (A) and the drone (B).

mission proceeds in four phases: vertical take-off, cable insertion, descent for inspection, and return-to-home. During the mission, the navigation module selects the target conductor on a 2D planning grid (conceived for mission control) [26] and then computes a collision-free path using obstacle detection. However, in testing near 315 kV conductors, sparse bursts of arc noise accumulated in the global OctoMap, were interpreted as static obstacles, and caused the planner to fail to converge. Removing arc noise is therefore essential for accurate mapping and for stable navigation toward conductors.

### B. Electric Arc Analysis

During tests near energized power lines, electric arcs were visually observed as the drone approached an energized cable during landing. In the corresponding point cloud, the high electric fields generated transient electric arcs that scattered LiDAR pulses, introducing noise (see Fig. 2).

We conducted two fully automated flights with identical home point, waypoints, and speed at a dedicated test site, using the same line–pylon geometry and matched wind conditions to minimize variability. The datasets from energized (with electric arc events) and non-energized power lines (without electric arc events) were aligned in a common frame using fused GNSS/IMU and LiDAR poses, then refined with the Iterative Closest Point (ICP) algorithm from the Point Cloud Library (PCL), to achieve centimeter-level co-registration. For analysis, raw points were directly accumulated into OctoMap voxels. Voxels corresponding to electric arcs were identified by comparing LiDAR acquired near energized conductors against ground truth from the otherwise identical, non-energized configuration. The power line under test was oriented along the X-axis, providing a

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

TABLE I
STATISTICAL DISTRIBUTION OF ARC-RELATED POINTS ALONG EACH AXIS

| Axis | Mean (m) | Std. Dev. (m) | Range (m) |
|---|---|---|---|
| X | 4.37 | 13.77 | 79.42 |
| Y | 2.69 | 12.29 | 62.27 |
| Z | 23.18 | 8.81 | 47.84 |

TABLE II
PARAMETERS AND RESULTS FOR DIFFERENT TESTS EVALUATING THE OPTIMAL TRANSIENT BUFFER DURATION $\tau_{\text{BUFFER}}$ BASED ON DRONE SPEED $V$, NOISE LIFETIME $\tau_n$, TRAVELED DISTANCE $d_{\text{BUFFER}}$, AND OVERLAP DISTANCE $D_{\text{OVERLAP}}$ FOR $\alpha_{\text{MIN}} = 0.7$

| $v$ (m/s) | $\tau_n$ (s) | $d_{\text{buffer}}$ (m) | $d_{\text{overlap}}$ (m) | $\tau_{\text{buffer}}$ (s) |
|---|---|---|---|---|
| 0.5 | 0.2 | 0.10 | 0.07 | 0.2 |
| 0.5 | 0.3 | 0.15 | 0.11 | 0.3 |
| 1.0 | 0.2 | 0.20 | 0.14 | 0.2 |
| 1.0 | 0.3 | 0.30 | 0.21 | 0.3 |

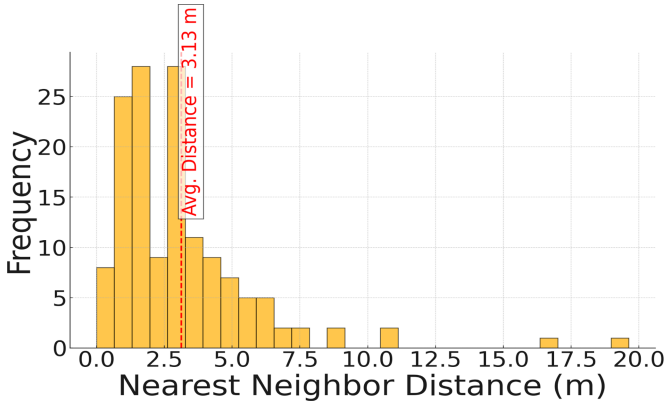



Fig. 3. Nearest Neighbor Distance (NND) distribution of arc-related points.

consistent spatial reference for arc noise distribution, exhibiting vertical bias and horizontal dispersion around the cables (see Table I). For example, tests indicate that arc noise comprises, on average, 0.021% of the total voxels and can propagate up to 39.71 meters from the drone-cable contact point within a 371-meter map range. Furthermore, the average nearest neighbor distance is 3.13 meters for arc points versus 0.71 meters for valid environmental points, indicating a sparser distribution relative to nearby real-world features (see Fig. 3). Although each arc event was brief, with an average point duration of 0.26 seconds in the point cloud, the cumulative effect of this noise in occupancy grids can mislead navigation algorithms by falsely classifying these points as obstacles (see Fig. 2).

## IV. PROPOSED APPROACH

Per-scan filtering fails on arc noise because returns are short-lived, neighborhoods are incomplete, and point clouds are large. We therefore adopt a dual-structure pipeline over sequential scans to maintain an accurate global map. A Velodyne VLP-16 continuously acquires data (Fig. 1). Pre-mission scans initialize a k-d tree transient buffer; subsequent scans use probabilistic radius searches to form correspondences, reject transient outliers, and fuse persistent points into an octree-based map [13], [18] used by the navigation module. This design suppresses electric arc artifacts and bounds map growth in large-scale power line environments. The next sections detail the data structures, spatial search, and noise filtering of the proposed method.

### A. Transient Buffer Construction

*1) Transient Buffer Structure:* The transient structure is a k-d tree based map that serves as a buffer [27]. To organize point cloud data hierarchically, the k-d tree recursively partitions the space along alternating coordinate axes, dividing the space into axis-aligned bounding boxes. Next, a neighborhood query is performed on the k-d tree to identify nearby points and estimate the local spatial density around each query point. For a query point $p_q = (q_x, q_y, q_z)$ and radius $r$, the algorithm recursively visits nodes where the Euclidean distance to the splitting plane does not exceed $r$. Let the current node's splitting axis be $a_l$, and let $p_n$ be the point stored at the node. If $d(p_q, p_n) \leq r$ condition is satisfied, $p_n$ is added to the batch of points to be inserted into the global octree. To determine which subtree(s) to visit, the algorithm compares the signed distance between $q_{a_l}$ and the splitting plane defined by the coordinate $p_{n,a_l}$:

$$d_{\text{plane}} = |q_{a_l} - p_{n,a_l}| \tag{1}$$

If $d_{\text{plane}} \leq r$, both the left and right subtrees are explored. Otherwise, only the subtree on the closer side of the plane is visited. This implemented pruning strategy efficiently reduces the number of nodes visited, yielding a search complexity of $O(\log N)$ for balanced trees, where $N$ is the total number of points. The k-d tree neighborhood query thus provides the geometric context required for the probabilistic neighborhood filtering and allows fast neighbor retrieval through hierarchical space partitioning.

*2) Probabilistic Neighbor Search:* Traditional radius-based outlier removal methods apply a hard threshold to determine whether a point is valid [8]. However, such a hard-thresholding approach can be sensitive to sudden dense bursts of arc noise in cluttered environments and, conversely, to sparse regions where valid points may be incorrectly removed. To improve robustness, a probabilistic formulation is adopted where each point $p_i$ is assigned a probability of retention $p_{\text{keep}}$ that depends on its local neighborhood statistics. For every query point, a k-d tree is used to find its $K$ nearest neighbors, and the squared distance to the $K$-th neighbor, $d_K^2$, is treated as an indicator of local sparsity or density. The probability of keeping the point is defined as:

$$p_{\text{keep}} = \max\left(p_{\text{floor}}, \frac{1}{1 + \exp\left(-\frac{r^2 - d_K^2}{\beta^2}\right)}\right), \tag{2}$$

where $\beta$ controls the softness of the transition between dense and sparse regions, and $p_{\text{floor}}$ is a small lower bound that preserves a minimal probability of retaining isolated points in low-density regions. When $d_K < r$, the local area is dense and $p_{\text{keep}} \approx 1$; when $d_K = r$, $p_{\text{keep}} \approx 0.5$; and when $d_K > r$, the region is sparse and $p_{\text{keep}}$ approaches $p_{\text{floor}}$. Finally, a pseudo-random function $h(i) \in [0, 1)$, deterministically derived from the point index, is used to determine whether a point is retained:

$$h(i) \leq p_{\text{keep}}. \tag{3}$$

Here, $h(i)$ acts as a reproducible generator that emulates a probabilistic sampling process. If the generated value $h(i)$ is less than or equal to $p_{\text{keep}}$, the point is retained in the filtered cloud; otherwise, it is discarded. This mechanism introduces controlled

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

randomness, rendering the decision probabilistic rather than binary.

*3) Transient Buffer Duration:* The transient buffer temporarily stores LiDAR data from $M$ consecutive scans. For a LiDAR with scan frequency $f_{\text{scan}}$ (Hz) and period $\tau_{\text{scan}} = 1/f_{\text{scan}}$, the buffer spans $\tau_{\text{buffer}} = M\,\tau_{\text{scan}} = M/f_{\text{scan}}$. The choice of $M$ (i.e., $\tau_{\text{buffer}}$) governs the trade-off between noise suppression and spatial coverage: shorter buffers reduce the accumulation of transient artifacts but limit inter-scan spatial overlap, whereas longer buffers capture more structure but may retain more noise. Let $\tau_n$ denote the mean lifetime of transient noise, $v$ the platform velocity, and $\alpha_{\min}$ the desired minimum overlap ratio between consecutive scans. We use $\Delta t \equiv \tau_{\text{scan}}$ for the inter-scan interval and set $\alpha_{\min} = 0.7$ to ensure sufficient common structure between frames. To balance spatial continuity and noise suppression, the buffer duration $\tau_{\text{buffer}}$ must satisfy two constraints. First, it should not retain transient noise beyond its mean lifetime:

$$\tau_{\text{buffer}} \leq \tau_n. \tag{4}$$

Second, to ensure sufficient spatial overlap, the buffer's spatial extent $d_{\text{buffer}} = v \cdot \tau_{\text{buffer}}$ must be at least a fraction $\alpha_{\min}$ of the inter-scan distance $d_{\text{scan}} = v \cdot \Delta t$, where the inter-scan interval is $\Delta t = 1/f_{\text{scan}}$:

$$\tau_{\text{buffer}} \geq \alpha_{\min}\,\Delta t. \tag{5}$$

Combining these bounds gives:

$$\alpha_{\min} \cdot \Delta t \;\leq\; \tau_{\text{buffer}} \;\leq\; \tau_n. \tag{6}$$

In our setup, $f_{\text{scan}} = 10\,\text{Hz}$ ($\Delta t = 0.1\,\text{s}$), $\alpha_{\min} = 0.7$, and $\tau_n = 0.26\,\text{s}$, yielding $0.07\,\text{s} \leq \tau_{\text{buffer}} \leq 0.26\,\text{s}$. We choose $\tau_{\text{buffer}} = 0.2\,\text{s}$, the largest integer multiple of $\Delta t$ that satisfies the bounds, so the buffer holds an integer number of scans. With this choice, the buffer stores two LiDAR scans, and the global map is updated every two scans.

### B. Global Map Construction

The global map is represented as an octree [18], which hierarchically partitions 3D space into cubic voxels via recursive subdivision into eight octants. To accelerate updates, batch insertion is employed in lieu of pointwise writes. Over the buffer window $\tau_{\text{buffer}}$, inlier points are accumulated as:

$$P = \{\, p_i = (x_i, y_i, z_i) \mid i = 1, \ldots, N \,\}. \tag{7}$$

These points are voxelized at resolution $r_{\text{octree}}$ and sorted by voxel index to promote spatial locality. For each voxel $(i, j, k)$, let $B_{ijk} \subset P$ denote its assigned points; a single update is then performed using the batch centroid:

$$p_{ijk} = \frac{1}{|B_{ijk}|} \sum_{p \in B_{ijk}} p. \tag{8}$$

This batching reduces octree traversals and dynamic node allocations, and yields a faster, more stable mapping pipeline as the octree expands to newly observed regions.

## V. Experimental Evaluation and Results

In this section, we present experimental results validating the proposed approach, which was implemented in the Robot Operating System (ROS). To assess the system's scalability and generalization, we conducted a total of 90 tests across three distinct power line environments, each featuring a different pylon and cable arrangements (see Table III). In each of the three environments, 10 tests were performed to ensure a comprehensive evaluation for each method. The computation times and mapping accuracy metrics, under the same hyperparameters, for each of the three environments are reported as averages over all 10 tests. The experiments were conducted on a Linux machine equipped with an 11th Gen Intel Core i7-11800H CPU (2.30 GHz, 8 cores/16 threads) and 16 GB of RAM.

To the best of our knowledge, no prior research has specifically addressed filtering electric arc noise. In this study, we compare our proposed method to two established filtering techniques: one intensity-based (Low-Intensity Outlier Removal, LIOR) and one neighbor-based (Radius Outlier Removal, ROR), both originally designed for filtering out dust and snow particles [8]. To enable fair comparison, we used the same hyperparameters across all three environments and aligned them with electric arc characteristics (mean nearest-neighbor distance 3.13 m; mean lifetime 0.26 s). The global map used an octree resolution of 1 m to balance fidelity and runtime, and the local buffer was 0.2 s (two scans at 10 Hz), consistent with the noise timescale. For neighborhood filtering we set the radius to 1.5 m with $K = 2$ and an intensity threshold of 10, and we ran additional sweeps at 0.5 m and 2.5 m to assess sensitivity. The 10 Hz scan rate (0.1 s inter-scan interval) captures scene dynamics without overloading computation. Moreover, the proposed mapping framework feeds into the navigation module, which operates at 10 Hz, allowing a theoretical upper bound of 100 ms per iteration, assuming the navigation module operates on a separate system. In our ROS node, we instrumented key code sections to log performance metrics, measuring iteration times and embedding timers around neighbor search routines to assess computational efficiency.

### A. Resource Consumption

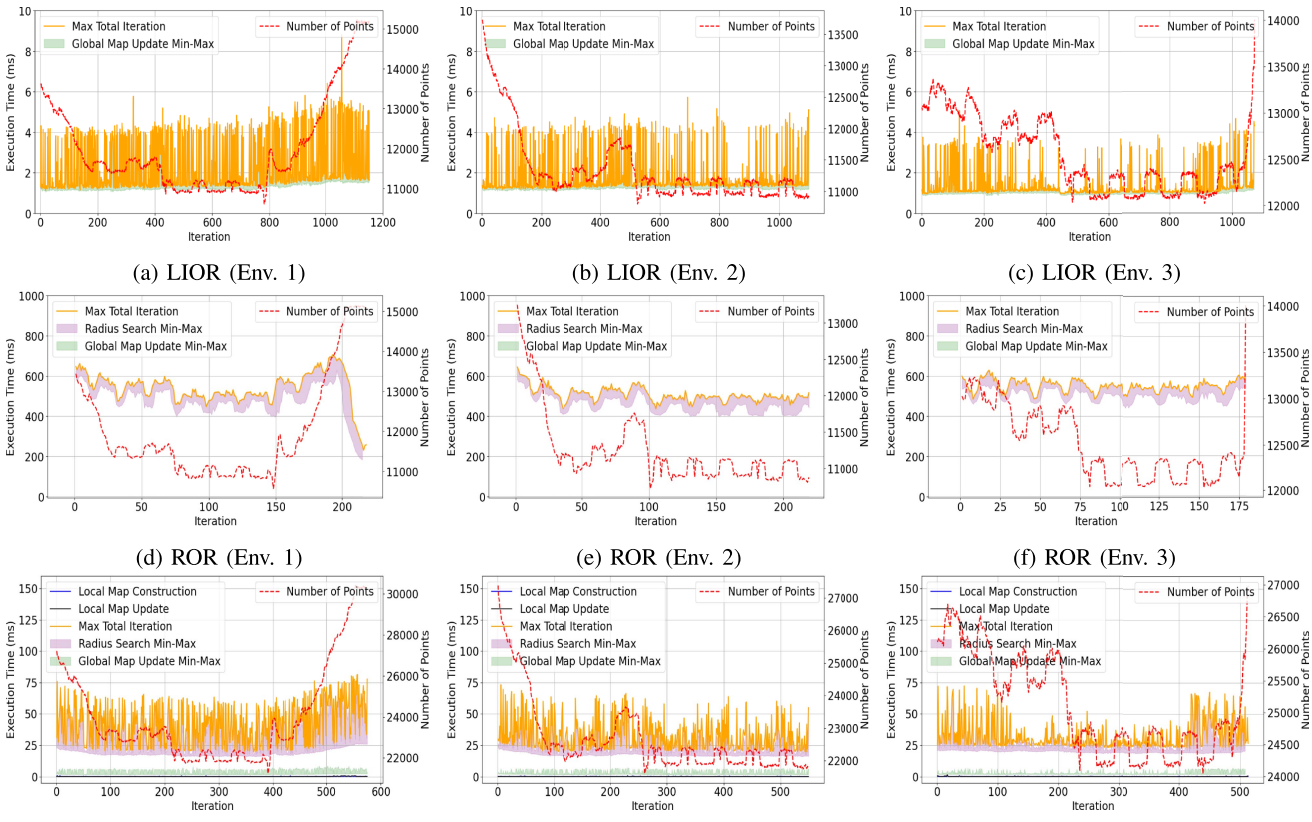
Table IV summarizes the computational performance of the three denoising filters, LIOR, ROR, and our method, across three different power line environments and search radii. LIOR achieves the lowest average total iteration and map update times, as it does not use radius-based searches. Although computationally light, its ability to filter electric arc noise due to unpredictable intensity patterns is limited, as shown in Fig. 5(d)–(f). ROR is computationally intensive: at $r = 1.5\,\text{m}$, it takes 479.94 to 524.54 ms per scan on average across environments, primarily due to expensive brute-force neighbor searches. As the search radius increases with the increasing points number, so do processing and map update times, with total iteration time peaking up to 706.06 ms, largely exceeding the 100 ms threshold. Ours consistently meets the 100 ms constraint. At $r = 1.5\,\text{m}$, it requires only 22.40 to 25.49 ms on average per scan. Its k-d tree enables stable and fast neighbor searches (21.75 to 22.40 ms), and global map updates remain stable (2.31 to 2.67 ms). Even under peak load, our method's total time (72.27 to 81.42 ms) stays within practical limits. Since k-d trees perform best below a size threshold, restricting the k-d tree to the transient buffer, where the point set remains small, constrains growth, yielding stable performance and preserving computational efficiency even under varying hyperparameters. Additionally, despite managing two maps, the batch insertion mechanism and hierarchical structure of OctoMap used in our filter ensures that updates add only few milliseconds to the total iteration time. Notably, our filter and LIOR occasionally

TABLE III
ENVIRONMENTS TESTS PARAMETERS

| Test Environments | Env.1 | Env.2 | Env.3 |
|---|---|---|---|
| Scene dimensions ($m^3$) | 96.98 × 26.76 × 38.03 | 76.51 × 33.87 × 32.89 | 112.43 × 27.85 × 9.51 |
| Average of rays (points) per scan | 11,982 | 12,553 | 11,325 |
| Number of scans | 1,160 | 1,060 | 1,180 |

TABLE IV
AVERAGE PERFORMANCE TIMES OF DENOISING FILTERS AT DIFFERENT RADII ACROSS THE THREE ENVIRONMENTS. THE VALUES ARE COMPUTED FOR $R_{\text{OCTREE}} = 1\,\text{M}$ AND $K = 2$. THE RESULTS REPRESENT THE AVERAGE OVER ALL CYCLES IN EACH ENVIRONMENT

| | Metric | LIOR | | | ROR | | | Ours | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $r = 0.5\,\text{m}$ | $r = 1.5\,\text{m}$ | $r = 2.5\,\text{m}$ | $r = 0.5\,\text{m}$ | $r = 1.5\,\text{m}$ | $r = 2.5\,\text{m}$ | $r = 0.5\,\text{m}$ | $r = 1.5\,\text{m}$ | $r = 2.5\,\text{m}$ |
| **Env. 1** | Transient Buffer Construction (ms) | N/A | N/A | N/A | N/A | N/A | N/A | 0.24 | 0.24 | 0.24 |
| | Transient Buffer Update (ms) | N/A | N/A | N/A | N/A | N/A | N/A | 0.18 | 0.18 | 0.17 |
| | Radius Neighbor Search (ms) | N/A | N/A | N/A | 559.78 | 506.26 | 460.99 | 22.79 | 22.40 | 22.73 |
| | Global Map Update (ms) | 1.46 | 1.47 | 1.51 | 1.13 | 1.29 | 1.36 | 2.77 | 2.67 | 2.84 |
| | Total Map Update (ms) | 1.46 | 1.47 | 1.51 | 1.13 | 1.29 | 1.36 | 2.95 | 2.85 | 3.01 |
| | Total Iteration (ms) | 1.46 | 1.47 | 1.51 | 560.09 | 507.55 | 462.35 | 25.98 | 25.49 | 25.98 |
| | Max Total Iteration (ms) | 6.43 | 8.80 | 5.27 | 807.20 | 706.06 | 622.61 | 80.00 | 81.42 | 74.83 |
| **Env. 2** | Transient Buffer Construction (ms) | N/A | N/A | N/A | N/A | N/A | N/A | 0.22 | 0.22 | 0.22 |
| | Transient Buffer Update (ms) | N/A | N/A | N/A | N/A | N/A | N/A | 0.14 | 0.14 | 0.14 |
| | Radius Neighbor Search (ms) | N/A | N/A | N/A | 529.97 | 478.73 | 470.91 | 19.50 | 19.60 | 18.94 |
| | Global Map Update (ms) | 1.36 | 1.35 | 1.41 | 1.09 | 1.21 | 1.27 | 2.52 | 2.44 | 2.38 |
| | Total Map Update (ms) | 1.36 | 1.35 | 1.41 | 1.09 | 1.21 | 1.27 | 2.67 | 2.58 | 2.52 |
| | Total Iteration (ms) | 1.36 | 1.35 | 1.41 | 531.07 | 479.94 | 472.54 | 22.38 | 22.40 | 21.68 |
| | Max Total Iteration (ms) | 4.46 | 5.72 | 4.85 | 673.00 | 644.73 | 561.40 | 60.50 | 73.30 | 65.16 |
| **Env. 3** | Transient Buffer Construction (ms) | N/A | N/A | N/A | N/A | N/A | N/A | 0.24 | 0.24 | 0.23 |
| | Transient Buffer Update (ms) | N/A | N/A | N/A | N/A | N/A | N/A | 0.17 | 0.17 | 0.16 |
| | Radius Neighbor Search (ms) | N/A | N/A | N/A | 601.51 | 523.43 | 491.40 | 22.00 | 21.75 | 22.26 |
| | Global Map Update (ms) | 1.16 | 1.06 | 1.13 | 1.06 | 1.11 | 1.74 | 2.34 | 2.31 | 2.48 |
| | Total Map Update (ms) | 1.16 | 1.06 | 1.13 | 1.06 | 1.11 | 1.74 | 2.51 | 2.47 | 2.71 |
| | Total Iteration (ms) | 1.16 | 1.06 | 1.13 | 602.57 | 524.54 | 492.62 | 24.75 | 24.60 | 25.13 |
| | Max Total Iteration (ms) | 2.51 | 5.06 | 5.16 | 699.59 | 628.75 | 586.94 | 70.96 | 72.27 | 69.59 |




(a) LIOR (Env. 1) (b) LIOR (Env. 2) (c) LIOR (Env. 3)

(d) ROR (Env. 1) (e) ROR (Env. 2) (f) ROR (Env. 3)

(g) Ours (Env. 1) (h) Ours (Env. 2) (i) Ours (Env. 3)

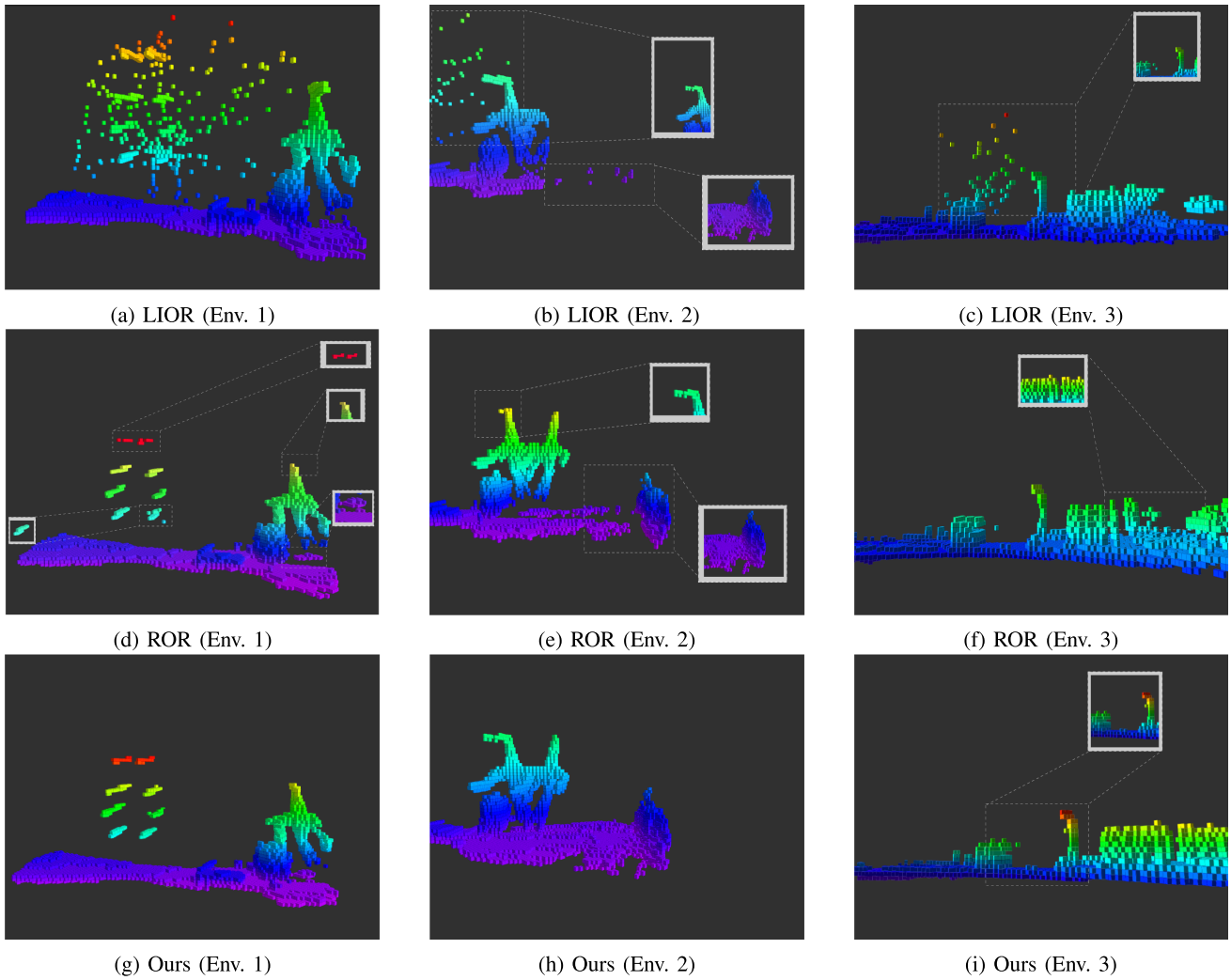
Fig. 4. Performance times over time of denoising filters at different radii across the three environments. The values are computed for $r_{\text{octree}} = 1\,\text{m}$, $K = 2$ and $r = 1.5\,\text{m}$. The results represent the average over all cycles in each environment.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

TABLE V

CONFUSION MATRICES OF DENOISING FILTERS AT DIFFERENT RADII ACROSS THE THREE ENVIRONMENTS. THE VALUES ARE COMPUTED FOR $R_{\text{OCTREE}} = 1$ M AND $K = 2$. THE RESULTS REPRESENT THE AVERAGE OVER ALL CYCLES IN EACH ENVIRONMENT. THE BEST VALUES FOR $R = 1.5$ M FOR EACH METRIC ARE HIGHLIGHTED IN BOLD. TRUE POSITIVES (TP) ARE ACTUAL ELECTRIC ARC VOXELS CORRECTLY IDENTIFIED AS NOISE VOXELS

| | Metric | LIOR | | | ROR | | | Ours | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $r = 0.5$ m | $r = 1.5$ m | $r = 2.5$ m | $r = 0.5$ m | $r = 1.5$ m | $r = 2.5$ m | $r = 0.5$ m | $r = 1.5$ m | $r = 2.5$ m |
| **Env. 1** | TP | 4 | 1 | 0 | 146 | 142 | 142 | 141 | **143** | 145 |
| | FN | 142 | 145 | 146 | 0 | 4 | 4 | 5 | **3** | 1 |
| | FP | 59 | 21 | 41 | 709 | 596 | 254 | 25 | **12** | 3 |
| | TN | 1,048,371 | 1,048,409 | 1,048,389 | 1,047,721 | 1,047,834 | 1,048,176 | 1,048,405 | **1,048,418** | 1,048,427 |
| **Env. 2** | TP | 1 | 0 | 3 | 32 | **32** | 32 | 32 | **32** | 32 |
| | FN | 31 | 32 | 29 | 0 | **0** | 0 | 0 | **0** | 0 |
| | FP | 186,750 | 239,223 | 221,056 | 186,520 | 121,561 | 172,560 | 26 | **17** | 19 |
| | TN | 703,820 | 651,347 | 669,514 | 704,050 | 769,009 | 718,010 | 890,544 | **890,553** | 890,551 |
| **Env. 3** | TP | 0 | 1 | 1 | 38 | **38** | 38 | 37 | 37 | 38 |
| | FN | 39 | 38 | 38 | 1 | **1** | 1 | 2 | 2 | 1 |
| | FP | 786 | 1,651 | 875 | 671 | 586 | 637 | 7 | **13** | 11 |
| | TN | 920,424 | 919,559 | 920,335 | 920,539 | 920,624 | 920,573 | 921,203 | **921,197** | 921,199 |



(a) LIOR (Env. 1) (b) LIOR (Env. 2) (c) LIOR (Env. 3)

(d) ROR (Env. 1) (e) ROR (Env. 2) (f) ROR (Env. 3)

(g) Ours (Env. 1) (h) Ours (Env. 2) (i) Ours (Env. 3)

Fig. 5. Experimental results after applying denoising filters to the electric arc noise voxels during a LineDrone-NADILE representative test in each power line environment ($r_{octree} = 1$ m, $K = 2$, $r = 1.5$ m). The results show that the LIOR filter fails to remove electric arc voxels. The results were compared to ground truth data, displayed alongside in gray boxes.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

TABLE VI
PERFORMANCE METRICS OF DENOISING FILTERS AT DIFFERENT RADII ACROSS THE THREE ENVIRONMENTS. VALUES COMPUTED FOR $R_{\text{OCTREE}} = 1\,\text{M}$ AND $K = 2$. THE VALUES ARE COMPUTED FOR $R_{\text{OCTREE}} = 1\,\text{M}$ AND $K = 2$. THE RESULTS REPRESENT THE AVERAGE OVER ALL CYCLES IN EACH ENVIRONMENT

| | Metric | LIOR | | | ROR | | | Ours | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $r = 0.5\,\text{m}$ | $r = 1.5\,\text{m}$ | $r = 2.5\,\text{m}$ | $r = 0.5\,\text{m}$ | $r = 1.5\,\text{m}$ | $r = 2.5\,\text{m}$ | $r = 0.5\,\text{m}$ | $r = 1.5\,\text{m}$ | $r = 2.5\,\text{m}$ |
| **Env. 1** | Accuracy (%) | 99.98 | 99.98 | 99.98 | 99.93 | 99.94 | 99.96 | 99.98 | **99.98** | 99.98 |
| | Precision (%) | 6.35 | 4.55 | 0.00 | 17.06 | 19.23 | 35.85 | 84.43 | **92.27** | 98.00 |
| | Recall (%) | 2.74 | 0.68 | 0.00 | 100.00 | 97.26 | 97.26 | 96.58 | **97.95** | 99.32 |
| | F1-score (%) | 3.94 | 1.31 | 0.00 | 29.17 | 32.13 | 52.46 | 89.99 | **95.00** | 98.65 |
| **Env. 2** | Accuracy (%) | 79.03 | 73.09 | 75.06 | 79.05 | 86.32 | 80.49 | 99.99 | **99.99** | 99.99 |
| | Precision (%) | 0.00 | 0.00 | 0.00 | 0.02 | 0.03 | 0.02 | 55.17 | **65.31** | 62.75 |
| | Recall (%) | 3.13 | 0.00 | 9.38 | 100.00 | **100.00** | 100.00 | 100.00 | **100.00** | 100.00 |
| | F1-score (%) | 0.06 | 0.00 | 0.00 | 0.05 | 0.05 | 0.05 | 71.09 | **79.01** | 77.14 |
| **Env. 3** | Accuracy (%) | 99.91 | 99.82 | 99.90 | 99.92 | 99.93 | 99.92 | 99.99 | **99.99** | 99.99 |
| | Precision (%) | 0.00 | 0.06 | 0.11 | 5.36 | 6.09 | 5.63 | 84.09 | **74.00** | 77.08 |
| | Recall (%) | 0.00 | 2.56 | 2.56 | 97.44 | **97.44** | 97.44 | 94.87 | 94.87 | 97.44 |
| | F1-score (%) | 0.00 | 0.11 | 0.21 | 10.16 | 11.42 | 10.66 | 89.25 | **83.15** | 86.00 |

show execution-time spikes attributable to ROS scheduling and system-level overhead. Short iteration times make even minor runtime jitter appear as large relative variance. In contrast, slower methods such as ROR are less sensitive because the same fluctuations are amortized over longer cycles. This highlights a trade-off: highly optimized algorithms can appear more sensitive to system noise not due to inefficiency, but precisely because they are fast. We also benchmarked runtime on the public LeGO-LOAM dataset to isolate per-iteration overhead. We evaluated our method and ROR over five consecutive cycles on a 100-m indoor map. The proposed filter processed up to 52,079 points per scan, with a mean radius-neighbor search time of 43.46 ms and a maximum iteration time of 101.49 ms. It also maintained lightweight transient buffer operations (3.61 ms to build; 1.14 ms to update), while the global-map update took 6.99 ms, compared with 2.65 ms for ROR, the latter being faster here because it discards many points, yielding a smaller OctoMap.

### B. Mapping Accuracy

Table VI reports performance across environments and search radii. Because the datasets are highly imbalanced (Env. 1: 146 arc vs. 1,048,430 non-arc voxels; Env. 2: 32 vs. 890,570; Env. 3: 39 vs. 921,210), overall accuracy is not informative for denoising quality; we therefore emphasize precision, recall, and F1. At a $1.5\,\text{m}$ search radius, our method yields the highest precision/F1 in all environments (Env. 1: $97.27\%$ / $95.00\%$; Env. 2: $65.31\%$ / $79.01\%$; Env. 3: $74.00\%$ / $83.15\%$). ROR attains perfect recall in Env. 2 ($100\%$) but low precision, resulting in reduced F1 due to many false positives. LIOR shows very low recall (Env. 1: $1.31\%$; Env. 2: $0\%$), indicating limited utility for arc suppression. Although ROR can gain precision at larger radii, its false-positive rate remains high. By contrast, our method exhibits a more stable F1-score across radii. The difference is architectural: both methods use radius-based neighborhoods, but our filter additionally leverages short-horizon temporal consistency with a probabilistic decision rule, retaining returns that persist across consecutive scans and down-weighting intermittent ones. Because arc noise is transient, this temporal-probabilistic treatment suppresses false positives relative to deterministic, per-scan ROR. Across methods, LIOR has negligible recall, yielding near-zero $F1$ for any $r$. ROR attains near $100\%$ recall but low precision, so $F1$ remains modest as $r$ grows. Furthermore, we assess our filter's sensitivity by varying $r \in [0.5, 2.5]\,\text{m}$, $K \in [2, 5]$, $\beta \in [0, 0.5]\,r$, $p_{\text{floor}} \in [0, 0.05]$, $\tau_{\text{buffer}} \in [0.1, 0.5]\,\text{s}$, and $r_{\text{octree}} \in [0.5, 2.0]\,\text{m}$. Across these settings, we observe: (i) $r$ governs the precision-recall trade-off: a small $r$ preserves thin structures and thus favors recall but admits more false positives, whereas a large $r$ favors precision but may miss thin geometries. Consequently, a moderate $r$, guided by the typical spacing between noise points and between valid points, tends to maximize $F1$. Our method attains a higher $F1$ than competing methods for all tested values of $r$. (ii) $K = 2 - 3$ reduces false positives with negligible runtime impact, whereas larger $K$ over-smooths thin structures; (iii) We set $\beta \approx 0.1 - 0.2\,r$ (10–20% of the neighbor radius $r$) so that the logistic function changes quickly but not abruptly around the threshold $d_K = r$. In other words, points just below $r$ are kept with high probability ($p_{\text{keep}} \geq 0.8$; e.g., $d_K \leq 0.984\,r$ for $\beta = 0.15\,r$), whereas points just above are assigned a low keep probability ($p_{\text{keep}} \leq 0.2$; e.g., $d_K \geq 1.015\,r$ for $\beta = 0.15\,r$), and the local slope $-r/(2\beta^2)$ indicates that the smaller $\beta$ is, the sharper the transition. Quantitatively, averaged over our three environments, F1 peaks at $\beta = 0.15\,r$ with $\text{F1} = 85.7$, remains between 84.4 and 87.0 for $\beta \in [0.10, 0.20]\,r$, and is 84.9 at $\beta = 0.10\,r$ and 84.5 at $\beta = 0.20\,r$. (iv) $p_{\text{floor}} = 0.02 - 0.05$ preserves thin features while limiting false positives; (v) $\tau_{\text{buffer}} \approx 0.2\,\text{s}$ (two scans at 10 Hz) balances transient suppression and inter-scan overlap. With $\tau_{\text{buffer}} = 0.1\,\text{s}$ the filtering is scan-by-scan, providing little temporal smoothing. Aggregating 2–3 scans improves robustness while keeping the map-update rate acceptable. By contrast, buffering 5 scans reduces the update rate to $f_{\text{update}} = \frac{10}{5} = 2\,\text{Hz}$, which hinders online operation. (vi) Choosing $r_{\text{octree}} = 1.0\,\text{m}$ offers a favorable fidelity–runtime trade-off. Coarser resolutions (larger voxels) fail to retain sufficient detail especially on thin cables, whereas very fine resolutions greatly increase processing time due to the larger number of voxels. Given the $\approx 0.71\,\text{m}$ spacing between valid environment points (vs. $\approx 3.13\,\text{m}$ for arc points), $1.0\,\text{m}$ strikes a practical balance.

## VI. CONCLUSION

In this letter, we introduced a dual-structure filtering framework that mitigates electric arc noise in drone-based LiDAR mapping near energized power lines using consecutive-scan neighborhood searches. The method combines k-d tree neighbor queries with octree insertions to process 3D point clouds efficiently. Experiments on custom datasets show that the proposed

filter outperforms existing methods. Future work will pursue tighter integrations to further improve mapping performance in large-scale, density-varying environments, addressing challenges for accurate range, depth estimation and reliable feature extraction.

## REFERENCES


[1] S. Thrun, W. Burgard, and D. Fox, "A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping," in *Proc. Millennium Conf. IEEE Int. Conf. Robot. Automat. Symposia Proc.*, 2000, vol. 1, pp. 321–328.

[2] F. Gul, W. Rahiman, and S. S. Nazli Alhady, "A comprehensive study for robot navigation techniques," *Cogent Eng.*, vol. 6, no. 1, 2019, Art. no. 1632046.

[3] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3D point cloud based object maps for household environments," *Robot. Auton. Syst.*, vol. 56, no. 11, pp. 927–941, 2008.

[4] M. H. Shariq and B. R. Hughes, "Revolutionising building inspection techniques to meet large-scale energy demands: A review of the state-of-the-art," *Renewable Sustain. Energy Rev.*, vol. 130, 2020, Art. no. 109979.

[5] H. Guan et al., "UAV-LiDAR aids automatic intelligent powerline inspection," *Int. J. Elect. Power Energy Syst.*, vol. 130, 2021, Art. no. 106987.

[6] D. McLeod, J. Jacobson, M. Hardy, and C. Embry, "Autonomous inspection using an underwater 3D LiDAR," in *Proc. OCEANS-San Diego*, 2013, pp. 1–8.

[7] A. Kurup and J. Bos, "DSOR: A scalable statistical filter for removing falling snow from LiDAR point clouds in severe winter weather," 2021, *arXiv:2109.07078*.

[8] A. Afzalaghaeinaeini, J. Seo, D. Lee, and H. Lee, "Design of dust-filtering algorithms for LiDAR sensors using intensity and range information in off-road vehicles," *Sensors*, vol. 22, no. 11, 2022, Art. no. 4051.

[9] B. Li, J. Li, G. Chen, H. Wu, and K. Huang, "De-snowing LiDAR point clouds with intensity and spatial-temporal features," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 2359–2365.

[10] X.-F. Han, J. S. Jin, M.-J. Wang, W. Jiang, L. Gao, and L. Xiao, "A review of algorithms for filtering the 3D point cloud," *Signal Process., Image Commun.*, vol. 57, pp. 103–112, 2017.

[11] M. Sze and M. Lachance, "A guide for partial discharge measurements on medium voltage (MV) and high voltage (HV) apparatus: Part 2 — measurement according to IEC 60270," OMICRON, [Online]. Available: https://www.omicronenergy.com/download/file/e5e8a545dfbaebe53f6487fe7f4886a9

[12] J.-R. Riba, "Application of image sensors to detect and locate electrical discharges: A review," *Sensors*, vol. 22, no. 15, 2022, Art. no. 5886.

[13] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems," in *Proc. ICRA Workshop Best Pract. 3D Percep. Model. Mobile Manipulation*, vol. 2, 2010, pp. 1–8.

[14] F. Mirallès et al., "LineDrone technology: Landing an unmanned aerial vehicle on a power line," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 6545–6552.

[15] C. Debeunne and D. Vivet, "A review of visual-LiDAR fusion based simultaneous localization and mapping," *Sensors*, vol. 20, no. 7, 2020, Art. no. 2068.

[16] L. Stanislas et al., "Airborne particle classification in LiDAR point clouds using deep learning," in *Proc. 12th Int. Conf. Field Service Robot., Results*, 2021, pp. 395–410.

[17] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. J. Leonard, "Dynamic pose graph SLAM: Long-term mapping in low dynamic environments," in *Proc. RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 1871–1878.

[18] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots*, vol. 34, pp. 189–206, 2013.

[19] L. Sun, Z. Yan, A. Zaganidis, C. Zhao, and T. Duckett, "Recurrent-octomap: Learning state-based map refinement for long-term semantic mapping with 3-D-LiDAR data," *IEEE Robot. Automat. Lett.*, vol. 3, no. 4, pp. 3749–3756, Oct. 2018.

[20] Y. Xu, B. Li, Y. Wang, and Y. Cui, "Filter fusion: Camera-LiDAR filter fusion for 3-D object detection with a robust fused head," *IEEE Trans. Instrum. Meas.*, vol. 73, 2024, Art. no. 2528812.

[21] G. Xie, J. Zhang, J. Tang, H. Zhao, N. Sun, and M. Hu, "Obstacle detection based on depth fusion of LiDAR and radar in challenging conditions," *Ind. Robot, Int. J. Robot. Res. Appl.*, vol. 48, no. 6, pp. 792–802, 2021.

[22] V. Valseca, J. Paneque, J. Martıńez-de Dios, and A. Ollero, "Real-time LiDAR-based semantic classification for powerline inspection," in *Proc. Int. Conf. Unmanned Aircr. Syst.*, 2022, pp. 478–486.

[23] Y. Shen, J. Huang, J. Wang, J. Jiang, J. Li, and V. Ferreira, "A review and future directions of techniques for extracting powerlines and pylons from LiDAR point clouds," *Int. J. Appl. Earth Observ. Geoinf.*, vol. 132, 2024, Art. no. 104056.

[24] J.-I. Park, J. Park, and K.-S. Kim, "Fast and accurate desnowing algorithm for LiDAR point clouds," *IEEE Access*, vol. 8, pp. 160202–160212, 2020.

[25] T. Parsons, J. Seo, B. Kim, H. Lee, J.-C. Kim, and M. Cha, "Dust de-filtering in LiDAR applications with conventional and CNN filtering methods," *IEEE Access*, vol. 12, pp. 22032–22042, 2024.

[26] L. Petit and A. L. Desbiens, "MOAR planner: Multi-objective and adaptive risk-aware path planning for infrastructure inspection with a UAV," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 8422–8428.

[27] B. C. Ooi, "Spatial KD-tree: A data structure for geographic database," in *Proc. Datenbanksysteme Büro, Technik Und Wissenschaft: GI-Fachtagung Darmstadt*, Apr. 1987, pp. 247–258.