LOADS: LiDAR-based Privacy-Preserving Queue Monitoring and Analysis

Saisricharan Malkireddy, **Sumedh Kane**, **Sourimitra Medepalli**, **Satvik Racharla**,
Bharg Barot, **Christian Badolato**, **Roberto Yus**
{s178, sumedhk1, s190, yv04378, bhargvb1, cbad1, ryus}@umbc.edu
University of Maryland, Baltimore County, USA

***Abstract*—Long queues in retail and public environments can frustrate customers and negatively impact user experiences. Traditional camera-based monitoring systems are effective in analyzing queues, however, the potential for identification raises privacy concerns. Other queue-counting methods (such as WiFi or RFID) depend on user-carried devices or tags. In contrast, LiDAR sensors strictly measure distances and angles, which drastically reduces privacy risks and does not require users to carry specialized hardware. We present LOADS, an end-to-end, single-sensor, LiDAR-based IoT solution for queue-occupancy and wait-time estimation. LOADS incorporates Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) to provide a robust method of accurately separating people from noise in real time. We employ a SARIMAX model to predict future queue lengths from historical data stored in a time-series database. A web interface shows real-time and historical queue information, enabling users to make informed decisions. We demonstrate the feasibility of LOADS in practical retail and conference scenarios, highlighting its privacy-preserving nature, accurate crowd estimation, and simple deployment.**

I. Introduction

The rapid growth of the Internet of Things (IoT) has created opportunities to improve customer experiences in public spaces such as retail shops, cafes, and event venues. One such opportunity is in queue management. Prolonged queue times can diminish customer satisfaction and reduce throughput [1]. Although camera-based systems have been widely studied for queue monitoring, they can capture identifiable features, such as faces, and raise serious privacy concerns [2]. Non-camera solutions using WiFi or RFID can mitigate some privacy issues, but they require users to carry mobile devices or radio tags and may suffer from interference of multiple devices in dense indoor spaces [3]. In contrast, LiDAR sensors emit laser pulses which are capable of measuring distances and angles without capturing private information such as facial images or demographic information. This allows for effective queue-counting while preserving user privacy and avoiding the need for users to carry specific hardware. A comparison of our approach with existing solutions is shown in Table I.

In this demo paper, we present LOADS (**L**iDAR **O**ccupancy **A**nalysis and **D**etection **S**ystem): an end-to-end IoT solution for accurate, real-time queue length measurement and wait-time prediction using a single LiDAR sensor. This sensor produces numerous measurement points in the form of a "point cloud". LOADS combines local data processing and a noise-resilient HDBSCAN clustering algorithm [4] to address the challenges related to distinguishing multiple people from environmental noise or background objects in a tightly packed and dynamically shifting queue. We store this cluster information in an InfluxDB[1] time-series database, enabling efficient archiving, analytics, and retrieval. Furthermore, we employ a Seasonal Autoregressive Integrated Moving Average with eXogenous regressors (SARIMAX)[2] model to forecast queue lengths, thus allowing end-users and venue managers to predict busy intervals. Our system also provides a web-based dashboard for real-time monitoring, historical visualizations, and queue predictions.

TABLE I
Comparison of queue-counting technologies.

Attribute	Camera-based	WiFi/RFID-based	LOADS (LiDAR-based)
Privacy	Poor	Good	Excellent
Requires user participation	No	Yes (device required)	No
Deployment cost	Medium	Low-Medium	Medium
Environmental robustness	Light-sensitive	Multiple Device issues	Unaffected

By showcasing LOADS in a live conference setting, we hope to emphasize three main contributions:

- **Privacy-preserving queue analytics:** Unlike cameras-based approaches, our LiDAR implementation does not collect any identifiable features.
- **Single-sensor design:** A single 360-degree LiDAR sensor is capable of covering the entire queue zone, which reduces deployment cost and complexity.
- **Occupancy Forecasting:** Our SARIMAX-assisted model provides short-term queue length predictions, enabling proactive decision-making.

II. System Architecture

In this section, we provide an overview of the components and functions of LOADS:

Data Observation: LiDAR Sensor. The core sensing component of our system is a 360-degree LiDAR sensor. LiDAR

[1]https://www.influxdata.com
[2]https://www.statsmodels.org/dev/

sensors operate by emitting rapid pulses of light and measuring the time it takes for the light to return after reflecting off an object. This process enables the sensor to gather precise measurements, which include the following.

- *Distance*: The distance between the sensor and objects in its range.
- *Angle*: The angular position of each detected object with respect to the sensor.
- *Signal Strength*: The level of intensity of the reflected light, which can be used to determine the surface properties of the detected object.

LOADS, at the moment, focuses on distance measurements, which allow it to identify the presence of individuals in the sensor's field of detection. LOADS uses an RPLiDAR A1 sensor with a 12-meter effective range which operates with a 5.5-Hz rotation speed, providing a reliable 360-degree scan.

Data Acquisition: Raspberry Pi. LOADS uses a Raspberry Pi 4 to collect data from the RPLiDAR sensor, which is directly connected through a UART serial interface. We use the RPLiDAR Python SDK[3] to seamlessly collect the observed data at 30-second intervals. This interval was chosen to balance real-time responsiveness and resource efficiency. Queues in retail environments are not expected to exhibit rapid changes within a shorter interval, however, longer intervals risk missing short-term changes in queue dynamics. The 30-second interval ensures timely updates while maintaining system efficiency and preserving computational resources.

The observed data consists of points, each representing the reflection of a LiDAR pulse off an object in the sensor's field of view, and is characterized by its distance from the sensor, its angle in relation to the sensor, and its signal strength (which can indicate the surface properties of the object). Together, the points within an observation form a "point cloud" representing the spatial layout of objects in the sensor's environment. In the context of LOADS, these points correspond to individuals and objects within the monitored area, which are subsequently processed to estimate queue lengths and waiting times. These scans are then processed locally on the Raspberry Pi to prepare them for further analysis and to prevent raw data from being transmitted across the network.

Furthermore, we have designed, using SOLIDWORKS[4], and 3D printed a prototype encasing for the data observation and acquisition components. The case holds the Raspberry Pi and the LiDAR sensor and allows us to angle the LiDAR to whichever angle is required in the deployment. This is done using two hinges and a screw that holds the sensor at the desired angle. The case also takes into account airflow, having openings on the side to deal with overheating experienced on the Raspberry Pi due to local processing.

Data Processing: HDBSCAN Clustering Algorithm. To robustly count the number of people in a queue, LOADS uses HDBSCAN [4]. HDBSCAN extends DBSCAN by converting the clustering procedure into a hierarchical tree before extracting clusters based on density persistence. It is well-suited for noisy spatial data as it:

- Identifies dense regions of points while marking outliers as noise.
- Requires minimal assumptions about cluster shape.
- Automatically determines the number of clusters.

In our setup, each data snapshot is clustered via HDBSCAN with parameters configured to reflect the typical spacing among people in a given queue. For instance, *min_cluster_size* is set based on the expected LiDAR point density on a human body at the chosen distance. The algorithm filters out spurious points and merges points corresponding to each person. We then label each cluster as a single individual, effectively estimating the queue length (see Fig. 1). This approach is naturally resilient to random noise and partially occluded views since the algorithm can handle clusters of varying shapes.

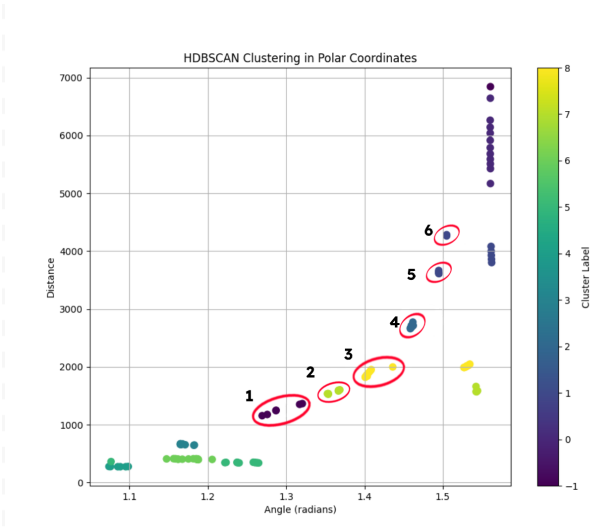

Fig. 1. HDBSCAN Clustering. Each number represents one person.

Data Transmission: MQTT Broker. We utilize the MQTT (Message Queuing Telemetry Transport) protocol[5] to facilitate communication between the networked system components of LOADS. MQTT is a publish-subscribe-based messaging protocol designed for lightweight machine-to-machine communication. Within MQTT, a broker, residing on a server (or a laptop during our demonstration) manages communication between the LOADS publishers (Raspberry Pi) and subscribers (the React front-end and timeseries database). To ensure data privacy and integrity, we use the TLS-encrypted version of MQTT (sometimes known as MQTTS). We selected MQTT instead of direct communication due to its low latency and scalability. An initial analysis shows that 20-30 sensors would be required to cover the main retail services on our campus which can be done efficiently with MQTT.

Once processed by the Raspberry Pi, the observation data is published via MQTT to a designated, per-sensor topic. For

[3]https://github.com/Lupin3000/RPLiDAR
[4]https://www.solidworks.com
[5]https://mqtt.org/

each observation, this data contains the number of detected clusters (individuals) in the observation along with the timestamp at which the observation was performed.

Data Storage: Timeseries Database. We use InfluxDB for storing and managing historical data observations. InfluxDB is a time series database known for its high performance, scalability, and advanced querying capabilities. This makes it particularly well-suited for systems like LOADS, which collects time-stamped data and requires efficient storage for historical analyses. The built-in compression mechanisms help minimize storage costs, even with continuous data collection from sensors. The InfluxDB database receives data directly from the Raspberry Pi using the described MQTT protocol.

Predictive Analytics: SARIMAX Algorithm. To extend the functionality of LOADS, we use a Seasonal Autoregressive Integrated Moving Average with eXogenous regressors model (SARIMAX) algorithm for predicting future queue lengths. SARIMAX analyzes historical data stored in InfluxDB to identify patterns and seasonal trends, such as daily or weekly peaks, and generates short-term forecasts by combining historical data with SARIMAX's forecasting capabilities; this helps users make proactive decision-making beyond real-time monitoring of wait times.

Queue Visualization: React Front-end. LOADS provides an engaging graphical user interface (GUI) developed in ReactJS. This GUI (see Fig. 2) is hosted as a publicly-accessible web application designed for accessibility and ease of use. The LOADS GUI allows users to view the real-time count of individuals in the queue. It also queries the InfluxDB database to provide a visualization of historical data and trends in line occupancy over time; this historical data helps users identify peak hours and plan accordingly.

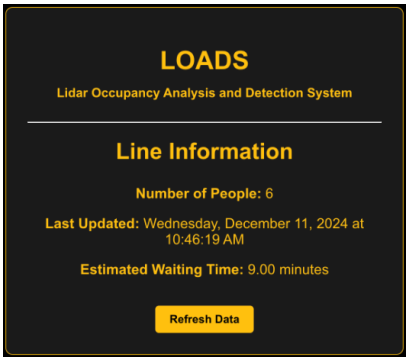

Fig. 2. Sample visualization.

III. Demo Overview

We plan to demonstrate LOADS through three distinct conference-queuing scenarios, which will enable attendees to interact with the system in differing environments (see Fig. 3).

- **Registration Desk**: Structured queues with intersecting exit paths. This demonstrates noise filtering during counter interactions. Real-time data can help attendees choose optimal check-in times and allow organizers to mitigate bottlenecks by opening more counters or shifting staff.
- **Lunch Area**: Assuming a buffet setup, here we also expect structured queues, possibly on both sides of a table. In this scenario, people progress through the line and exit at the front, presenting a scenario that is less complex than the registration desk. Real-time queue monitoring in this scenario might help attendees choose less crowded stations or better times to eat, enhancing their experience and reducing frustration during busy intervals.
- **Demo Zone**: Expected to be the most chaotic scenario, with unstructured crowd movements as attendees move freely in the crowded space. This will validate the system's adaptability in dynamic environments with unstructured queues, which present the largest challenges to cluster tracking. We will use the collected queue data to evaluate attendee engagement.

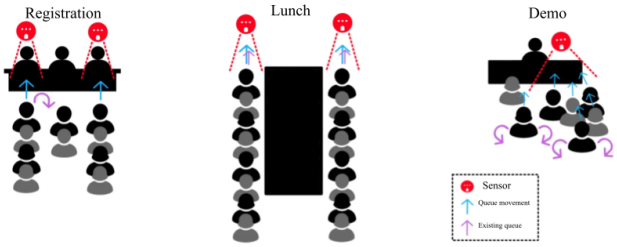

Fig. 3. Demo Scenarios: (1) Registration, (2) Lunch Area, (3) Demo Zone

Implementation: The LiDAR sensor will be mounted on an adjustable stand and will stream processed data via a Raspberry Pi to an interactive dashboard. The 30-second update cycle will enable real-time visualization of:

- Current queue lengths and predicted wait times.
- Historical trends and SARIMAX forecasts.
- Cluster visualization overlays (without raw point clouds).

Attendees can test detection accuracy by forming experimental queues and comparing system outputs with ground truth counts. They will also be able to view aggregated metrics, demonstrating privacy-preserving analytics. This hands-on experience showcases LOADS' operational integrity from sensing to prediction while maintaining anonymity and following privacy-by-design concepts.

References

[1] M. M. Davis and J. Heineke, "Waiting time influences on satisfaction, decisions, and attributed value," *Journal of the Academy of Marketing Science*, vol. 24, no. 4, pp. 338–347, 1996. [Online]. Available: https://link.springer.com/article/10.1177/0092070396244005

[2] H. Zhang, M. Zhou, H. Sun, G. Zhao, J. Qi, J. Wang, and H. Esmaiel, "Que-fi: A wi-fi deep-learning-based queuing people counting," *IEEE Systems Journal*, vol. 15, no. 2, pp. 2926–2937, 2021.

[3] M. De Sanctis, S. D. Domenico, D. Fioravanti, E. B. Abellán, T. Rossi, and E. Cianca, "Rf-based device-free counting of people waiting in line: A modular approach," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 10, pp. 10 471–10 484, 2022.

[4] L. McInnes and J. Healy, "Accelerated hierarchical density based clustering," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, 2017, pp. 33–42.