Pedestrian Detection, Tracking and Path Prediction using Ultrasonic Point Clouds

By

McManamon, Colm Hugh

MSc Robotics Dissertation

School of Engineering Mathematics and Technology
UNIVERSITY OF BRISTOL

& Department of Engineering Design and Mathematics
UNIVERSITY OF THE WEST OF ENGLAND

A MSc dissertation submitted to the University of Bristol and the University of the West of England in accordance with the requirements of the degree of MASTER OF SCIENCE IN ROBOTICS in the Faculty of Engineering.

29th August 2025

Declaration of own work

I declare that the work in this MSc dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

Colm Hugh McManamon 29th August 2025

Ethics statement

"This project did not require ethical review, as determined by my supervisor Dr Amina Hamoud".

Colm Hugh McManamon 29th August 2025

Pedestrian Detection, Tracking and Path Prediction using Ultrasonic Point Clouds

Colm Hugh McManamon
MSc Robotics Dissertation, 2025

***Abstract*—This paper presents a processing pipeline for pedestrian detection, tracking, and path prediction using ultrasonic time-of-flight point clouds. An investigation into whether techniques originally developed for Light Detection and Ranging (LiDAR) can be adapted to acoustic sensing was conducted. The system operates on 2D range–azimuth images generated by the Calyo ultrasonic sensor and its supporting software. The sensor supports 3D output but this work focuses on 2D projections to leverage computer vision techniques to perform the task. Detection is performed via adaptive thresholding and Density-Based Spatial Clustering of Applications with Noise (DBSCAN), followed by multi-target tracking with ID management and strategies for occlusion handling and cluster separation. A Long Short-Term Memory (LSTM) network — a form of recurrent neural network — is trained to predict pedestrian positions one second into the future based on motion history. The system is evaluated on both simulated and real-world data, achieving sub-0.3m accuracy in simulation and demonstrating the feasibility of ultrasonic sensing as a redundant or self-contained perception strategy for robots operating in social environments.**

I. Introduction

A. Motivation

Autonomous and semi-autonomous robotic systems are increasingly expected to operate in environments shared with people, ranging from outdoor contexts such as self-driving vehicles to indoor domains including healthcare facilities, office buildings, and warehouses. In these settings, robots must be able to detect pedestrians, track their motion, and predict their short-term trajectories in order to plan safe navigation paths. Failing to detect or anticipate pedestrian movement can result in unsafe manoeuvres, abrupt stops, or even collisions, all of which undermine safety and erode user trust.

Recent work has shown that LiDAR point clouds can be applied for pedestrian detection, multi-object tracking, and short-term trajectory prediction [1]. By combining LiDAR's dense spatial resolution with machine learning methods, accurate and reliable pedestrian forecasts were achieved. However, LiDAR hardware remains expensive and bulky, limiting its practicality for indoor spaces. Cameras, while more accessible, are susceptible to fog, rain, and poor lighting, which can critically reduce their effectiveness.

These limitations motivate the exploration for alternative sensing technologies. Ultrasonic time-of-flight sensors offer an attractive option: they are inexpensive, compact, and resilient to these weather conditions. Crucially, they have the potential to act as a redundancy layer when vision-based systems degrade or fail. This dissertation therefore examines whether techniques originally developed for LiDAR-based pedestrian perception [1] can be adapted to ultrasonic point clouds, providing both a low-cost substitute for LiDAR and a fallback sensing approach when cameras are impaired.

Fig. 1: The Calyo pulse sensor used in this experiment.

B. Aims

The aims of this dissertation are:

- To adapt LiDAR-based pedestrian perception methods for ultrasonic point cloud data, and to design a complete system for detection, tracking, and short-term trajectory prediction using low-cost hardware.
- To validate the system in indoor environments, while demonstrating its potential for wider use in outdoor settings as either a primary low-cost solution or a redundant backup in safety-critical applications.

C. Objectives

To meet the above aims, the following technical objectives were defined:

1) **Experimental setup**
   Build and configure an ultrasonic time-of-flight platform to generate range–azimuth point cloud data in an indoor environment.
2) **Detection**
   Apply thresholding and clustering methods to segment raw point clouds into pedestrian candidates.

3) **Tracking**
   Develop a multi-target tracking scheme with occlusion handling and stable ID management.
4) **Dataset**
   Construct training, validation, and test sets from both simulated trajectories and real-world sensor data.
5) **Prediction**
   Train a Long Short-Term Memory (LSTM) network to forecast pedestrian positions one second ahead from motion histories.
6) **Evaluation**
   Assess system performance using metrics such as root mean square error (RMSE) and hit-rate, and compare real sensor obtained data with simulated data.

D. Key Contributions

This dissertation is the first to demonstrate a processing pipeline for pedestrian detection, tracking, and trajectory prediction using ultrasonic point cloud data. It adapts methods originally developed for LiDAR-based perception [1], showing that although ultrasonic point clouds have lower resolution, they can support the same fundamental stages of perception. Several refinements to detection and tracking are introduced—including in-frame cluster consolidation, ID protection radii, and multi-ID confirmation strategies—which enable stable pedestrian tracks to be maintained in noisy and fragmented ultrasonic data, a capability not achieved in prior work.

The project evaluates this pipeline on both simulated and real-world ultrasonic datasets, highlighting the gap between idealised trajectories and those extracted from sensor data. This comparison reveals practical challenges of adapting LiDAR-inspired methods to ultrasonic sensing, such as centroid drift, velocity distortion, and fragmented tracks. By framing ultrasonic sensing as both a low-cost alternative and a redundancy layer within the wider perception system, the work supports future development of robust, safety-critical robotic systems.

II. Background

Clustering and Detection in Point Cloud Data

Detecting pedestrians in 3D point cloud data usually starts by grouping nearby points that likely belong to the same object. In LiDAR or radar sensor systems, a conventional method is DBSCAN. DBSCAN is a clustering method that groups points based on proximity to each other, using two simple settings: the distance allowed between points (Eps) and the minimum number of points to form a group (minPts) [2]. It's especially effective because it can find groups of any shape.

This makes DBSCAN a strong candidate for real-time pedestrian detection. For example, Santos et al. [6] used it with radar point cloud data, and once the pedestrian points were clustered, they used the centroid as the tracked location for each one.

In practice, DBSCAN can filter out random or isolated points automatically, which is useful in real-time environments. However, it does have some drawbacks. In crowded scenes, pedestrians who are close together might get grouped into a single cluster, especially if the point cloud resolution is low. Wang et al. [2] noted this issue, observing that DBSCAN sometimes merges multiple people into one when they're standing near each other.

To fix this, they used Kernel Density Estimation (KDE), which analyses how densely packed the points are. Since each person usually has a dense area around their torso, KDE can help find these peaks and separate merged clusters into individual pedestrians [2]. This made detection much more accurate in busy scenes.

Other researchers have tried machine learning for this task. Na and Park [1], for instance, used a small neural network (an autoencoder) to learn how pedestrian shapes appear in 3D LiDAR data. After that, they applied a simple connected-components method to pull out each pedestrian individually. This combination worked well, even with sparse data.

In short, clustering methods like DBSCAN offer a reliable starting point for detecting people in point clouds. Based on past research [2], crowded situations may need extra refinement. However, studies suggest that DBSCAN alone is often sufficient for accurate detection in lower-resolution or resource-constrained sensing systems, making it a strong candidate for adaptation to ultrasonic data.

Multi-Target Tracking in Point Clouds

Tracking involves assigning an identifier to each person and maintaining that identity as they move, frame by frame. In most point cloud tracking systems, this can be done by treating each detected pedestrian as a track and updating their estimated position as new data comes in.

A common method used in the literature is the Kalman Filter (KF), a mathematical tool that predicts where an object is likely to be next based on its current position and velocity, while accounting for possible noise in the measurements. Many systems use it to smooth out pedestrian movement and provide short-term predictions between frames. For example, Santos et al. applied DBSCAN to radar point clouds and then used a KF to follow the motion of each pedestrian by updating the track from frame to frame [6].

Shen et al. designed a system with a low-cost 2D LiDAR sensor where the KF helps to stabilise detection by predicting where a person is likely to appear next. This prediction assists the clustering algorithm, even when the sensor momentarily loses track of a person due to noisy data [5].

Another important step in multi-person tracking is data association. This is typically done using either a nearest-neighbour method or a more advanced Hungarian algorithm, which match predicted positions with new detections based on how close they are. This process helps ensure that each pedestrian keeps the same ID over time, even as they change position [2].

Some studies also incorporate appearance-based features like shape or surface reflectance to help distinguish people in crowded scenes. Others use multiple motion models, such as switching between constant-speed and stop-and-go behaviours, to improve prediction accuracy when people change direction or speed suddenly [1].

While KFs are widely used to stabilise trajectories [5][6], some researchers have shown that simpler nearest-neighbour association strategies can be effective in systems with high frame rates [2]. This reflects a design trade-off that becomes particularly relevant when dealing with lower-resolution sensors, where computational simplicity can outweigh the benefits of predictive filtering.

Trajectory Prediction with LSTM Networks

Predicting a pedestrian's trajectory is needed for systems like autonomous robots. In recent years, researchers have applied deep learning networks, particularly Long Short-Term Memory (LSTM) networks, because of their ability to learn from past patterns of movement [3], [4].

Unlike basic models that assume constant speed or direction, LSTM networks can learn more complex patterns by analysing how people move over time. LSTMs are a type of recurrent neural network, and are designed to handle sequences—like the series of positions a person takes as they move. They're particularly effective at remembering important movement trends while ignoring less relevant details. Sighencea et al. [3] showed that these networks can learn from real-world pedestrian paths and predict future movement with good accuracy.

One of the most influential models in this space is the "Social LSTM" by Alahi et al. [4], where each pedestrian is tracked with its own LSTM, but nearby pedestrians share some information via a "social pooling" layer. This lets the system learn how people adjust their path when others are nearby—for example, avoiding collisions or walking in groups.

These systems rely on the LSTM's ability to process sequences of past positions [3]. These models have consistently shown strong performance in predicting short-term movement and can adapt to different walking patterns or crowded situations.

While most prior research has focused on LiDAR and radar systems [1], [2], [4], the potential for LSTM-based trajectory models to work effectively with ultrasonic point clouds has not yet been explored. This represents a novel research direction, since it combines an established learning architecture with a sensing modality that has not previously been used in this context.

Applying Computer Vision (CV) Techniques to 2D Sensor-Generated Images

Some recent work has projected complex 3D sensor data (e.g., LiDAR point clouds) into 2D image representations, allowing researchers to leverage fast, mature image-space algorithms for detection and tracking [11,16]. This practice has been reported in LiDAR studies for efficiency and real-time performance [2]. In effect, treating the 2D outputs of LiDAR or ultrasound as conventional images—and using classical steps such as grayscale conversion, thresholding, and morphology—isolates key features and improves throughput [16,11,2].

Grayscale conversion and its benefits

Many systems convert frames to grayscale (0–255) so that downstream routines operate on a single intensity channel. Vision primitives such as OpenCV's thresholding often assume or require grayscale input [9]. Processing one channel instead of three reduces both memory usage and arithmetic operations by approximately two-thirds, while still preserving the salient structure for edge, blob, and contour analysis [8]. This choice also matches the data: LiDAR intensity maps, projected depth, and ultrasound all encode signal strength rather than colour, so grayscale preserves the meaningful content and streamlines subsequent steps [8,9].

Isolating features with thresholding and morphology

After grayscale conversion, frames can be segmented via thresholding to obtain a binary foreground/background mask—an established first step when intensity contrast is informative [9,15]. These masks can then be refined with morphological operations (erosion, dilation, opening, closing) to remove speckle, reconnect thin gaps, and regularise object shapes; these operators are a staple of practical CV pipelines [10]. In ultrasound specifically, combining Otsu thresholding with morphology (or active-contour refinement) improves boundary definition in noisy imagery, producing stable blobs for tracking [15,10].

Leveraging 2D projections for LiDAR data (case examples)

Working directly on 3D point clouds is computationally heavy; many systems therefore project point clouds into 2D grids/maps and apply image-space detectors to meet real-time constraints [11]. Industry guidance also notes that most mature object-identification algorithms are 2D, so structured LiDAR images (e.g., reflectivity frames) can plug into those pipelines directly [16]. Concretely, researchers have formed one-channel grayscale depth images from point-cloud projections [12], and generated 2D 16-bit grayscale LiDAR images to drive pedestrian recognition with deep models [13]. These patterns support the design choice to operate in 2D for efficiency while retaining task-relevant structure [11,16,2,12,13].

Applying CV techniques to ultrasound images (case examples)

Ultrasound frames are inherently grayscale, making them suitable for standard CV. With appropriate preprocessing, classical methods transfer effectively: optical-flow tracking has been adapted to ultrasound sequences to follow tools (e.g., needles) across frames, sometimes paired with Hough-based geometric detection to localise tips [14]. For segmentation, Otsu thresholding plus morphological cleanup (or active contours) delineates structures such as ovarian follicles in noisy scans, improving countability and reducing manual workload [15]. These studies reinforce the general principle that regardless

of source, 2D sensor images can be enhanced with the same CV toolkit—grayscale conversion, thresholding, morphology—and then passed to trackers or higher-level detectors [14,15,10,9].

III. Research Methodology

A. Experimental Setup and Sensor Configurations

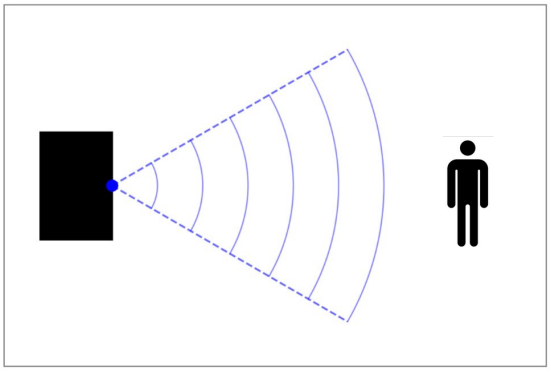


Fig. 2: Top view of the experimental setup showing ultrasonic sensor placement and coverage area

The experimental setup is illustrated in Fig. 2. A stool was used to represent the robotic platform, with an ultrasonic sensor mounted on its top-right edge. The sensor emits ultrasonic waves toward the detection area and generates a point cloud relative to its position. The room height was approximately 2.41 m, and the sensor was positioned 0.595 m above the floor.

B. Image Acquisition

TABLE I: PrecisionImager2d-cuda Configuration

Parameter	**Value**
Device Reader	`DeviceReader`
Serial Number	any
Cycles per Frame	20
Pixel Size	0.010 m
Output Writer	`PNGWriter`
Processor Stage	log_conversion
Algorithm	precision
Dimensions	2
Field of View x (min, max)	$[-3.0, 3.0]$ m
Field of View y (min, max)	$[0.1, 3.6]$ m
Range Window (min, max, step)	$[0.8, 10.0]$ m, 0.1 m
Azimuth (min, max, steps)	$[-1.57, 1.57]$ rad, 150
ROI Radius	10.0 m
Normalization Mode	norm2max
Normalization Value	0.02
dB Conversion Enabled	true
dB Scale	–20.0 dB
CFAR Enabled	true
CFAR Ratio	1.0

Image acquisition was managed using the `SENSUS Viewer`, with frames exported through the `PNGWriter` setting as grayscale images (0–255) to be later processed on Google Colab. The sensor produced outputs at 4 Hz, with frame rate linked directly to pixel resolution. A pixel size of 0.010 m was selected as a trade-off: smaller values preserved spatial fidelity but slowed acquisition, while larger sizes (0.015–0.020 m) yielded faster capture (up to 6 Hz) but degraded pedestrian blobs, breaking track continuity. The field of view was defined as $x \in [-3.0, 3.0]$ m, $y \in [0.1, 3.6]$ m, with an azimuth sweep of $[-1.57, 1.57]$ rad over 150 steps, chosen to align with the walking area of the test environment. These bounds maximised pedestrian coverage while limiting background clutter.

Runtime parameters were also critical for producing stable images. Intensity normalisation (`norm2max`, value 0.02) prevented saturation and ensured consistency across frames, decibel scaling (-20 dB) reduced low-energy noise, and CFAR thresholding (ratio 1.0) suppressed static clutter while preserving moving pedestrian returns. Together, these acquisition settings balanced spatial detail with temporal resolution, yielding clean inputs for the subsequent preprocessing and trajectory prediction pipeline.

C. Preprocessing and Tracking

As shown in Figure 3, the sensor scans the field of view, within this region, reflections from objects are captured and aggregated into intensity maps.
Figure 4 presents a representative raw output frame, where clusters of brighter points correspond to objects in the environment. These raw detections form the starting point for the preprocessing pipeline.

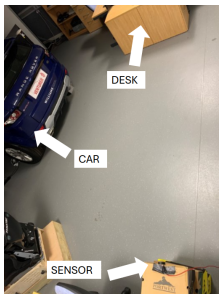


Fig. 3: Sensor scanning area

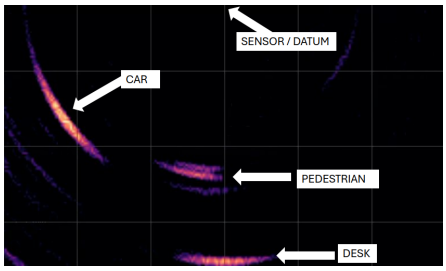
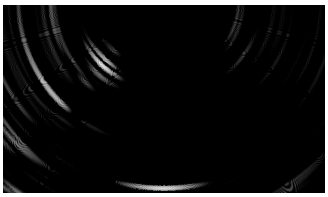


Fig. 4: Example sensor output frame from the `imager_2d` pipeline



(a) Excess noise in close range

(b) Reflections from nearby car

(c) Merged clusters

Fig. 5: Examples of common challenges in raw sensor output: (a) close-range noise, (Top of image) (b) Reflections (Left side of image) (c) cluster merging (bottom of image).

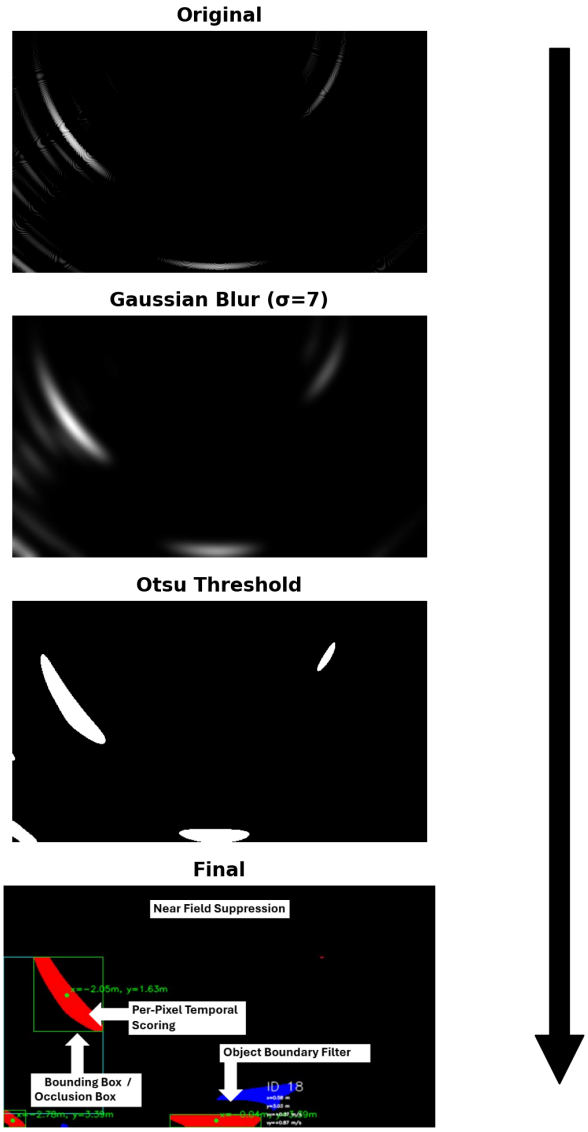


Fig. 6: Processing pipeline: original input, Gaussian blur, Otsu threshold, and final filtered output.

Common challenges

Fig. 5 shows typical issues that affect the preprocessing: (a) close-range noise appears as static that obscures true targets, (b) strong reflections from nearby cars create false pedestrian IDs, and (c) cluster merging causes pedestrians to be absorbed into neighbouring detections. These effects require additional corrective steps to ensure robust tracking.

Denoising and binarisation

As illustrated in Fig. 6, each frame is first smoothed with a Gaussian filter (σ=7 px) to suppress speckle, then binarised by Otsu's method. The result is a clean foreground/background mask that feeds subsequent processing. The final panel also highlights the effect of near-field suppression, where the bright band at the top of the image is removed to prevent close-range noise overwhelming the scene.

Near-field suppression (close-range noise)

Pixels corresponding to the first 1.0 m of range are zeroed in the binary mask (a fixed offset from the sensor), removing bright splash directly in front of the transducer. This specifically targets the behaviour shown in final image of Fig. 6 and stabilises downstream clustering.

PaTS: Per-pixel temporal scoring

On the 600×350 grid, each frame (Δt=250 ms) updates a per-pixel score $s(x, y)\in[0, 10]$ from the Otsu binary mask:

$$s \leftarrow \begin{cases} \min(10,\ s+1), & \text{if mask} = 1 \text{ (foreground/white)} \\ \max(0,\ s-1), & \text{if mask} = 0 \text{ (background/black)}, \end{cases}$$

with a cluster-aware hold ($\geq$60% of a previously stable cluster remains active) that skips the decrement. Pixels with $s \geq 5$ are treated as persistent structure (*red band*); $1 \leq s \leq 4$ indicates transient activity (*blue band*); $s = 0$ is suppressed. An ID-protection mask caps $s \leq 4$ within a 0.5 m radius of confirmed tracks to prevent "red creep." This across-frames accumulation promotes stationary objects to red while movers remain blue, supplying stable seeds for DBSCAN and subsequent tracking. Fig. 6 illustrates these red and blue shifts across frames.

Map referencing placeholder

The first four frames act as a warm-up period allowing persistent reflections to settle as background before confirming any pedestrian IDs. Conceptually, this serves as a placeholder for a deployed system that would use a prebuilt map to reference returns at known coordinates. The static scene is

updated sequentially. After warm-up, only clusters that exhibit motion over frames and remain in the blue band (score ≤ 4) are eligible to become tracks.

DBSCAN clustering of persistent structure (red)
Stable red pixels are converted to metric coordinates using the image–to–metres scale, then grouped with DBSCAN (eps 0.15 m, min_samples 15). Tiny or elongated artefacts are rejected using an oriented extent test (≥ 0.3 m for static objects). Each retained cluster yields a tight axis-aligned bounding box and a centroid shown in final image of Fig. 6. This step provides reliable geometry for later masking and association.

Proportional occlusion boxes (nearest-edge aligned)
as shown Fig. 6, after DBSCAN defines a stable cluster and its bounding box (x_0, x_1, y_0, y_1), an occlusion rectangle is grown so its area is $\approx 3\times$ the bbox area (width/height multiplied by $\sqrt{3}$). The growth direction is chosen relative to the image midline at $x = \lfloor (W-1)/2 \rfloor$:

Let $x_c \lfloor (W-1)/2 \rfloor$. Grow the occlusion box as

$$\text{direction} = \begin{cases} \text{rightward to } x = W-1, & \text{if } x_0 > x_c, \\ \text{leftward to } x = 0, & \text{if } x_1 < x_c, \\ \text{downward from } y_0, & \text{otherwise.} \end{cases}$$

Widths/heights are clipped to the remaining image limits (width limited by distance to the nearest side; height limited by $H - y_0$) and never smaller than the original bbox. Pixels inside this rectangle are masked *before* candidate (*blue*) selection and score updates, preventing spurious moving detections immediately "behind" strong reflectors (cf. Fig. 5b). Nested suppression avoids redundant occlusion rectangles when objects overlap.

Object Boundary Filter
Candidate blue pixels are suppressed in a small margin around red structure to avoid boundary bleed-through (a black rim of $\sim$10–20 px; 20 px used). DBSCAN is then applied again to blue candidates (eps 0.15 m, min_samples 15), with a relaxed oriented extent threshold for movers (≥ 0.2 m). If multiple blue clusters fall within 1.0 m, they are consolidated by keeping the most supported/extended cluster. This guards against the merged-cluster failure mode in Fig. 5(c), preserving separability between a moving pedestrian and nearby static structure.

Association and ID management
Candidate detections arise from the *blue band* (per-pixel scores in $[1, 4]$ after temporal scoring across frames). Before data association, detections in close proximity are consolidated: groups whose centroids lie within 1.0 m are merged, and the *representative* is the largest cluster by point count and spatial extent. This choice typically corresponds to the strongest, most stable return when multiple fragments reflect the same object. A nearest-neighbour association then links consolidated detections to existing tracks using a gating window of 0.00–1.20 m per frame. A candidate becomes a confirmed track after 2 consecutive hits; tracks are retired after 4 consecutive misses. IDs are assigned only on confirmation (delayed numbering) to avoid label churn.
Track states are defined at the cluster centroid in metric coordinates (x_k, y_k). Per-axis velocity is estimated online by a backward finite difference at 4 Hz ($\Delta t = 0.25$ s):

$$v_x[k] \approx \frac{x_k - x_{k-1}}{\Delta t},$$
$$v_y[k] \approx \frac{y_k - y_{k-1}}{\Delta t}.$$
$$v[k] \;=\; \sqrt{v_x[k]^2 + v_y[k]^2}\,.$$

To keep movers from being absorbed into static structure, an ID-protection mask caps nearby pixel scores to ≤ 4 within a 0.5 m radius of each confirmed track, ensuring those regions remain in the blue band. Together, consolidation, gated association, and protection yield stable IDs and centroid-based kinematics for pedestrians despite clutter and fragmented returns. As shown in Fig. 6, this mechanism allowed IDs such as ID 18 to persist independently without merging into a nearby object, thanks to its lower per-pixel score and the boundary filter.

D. Dataset Structure

TABLE II: Dataset Showing Input Features and Outputs +1 s Position

t (s)	**ID**	x (m)	y (m)	v_x (m/s)	v_y (m/s)	x_{+1s}	y_{+1s}
0.25	1	2.43	2.73	-2.29	0.27	0.18	2.94
0.50	1	1.86	2.80	-2.27	0.27	-0.38	2.96
0.75	1	1.29	2.85	-2.26	0.20	-0.94	2.98
1.00	1	0.73	2.89	-2.25	0.16	-1.50	2.96
1.25	1	0.18	2.94	-2.22	0.19	-2.06	2.94
1.50	1	-0.38	2.96	-2.23	0.10	-2.62	2.90

The dataset was organized to support short-term trajectory prediction based on sequential motion patterns. As shown in Table II, each row represents a single observation captured at 4 Hz, matching the frame rate of the `PNGWriter`. This sampling rate ensures high temporal resolution for accurately capturing pedestrian movement.
Each observation includes the pedestrian's current position (x, y), velocity components (v_x, v_y), and the corresponding future position after one second (x_{+1s}, y_{+1s}). These future positions serve as ground-truth labels for supervised learning.

To maintain temporal continuity, all observations for a given pedestrian are grouped into sequences ordered by time. This structure allows the learning model to leverage historical context when predicting the next position, enabling it to capture spatial-temporal patterns such as direction changes, speed variations, and general movement trends.

D. Dataset Generation

To generate realistic trajectory data, I designed a simulation representing the operating area as a two-dimensional grid of area x = (-3m to 3m), y = (0.1m to 3.6m). Virtual pedestrians were introduced at random boundary positions, emulating entries from different directions. Each pedestrian was assigned a random heading and a walking speed between 0.5 m/s and 2.0 m/s. Natural variability was added using Gaussian noise on speed and heading, with a small probability (1%) of sharp turns up to $\pm 45°$, and occasional pauses (0.5%) lasting 2–20 seconds to reflect realistic behaviour.

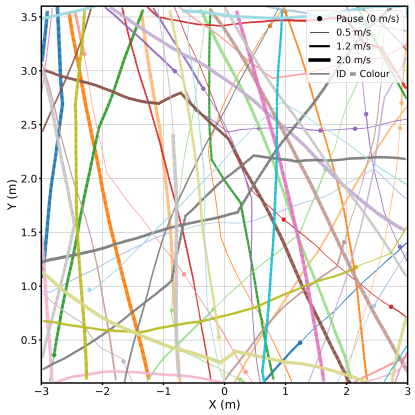


Fig. 7: Visualisation of pedestrian trajectory simulation (5 minutes).

The simulation ran at 4 Hz, matching the frame rate. At each time-step, positions and velocities were updated and recorded; pedestrians leaving the area were removed, and new ones spawned every 2–8 seconds. This produced continuous, dynamic trajectories with periods of movement and rest. To validate if the trajectories were realistic, I visualized the first five minutes (Fig. 7), where each pedestrian is shown in a unique colour, line thickness encodes speed (thin: 0.5 m/s, thick: 2.0 m/s), and pauses appear as dots. These patterns—straight paths, occasional turns, and stops—closely resemble real pedestrian motion, ensuring suitability for trajectory prediction.

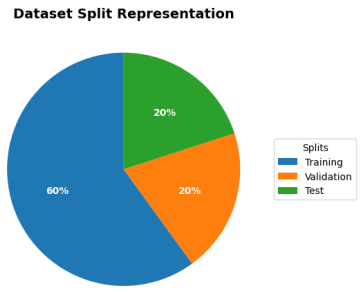


Fig. 8: Dataset Distribution

Training Set
This dataset spans 90 minutes of continuous pedestrian activity and comprises 4,599 individual pedestrian sequences. Sequence lengths vary based on pedestrian movement patterns and exit times within the defined area.

Validation Set
This consists of 30 minutes of data with 1,137 pedestrian sequences and adheres to the same structure and feature representation as the training set. It was primarily used for hyperparameter tuning and to monitor overfitting during model training.

Test Set
This was constructed from real-world ultrasonic sensor readings. For each frame, the raw sensor point cloud was preprocessed to extract pedestrian positions and group them into individual sequences. This dataset provides an unbiased evaluation of the trained model under real operating conditions and contains 180 unique pedestrian IDs. The distribution of these sets is shown in Fig. 8 and is segmented by time.

E. Model Training and Hyper-parameters

A Long Short-Term Memory (LSTM) model was chosen due to its capability to capture temporal dependencies inherent in pedestrian motion. The architecture comprised four LSTM layers with a hidden size of 80 units and a dropout of 0.1 to mitigate overfitting. Each input vector consisted of four features: (x, y, v_x, v_y), representing position and velocity components, while the network output predicted the future position after a 1-second horizon.

Data normalization was performed using a physics-based approach, scaling positions and velocities to maintain numerical stability and ensure consistency across sequences.

The model was optimized using the Adam optimizer with an initial learning rate of 10^{-3} and a weight decay of 10^{-5} for regularization. Gradient clipping was applied at 1.0 to prevent gradient explosion. A learning rate scheduler (`ReduceLROnPlateau`) was implemented to adaptively reduce the learning rate upon validation loss plateau. Early stopping with a patience of 5 epochs was employed to avoid overfitting. Training was conducted with a batch size of 32 for a maximum of 5 epochs, although convergence occurred earlier through early stopping.

TABLE III: Model Hyperparameters

Parameter	Value
Input Features	(x, y, v_x, v_y)
Hidden Size	80
LSTM Layers	4
Dropout	0.1
Batch Size	32
Learning Rate	10^{-3}
Weight Decay	10^{-5}
Optimizer	Adam
Loss Function	Masked MSE
Gradient Clip	1.0
Early Stopping Patience	5
Maximum Epochs	5

F. Evaluation metrics for LSTM

Loss function (MSE)
Mean Squared Error was computed in the normalised feature space during training. The best validation loss corresponds to the checkpoint with the lowest error, indicating the point of best generalisation under simulation conditions. In visual terms, a lower MSE means predicted points cluster tightly around the ground truth trajectory with minimal spread.

RMSE (1 s horizon)
Root Mean Square Error in metres was calculated after inverse normalisation. This metric quantifies the typical distance error between predicted and actual pedestrian positions. For example, an RMSE of 1.0 m implies that predictions typically land about one metre away from the ground truth.

Hit rate (within 0.3 m)
The hit rate is defined as the percentage of predictions that fall within 0.3 m of the ground truth. This tolerance approximates the average width of a human chest.

IV. Results

A. Trajectory Prediction Accuracy (LSTM) — Simulation Results

The final model achieved the following performance on the validation set:

TABLE IV: Validation Set Performance Metrics for the Best Model

Metric	Value
Best Validation Loss (MSE)	0.003575
Validation RMSE (1 s horizon)	0.212 m
Hit Rate ($\leq$ 0.30 m)	40,956 / 43,106 (95.01%)

These results indicate that the LSTM model provides accurate short-term trajectory predictions, with the majority of predictions falling within the operational tolerance threshold.

B. Qualitative Visualisations — Simulation

To visualise the typical predictions in the dataset, Figs. 9 shows predicted vs. ground-truth trajectories with the 0.3 m tolerance zone overlaid.

C. Detection (DBSCAN) and Tracking (nearest-neighbour) Performance

Out of **180** total ID sequences extracted from the test set:

- **70** were identified as **Noise** (**38.89%**)
- **110** were considered **Valid Pedestrians** (**61.11%**)

To ensure a fair and representative evaluation of pedestrian motion prediction, these noise-classified sequences were excluded from model predictions and metrics. Noise IDs typically corresponded to spurious detections or stationary reflectors, which do not exhibit meaningful trajectory behaviour. This was done via manual inspection.

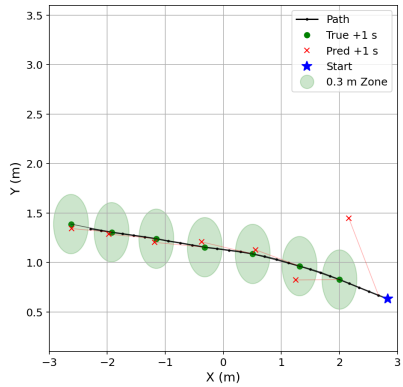


Fig. 9: Predicted vs. ground-truth trajectory with 0.3 m tolerance (simulation), example ID 364 (Validation Set).

TABLE V: Test Set Performance Metrics

Metric	Value
Best Validation Loss (MSE)	0.003575
Test RMSE (1 s horizon)	0.689 m
Hit Rate ($\leq$ 0.30 m)	$162/549 = 0.2951$ (29.51%)

D. Trajectory Prediction Accuracy (LSTM) — Test Set Results

Table V summarises the final model's performance on the real-world test set. Three key metrics are reported:

On the test set, the model achieved a root-mean-square error of 0.689 m and a hit rate of 29.51%. For comparison, performance on the validation set (based on simulation data) showed a significantly lower RMSE of 0.212 m and a hit rate of 95.01% (Table IV).

These results show that although the model generalised well in simulation, its performance deteriorated when applied to real-world trajectories extracted from sensor data.

Qualitative Visualisations — Test Set

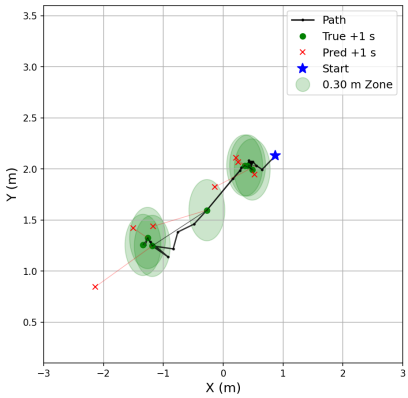


Fig. 10: Extracted trajectory from the test set showing Pedestrian 46 (Test Set).

F. Test Set Results analysis

While the LSTM achieved strong accuracy in simulation (Fig. 9), performance on real-world data (Fig. 10) deteriorated due to several upstream issues in the preprocessing and tracking pipeline:

Simulation/label mismatch: Simulation used smooth kinematic labels, while test set labels came from noisy centroids. The model was therefore trained and evaluated on different target distributions.

Centroid drift: DBSCAN centroids shifted when clusters were asymmetric, clipped, or incomplete, producing zig-zags and spurious velocity changes even for static pedestrians.

Binary-mask instability: Frame-by-frame Otsu thresholding made clusters sensitive to intensity fluctuations, causing marginal pixels to flicker and centroids to wander.

CFAR suppression: A fixed CFAR ratio of 1.0 sometimes removed weaker valid reflections, creating detection gaps and sudden jumps in tracking.

Occlusion and boundary filters: Large occlusion boxes and 20,px margins suppressed reflections but also masked valid pedestrian returns, reducing track stability.

Cluster consolidation: In-frame merging occasionally collapsed fragmented clusters into a single representative, causing centroid "teleports" between fragments.

Velocity amplification: Frame-to-frame differencing at 4,Hz exaggerated centroid jitter, inflating velocity estimates and propagating noise into model predictions.

Nearest-neighbour limits: Without a motion model (e.g., Kalman), association was brittle; intermittent misses caused ID switches and fragmented trajectories.

Together, these effects explain the wider spread of predicted points and lower hit rate in the test set.

V. Discussion

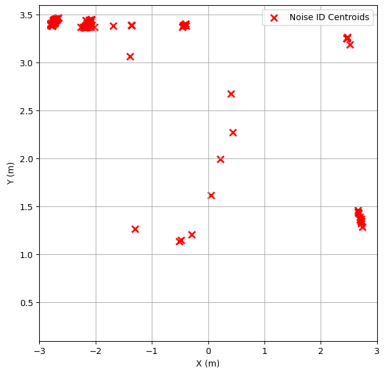


Fig. 11: Noise map of the recording area derived from per-pixel temporal activity.

TABLE VI: Dataset showing input features and outputs at $+0.25\,\mathrm{s}$ (250 ms) and $+1\,\mathrm{s}$.

t (s)	**ID**	x (m)	y (m)	v_x (m/s)	v_y (m/s)	$x_{+0.25s}$	$y_{+0.25s}$	x_{+1s}	y_{+1s}
0.25	1	2.43	2.73	-2.29	0.27	1.86	2.80	0.18	2.94
0.50	1	1.86	2.80	-2.27	0.27	1.29	2.85	-0.38	2.96
0.75	1	1.29	2.85	-2.26	0.20	0.73	2.89	-0.94	2.98
1.00	1	0.73	2.89	-2.25	0.16	0.18	2.94	-1.50	2.96
1.25	1	0.18	2.94	-2.22	0.19	-0.38	2.96	-2.06	2.94
1.50	1	-0.38	2.96	-2.23	0.10	-0.94	2.99	-2.62	2.90

A. Occlusion and Boundary Handling

Proportional occlusion boxes at $\sim 3\times$ the object bounding box suppressed the area of the car to reduce noise at this site but are specific to this scenario. Manually setting these occlusion zones sizes may be more practical as having occlusion zones to large will reduce the reliability of detection and tracking.

B. Early-Frame Context and Map Prior

A short warm-up stabilises early IDs before confirmation. The "red@5" threshold also serves as a placeholder for a pre-made map: in a known environment, fixtures would be capped (score ≤ 4) so only potentially dynamic entities are promoted to trackable IDs, reducing false starts near static reflectors.

C. Predictive Gating at $+0.25\,s$ (Proposed)

The nearest-neighbour test was too forgiving in clutter. A practical fix is *predictive gating*: instead of matching a detection to the last centroid, it can be matched to where the track is expected to be one frame later (Δt=0.25 s at 4 Hz). Draw a small "gate" (a circle) around that predicted point and only accept detections that fall inside it.

Two additional safeguards can be put in place to make the association more robust in clutter: first, by requiring a larger displacement before creating a new ID, it will make it less likely for noise to be mistaken for an ID; second, if i make a shorter protection radius around confirmed tracks it will prevent stray point clouds from stealing the match. This setup aligns with Table VI: the inputs reflect the $+0.25\,\mathrm{s}$ step used for predictive gating, while the targets provide the $+1\,\mathrm{s}$ forecast.

D. Limitations and Next Steps

the otsu thresholds, occlusion scaling, and suppression rules was tested in a single area. the training targets simulated were smooth and did not mimic the noisy centroids in the test set. Simulating test trajectories is difficualt to replicate; the next step is to collect a larger real dataset and retrain/fine-tune the LSTM model so the data matches real trajectory behaviour, then extend to multi-pedestrian outdoor scenes and real-time deployment.

VI. Conclusion

Ultrasonic point clouds are suitable for a detect–track–predict pipeline for pedestrian trajectories. In simulation, the LSTM

achieved sub-0.3 m RMSE; on real data, accuracy declined due to centroid jitter, permissive nearest-neighbour matches, and masking side-effects. To address these shortcomings, the proposals in the Discussion warrant empirical testing: predictive gating at $+0.25$ s to anchor associations and reduce ID switches; imposing a minimum-motion threshold before ID creation and keeping a small protection radius around confirmed tracks to suppress noise promotions; site-calibrated occlusion and boundary masks guided by a noise map to limit reflector bleed-through without hiding valid returns; retraining/fine-tuning on real trajectory data so the model better learns centroid dynamics; and ego-motion compensation to keep tracks consistent in sensor coordinates. If validated, these changes may be able to improve robustness and establish ultrasonic sensing as a practical redundancy when camera or LiDAR systems fail.

Code and Materials

Repository (open): https://github.com/ADAR-cell29/Pedestrian-Detection-Tracking-and-Path-Prediction-Ultrasonic

REFERENCES

[1] **Na, K. I. and Park, B.** (2023). Real-time 3D multi-pedestrian detection and tracking using 3D LiDAR point cloud for mobile robot. *ETRI Journal*, 45(5), pp. 836–846.

[2] **Wang, W., Chang, X., Yang, J. and Xu, G.** (2022). LiDAR-based dense pedestrian detection and tracking. *Applied Sciences*, 12(4), p. 1799.

[3] **Sighencea, B. I., Stanciu, R. I. and Căleanu, C. D.** (2021). A review of deep learning-based methods for pedestrian trajectory prediction. *Sensors*, 21(22), p. 7543.

[4] **Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L. and Savarese, S.** (2016). Social LSTM: Human trajectory prediction in crowded spaces. In *Proc. IEEE Conf. on Comp. Vision and Pattern Recognition*, pp. 961–971.

[5] **Shen, S., Saito, M., Uzawa, Y. and Ito, T.** (2023). Optimal clustering of point cloud by 2D-LiDAR using Kalman filter. *Journal of Robotics and Mechatronics*, 35(2), pp. 424–434.

[6] **Santos, B., Oliveira, A. S., Carvalho, N. B., Fernandes, R., Cannizzaro, A. and Cruz, P. M.** (2021). FMCW radar point cloud multiperson tracking using a Kalman filter–based approach. *URSI Radio Science Letters*, p. 21-0068.

[7] **Zhao, J., Xu, H., Wu, J., Zheng, Y. and Liu, H.** (2019). Trajectory tracking and prediction of pedestrian's crossing intention using roadside LiDAR. *IET Intelligent Transport Systems*, 13(5), pp. 789–795.

[8] **Nelson, J.** (2020). When to Use Grayscale as a Preprocessing Step. *Roboflow Blog*. Available at: https://blog.roboflow.com/ (Accessed: 17 August 2025).

[9] **OpenCV Developers** (n.d.). Image Thresholding (documentation). *OpenCV*. Available at: https://docs.opencv.org/ (Accessed: 17 August 2025).

[10] **Rosebrock, A.** (2021). OpenCV Morphological Operations. *PyImageSearch*. Available at: https://pyimagesearch.com/ (Accessed: 17 August 2025).

[11] **Stronzek–Pfeifer, D.** *et al.* (2023). LiDAR-based Object Detection for Agricultural Robots (abstract). *ResearchGate*. Available at: https://www.researchgate.net/ (Accessed: 17 August 2025).

[12] **Chung, W. Y.** *et al.* (2022). Image Dehazing Using LiDAR-Generated Grayscale Depth. *PubMed Central*. Available at: https://www.ncbi.nlm.nih.gov/pmc/ (Accessed: 17 August 2025).

[13] **Chávez Plascencia, A.** *et al.* (2023). Deep Learning Sensor Fusion for Pedestrian Detection. *ResearchGate*. Available at: https://www.researchgate.net/ (Accessed: 17 August 2025).

[14] **Ayvali, E.** *et al.* (2015). Optical Flow-Based Needle Tracking in Ultrasound. *ResearchGate*. Available at: https://www.researchgate.net/ (Accessed: 17 August 2025).

[15] **Nazarudin, A. A.** *et al.* (2023). Follicle Segmentation in Ovaries: Otsu Thresholding and Morphological Methods. *ResearchGate*. Available at: https://www.researchgate.net/ (Accessed: 17 August 2025).

[16] **Ouster, Inc.** (2022). Object Detection and Tracking with Ouster LiDAR. *Ouster Blog*. Available at: https://ouster.com/ (Accessed: 17 August 2025).