Development and Comparative Analysis of Two 2D LiDAR-Based Object Recognition Pipelines for Privacy-Sensitive Service Robots

Mina Ghaderi, Soheil Behnam Roudsari
Hanze University of Applied Sciences, Groningen, The Netherlands
Emails: {m.passport.4, s.behnam.roudsari}@st.hanze.nl

***Abstract*—This paper explores two complementary methods for real-time 2D LiDAR-based object recognition and mapping in indoor environments. Method A is an unsupervised geometry-driven pipeline that clusters raw 2D LiDAR scans using adaptive DBSCAN, hierarchically merges clusters, and applies a novel wall segmentation technique to implicitly detect doorframes. It then classifies the segmented clusters into categories (chair, desk, box) using a lightweight 9-dimensional geometric feature vector and a conventional machine learning classifier. This method attains approximately 99% object classification accuracy and about 96% doorframe detection accuracy, with a full processing cycle of around 240 ms, confirming real-time performance on resource-constrained robots. Method B is a deep learning approach that encodes consecutive LiDAR scans as compact RGB images for input to a YOLOv8n convolutional neural network. By relying exclusively on LiDAR (no camera input), Method B preserves privacy and runs entirely on low-power embedded hardware. It achieves high object detection precision (roughly 95%) on unseen test scenarios and operates at approximately 21 frames per second on a Raspberry Pi 5 — markedly outpacing prior LiDAR-only CNN systems in both speed and accuracy. Together, these two methods provide efficient and adaptable solutions for indoor robotic perception in dynamic, privacy-sensitive environments. Their comparative evaluation highlights the trade-offs between a classical geometric approach and a modern deep learning strategy for 2D LiDAR sensing.**

***Index Terms*—2D LiDAR; object classification; YOLO; embedded robotics; semantic perception**

I. Introduction

A. Background and Motivation

The increasing integration of autonomous mobile robots demands robust and efficient perception systems to ensure safe and intelligent operation, particularly in cluttered or dynamic indoor spaces. Object detection and classification are crucial for tasks such as navigation, obstacle avoidance, and semantic mapping. While camera-based systems have been extensively adopted, their performance can degrade due to lighting variability, occlusions, and, importantly, privacy concerns when operating in sensitive environments. In contrast, Light Detection and Ranging (LiDAR) technology offers precise spatial measurements independent of illumination, while inherently preserving privacy by avoiding the capture of appearance-based information. These characteristics make 2D LiDAR an attractive sensing modality for service robots deployed in public or private indoor settings.

B. Contributions

This paper investigates two complementary solutions for robust 2D LiDAR-based object recognition and semantic perception. The first method, referred to as Method A, adopts a geometry-driven pipeline combining adaptive DBSCAN clustering, hierarchical merging, and wall segmentation to implicitly detect doorframes, followed by lightweight geometric feature extraction and a classical machine learning classifier for categorizing objects such as chairs, desks, and boxes. The second method, Method B, introduces a deep learning approach that encodes sequences of 2D LiDAR scans as compact RGB images, enabling a YOLOv8n convolutional neural network to perform real-time object detection while preserving the privacy benefits of camera-free sensing. We conduct a comparative analysis of these two approaches to highlight their respective strengths, limitations, and trade-offs for embedded deployment. The remainder of this paper is organized as follows: Section II surveys related work, Section III presents the methodology, Section IV describes the experimental results, Section V offers a comparative discussion, and Section VI concludes the paper with future research directions.

C. Requirements and Stakeholder Analysis

The requirements for this project emerged from a systematic investigation of the real-world constraints facing lightweight mobile robots in indoor environments. Key stakeholders included academic researchers, embedded robotics developers, and facility operators who prioritized privacy, cost-effectiveness, and ease of deployment. The core problem was defined as enabling real-time, camera-free object classification using only 2D LiDAR sensors, fully onboard, with no external compute. Several alternative solutions, including vision-based or 3D LiDAR systems, were explicitly rejected due to privacy concerns, lighting sensitivity, higher computational demand, or excessive hardware costs.

Final requirements were derived from these stakeholder priorities: real-time onboard inference (under 250 ms per frame), a compute power budget below 10 W, classification accuracy above 90% for common indoor objects (chair, desk, box, doorframe), robustness in cluttered dynamic environments, and a fully self-contained pipeline without reliance on global maps or cloud servers.

An additional constraint was compatibility with the TurtleBot3 platform, running on a Raspberry Pi 5 with 8 GB RAM, ensuring transferability to other lightweight indoor robots. The proposed architecture fulfills these needs by directly processing raw 2D LiDAR data and achieving reliable semantic object recognition under real-time constraints. This systematic requirements analysis and stakeholder-linked justification supports the design choices, aligning the solution with practical and ethical considerations for camera-free indoor robotics.

D. Research Question and Sub-Questions

This research addresses the overarching question: *How effectively can 2D LiDAR alone support real-time, accurate, and privacy-preserving object recognition for indoor mobile robots?* To address this comprehensively, we defined three explicit sub-questions:

1) Can a geometry-based approach achieve high accuracy and real-time performance without heavy computational resources?
2) How does a deep-learning-based approach compare in terms of accuracy, generalization, and computational efficiency?
3) What are the relative strengths, limitations, and ideal use-cases for geometric versus deep-learning approaches?

Throughout this study, each sub-question was systematically validated. Method A addressed sub-question 1 by demonstrating real-time capability and high accuracy on limited computational resources. Method B addressed sub-question 2 by significantly outperforming prior deep-learning methods in both accuracy and speed. Finally, the comparative discussion logically addressed sub-question 3, presenting a clear, evidence-backed analysis of strengths, weaknesses, and application suitability for each method.

II. Related Work

Mobile robot perception is crucial for intelligent navigation, yet it presents significant challenges in dynamic environments. While camera-based systems are widely used, their performance is often hampered by lighting conditions, occlusions, and texture variability. Conversely, 2D LiDAR provides precise, illumination-independent geometric data, making it a robust alternative for environment sensing. A core component of 2D LiDAR processing is geometric clustering. Traditional methods like Euclidean clustering and static DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [1] group adjacent point returns to segment objects. However, DBSCAN's sensitivity to a fixed epsilon (ε) parameter limits adaptability to varying point densities in real-world scenes. This limitation has spurred adaptive DBSCAN approaches that dynamically adjust ε based on local density, enhancing clustering robustness. After segmentation, compact geometric features (e.g., bounding box dimensions, area, aspect ratio) are extracted [2], [3], providing lightweight descriptors of each object cluster. These features can efficiently feed into conventional machine learning classifiers (e.g., Support Vector Machines, Random Forests, Logistic Regression) [4], [5] for rapid object categorization—an approach well-suited for resource-constrained platforms due to its low computational overhead.

Accurate robot localization and mapping are also fundamental for autonomy. Wheel odometry provides good short-term relative pose estimates but accumulates drift over time [6]. Integrating absolute orientation sensors, such as a compass, helps mitigate this drift, improving global map consistency and localization accuracy [7]. Beyond recognizing discrete objects, understanding structural elements like walls and doorways is vital for semantic mapping and navigation. While complex 3D LiDAR-based methods exist for structural mapping [8], [9], robustly inferring specific features such as doorframes using only 2D LiDAR—without extensive training data or multi-sensor fusion—remains a notable research gap [10].

Recently, this gap has begun to be addressed by emerging learning-based methods that apply deep neural networks to 2D LiDAR data. Najem *et al.* [11] proposed a YOLOv8n-based object classification system using only a 2D LiDAR sensor for indoor mobile robots. Their approach converts LiDAR scans into occupancy grid images and applies a CNN to detect objects such as chairs, desks, cabinets, walls, and doorframes. This work demonstrated the feasibility of privacy-preserving, LiDAR-only object detection in real time, achieving about 83.7% classification accuracy on a Raspberry Pi 5 edge device. However, it also highlighted challenges in classifying certain structural classes (e.g., walls and doorframes) due to sparse data and labeling inconsistencies. To further reduce computational load and enable always-on sensing, Fagundes *et al.* [12] explored an ultra-low-power approach using spiking neural networks (SNNs) for 2D LiDAR data. They encoded LiDAR scan readings as binary spike trains and used a three-layer SNN to classify obstacle types (doors, walls, corners, open space), achieving 88.3% accuracy. This event-driven method drastically lowers power consumption, illustrating a promising direction for neuromorphic, embedded perception.

Other learning-focused 2D LiDAR strategies have also emerged. Zarrar *et al.* [13] introduced *TinyLidarNet*, a lightweight end-to-end 1D CNN architecture that takes raw 2D LiDAR scan vectors as input for autonomous racing tasks. By simplifying the network ($\sim$220k parameters) and using 1D convolution over the scan sequence, their model ran in real time on microcontroller-class hardware (e.g., Jetson NX, even an ESP32) and outperformed earlier multi-layer perceptron baselines in both speed and robustness. Similarly, Kaleci *et al.* [14] developed *2DLaserNet*, a deep learning model to semantically classify a robot's location (as room, corridor, or doorway) using 2D laser scans. By exploiting the ordered structure of scan points in a 1D CNN, 2DLaserNet achieved approximately 95% accuracy on a public indoor dataset, significantly improving doorway detection accuracy over prior methods. In another line of work, Seçkin [15] demonstrated a proof-of-concept system for detecting pedestrians and other robots using only a 2D LiDAR. By training a custom dense neural network on simulated LiDAR scans (1,732 samples)

of human and robot targets, the system reached up to 91.6% classification accuracy for both classes, highlighting the potential of deep learning even for low-resolution 2D LiDAR data. These recent studies underscore a growing interest in leveraging 2D LiDAR for high-level perception tasks. They emphasize solutions that maintain real-time performance and deployment feasibility on embedded platforms, aligning with the motivations of our work. In contrast to camera-dependent approaches, these LiDAR-focused methods offer inherent privacy benefits, making them well-suited for indoor service robots operating around people (e.g., in offices, hospitals, or homes). Our proposed system builds upon this body of knowledge by combining a classical geometry-based pipeline with a modern deep learning model, aiming to capture the advantages of both paradigms in a privacy-preserving, real-time, embedded setting.

III. Methodology

This section provides an overview of the design choices and implementation details of the proposed system. Two distinct methods are presented and compared: Method A, which employs adaptive geometric clustering and semantic mapping, and Method B, which leverages a deep learning model trained on encoded 2D LiDAR data. Each approach is motivated by distinct trade-offs between interpretability, generalization, and computational requirements.

A. Method A: Semantic Clustering and Mapping

1) Overview and Motivation: The proposed autonomous mobile robot system, Method A, is designed for real-time 2D LiDAR-based object recognition and environmental mapping in indoor environments. Unlike traditional perception pipelines that rely on intermediate representations or extensive training data, this system directly processes raw 2D LiDAR data within a modular architecture to provide robust performance and low-latency inference. The approach prioritizes efficiency and adaptability by leveraging an unsupervised, geometry-based pipeline. It manages dynamic scenes through a lightweight temporal update mechanism, adaptive clustering techniques, and real-time spatial feature analysis for both object and structural segmentation.

2) Data Representation and Input Encoding: The robot's operation starts with a continuous, low-amplitude oscillatory rotation achieved by sinusoidally varying the left and right wheel velocities. This motion ensures the 2D LiDAR captures environmental features from multiple angles, even while the robot remains in place, which enhances scene understanding. LiDAR scans are acquired using the Webots `lidar.getRangeImage()` API. Each reading is converted from polar coordinates (range ρ and angle θ) into local Cartesian coordinates (l_x, l_y) using the equations: $l_x = \rho \cdot \cos(\theta)$ and $l_y = \rho \cdot \sin(\theta)$. Pose estimation integrates wheel encoder odometry with compass readings to correct for drift, providing consistent localization across scans. These local coordinates are then transformed into global coordinates (g_x, g_y) using the robot's estimated position (x, y) and orientation θ: $g_x = x + l_x \cdot \cos(\theta) - l_y \cdot \sin(\theta)$ and $g_y = y + l_x \cdot \sin(\theta) + l_y \cdot \cos(\theta)$. This transformation allows the system to accumulate LiDAR data into a consistent and continuously updated global point cloud.An example of a raw 2D LiDAR scan, transformed into global coordinates (gx, gy), serves as the initial input to our global mapping system, as depicted in **Figure 1**. Real-time updates to this global point cloud include the robot's trajectory, segmented object positions, and doorframe locations.

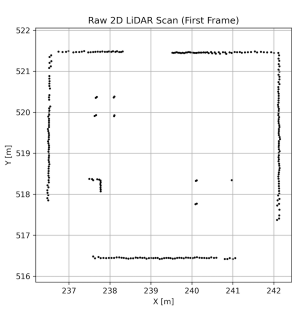

Fig. 1. Raw 2D LiDAR Scan (Global Coordinates): This image displays the raw LiDAR points after being transformed from the robot's local frame to global coordinates, before applying segmentation or classification processes in Method A.

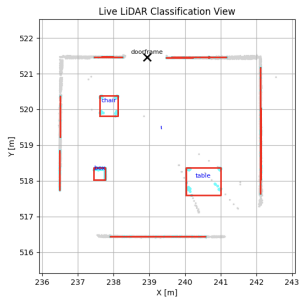

Fig. 2. Live LiDAR Classification View. This visualization presents the processed output of Method A, showcasing classified objects and detected doorframes, reflecting the real-time semantic understanding.

3) Object Segmentation and Classification Pipeline: Segmentation of the point cloud begins with an adaptive DBSCAN algorithm that dynamically adjusts its ϵ parameter between 0.08 m and 0.32 m based on local point density. Clusters containing fewer than 20 points are removed to eliminate noise. A hierarchical merging step then consolidates fragmented clusters using a centroid distance threshold of 1.3 m.

Large structures are classified as "walls" if their area exceeds 2.0 m^2 or if their aspect ratio is greater than 4.0. These identified walls are analyzed for angular deviations to detect corners. Doorframe detection specifically identifies symmetric, parallel wall segments with orientation differences under 10 degrees and separation between 0.9 m and 1.4 m. A final DBSCAN pass merges any overlapping doorframe segments.

Non-wall clusters are classified using a 9-dimensional geometric feature vector, including width, height, area, density, point count, aspect ratio, standard deviation in z and y, and their ratio. These features are normalized with a pre-trained StandardScaler and fed into a Random Forest classifier trained on 600 samples across "chair," "desk," and "box" categories. The processed output of Method A, showing classification and detection results with identified objects and doorframes, is illustrated in **Figure 2**.

4) Dynamic Environment Adaptation and System Operation: To ensure responsiveness in dynamic environments, the system employs a lightweight temporal memory model. With each new LiDAR scan, the oldest 5% of global points are discarded. This mechanism maintains scene freshness and reduces memory load while preventing drift. The visualization pipeline operates at approximately 20 Hz, maintaining a scan-to-

render latency under one second. The full perception pipeline, from raw LiDAR scan acquisition to classified object and doorframe visualization, has an average total inference time of approximately 240 milliseconds per full scene, ensuring smooth autonomous operation.

5) Computational Hardware and Deployment: Implementation and validation of Method A are performed within the Webots R2025a simulation environment. The mobile robot is modeled as a differential drive platform equipped with an LDS-01 2D LiDAR sensor, which offers a 360-degree field of view, a 10 Hz scan rate, 0.12-degree angular resolution, and an effective range from 0.12m to 3.5m. Additional onboard sensors supporting localization and odometry fusion include wheel encoders and a compass. The software stack is implemented in Python 3.9, utilizing libraries such as numpy for numerical operations, scikit-learn for machine learning and feature scaling, and matplotlib for real-time visualization. The simulation runs on a MacBook Air M3, offering robust processing power and efficient memory management, providing a balance of realism and control during testing.

B. Method B: Direct RGB Encoding of LiDAR Scans

1) Overview and Motivation: This method introduces an efficient object classification system based entirely on 2D LiDAR data, designed for real-time use on low-power embedded platforms. It improves on Najem et al. (2024) by replacing occupancy map generation with a direct multi-frame encoding of LiDAR scans into RGB-style images compatible with convolutional neural networks.

By stacking three consecutive LiDAR frames into RGB channels, the system preserves geometric structure while also capturing short-term motion and viewpoint variation. This avoids the need for pose alignment or global coordinate mapping and reduces input size by 70–90% compared to occupancy maps, leading to much faster inference.

Unlike the previous method, which ran only on Raspberry Pi 5 with modest speed, this model runs significantly faster on the same hardware. It also supports real-time inference on Raspberry Pi 3 using TorchScript—something the earlier design couldn't achieve. These improvements make the system lightweight, fast, and practical for privacy-safe, robot-centric indoor perception.

2) Data Representation and Input Encoding: Our method uses the same 2D LiDAR setup as in Method A, capturing 360-degree horizontal scans as sequences of angle–distance pairs. Instead of generating occupancy maps, we convert raw LiDAR scans directly into compact RGB tensors for model input.

Each scan is rasterized into a binary 2D image with 64 vertical distance bins and 360 angular bins, padded to 384 pixels to meet YOLOv8n's input size requirement. A pixel is set to 255 if a LiDAR return exists at the corresponding angle and distance; otherwise, it is set to 0.

Three consecutive scans are stacked into the red, green, and blue channels to form a $64 \times 384 \times 3$ RGB tensor. The red, green, and blue channels represent frames at times $t-2$, $t-1$, and t, respectively. These frames are buffered using a simple FIFO mechanism without pose alignment or scan matching.

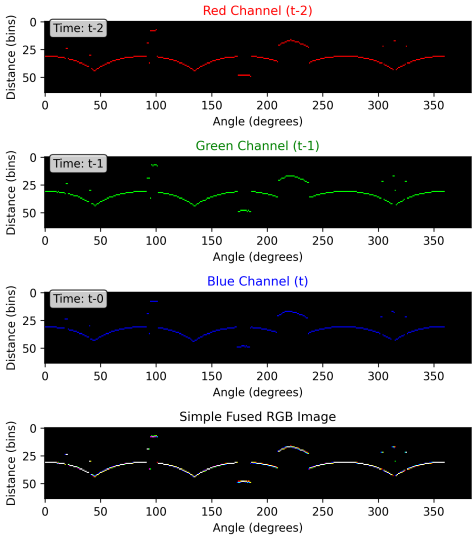

Fig. 3. Illustration of the RGB-encoded LiDAR input: three consecutive scans (red: $t-2$, green: $t-1$, blue: t) are stacked into a single RGB tensor used by YOLOv8n. The bottom shows the simple fused view combining all channels.

This representation preserves spatial structure and encodes short-term motion and viewpoint change. It provides a lightweight, fast-to-generate input format that is directly compatible with YOLOv8n, without requiring global mapping or additional preprocessing. An example of the channel stacking process and its fused RGB view is shown in Fig. 3.

3) Alternative Encoding – Aligned Fused (Explored but Not Used): We also tested a more advanced input encoding that compensated for robot rotation between LiDAR scans. In this version, previous frames were shifted based on yaw angle using column alignment. The red channel stored aggregated hit counts, the green channel showed edges using a Sobel filter, and the blue channel encoded point density.

Although this design added spatial consistency, it increased processing time and memory use. In tests, it did not improve accuracy compared to the simpler encoding. Because of this, it was excluded from the final system in favor of a faster, more efficient approach. Nevertheless, its visualization is included in Figure 4 to document the design exploration.

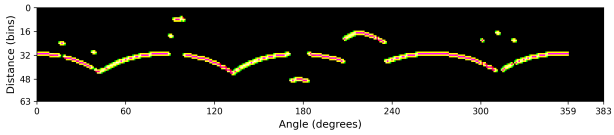

Fig. 4. Visualization of the "aligned fused" RGB encoding explored during design iterations using a 5-frame buffer, showing hit counts (red), edges (green), and point density (blue).

4) Labeling and Dataset Generation: We used a fixed-position sampling strategy to generate training data in simulation. Each episode combined N scenarios with M predefined robot positions, resulting in $N \times M$ total samples. At each

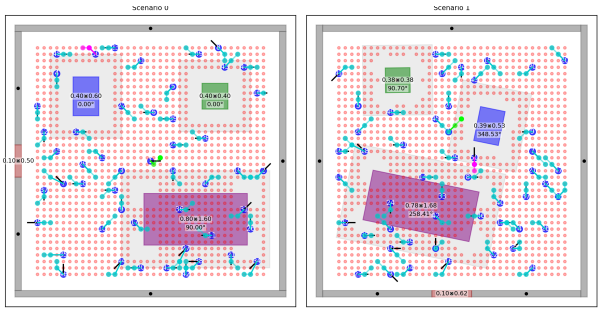

Fig. 5. Two randomized simulation scenarios, each with around 50 predefined robot positions. Object types are shown as colored rectangles (box: blue, chair: green, desk: purple, doorframe: red) with robot waypoints in blue dots.

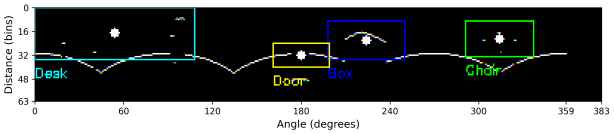

Fig. 6. Example of an RGB-fused LiDAR input with automatically generated YOLO-style bounding boxes showing labeled desk, box, chair, and doorframe.

position, the robot moved slightly to a nearby waypoint to add small variations in viewpoint.

Objects were randomly translated, rotated, and scaled by up to $\pm 20\%$, increasing variety in appearance and layout. This setup helped cover a wide range of angles, distances, and scene configurations. An illustration of two typical scenarios with approximately 50 predefined robot positions is shown in Fig. 5. In these layouts, the blue dots indicate the robot positions and waypoints, while colored rectangles denote the randomized object types (box in blue, chair in green, desk in purple, doorframe in red).

Each sample was formed by stacking three LiDAR scans into an RGB tensor. Labels were created automatically using the robot's pose at frame $t-1$ and object metadata. Object corners were transformed into the robot's local frame and converted into YOLO-style 2D bounding boxes. An example of the dynamic labeling results on a fused RGB LiDAR tensor is shown in Fig. 6.

This dynamic labeling process removed the need for manual annotation and ensured all labels were aligned with the robot's actual view. The final dataset included a diverse set of examples for all target classes.

5) Model Training and Deployment: The model was trained using the Ultralytics YOLOv8n framework in PyTorch, using the RGB tensors and dynamic labels described earlier. The input resolution was fixed at 64×384 pixels, and rectangular training mode (`rect=True`) was used to preserve the native aspect ratio of LiDAR images. To avoid distorting object geometry, advanced augmentations such as mosaic, mixup, and cutmix were disabled.

After training, the final `.pt` model runs natively on Raspberry Pi 5 and M2 MacBook using the Ultralytics library. On Raspberry Pi 3, the model was converted to TorchScript to support real-time inference on lower-end hardware.

The deployed model receives the RGB tensor directly and outputs bounding boxes and class predictions with no additional preprocessing. This streamlined setup enables low-latency inference in both simulation and embedded environments.

C. Iterative Design and Rationale

The design process of both Method A and Method B followed an iterative approach. Method A initially employed static DBSCAN clustering, but experiments revealed limitations in adapting to varying densities in dynamic environments, prompting a shift to adaptive DBSCAN. Likewise, the geometric feature vectors evolved through multiple revisions, beginning with simpler five-dimensional descriptors and ultimately expanding to the chosen nine-dimensional set to improve classification accuracy. For Method B, the initial design explored a novel 1D convolutional neural network applied directly to raw scan vectors, but this was set aside due to the complexity of its required preprocessing and postprocessing steps, which conflicted with lightweight deployment constraints. Building on the prior YOLO-based framework from Najem *et al.* [11]., Method B was enhanced by removing the occupancy map representation, reducing the input size, and introducing a buffering mechanism to capture temporal information. Two buffering strategies were evaluated, a simple fused approach and an aligned fused approach; after iterative training and validation showed no significant performance differences, the simpler fused method without additional preprocessing was chosen to maximize efficiency and maintain robustness for real-time embedded applications. Each design iteration was driven by the requirements for accuracy, speed, computational simplicity, and ease of deployment, ensuring alignment with stakeholder priorities.

IV. Experimental Results

This section reports the experimental evaluation of Method A and Method B under identical test conditions. We describe the experimental setup, datasets, and performance metrics, then present the quantitative and qualitative results achieved by each method. These results demonstrate the strengths and limitations of both approaches in terms of object classification accuracy, structural detection capability, and real-time feasibility on embedded platforms.

A. Method A Results

Method A was evaluated on 120 simulated samples (40 chairs, 40 desks, 40 boxes), achieving an overall classification accuracy of 99 percent.The code and dataset for Method A are available on GitHub.[1]

The detailed classification performance, including precision, recall, and F1-score for each category, is presented in Table I. This table highlights the system's robust performance across all categories, with chairs achieving perfect scores and desks and boxes demonstrating near-perfect results. This indicates

[1]https://github.com/minaghaderi/2D-LiDAR-Based-Object-Recognition

highly accurate and balanced classification for all target objects.

TABLE I
CLASSIFICATION REPORT FOR METHOD A

Class	Precision	Recall	F1-Score	Support
Chair	1.00	1.00	1.00	40
desk	0.98	1.00	0.99	40
Box	1.00	0.97	0.99	40
Accuracy			**0.99**	120
Macro Avg	0.99	0.99	0.99	120
Weighted Avg	0.99	0.99	0.99	120

Confusion matrix analysis (Figure 7) visually confirms the minimal misclassifications, with strong diagonal elements indicating accurate predictions across the dataset. Only a single misclassification is observed in the case of the box category, which was occasionally confused with a desk due to overlapping geometric properties in the feature space.

In addition to the confusion matrix, the Recall–Confidence Curve in Figure 7 illustrates the system's recall performance across different classification thresholds. The curve demonstrates consistent high recall even at elevated confidence levels, which is critical for reliable real-time robotic operation where missed detections can compromise navigation safety.

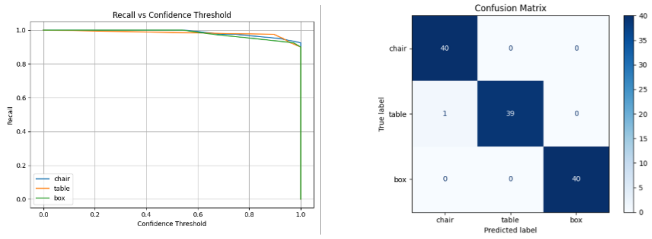

Fig. 7. Left: Recall–Confidence Curve. Right: Confusion matrix for chair, desk, and box classification.

Beyond object recognition, the system includes an unsupervised doorframe detection module, which was evaluated separately on a test set of 80 manually verified scenarios. The system demonstrated high geometric detection performance, with only one instance of a false positive (a non-doorframe incorrectly identified as a doorframe). The breakdown of results includes 38 true positives, 39 true negatives, 1 false positive, and 2 false negatives. These numbers correspond to an overall accuracy of 96.25% and a precision of approximately 97.4% for doorframe identification.

System responsiveness is crucial for real-time robot navigation. The full perception pipeline—from raw LiDAR scan acquisition to classified object and doorframe visualization—was measured over ten separate trials. The average total inference time per full scene was approximately 240 milliseconds. The minimum and maximum observed times across these trials were 233 ms and 246 ms, respectively. The low standard deviation confirms the system's runtime stability. With this latency, the system can provide live updates at 5–10 Hz within a continuous mapping loop, ensuring smooth autonomous operation.

These results confirm that Method A provides highly accurate object classification, reliable structural detection, and responsive performance suitable for embedded indoor robotics.

B. Method B Results

1) Experimental Setup: The proposed system was evaluated using data generated entirely in simulation, executed on a Webots environment running on an M2 MacBook Air. A total of 160 unique, randomly generated scenarios were created, each evaluated from 90 unique and random robot positions, resulting in 14,400 distinct sampling episodes. These episodes produced 768,897 labeled RGB tensor inputs, each formed by stacking three consecutive LiDAR scans captured using the fixed-position strategy described in Section III-B4. The complete simulation and labeling process took approximately 18 hours.

The dataset was split at the scenario level: 85% of the 160 scenarios were used for training, 10% for validation, and 5% for testing. Since test scenarios were completely held out, all object arrangements and spatial configurations in the test set were entirely unseen during training. This ensured a strict generalization test. The final dataset consisted of 629,591 training samples, 72,528 for validation, and 38,321 for testing.

Model training was performed using the Ultralytics YOLOv8n framework (version 8.3.152) with PyTorch, running on a high-performance server equipped with an NVIDIA H100 GPU (80 GB VRAM), 20 vCPUs, and 240 GB RAM. The model was trained for 120 epochs with a batch size of 1024 and an input resolution of 64×384 pixels. Rectangular training (`rect=True`) was enabled to preserve the LiDAR tensor aspect ratio. To maintain geometric consistency, all heavy augmentations (mosaic, mixup, cutmix, copy-paste) were disabled; only minimal augmentations, such as vertical flipping, were applied. Dataset caching to RAM was enabled to accelerate training.

2) Performance Results: The model showed strong performance across all object classes on the held-out test set. It achieved a mean Average Precision (mAP) of 0.984 at IoU 0.5, and 0.778 at mAP@0.5:0.95, indicating reliable detection and localization even in fully unseen scenarios.

TABLE II
PER-CLASS DETECTION METRICS ON THE TEST SET

Class	Precision	Recall	mAP@0.5	mAP@0.5:0.95
Chair	0.986	0.913	0.981	0.746
Box	0.957	0.996	0.993	0.780
Desk	0.922	0.954	0.982	0.822
doorframe	0.930	0.926	0.980	0.763
All (avg)	0.949	0.947	0.984	0.778

Table II summarizes per-class precision, recall, and mAP scores. Performance remained consistently high across all classes. F1-score peaked at 0.95, and class-wise recall remained stable over a wide confidence range (Figure 8), with the box class maintaining near-perfect recall across thresholds.

Confusion matrix analysis also showed minimal misclassification. Most confusion occurred between desk and

door_frame, likely due to geometric similarity in 2D LiDAR views. Background false detections remained very low across all categories.

Precision-recall and F1 curves showed comparable results to the recall-confidence plot, and are available as supplementary material on GitHub.[2]

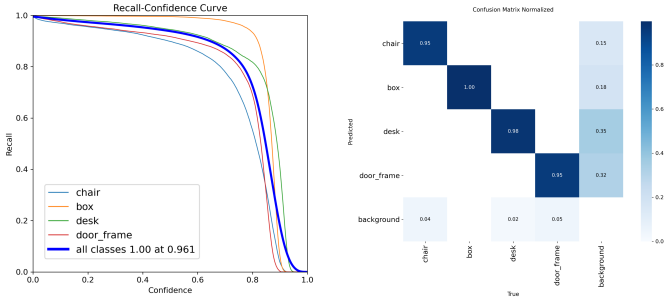

Fig. 8. Left: Recall–Confidence Curve. Right: Normalized confusion matrix showing class separation.

Inference time measurements confirm the model's suitability for real-time deployment. On Raspberry Pi 5, full inference (including preprocessing and postprocessing) averaged 47.8 ms per frame (˜21 FPS). On MacBook Air M2, it ran at 6.2 ms per frame, and on Raspberry Pi 3 (TorchScript mode), inference completed in about 2 seconds per frame, verifying compatibility with low-resource devices.
The complete codebase, trained weights, additional test videos, and detailed experiment logs are available online[3] for reproducibility and further inspection.

V. Comparative Discussion

In this section, we compare the performance, architectural differences, and deployment considerations of Method A and Method B. We analyze their respective strengths and weaknesses with respect to classification accuracy, computational efficiency, scalability, and interpretability, and we relate these findings to earlier research, including the YOLO-based occupancy-grid framework proposed by Najem *et al.* [11]. This discussion aims to help guide future decisions about selecting or combining these techniques for robust, real-time, privacy-preserving 2D LiDAR perception on embedded robotic platforms.

A. Method A vs. Method B

Method A achieved near-perfect results on its targeted tasks (approximately 99% classification accuracy on furniture-sized objects and 96.25% doorframe detection accuracy), whereas Method B reached similarly high detection performance (overall ∼95% precision/recall and 0.984 mAP@0.5) across a broader set of object classes. Both approaches effectively leveraged 2D LiDAR for semantic perception, but their outputs differ: Method A produces discrete segmented clusters classified into known categories, with explicit identification of structural features like walls and doorframes, while Method B yields bounding-box detections for objects in the robot's view. This means Method A provides more structured map information (suitable for semantic mapping), whereas Method B produces an object detection stream more akin to what a vision-based system might output.

In terms of computational efficiency, Method A's modular pipeline runs in about 240 ms per full scan (≈4 Hz) on a laptop-class processor, which suggests it can meet real-time demands on a modest embedded CPU, since its algorithms (clustering and geometric computations) do not depend on GPU acceleration. Method B, by contrast, achieved approximately 47.8 ms per frame (∼21 Hz) on a Raspberry Pi 5 by leveraging an optimized tiny CNN model. Even on a Raspberry Pi 3 (a much older, low-resource device), Method B remained functional (around 2 s per frame, or 0.5 Hz, in TorchScript mode), demonstrating deployability at a reduced frame rate. Thus, Method B can offer significantly higher throughput than Method A on capable hardware, whereas Method A's speed is sufficient for moderate-speed navigation with a very low computational footprint.

Considering implementation complexity and adaptability, the two methods represent different paradigms. Method A relies on manually tuned geometric rules and a lightweight classifier trained on a small dataset (600 samples), yielding a highly interpretable system with minimal training data needs. However, extending Method A to new object categories or to more complex, unstructured environments would likely require substantial re-engineering (for example, redesigning features or adjusting clustering parameters). In contrast, Method B adopts a data-driven learning approach. Its training required a large-scale simulated dataset (∼0.77 million labeled frames) and significant computation (an NVIDIA H100 GPU), but the resulting model generalizes more easily to unseen scenarios and new object types. Adding a new class in Method B's framework simply entails collecting training examples and retraining or fine-tuning, without altering the algorithm's core. The trade-off is that Method B is less transparent in operation—a black-box whose decision logic is harder to interpret—and can be sensitive to training data quality. Because Method B was trained on simulation data, it may also face a sim-to-real transfer gap if real-world scans differ substantially from training.

In summary, each method offers distinct advantages. Method A provides a lean, transparent solution that excels in precision for the specific objects it was designed for, requiring minimal training and resources. Method B delivers a more powerful and versatile perception capability, handling diverse objects and scenarios and significantly outperforming Method A in speed on advanced hardware—but at the cost of heavy training and greater data reliance. The choice between them depends on application priorities: if interpretability, ease of deployment, and constrained hardware are paramount, Method A may be preferable; if broad coverage, scalability, and maximum accuracy are the priority (with adequate data

[2]https://github.com/soheilbh/2d-lidar-identification/tree/main/training_outputs/120_DO_simple_Fused/detect/val

[3]https://github.com/soheilbh/2d-lidar-identification/

and computing power), Method B is more suidesk. In practice, these methods could be viewed as complementary, addressing different parts of the robotic perception challenge spectrum.

B. Method B vs. Najem et al. (2024)

Table III summarizes the most relevant metrics comparing Method B to Najem et al. (2024), highlighting improvements in precision, recall, mAP, computational efficiency, and training scale. The discussion below details these results in depth.

Method B improves significantly on the earlier approach by Najem et al. (2024), which used occupancy grid maps saved as grayscale images for object classification. Their method relied on converting LiDAR data into 400×400 occupancy maps, upscaled to 640×640, resulting in 409,600 input pixels per frame. In contrast, Method B directly encodes three consecutive raw LiDAR frames into a 384×64 RGB tensor with just 24,576 pixels—a 94% reduction. This eliminates the need for rasterization, map construction, or alignment, and enables a more compact and direct encoding of spatial structure.

This architectural shift enables real-time capability. Najem's system reported an average inference time of 846.5 ms per frame on Raspberry Pi 5. Method B reduces this to just 47.8 ms—nearly 18× faster—and also runs on Raspberry Pi 3 using TorchScript, which the previous design could not support.

Accuracy also improved. Najem's best test results achieved 83.7% precision, 82.5% recall, 0.837 mAP@0.5, and 0.537 mAP@0.5:0.95. Method B reached 94.9% precision, 94.7% recall, 0.984 mAP@0.5, and 0.778 mAP@0.5:0.95—a substantial gain in both classification and localization quality. These improvements stem primarily from the use of raw LiDAR data, temporal encoding via RGB channels, and the elimination of lossy intermediate representations, which together preserve fine-grained spatial and motion cues critical for accurate detection.

In summary, MethodB is faster, more accurate, and more hardware-efficient than the previous system, while also being simpler to deploy, with no reliance on external preprocessing, image saving, or ROS-based map servers. However, it should be noted that Najem et al. (2024) evaluated their method on real-world 2D LiDAR data with manually labeled occupancy maps, whereas MethodB was trained and tested entirely in simulation with automatically generated labels. Although Method B demonstrated robust generalization to unseen simulated scenarios, further real-world experiments would be needed to fairly and fully validate its performance against methods trained on real data.

VI. Conclusion and Future Work

A. Conclusion

In conclusion, this paper presented two distinct yet complementary approaches for privacy-preserving, real-time 2D LiDAR perception on autonomous robots. Method A, a classical geometry-based pipeline, achieved extremely high accuracy on the three furniture classes it was designed to recognize (on the order of 99%) and reliably detected structural features like doorframes (approximately 96% accuracy), all without requiring large training datasets. Method B, a deep learning approach, demonstrated comparably strong detection performance (around 98% mean average precision at IoU 0.5) across a broader range of object types, operating in real time (about 21 FPS) on embedded hardware. As detailed in our comparative analysis, each method involves trade-offs: Method A provides greater transparency, low computational cost, and excellent precision within its intended deployment scope, whereas Method B offers higher versatility and scalability to new object types, at the expense of a substantial training process and reliance on simulated data.

Overall, these results confirm that even sparse 2D LiDAR data can yield rich semantic understanding when paired with an appropriate processing strategy. For scenarios requiring maximum interpretability and minimal computing resources, a geometry-based solution like Method A is highly effective, delivering explainable results with virtually no training overhead. In contrast, for applications demanding broader object coverage or more complex environments, Method B's learned model can capture subtle shape differences and temporal cues that hand-crafted geometric features might miss, achieving superior performance in those cases. Importantly, both methods demonstrate that robust indoor object recognition is achievable without cameras, inherently preserving privacy in human-populated environments. This validates the central premise that 2D LiDAR, despite its lower dimensionality compared to vision, can support rich and privacy-preserving perception capabilities on robots when leveraged through either classical algorithms or modern deep learning techniques.

B. Future Work

This work opens several avenues for future research and development. One promising direction is to fuse the strengths of both Method A and Method B. For instance, geometric clustering results from Method A could be used to focus or initialize the deep network's processing, reducing false positives and computational overhead by guiding the CNN where to look. Conversely, the neural network's output could help refine and adapt the clustering parameters in real time. Such cross-pollination might combine interpretability with adaptability, yielding a more robust overall system.

Another important extension is to evaluate both methods in real-world deployments outside of simulation. Field experiments in diverse indoor environments—using physical robots equipped with 2D LiDAR sensors and encountering actual people and moving obstacles—would validate the systems' robustness and help identify any performance gaps caused by the sim-to-real transition. This would confirm whether the simulation-trained Method B generalizes to real sensor data and whether Method A's tuned parameters hold under real-world variability.

Future work could also broaden the range of object categories and scenarios considered. This includes incorporating dynamic agents like pedestrians or pets into the detection

TABLE III
COMPREHENSIVE COMPARISON BETWEEN NAJEM ET AL. AND METHOD B.

Method	Precision	Recall	mAP@0.5	mAP@0.5:0.95	Input Pixels	Avg. Inf. Time (PC/RPi5/RPi3, ms)	FPS (RPi5)	Train/Test Samples
Najem et al.	83.7%	82.5%	0.837	0.537	409,600	328.5 / 846.5 / –	1.2	40,000 / 10,000
Method B	**94.9%**	**94.7%**	**0.984**	**0.778**	24,576	**6.2 / 47.8 / 2000**	**21.0**	629,591 / 38,321

framework, beyond static furniture and structural features, as well as exploring long-term autonomy scenarios where the robot operates continuously for extended periods. Prolonged operation may require the system to adapt online—for example, through incremental learning or self-tuning—to maintain high performance as conditions or sensor characteristics change over time.

Finally, given the promising results of a spiking neural network-based LiDAR approach in [12], an intriguing avenue is to explore neuromorphic hardware implementations of the learned method. Deploying Method B's LiDAR processing on event-driven processors or low-power FPGAs could drastically reduce power consumption and latency, pushing the limits of energy-efficient, always-on 2D LiDAR perception. This direction complements the privacy-preserving nature of LiDAR and could enable continuous sensing in resource-constrained, wearable, or long-deployment scenarios without requiring conventional high-power GPUs.

Beyond these specific performance results, the outcomes presented here also suggest that 2D LiDAR, conventionally regarded as a lower-dimensional and limited sensor, can deliver semantic perception capabilities comparable to richer modalities such as cameras or 3D LiDAR, while preserving privacy. This finding supports a broader perspective in which simple, low-cost, and privacy-respecting sensors may be sufficient for robust indoor perception tasks, potentially expanding the accessibility and societal acceptance of service robots in sensitive environments.

In summary, this comparative study demonstrates that 2D LiDAR—harnessed through both classical geometric techniques and modern deep learning—is a viable and effective sensor modality for indoor robotic perception. Method A and Method B offer complementary strengths, and together they highlight that camera-free, LiDAR-centric solutions can achieve accurate, real-time understanding of indoor environments while respecting privacy. We hope these findings inspire the development of more privacy-preserving, computationally efficient LiDAR-based perception systems in the next generation of autonomous service robots.

C. Lessons Learned

The comparative evaluation in this study highlights a broader lesson for the robotics community: there is often a fundamental trade-off between classical, model-driven methods and modern learning-based methods. Simpler geometry-based pipelines can deliver remarkable performance in constrained domains with minimal data and computational resources, while providing high interpretability and ease of deployment. In contrast, deep learning–based approaches can generalize further and achieve higher overall performance, given sufficient data and computing resources, but at the expense of transparency and training complexity. This work illustrates that evaluating two fundamentally different paradigms on a common task can reveal complementary strengths and weaknesses, encouraging future systems that combine both. Furthermore, these results demonstrate the feasibility of privacy-preserving perception using 2D LiDAR alone, which is increasingly valuable for robots operating around people in sensitive indoor environments.

REFERENCES

[1] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining*, 1996, pp. 226–231.

[2] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line nonlinear/non-gaussian bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, 2002.

[3] M. Joho, J. Horn, and K. Konolige, "2d laser scan matching with feature extraction," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2006, pp. 3020–3025.

[4] T. Kanda, T. Hirano, H. Ono, and Y. Nishida, "A new classification method for 2-d laser scan data using support vector machine," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2002, pp. 297–302.

[5] L. Spinello, R. Siegwart, and S. Thrun, "A mobile robot system for fast, accurate, and autonomous laser-based object recognition," *J. Field Robotics*, vol. 28, no. 6, pp. 940–962, 2011.

[6] J. Borenstein, H. R. Everett, and L. Feng, *Where am I? Sensors and Methods for Mobile Robot Positioning*. University of Michigan, 1996.

[7] A. Martinelli, "The use of the compass in mobile robot localization and mapping," *Robotics Auton. Syst.*, vol. 60, no. 9, pp. 1261–1270, 2012.

[8] D. Hähnel, W. Burgard, and S. Thrun, "Learning compact 3d models of indoor environments with a mobile robot," *Robotics Auton. Syst.*, vol. 44, no. 2–3, pp. 205–217, 2003.

[9] Z. Li, C. Lv, W. Li, and J. Liang, "Real-time semantic mapping for mobile robots in dynamic indoor environments," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 4, pp. 1957–1969, Oct. 2021.

[10] C. Stachniss, G. Grisetti, and W. Burgard, "An efficient approach to scan matching for mobile robots," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2007, pp. 410–415.

[11] A. Najem, L. Kuiper, T. Jansen, and F. N. Martins, "Object classification using 2d-lidar and yolo for robot navigation," Master's thesis, Hanze Univ. of Applied Sciences, Groningen, Netherlands, 2024.

[12] L. A. Fagundes, A. G. Caldeira, M. B. Quemelli, F. N. Martins, and A. S. Brandão, "Analytical formalism for data representation and object detection with 2d lidar: application in mobile robotics," *Sensors*, vol. 24, no. 7, p. 2284, 2024.

[13] M. M. Zarrar, Q. Weng, B. Yerjan, A. Soyyigit, and H. Yun, "Tinyl-idarnet: 2d lidar-based end-to-end deep learning model for f1tenth autonomous racing," arXiv:2410.07447 [cs.RO], Oct. 2024.

[14] B. Kaleci, K. Turgut, and H. Dutağacı, "2dlasernet: A deep learning architecture on 2d laser scans for semantic classification of mobile robot locations," *Eng. Sci. Technol., Int. J.*, vol. 28, p. 101027, 2021.

[15] A. Seçkin, "Pedestrian and mobile robot detection with 2d lidar," *Eur. J. Sci. Technol.*, vol. 23, pp. 583–588, 2021.