Contents lists available at ScienceDirect

Measurement

journal homepage: www.elsevier.com/locate/measurement

A smart container for real-time load occupancy estimation using embedded neural inference

Diego Remón, Alberto Gascón, Álvaro Marco, Teresa Blanco, Roberto Casas *

Aragon Institute of Engineering Research, University of Zaragoza, 50018, Zaragoza, Spain

ARTICLE INFO

Dataset link: https://doi.org/10.5281/zenodo.18614817

Keywords:
Smart container
Time-of-flight sensing
Occupancy estimation
Embedded inference
Edge AI
Convolutional neural network
ESP32-S3

ABSTRACT

The increasing demand for sustainable urban delivery solutions has driven the adoption of cargo bikes due to their environmental benefits and adaptability to congested urban environments. These operations benefit from monitoring systems capable of estimating load occupancy (volume) to support logistical decision-making. This study presents a smart-container approach for real-time occupancy estimation using two Time-of-Flight (TOF) sensors and a compact neural model deployed on an ESP32-S3 microcontroller. TOF sensors generate distance matrices of the internal cargo space, which are processed to estimate occupied volume via the normalized *FreeSpace* target. In two inference stress tests, the system achieves $R^2 = 0.929$ and 0.923, with mean absolute error (MAE) = 0.044 on the normalized *FreeSpace* scale (0–1), equivalent to 8.1 dm^3 (4.4% of container capacity). The results support the feasibility of low-cost embedded inference for operational capacity checks in cargo-bike logistics.

1. Introduction

In light of climate change and increasing environmental awareness [1], sustainable transportation methods [2,3] are gaining traction. This shift is reinforced by the growing pressure on last-mile logistics: global e-commerce sales exceeded USD 6.4 trillion in 2025, and last-mile delivery can account for roughly 41% of total supply-chain costs [4]. Cargo bikes, in particular, have become increasingly popular, with more than 100,000 units sold in Germany in 2020 [5,6], 78,000 of which were electric. Similarly, in the United States in 2019 [7], 400,000 electric bicycles were sold, representing a 78% increase over the previous year. In Switzerland, between 2019 and 2021 [6], e-cargo bike sales increased by 184%, reaching 4218 units. In parallel, market demand in courier, express, and parcel (CEP) operations remains strong (e.g., Germany's CEP sector is projected at around USD 29.5 billion in 2025), supporting continued investment in electrification and micromobility solutions [4,8]. Market indicators point in the same direction: in Switzerland the e-bike market is estimated at USD 486.3 million in 2025 and projected to reach USD 621.4 million by 2029, signaling sustained demand [9]. At the global level, analysts value the cargo bike segment at around USD 3–4 billion in 2024 and expect steady growth through 2031 [10]. Across Europe, the e-cargo bike market is estimated at USD 1.16 billion in 2025 and is projected to grow to USD 1.46 billion by 2031 (3.85% CAGR over 2026–2031) [11]. Beyond freight, recent evidence indicates that e-cargo bikes can cover a meaningful share of urban transport demand (e.g., up to 20% of urban freight demand in suitable contexts) [12]. Recent research (2023–2025) also shows that e-cargo bikes are not only a logistics solution but are increasingly adopted for personal transport, filling a gap between cycling and family cars [13,14]. This trend is also visible in shared cargo-bike services, where reported impacts include reduced car ownership among active users (7.4%–18.1%) [15]. Forward-looking national analyses project continued fleet growth; for instance, a high-uptake scenario estimates 31,700 light electric freight vehicles operating in the Netherlands by 2027 [16]. A global increase in their use is therefore expected in the coming years [12,17–19], driven by environmental benefits [20] and advantages in cost, health, and urban traffic efficiency [5,21].

Cargo bikes can be classified in various ways based on their design, load position, etc. Examples [3,22] include *Mini cargo bikes* or *Short Johns*, which are similar to traditional bicycles and support small boxes on the front or rear; *Box bikes*, also known as *Long-John* or *Bakfiets*, which feature a two-wheeled wheelbarrow-like design with front-load capacity; *Longtails/Rear loaders*, which generally carry the load on a

Abbreviations: ANN, Artificial Neural Network; CNN, Convolutional Neural Network; DC, Direct Current; DOE, Diffractive Optical Elements; FOV, Field of View; LiDAR, Light Detection and Ranging; MAE, Mean Absolute Error; MLP, Multilayer perceptron; MSE, Mean Squared Error; PCA, Principal Component Analysis; RMSE, Root Mean Squared Error; SPAD, Single Photon Avalanche Diode; SoC, System on Chip; TOF, Time-of-Flight

* Corresponding author.

E-mail addresses: dremon@unizar.es (D. Remón), algaroche@unizar.es (A. Gascón), amarco@unizar.es (Á. Marco), tblanco@unizar.es (T. Blanco), rcasas@unizar.es (R. Casas).

https://doi.org/10.1016/j.measurement.2026.120785
Received 8 October 2025; Received in revised form 16 January 2026; Accepted 9 February 2026
Available online 10 February 2026

rear platform and, while similar to the previous type, are lighter and easier to handle due to their similarity to traditional bicycles; and *Cargo trikes/Cargo tricycles*, which are three-wheeled models.

One of the main reasons for using vehicles such as cars or vans in logistics is their load-carrying capacity. *Cargo bikes* with high payload capabilities, when combined with intelligent delivery route optimization systems [20,23,24], can serve as a flexible alternative. They can access bike lanes and pedestrian areas, making them suitable for last-mile delivery.

Consequently, new mobility solutions, possibly *electric*, improve consumption and efficiency, especially for short distances. Recent studies have analyzed the integration of cargo bikes in logistics networks, emphasizing their potential for reducing congestion and emissions while maintaining operational efficiency [17,19]. Their prevalence in cities has been increasing due to technological advancements, such as enhanced autonomy, which ranges from 39.3 km to 74.7 km with 250 Wh and 625 Wh batteries [25].

However, the use of cargo bikes faces limitations in load capacity regarding weight and volume, as well as restrictions on autonomy, battery recharge times, speed, and adaptability to terrain [12,18]. The load capacity varies among different types of bikes and configurations – traditional bicycles, trailers, tricycles, and quads [26] – which can carry between 24 kg and 600 kg, depending on the design and structure of the vehicle. The typical battery range [26] varies between 19 km and 100 km of autonomy. Similarly, the battery charging time typically ranges from 4 to 8 h for a full charge.

One major challenge in cargo bike logistics is the lack of *real-time monitoring* and *accurate volume estimation* of the cargo compartment. While cargo bikes offer a sustainable alternative for urban logistics, studies emphasize the need for better technological integration, including automated volume estimation systems to support space usage monitoring and route planning [17,19]. Unlike load cells, which accurately measure weight, there is no widespread automated technology that quantifies the occupied volume in cargo bikes. Currently, package arrangement is left to judgment of human operators, which often results in inefficient space utilization, suboptimal loading sequences, and unnecessary reconfigurations during delivery operations [13,15].

In this context, ultrasonic distance sensing has been explored in some commercial solutions due to its low cost and simplicity [27–29]. However, its performance degrades significantly in logistics scenarios involving dense and stacked cargo [29]. When multiple objects are closely packed, ultrasonic waves are prone to multipath propagation and secondary reflections on edges and inclined surfaces, leading to unstable echo detection and spurious distance readings [27,30, 31]. These effects are exacerbated at high occupancy levels, where the acoustic path is increasingly dominated by partial occlusions and overlapping reflections, rather than direct line-of-sight returns. As a result, ultrasonic-based systems can exhibit unstable distance readings and limited reliability as a volumetric signal under dense stacking conditions.

More broadly, recent work in measurement-oriented engineering increasingly relies on neural models to infer latent physical states from sparse, noisy or indirect sensor observations. Recent contributions in the field of load and stress estimation [32,33] and neural-network-based diagnostic frameworks for structural and mechanical systems [34–36] illustrate the growing adoption of data-driven inference approaches for monitoring and decision support under uncertainty. These works highlight the suitability of compact neural models for extracting actionable information from limited sensing configurations, aligning with the measurement-driven and resource-constrained context addressed in this study.

This paper is organized as follows: Section 2 presents the designed system, including both its electronic configuration and physical structure. Section 3 describes the procedure for creating the dataset from sensor data. Section 4 covers the methodology for neural network modeling, while Section 5 presents and analyzes the modeling and inference results. Section 6 discusses the findings and outlines the study's limitations. Finally, Section 7 offers the conclusions.

2. System design

2.1. State of the art

2.1.1. Sensing techniques for volume estimation

To address this challenge, various sensor technologies enable volume detection and estimation, such as distance sensors [37], spatial mapping [38,39], or cameras [37,40]. By extrapolating methodologies used to determine the volume of boxes, other more complex geometries can also be characterized [41,42]. This is particularly useful when measuring the dimensions of various objects. Cameras are commonly used and can be coordinated with other sensors, such as lasers, to capture multiple views of the environment. Appropriate image processing methods then yield the desired results.

LiDAR (Light Detection And Ranging) devices, also known as Laser Range Finders (LRFs), are notable for their efficiency and accuracy. These devices can be used individually or in combination with cameras to precisely detect distances. The sensor defines the volume of interest by providing relative distance measurements from the origin point [39,43,44]. With LiDAR sensors, it is possible to obtain 2D (Two-Dimensional) data images and approximate the 3D (Three-Dimensional) space using point clouds generated by projected laser lines. The cost and power consumption of these sensors depend on the format and density of the generated point cloud [45]. These devices are also used in robotics, environmental detection, location mapping, safe driving [45,46], and urban areas for detecting pedestrians and obstacles.

There are other types of sensors that project laser beams to measure distances, dimensions, and spatial characteristics, enabling precise estimation of the volume of various objects. These high-precision sensors, when properly calibrated, facilitate the use of computer vision and structured light techniques. Unlike LiDAR, they use triangulation of laser projections [47], where the modulated lines drawn on the object can be captured by a camera to calculate its dimensions [48,49]. This method [37,40] allows for accurate 3D object dimensioning, even under adverse conditions.

Time-of-Flight (TOF) sensors represent another type of laser-based sensors that can be used individually. Similar to LiDAR, they estimate distances by measuring the time it takes for a light pulse to reflect off an object [38]. However, unlike LiDAR, which generates 3D non-structured point clouds in open spaces, TOF sensors [50] produce a 2D distance matrix, displaying the distance to the laser contact point at each pixel. This structure makes TOF sensors more compact and energy-efficient, ideal for controlled or small-scale applications [51–53].

Ultrasonic sensors are known for their ability to measure distances quickly and easily, though their accuracy is generally low [54]. There are two primary methods for measuring distances with these sensors [55]: pulse reflection (echo) and continuous wave [56]. A single device can act as both transmitter and receiver, simplifying the measurement complexity. To improve accuracy [55], additional equipment and advanced processing techniques, such as Kalman filters, neural networks, and probability theories, are required, as is the case with other sensors.

Infrared sensors are widely used to measure distances, detect motion, and measure temperature, although they are less developed for volume detection [57,58]. However, variants of TOF sensors use infrared light to measure distances accurately. Unlike TOF sensors, infrared sensors do not capture a matrix-format distance map but instead typically measure point distances or smaller areas, depending on the sensor's configuration. A common method for utilizing infrared sensors is similar to the triangulation technique used by laser sensors [59]. Other types of sensors for distance and volume detection include photoelectric sensors, which operate using structured light and detect intensity differences in various objects [60–62]. A comparison

Table 1
Comparison of laser sensors for distance detection.

Type of sensor	Description	Accuracy	Performance in different conditions	Applications
LiDAR	Laser scanner that measures distances and generates point clouds	High (±1 mm)	Sensitive to dust, works in darkness, higher power consumption	3D mapping, robotics, autonomous driving
Laser sensors	Sensors projecting laser beams to measure distances and dimensions	High (±1–2 mm)	Sensitive to surface reflectivity, affected by ambient lighting	Volume measurement, quality control
TOF sensors (laser)	Measure distances using the time of flight of the laser	Distance-dependent (typically ∼5%–6% at meter-scale ranges; mm-level only at short range)	Works in darkness, less affected by dust, occlusion handling via ML processing	2D and 3D distance mapping
Ultrasonic sensors	Measure distances using sound pulses	Low (±10 mm)	Unaffected by light, highly affected by dust and obstructions	Liquid level measurement, simple proximity sensing
Infrared sensors	Use infrared light to measure distances and detect motion	Moderate (±5–7 mm)	Affected by ambient light, moderate performance in dusty environments	Motion detection, temperature measurement
Photoelectric sensors	Measure distances by detecting differences in light	Moderate (±5–8 mm)	Affected by ambient light, low performance with transparent or reflective objects.	Object and distance detection

of accuracy and robustness of these sensor types under varying environmental conditions, including dust presence, different lighting levels, and occlusions is summarized in Table 1.

While LiDAR sensors offer the highest accuracy, their high power consumption and cost make them impractical for lightweight, energy-efficient applications like cargo bikes. Additionally, LiDAR is more sensitive to dust and requires complex data processing to extract relevant volumetric information [63,64].

Ultrasonic sensors, while low in power consumption, suffer from significant accuracy limitations and are highly susceptible to environmental noise such as dust and vibrations [65], making them unreliable for detailed volumetric estimation. Similarly, infrared sensors struggle with ambient light interference [66], reducing their reliability in dynamic urban settings.

To enable on-device occupancy estimation in a compact cargo-bike container, this study employs multizone TOF sensors, which provide a low-power depth signal in matrix form and fit the cost and integration constraints of lightweight platforms.

The multizone TOF sensors used in this work provide per-zone distance readings with distance-dependent accuracy that varies with target reflectance and ambient conditions. For devices such as the VL53L7CX, ranging performance is typically reported as a relative error on the order of 5%–6% at meter-scale distances rather than as a fixed millimeter precision across the full operating range.

2.1.2. Data-processing methods

In this context, several data processing methods can be applied to optimize the use of distance matrices and estimate cargo volume. One such method is *Point cloud fusion*, which integrates data from multiple sensors such as LiDAR or TOF into a unified spatial model. This fusion transforms local distance measurements into global coordinates through rotation and translation. There are three main fusion strategies [45]: *Early Fusion Scheme*, which merges the point matrices from all sensors before detection, improving accuracy in low-visibility areas; *Late Fusion Scheme*, where each sensor detects objects separately, and the results are merged to eliminate duplicates; and *Hybrid Fusion Scheme*, which combines early fusion in complex zones with late fusion in more visible areas. This strategy balances precision and efficiency, especially under occlusion.

Another method is *voxelization* [38]. This technique converts geometric data into a 3D grid made up of discrete units called *voxels* (volumetric pixels). Sensor readings are transformed into coordinates relative to a fixed reference – usually the sensor itself – and then mapped into a voxel grid. Each voxel represents a small, uniform volume of space. The total cargo volume is then calculated by summing all occupied voxels within the region of interest [39], providing a detailed and structured spatial representation.

Top-plane detection is also commonly used to extract object dimensions [38,44]. The process starts by identifying objects in a depth map. The *Flood Fill algorithm* is applied to group areas of similar depth, isolating the top surface of each object. The contour of this surface is traced and stored as a set of coordinates. The algorithm then removes the processed plane and iterates to detect subsequent surfaces. The object's depth is determined by measuring the distance from its top plane to the background. Width and length are obtained by analyzing laser line intersections on the object's contour [40], while height is derived from laser deviations along the object's edges.

Neural networks and deep learning are widely applied today, with deep learning [67] being a subset of machine learning. These are neural networks with a large number of layers and parameters [43,52]. A common implementation is the *convolutional neural network (CNN)*, which employs convolutional computation and has a deep structure [53,58]. Essentially [60], a CNN is a *multilayer perceptron (MLP)* that adopts local connections and shared weights, significantly reducing the number of parameters compared to a traditional network, facilitating optimization, and decreasing the risk of overfitting. A basic CNN [60] consists of a convolutional layer that extracts features using a *kernel*, followed by a pooling layer that reduces the network scale and simplifies computational complexity. Additionally, *transfer learning* can be applied, using pre-trained networks such as *VGG16 (Visual Geometry Group - 16 layers)* from University of Oxford, and fine-tuning their final layers for the specific project goal. By modifying the network's structure and implementing appropriate hyperparameters and activation functions [47,68], the network can be trained and its performance evaluated through final tests.

2.2. Practical challenges

The landscape of last-mile delivery monitoring solutions is diverse, with each provider offering distinct functionalities adapted to specific operational scenarios.

The comparison of existing market solutions reveals a fragmented landscape in last-mile delivery monitoring systems. Different providers as Sensolus [69], Velco [70], Frotcom [71], Road Ready [72], TFT-100 [73], or Didcom [74] offer basic location tracking, but only a few incorporate volume measurement features. Among them, Road Ready, TFT-100 and Didcom include ultrasound-based volume sensing, while no solutions currently deploy Time-of-Flight (TOF), camera, or LiDAR technologies for this purpose. In terms of vehicle status monitoring, most of providers – Velco, Frotcom, Road Ready, TFT-100, and Didcom – offer this functionality. Weight measurement is notably scarce, present only in Frotcom and Road Ready. These findings highlight a clear technological gap in the market: current systems prioritize location and basic telemetry, while real-time occupancy (free-space) estimation remains largely unaddressed, particularly with advanced depth-sensing technologies.

To better understand the operational implications of this gap, a focus group was conducted with one logistics CEO, two fleet managers, and four last-mile couriers. The discussion revealed several key challenges.

- **Minimizing manual workload and maximizing automation**: Couriers, particularly in last-mile delivery scenarios, operate under significant time constraints, making manual data entry tasks impractical. The integration of automated systems is essential to streamline operations and reduce cognitive load on couriers [75,76].

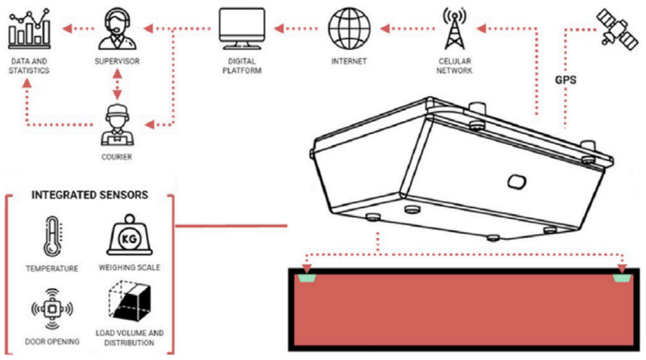


Fig. 1. Integrated smart sensing unit scheme [78].

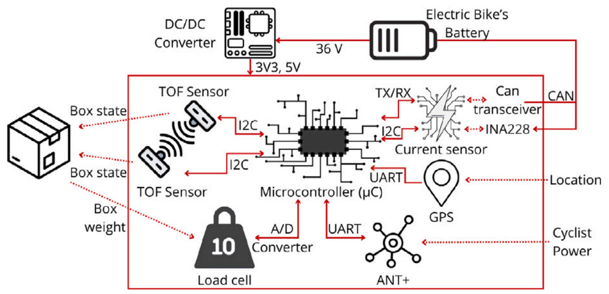


Fig. 2. Block diagram of the electronic system highlighting the main components involved in the volume estimation pipeline. Selected supporting elements of the overall device architecture are included, while auxiliary subsystems not used in the experiments reported in this study (e.g., temperature and door-status sensing) are omitted for clarity.

- **Integration of artificial intelligence (AI) and Internet of Things (IoT) technologies in logistics operations facilitates real-time decision-making and predictive analytics**: These advancements contribute to more responsive and silent delivery networks, capable of adapting to fluctuating demand and traffic conditions [77].
- **On-device computation requirement**: Edge processing is essential, as relying solely on cloud connectivity may introduce latency, additional costs, or vulnerability to coverage gaps (e.g., in tunnels, underground areas, or dense urban environments). Therefore, inference must be executed directly on the embedded device to ensure real-time operation and independence from network conditions.
- **Prevalence of cubic cargo forms**: Participants in the focus group reported that, in their day-to-day last-mile operations, a large share of shipments arrives in *rectangular cartons/boxes*.
- **Cost acceptability**: An integrated sensing system – capable of reporting in real-time cargo volume, GPS, position, temperature, and weight – would be economically viable for last-mile logistics operations if it remained within a target cost range of 250€–300€ per unit. This threshold is considered acceptable given the operational value provided by such systems in terms of real-time decision-making, dynamic routing, and cargo optimization.

2.3. System description

Taking all these into consideration, we propose an integrated smart sensing unit that integrates sensors to determine load volume as its core function. In addition, the unit includes embedded sensors for weight measurement, temperature, and door status, enabling extended monitoring capabilities. It is also equipped with GPS for geolocation and communicates via cellular networks to a central digital platform. This platform enables supervisors to access real-time data, support delivery routing decisions, and manage courier performance. This is summarized in Fig. 1.

Fig. 2 shows a simplified overview of the electronic system, presenting the main components involved in the volume estimation pipeline together with selected supporting elements of the overall device architecture.

In addition to the sensing and processing elements involved in volume estimation, the overall system architecture includes components for power and system-level monitoring. In particular, a CAN (Controller Area Network) interface is incorporated to enable robust integration with the electrical ecosystem of a cargo bike or similar vehicle platform, where CAN is commonly used for battery management and vehicle-state data exchange.

An *INA228* current and voltage monitor is included to measure battery-side electrical parameters, allowing the system to track power consumption and energy usage during operation. This information is relevant for deployment considerations such as autonomy, power budgeting, and long-term system monitoring.

The load volume sensor selected was the laser Time-of-Flight (TOF) sensor *VL53L7CX* [79]. This is a multizone range sensor integrated in a compact chip that includes a SPAD (Single Photon Avalanche Diode) array, physical infrared filters, and DOE (Diffractive Optical Elements). These components work together to achieve optimal range performance under varying ambient light conditions. The DOE positioned above the emitter laser projects a square FOV (Field Of View) onto the scene, while the receiver's lens focuses the reflected light onto the SPAD array.

The sensor supports per-zone ranging up to 350 cm and operates at speeds of up to 60 Hz, making it among the fastest miniature multizone TOF sensors available [79]. Its ranging performance is distance-dependent and is typically characterized by a relative error on the order of 5%–6% at meter-scale distances (with higher precision at short range). While this study focused on static conditions, the high sampling rate suggests that the system may handle minor load movements during transport by averaging consecutive measurements. It measures distances in up to 8 × 8 zones with a wide diagonal FOV of 90° (reducible via software, 60° both vertically and horizontally). Unlike other models, it performs well under different ambient lighting conditions. It operates on a 3.3 V power supply, consuming 4 mA in standby mode and 95 mA during active data acquisition. Communication with the microcontroller is managed via the I2C (Inter-Integrated Circuit) protocol, and multiple sensors can be assigned distinct addresses for simultaneous use.

The selected microcontroller is the *ESP32-S3* SoC (System on Chip), developed by *Espressif* [80]. It integrates a dual-core *Xtensa LX7* processor and supports WiFi (Wireless Fidelity) and Bluetooth Low Energy (BLE), enabling both local processing and wireless connectivity when required. The device provides 41 multifunctional GPIOs (General Purpose Input Output) and includes an integrated PCB (Printed Circuit Board) antenna, making it suitable for compact, embedded implementations. Fig. 3 shows the proposed design of the electronic system.

The assembled prototype has approximate external dimensions of 85 mm × 45 mm × 21 mm, excluding cabling and temporary mounting fixtures.

Table 2 provides an approximate bill-of-materials (BOM) breakdown of the key hardware components considered in the proposed system design, together with estimated mounting and enclosure costs. The reported values correspond to component-level price estimates and basic assembly elements, and do not account for additional factors typically included in commercial products, such as large-scale manufacturing, packaging, logistics, profit margins, certification, customer support, or long-term maintenance.

To provide context for these figures, commercially available fleet-tracking solutions in the cargo and logistics domain typically price

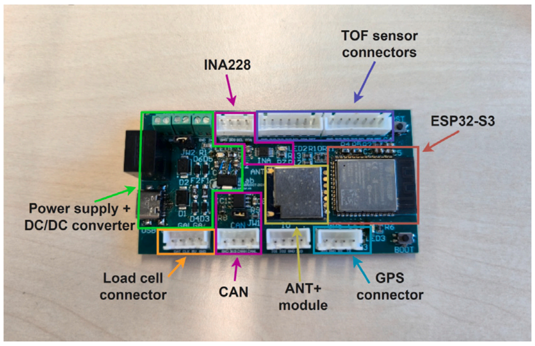


Fig. 3. Prototype implementation of the electronic system. The figure shows the main hardware components physically realized in this prototype, corresponding to the sensing and processing elements described in Figs. 1 and 2 (ESP32-S3 microcontroller, dual TOF sensor connections, power regulation, and system monitoring). Communication interfaces and auxiliary components included in the broader system design but not used in the reported experiments are either omitted or only indicated for completeness.

Table 2
Approximate bill-of-materials (BOM) and mounting cost estimate for the proposed system design (component-level prices).

Component	Model	Approximate cost
Time-of-Flight Sensor	VL53L7CX	8€
Load Cells and ADC	HX711	12€
GPS	ATGM336H	11€
TH sensor	DHT22	2€
Door-status sensor	59155-1-S-02-A	3€
NB-IoT module	SIM6700	15€
Microcontroller	ESP32-S3	5€
Power Supply	DC-DC Converter	5€
Various electronics	–	3€
Plastic case	–	10€
Mounting	–	10€
Total (BOM estimate)	–	84€

hardware and software separately. For example, Teltonika's FFM920 4G Cat-M1 tracker retails around 55€-66€ per unit through European distributors, while devices used in solutions such as Frotcom are available in retail channels between USD 127 and USD 250 per unit, depending on configuration and supplier. Platform fees are commonly charged per asset per month, typically ranging from 1€ to 7€, and often require additional external sensors for functions such as door status, temperature, or cargo monitoring [69,71–74].

Some components listed in the BOM (e.g., temperature or door-status sensing) are part of the overall system design but were not used in the experimental evaluation reported in this study. Door-status sensing is implemented using a dedicated low-cost sensor, independent of the TOF sensing subsystem, and is therefore accounted for separately in the bill of materials.

The component labeled *Various electronics* groups low-cost discrete and interconnect components required for the custom PCB but not listed individually, including resistors, capacitors, protection diodes, connectors, headers, jumpers, and minor consumables such as LEDs (Light Emitting Diodes) and switches. The *Mounting* cost includes the mechanical fixtures required for installation inside the container, including fasteners and the two 3D-printed sensor wedges used to set the inclination of the TOF sensors, as well as basic assembly elements. These items are separated from the plastic enclosure to distinguish structural housing from installation-specific components.

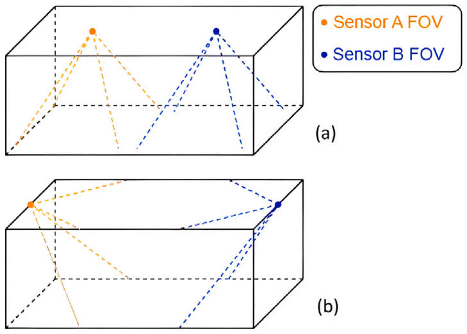


Fig. 4. Placement of sensors in different configurations. The first representation (a) shows the two sensors positioned on the top lid of the box, while the second (b) illustrates both sensors placed on the side edges of the box.

2.4. Physical sensor layout

A prototype box with dimensions 9.70 dm × 4.05 dm × 4.64 dm (length, width, height) was used to simulate the cargo box of the bike. This box was chosen for its challenging complexity, requiring two sensors compared to smaller or taller designs which just require one sensor.

The cargo volume estimation system is designed to be adaptable to a variety of cargo bike configurations currently used in urban logistics. Typical cargo box dimensions in commercial cargo bikes range between 8.50 dm × 7.00 dm × 5.00 dm for compact, high-capacity transport solutions, 12.00 dm × 8.00 dm × 5.00 dm for mid-sized cargo bikes used for parcel deliveries, and 12.50 dm × 8.00 dm × 6.00 dm for large cargo bikes with modular storage compartments. Given these ranges, the system is compatible with real-world logistics applications without requiring major modifications.

As mentioned, in this design a single sensor was insufficient for accurate measurements, since blind spots and occlusions caused by the introduced load were likely to occur. To address this, two sensors were employed, minimizing cost and computation time. Proper placement of the sensors mitigates occlusion issues. Two possible configurations for sensor placement are proposed:

1. **Sensors on the top lid of the box (Fig. 4.a)**: Placing both sensors on the top lid pointing inward. By dividing the box into two equal parts lengthwise, each sensor is positioned at the center of its respective half, providing a symmetric top-down viewpoint.
2. **Sensors on the side edges of the box (Fig. 4.b)**: Placing the sensors on the wide side edges of the box. This configuration covers nearly the entire volume. Although each sensor has blind spots at its lower region, the other sensor can cover these areas. In terms of occlusions, if one box occludes another and is not visible to one sensor, the other one can detect it by viewing the opposite side of the boxes.

The two sensor placement configurations shown in Fig. 4 were compared in terms of their geometric field-of-view coverage (90° diagonal FOV, corresponding to a 60° × 60° square FOV projected onto the scene) under empty-container conditions, considering only line-of-sight constraints imposed by the container geometry and the nominal sensor field of view. In the top-mounted configuration (Fig. 4.a), each sensor observes approximately 37.35 dm^3 of the 182.282 dm^3 container volume. When both sensors are considered jointly, the covered volume reaches 74.37 dm^3, corresponding to 40.98% of the container (i.e., 59.02% remains outside the geometric sensing range in the empty-container case). This limited coverage is primarily caused by the predominantly downward viewing direction and the significant

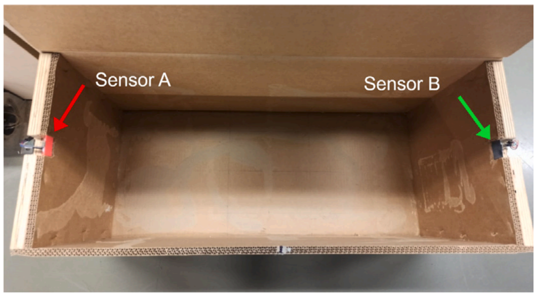


Fig. 5. Load box prototype (9.70 dm × 4.05 dm × 4.64 dm) with sensors positioned at its side edges.

overlap between the two fields of view, which leaves large regions of the container unobserved.

In contrast, the lateral configuration (Fig. 4.b) yields near-complete geometric coverage in the empty-container case: while each individual sensor has a lower blind region directly beneath its mounting side (approximately 42.17 dm^3, i.e., 23.13% of the container volume), this region is largely visible from the opposing viewpoint, and the union of both fields of view covers essentially the full container volume. Importantly, this coverage analysis characterizes the geometric sensing capability in the absence of cargo. Once packages are introduced, occlusions and shadow regions can still reduce effective visibility, particularly at high occupancy. Nevertheless, the lateral two-sensor layout minimizes global blind zones and provides complementary viewpoints, which is critical for reducing unobserved regions in practical loading scenarios.

Based on this geometric coverage analysis, the sensors were placed on the side edges of the container to exploit complementary viewpoints that minimize global blind regions and reduce occlusion effects observed in the top-mounted configuration. Mounted near the upper edge of the box, the sensors were tilted downward by 30° so that the upper boundary of the effective field of view coincided with the top plane of the container. In this work, the tilt mainly serves as a geometric alignment to maximize in-container coverage (i.e., to avoid wasting part of the FOV above the usable cargo volume), rather than as a calibration step that would alter the intrinsic ranging accuracy of the TOF device. Consequently, moderate misalignment primarily results in reduced effective coverage (more unseen regions and stronger shadowing/occlusion), whereas the distance measurements within the observed regions remain governed by the sensor's internal calibration and filtering. Large misalignment could additionally increase the fraction of rays intersecting the lid or the upper rim, which may generate invalid returns. These would be handled in future implementations through explicit masking of out-of-container rays and additional consistency checks. Locating the sensors on the lateral walls rather than on the lid further improves system robustness, as this configuration avoids exposure to mechanical stresses associated with a movable element and reduces the likelihood of long-term misalignment.

Regarding reflective effects, the current design does not exploit multipath or lid reflections to infer hidden geometry. Although reflective surfaces may produce secondary returns, such signals are highly dependent on material properties and incidence angle and can lead to unstable readings. Therefore, they are treated as a potential disturbance rather than as a reliable information source in the present pipeline.

To implement this fixed 30° downward tilt in a repeatable and mechanically stable way, small 3D-printed pyramidal mounts were created for each sensor, allowing fine adjustment of the orientation to match the target coverage area. This placement is illustrated in Fig. 5.

The robustness of the TOF sensors in different environmental conditions is a key factor in ensuring the reliability of the volume estimation system. These sensors are designed to operate under various temperature and lighting conditions, and certain measures can be taken to mitigate the effects of dust and obstructions. In addition, the protective enclosure provides IP6x-rated resistance against dust, further enhancing durability in logistics environments. Table 3 summarizes the environmental specifications and operational characteristics of the TOF sensors used in this study.

Table 3
Table of sensor environmental parameters.

Aspect	Description
Operating temperature range	−20 °C to 60 °C
Resistance to Dust	Protective casing (IP6x protection); lens cleaning recommended
Impact of obstructions	Reduction through data preprocessing and averaging
Sampling frequency	Up to 60 Hz allowing compensation
Interference handling	Uses infrared filtering to minimize light interference

Fig. 6. Representation of the six cargo boxes.

Table 4
Table of boxes with their dimensions and volume.

Boxes	Dimensions (dm)	Volume (dm^3)	Size
1	0.56 × 1.46 × 2.05	1.676	S
2	1 × 2.3 × 1.97	4.531	S
3	1.23 × 3.44 × 2.24	9.478	M
4	2.72 × 2.31 × 2.32	14.577	M
5	1.76 × 2.9 × 4.47	22.815	L
6	5.56 × 3.5 × 4.19	81.537	XL

3. Dataset creation

3.1. Box combinations

After an initial analysis and consideration of different box sizes, a reduced yet representative set of boxes was selected. Two boxes of size ‘S’ (Small), two of size ‘M’ (Medium), one of size ‘L’ (Large), and one ‘XL’ (Extra Large) were chosen, following a similar approach to the box selection described in [44]. In total, six boxes were selected to provide a diverse range of sizes, allowing for combinations that capture all relevant information without requiring an excessive number of boxes. A representation of the selected boxes is shown in Fig. 6.

The container and the six boxes mentioned above were initially measured. The container had a total available volume of 182.282 dm^3, and the respective cargo boxes had the volumes detailed in Table 4.

From the selection of boxes, the total number of possible combination cases is determined. A dataset was created, including measurements of the boxes placed individually and combined in groups of 2, 3, 4, 5, and finally, all 6 together. These combinations resulted in a total of 63 cases to measure. For each of the 63 box combinations, 60 manually re-arranged placements were tested following a pseudo-random procedure constrained by the container geometry and box dimensions. No orthogonal or predefined layouts were enforced, but the achievable placement variability naturally decreased as occupancy increased, particularly for larger box combinations. This resulted in 3780 different measurement configurations. A placement was counted as a new configuration only after manually re-positioning at least one box by a visible translation and/or rotation, and re-stacking when applicable; minor incidental perturbations during lid opening were not considered new placements. At each placement, 40 measurements were taken. This combination of cases, placements per case, and samples per placement leads to a dataset with *151,200 records*, 2400 per case.

Because the dataset is built from a finite set of six predefined boxes, each *case* (box combination) yields a discrete total occupied volume $VolT$ that is typically unique by construction (see Table 7). This does not make the learning task a trivial lookup, since the input consists only of sparse TOF depth patterns affected by occlusions and viewpoint limitations, and each case is measured under multiple distinct spatial placements. In practice, the mapping from $VolT$ to the TOF input is many-to-one and highly variable, because different placements of the same case can produce substantially different depth patterns due to self-occlusions and limited viewpoints. However, the present real-world dataset does not explicitly include different box compositions that lead to the same $VolT$ (volume-matched compositions). Therefore, invariance to volume-matched compositions is not evaluated in the present experimental scope and is left for future investigation.

The repeated measurements acquired at each placement capture sensor-level variability and short-term noise. They are replicates of the same underlying geometry and are not used to claim additional geometric diversity.

All measurements were conducted under static and controlled conditions using cubic and rectangular objects, allowing for a robust baseline assessment before extending the approach to more complex geometries. Disturbances due to cargo bike motion were intentionally excluded at this stage to establish a baseline evaluation.

To promote placement diversity within the physical constraints of the container, re-arrangements were performed to avoid repeated symmetric layouts. Configurations with smaller or fewer boxes allowed greater variability, whereas higher-occupancy combinations required more structured placements due to reduced degrees of freedom.

One of the inherent challenges in using Time-of-Flight (TOF) sensors for volume estimation is the presence of occlusions and undetected empty spaces when large or irregularly shaped objects are stacked. Since TOF sensors are mounted at the top of the cargo compartment, they primarily capture surface-level depth information and cannot penetrate objects to detect voids beneath them.

To mitigate this limitation, the dataset was designed to expose the neural network to occlusion-prone configurations through:

1. **Varied (pseudo-random) box arrangements**: Manual re-arrangements were performed between trials to generate diverse stacking conditions within the physical constraints of the container, increasing the likelihood of partial occlusions and shadowed regions.
2. **Sensor Placement Considerations**: The dataset was collected using two TOF sensors positioned at the edges of the cargo area, rather than a single central sensor, to reduce blind spots and improve depth estimation from different angles.
3. **Multiple placements per box combination**: For each box combination, 60 distinct placements were recorded, avoiding orthogonal layouts to increase variability and maximize the likelihood of capturing occlusion-prone configurations.

Fig. 7 shows several examples of boxes placed inside the load space.

While these strategies improve the dataset's robustness, some occlusion-related challenges may persist, particularly when dealing with highly irregularly shaped objects or asymmetric stacking configurations.

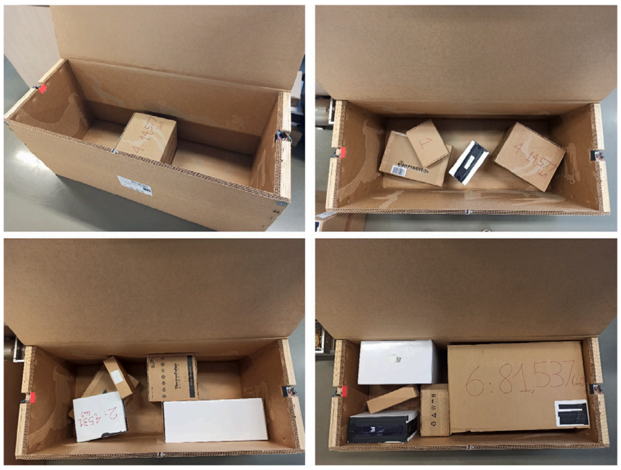


Fig. 7. Box placed inside the load space. The dataset includes various cargo loading arrangements, ranging from loosely stacked boxes to more tightly packed configurations, simulating real-world cargo bike loading conditions.

Table 5
Environmental conditions during data acquisition.

Parameter	Observed condition/range
Ambient temperature	20 °C – 25 °C
Relative humidity	40%–60%
Lighting conditions	Stable artificial indoor lighting
Presence of dust	Low (clean lab environment)
Enclosure state	Box closed with lid during all measurements

3.2. Data collecting

Once the TOF sensors were placed in the container, they were connected to the electronic prototype as shown in Figs. 2 and 3. In the experiments reported in this paper, raw TOF frames were streamed to a PC via a wired USB connection (USB-to-UART/CDC serial link) for logging and offline analysis, while the embedded platform was used to configure the sensors and acquire the measurements. Although the overall system design supports additional communication interfaces (e.g., ANT+ and cellular connectivity as depicted in Figs. 1–2, and WiFi/BLE capabilities of the ESP32-S3), these were not used during the dataset acquisition and validation in this study.

In addition, although the overall system architecture includes weight sensing via load cells (as detailed in the bill of materials), these sensors were not instrumented or used during the experimental campaign reported in this paper. The present validation therefore focuses exclusively on TOF-based volume and occupancy estimation. Integrating volume estimates with weight measurements for hybrid volume-weight validation and improved robustness is identified as future work.

All data collection was conducted indoors under controlled laboratory conditions. Environmental factors were maintained within stable ranges to minimize their influence on sensor performance. Table 5 summarizes the main parameters observed during data collection.

These controlled conditions do not capture the environmental stressors of urban deployment (e.g., vibration, wider temperature swings, dust/rain ingress, and outdoor illumination variability such as direct sunlight or rapid transitions between shaded and sunlit environments), which were not evaluated in this study.

The stability of these environmental factors ensured consistent sensor behavior during data acquisition, and no noticeable measurement degradation was observed in the laboratory setting. However, the impact of these stressors on ranging stability and long-term reliability remains to be validated in field conditions.

3.2.1. Sensor data transmission firmware

Both TOF sensors were assigned appropriate addresses for I2C communication, and the microcontroller read their data and transmitted the frames to the PC via a USB serial link (UART/CDC) every 200 ms (5 Hz), as shown in Algorithm 1. The microcontroller was programmed in C using Visual Studio Code with the PlatformIO extension.

Algorithm 1: Reading and sending data from the VL53L7CX sensor. The data is transmitted via the I2C protocol, with the first sensor assigned to address 0x44 and the second sensor to address 0x29.

```
Input: Keys s, t, e.
Output: Sensor data (2x[8×8 Matrix]).
Setup: Variables, Timer, USB serial (UART/CDC) and I2C protocol initialization.
       Initializing and configuring sensors.
while 1 do
    if TimerInterrupt == true then
        if Reading ready in sensor at ADDR 0x44 then
            Sending data from 0x44 via USB serial (UART/CDC);
        end
        if Reading ready in sensor at ADDR 0x29 then
            Sending data from 0x29 via USB serial (UART/CDC);
        end
    end
end
```

This transmission period was deliberately selected to ensure deterministic synchronization and logging stability during dataset acquisition. This sampling strategy is optimized for the static conditions of the current study, while the hardware remains capable of up to 60 Hz (≈16.7 ms per frame) for high-frequency dynamic monitoring applications.

Algorithm 1 reflects the static acquisition protocol used in this study. For dynamic or vibration-prone scenarios, concurrent acquisition strategies and reduced polling intervals would be required.

Once the data had been successfully transmitted, the analysis began, followed by visualization and attempts to represent the data in both 2D and 3D formats.

3.2.2. Data reception at PC

The first step was to retrieve data from each sensor regarding its captured point matrix image, separated into two 8×8 lists. Since the data format on the USB serial (UART/CDC) was a *String*, it had to be split into two parts corresponding to each sensor's data.

To analyze the data obtained from the sensors, various functions were applied, such as data scaling, averaging a series of samples, and calculating their standard deviation. These operations helped evaluate the sensor's reading performance and the margin of error in their results.

For visualization, the raw TOF distance measurements (expressed as absolute distances in millimeters) were linearly normalized to a [0,1] range. This normalization does not alter the physical meaning of the measurements, but provides a bounded and dimensionless representation that facilitates visualization. Specifically, for each pixel distance d, visualization normalization was computed as $x = (d - d_{\min})/(d_{\max} - d_{\min})$, where $d_{\min}$ and $d_{\max}$ denote the minimum and maximum distances observed within the sensor's 8×8 depth matrix for the displayed frame (or visualization window). This per-frame normalization does not affect the absolute-distance values used in the experimental pipeline. As a result, normalized values depend on the specific scene geometry and do not necessarily span the full [0,1] interval in all examples.

For these visualization purposes, the normalized values were inverted ($y_i = 1 - x_i$) so that closer points appear with higher values, providing a more intuitive depth representation.

Detailed plots were created using Python libraries such as *Matplotlib* to display the scaled measured distances. These plots, in both 2D and 3D, depicted contours or surfaces based on the measured distances.

In Fig. 8, a 3D representation of the distances measured by the sensor matrix is shown, along with other possible representations. On

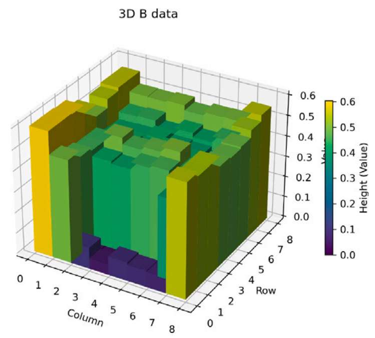


Fig. 8. 3D visualization of the depth values measured by sensor B for the configuration shown in Fig. 7, shown from the sensor viewpoint (i.e., as if the sensor were taking a top-down depth snapshot). Bar height represents the visualization-normalized depth (inverted so that closer points appear higher).

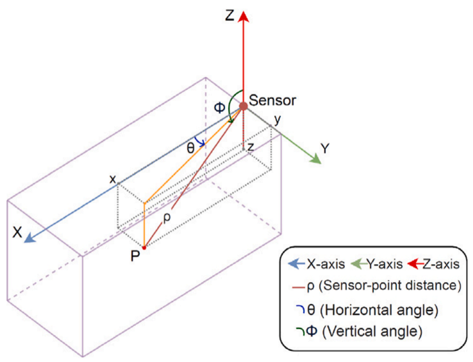


Fig. 9. Representation of a point captured by the sensor (sensor frame).

the *column* axis, the lateral edge positions ('0' and '8') exhibit taller bars and lighter colors, indicating shorter distances in those areas. Toward the center, the bar heights decrease, and the color shifts to greenish tones, representing greater distances toward the back and bottom of the box. On the row axis, lower values (purple tones) correspond to larger detected distances. However, an elevation in the center of the matrix suggests the presence of an obstacle, as a linear decrease in bar height toward the back of the box would be expected in its absence (indicating farther detected distances).

As mentioned earlier, TOF sensor provides a matrix of distance values ρ, each corresponding to a pixel defined by its vertical and horizontal position within the sensor's field of view. This measurement can be expressed in *spherical coordinates*, where ρ is the radial distance, ϕ the polar angle, and θ the azimuth angle. In the case of the VL53L7CX sensor (8×8 resolution), the field of view is approximately 60° in both directions, so each pixel represents an angular step of about 7.5°.

The mapping from *spherical* to *Cartesian coordinates* in the sensor frame follows the standard formulation (see Eqs. (1), (2), and (3)) [81]:

$$x = \rho \sin\phi \cos\theta \tag{1}$$

$$y = \rho \sin\phi \sin\theta \tag{2}$$

$$z = \rho \cos\phi \tag{3}$$

This representation yields the three-dimensional position of each measured point in the sensor's local reference system. To align these measurements with the global coordinate system of the box, a rigid-body transformation (rotation and translation) is applied, accounting for the sensor's position and orientation on the container. A schematic of the measurement geometry in the *sensor frame* is shown in Fig. 9.

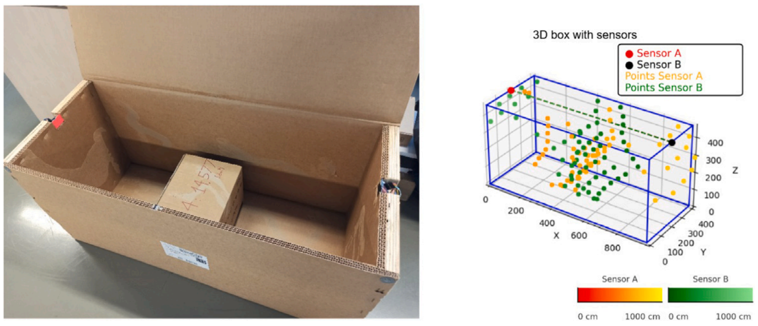


Fig. 10. 3D representation of the points captured by both sensors.

In Fig. 10, the representation of the box is shown, with points detected by each sensor (sensor 'A' in orange and sensor 'B' in green). Each 3D point cloud is color-coded according to distance, with the accompanying legend indicating the measured depth values from the TOF sensor. This improves the interpretability of spatial patterns and occlusions within the cargo space.

Following this visualization process, the data were stored either automatically or manually, depending on the selected mode. In manual mode, a single sample was saved, while in automatic mode, data were continuously stored until the point at which the box positions needed to be changed, without user intervention.

In Algorithm 2, the storage procedure is detailed, where the box or combination of boxes is recorded. The random arrangement of the boxes ensured more diverse measurements for the neural network in each case and increased the likelihood of encountering occlusions and challenging configurations, thereby improving the network's learning process.

Algorithm 2: Reading from USB serial (UART/CDC) and saving to dataset.

```
Input: Keys s (Start), a (Auto), m (Manual), Sensor data (2x[8×8 Matrix]).
Output: CSV Volume Dataset.
Setup: Initialize the USB serial (UART/CDC) and keys reading.
while True do
    if s key pressed and Data on the USB serial (UART/CDC) then
        Read USB serial (UART/CDC);
        if a key pressed then
            if samples < 40 then
                Split Data;
                Flatten Data; /*1 row, 128 val                    */
                CSV Dataset ← Data and Target values;
            else if positions < 60 then
                print("Change box!");
            else
                print("Change position!");
            end
        else if m key pressed then
            if samples < 20 then
                Split Data;
                Draw box and points;
                if Save sample then
                    Flatten Data; /*1 row, 128 val                */
                    CSV Dataset ← Data and Target values;
                end
            end
        end
    end
end
```

3.3. Dataset design

Each entry in the dataset consisted of sensor data as described previously, along with target values corresponding to each box combination. First, a list of 128 values was recorded, representing the matrices from the two sensors (2×64 distances) stacked together. Then, the entry included target values that described the data: the *case number*, indicating the specific combination of boxes being measured.

Table 6
Simplified representation of the dataset. The values a00 and a77 represent the readings from row 0, column 0, and row 7, column 7 of sensor 'A', respectively. Similarly, b00 and b77 correspond to the readings from row 0, column 0, and row 7, column 7 of sensor 'B'. *Target values* refers to the same target values mentioned in Table 7. The dataset comprises a total of 63 cases, each containing 2400 samples, amounting to a total of 151,200 samples.

a00 ⋯ a77, b00 ⋯ b77	Target values
2×64	1
…	…
2×64	63
Total samples: 151,200	

Table 6 provides a simplified representation of the stored dataset. Additionally, Table 7 outlines the various measurement cases, specifying the introduced volumes, the total sum of these volumes, and the percentage of free space, which is scaled between '0' and '1' and calculated using Eq. (4).

$$\%FreeSpace = \frac{VolBox - VolT}{VolBox} \tag{4}$$

In Eq. (4), we define *FreeSpace* as the internal volume of the container that remains unoccupied by any object. It is calculated by subtracting the total occupied volume ($VolT = \sum vol_i$) from the container's available space (VolBox).

3.4. Dataset scope and generalization limits

The experimental dataset used in this study is intentionally constrained to a finite set of structured cuboidal packages. All real-world measurements are derived from combinations and spatial arrangements of six predefined box sizes, which define the effective training and evaluation domain of the model.

Importantly, the performance of the model on previously unseen box dimensions (e.g., a cuboidal package of 7 dm^3 not included in the original set) or on substantially different object geometries is not validated by the present experimental dataset. The reported results should therefore be interpreted as representative of this constrained operational domain.

To explore this features and assess the potential of extending the dataset beyond the original box set, additional physics-based simulations were conducted using synthetic data. These simulations are not used to claim generalization performance, but rather to investigate the feasibility of simulation-based extensions and to characterize the expected sim-to-real gap. The simulation methodology and results are reported in Appendix and discussed as directions for future work.

Table 7
Measurement cases data. The first value indicates the combination, the following six values represent the volumes of each introduced box, the penultimate value is the sum of the previous volumes, and the last value is the percentage of free space calculated according to (4), where VolBox is the volume of the main box (182.282 dm^3) and VolT is the total load volume in each case.

Target values	Vol1	Vol2	Vol3	Vol4	Vol5	Vol6	VolT	%FreeSpace
1	1.676	0	0	0	0	0	1.676	0.990805455
2	0	4.531	0	0	0	0	4.531	0.97514291
3	0	0	9.478	0	0	0	9.478	0.948003643
4	0	0	0	14.577	0	0	14.577	0.920030502
5	0	0	0	0	22.815	0	22.815	0.874836791
6	0	0	0	0	0	81.537	81.537	0.552687594
7	1.676	4.531	0	0	0	0	6.207	0.965948366
8	1.676	0	9.478	0	0	0	11.154	0.938809098
9	1.676	0	0	14.577	0	0	16.253	0.910835957
10	1.676	0	0	0	22.815	0	24.491	0.865642247
11	1.676	0	0	0	0	81.537	83.213	0.543493049
12	0	4.531	9.478	0	0	0	14.009	0.923146553
13	0	4.531	0	14.577	0	0	19.108	0.895173413
14	0	4.531	0	0	22.815	0	27.346	0.849979702
15	0	4.531	0	0	0	81.537	86.068	0.527830504
16	0	0	9.478	14.577	0	0	24.055	0.868034145
17	0	0	9.478	0	22.815	0	32.293	0.822840434
18	0	0	9.478	0	0	81.537	91.015	0.500691237
19	0	0	0	14.577	22.815	0	37.392	0.794867294
20	0	0	0	14.577	0	81.537	96.114	0.472718096
21	0	0	0	0	22.815	81.537	104.352	0.42752438
22	1.676	4.531	9.478	0	0	0	15.685	0.913952008
23	1.676	4.531	0	14.577	0	0	20.784	0.885978868
24	1.676	4.531	0	0	22.815	0	29.022	0.840785157
25	1.676	4.531	0	0	0	81.537	87.744	0.51863596
26	1.676	0	9.478	14.577	0	0	25.731	0.8588396
27	1.676	0	9.478	0	22.815	0	33.969	0.813645889
28	1.676	0	9.478	0	0	81.537	92.691	0.491496692
29	1.676	0	0	14.577	22.815	0	39.068	0.785672749
30	1.676	0	0	14.577	0	81.537	97.79	0.463523551
31	1.676	0	0	0	22.815	81.537	106.028	0.418329841
32	0	4.531	9.478	14.577	0	0	28.586	0.843177055
33	0	4.531	9.478	0	22.815	0	36.824	0.797983344
34	0	4.531	9.478	0	0	81.537	95.546	0.475834147
35	0	4.531	0	14.577	22.815	0	41.923	0.770010204
36	0	4.531	0	14.577	0	81.537	100.645	0.447861007
37	0	4.531	0	0	22.815	81.537	108.883	0.402667296
38	0	0	9.478	14.577	22.815	0	46.87	0.742870936
39	0	0	9.478	14.577	0	81.537	105.592	0.420721739
40	0	0	9.478	0	22.815	81.537	113.83	0.375528028
41	0	0	0	14.577	22.815	81.537	118.929	0.347554887
42	1.676	4.531	9.478	14.577	0	0	30.262	0.8339825
43	1.676	4.531	9.478	0	22.815	0	38.5	0.788788
44	1.676	4.531	9.478	0	0	81.537	97.222	0.466639602
45	1.676	4.531	0	14.577	22.815	0	43.599	0.760815659
46	1.676	4.531	0	14.577	0	81.537	102.321	0.438666462
47	1.676	4.531	0	0	22.815	81.537	110.559	0.393472751
48	1.676	0	9.478	14.577	22.815	0	48.546	0.733676392
49	1.676	0	9.478	14.577	0	81.537	107.268	0.411527194
50	1.676	0	9.478	0	22.815	81.537	115.506	0.366333483
51	1.676	0	0	14.577	22.815	81.537	120.605	0.338360343
52	0	4.531	9.478	14.577	22.815	0	51.401	0.718013847
53	0	4.531	9.478	14.577	0	81.537	110.123	0.395864649
54	0	4.531	9.478	0	22.815	81.537	118.361	0.350670938
55	0	4.531	0	14.577	22.815	81.537	123.46	0.322697798
56	0	0	9.478	14.577	22.815	81.537	128.407	0.29555853
57	1.676	4.531	9.478	14.577	22.815	0	53.077	0.708819302
58	1.676	4.531	9.478	14.577	0	81.537	111.799	0.386670105
59	1.676	4.531	9.478	0	22.815	81.537	120.037	0.341476394
60	1.676	4.531	0	14.577	22.815	81.537	125.136	0.313503253
61	1.676	0	9.478	14.577	22.815	81.537	130.083	0.286363985
62	0	4.531	9.478	14.577	22.815	81.537	132.938	0.270701441
63	1.676	4.531	9.478	14.577	22.815	81.537	134.614	0.261506

4. ANN modeling

This section outlines the methodology and data processing obtained from the sensors to determine the load volume of the cargo bike. For this part of the project, the Python programming language was employed, utilizing tools such as Spyder and Google Colab. The following libraries were used:

- ***Scikit-learn***: A machine learning library optimized for implementing classical machine learning techniques [82–84].
- ***TensorFlow***: A deep learning library developed by Google, widely used due to its flexibility and efficiency. Additionally, its optimized version, *TensorFlow Lite*, is particularly suited for development on embedded devices [82,85,86].

The procedure for processing using artificial neural networks is illustrated in Fig. 11.

The procedure began with an initial phase where the complete dataset was preprocessed and used to train the neural network. The objective of this phase was to identify the type of model best suited to the task, focusing on performance. Classic *Machine Learning* models were considered alongside *deep learning* models with both dense and convolutional layers. Hyperparameter tuning was performed to optimize the models. As this task focuses on continuous value estimation rather than classification, regression-based models were prioritized.

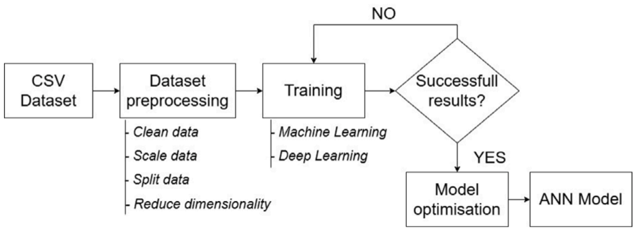


Fig. 11. Procedure for processing with ANN.

In a second phase, a reduced version of the candidate model was developed, prioritizing minimal memory footprint while preserving decision-grade performance for embedded deployment. Rather than aiming at precise volumetric metrology, the operational requirement in last-mile delivery is typically to distinguish coarse occupancy regimes (e.g., whether the load is closer to 30% or 80% of the available capacity) to support routing and handling decisions. On this basis, a total error margin of approximately 10%–15% was set as an acceptable design target for the application. Consequently, training in this phase focused on the *FreeSpace* target, expressed as a normalized ratio in the $[0, 1]$ range, which directly represents the fraction of unoccupied volume and is better aligned with the intended use of the system than an absolute volume output.

4.1. ANN modeling objectives

To evaluate the neural network's performance under different inference conditions within the predefined experimental domain, distinct modeling objectives were defined for the selected model. These objectives assess the model's inference capability under instance-level exclusion scenarios (e.g., unseen box instances or restricted training cases), rather than generalization to novel box geometries, dimensions, or sensing layouts:

- ***Inference capability of unseen box instances within the predefined set.*** Cases involving boxes '2' and '4' were excluded from training. The network was trained on the remaining combinations of boxes.
- ***Inference capability under restricted training cases within the same geometric domain.*** For a stricter evaluation, the model was trained exclusively on cases 1, 3, 5, 6, which correspond with their respective individual boxes, as well as the combination of these four boxes (case 50). Boxes '2' and '4' were excluded to ensure that the test set included boxes never seen during training. All other cases were reserved for testing, providing a challenging assessment of the model's ability to infer unseen combinations.

Table 8 summarizes the inference tests and their objectives.

4.2. Dataset preprocessing

The dataset was loaded into the software and preprocessed using the machine learning and deep learning libraries previously mentioned. The preprocessing included data validation, cleaning, partitioning, scaling and dimensionality reduction. These steps ensured the dataset was ready for training and testing. The dataset was processed as shown in Fig. 11.

Table 8
Table of inference tests with training data, test data, and the intended objective.

Test	Train data	Test data	Objective
Inference capability of unseen boxes	Cases where boxes '2' and '4' are missing	Cases with boxes '2' and '4' introduced	Load volume detection without prior knowledge of the size of specific boxes
Inference capability of selected cases	Cases 1, 3, 5, 6 and 50 (cases with individual boxes selected and their combination)	Other cases	Extreme test to check if it can infer on combinations of boxes and other cases not seen during training

4.2.1. Dataset partitioning

Model selection and hyperparameter tuning were performed using training-only data, while final performance is always reported on held-out samples defined by the inference objectives in 4.1. Accordingly, samples belonging to cases reserved for testing were never used during model selection or training.

The target value columns were selected and excluded from the training set, leaving only the 128 sensor measurement values per sample. A specific target value was then selected for the neural network. Initially, the target value 'volTotal' was used, and subsequently the 'FreeSpace' target value was considered. This dual-target value approach enables a comparative analysis of training outcomes.

4.2.2. Cleaning and scaling

To prepare the dataset for optimal neural network input, missing or anomalous values were addressed. The dataset was normalized to standardize the input range and enhance model performance, a crucial step for effective neural network training. The *StandardScaler()* function from the *sklearn* library was used to subtract the mean and divide by the standard deviation for each feature. This scaler was integrated into a preprocessing pipeline to uniformly process both training and test sets.

Additionally, Principal Component Analysis (PCA) [87] was performed to reduce data dimensionality and capture the most significant features [88].

4.3. Training

The objective was to estimate the volume of one or more boxes within an environment, framing the problem as a regression task. After partitioning the training set, both classical machine learning and deep learning methods were employed to train the models.

To assess model performance, metrics such *Mean Squared Error (MSE)* [82,89], *Mean Absolute Error (MAE)*, and *Root Mean Squared Error (RMSE)* were used. These metrics calculate the average error between the actual values (y_i) and model predictions ($\hat{y}_i$), as defined in Eqs. (5), (6), and (7).

$$MSE = \frac{1}{n}\sum (y_i - \hat{y}_i)^2 \tag{5}$$

$$MAE = \frac{1}{n}\sum |y_i - \hat{y}_i| \tag{6}$$

$$RMSE = \sqrt{\frac{1}{n}\sum (y_i - \hat{y}_i)^2} \tag{7}$$

Additionally, the *Coefficient of Determination (R^2)* was included to evaluate the proportion of variance in the target variable that is captured by the model. Unlike the previous error-based metrics, R^2 provides an interpretable score ranging from 0 to 1 (or negative if the model performs worse than a constant baseline), where a value closer to 1 indicates better predictive performance and fit. This metric is defined in Eq. (8).

$$R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2} \tag{8}$$

Where $\hat{y}_i$ is the predicted value, and $\bar{y}$ is the mean of the actual values. This metric complements the error-based analysis and provides additional insight into the generalization capability of the trained model.

Given that the dataset size allowed manageable computation times, *cross-validation* was applied to classical machine learning models. The training set was divided into k equal parts (k-folds), typically $k = 5$ or $k = 10$ [82]. In each iteration, one partition was reserved for testing while the model was trained on the others, ensuring robust model evaluation.

Once the most promising models were identified, hyperparameters were fine-tuned using *RandomSearchCV*. This method selects random values within predefined parameter ranges [82], offering a computationally efficient alternative to exhaustive approaches. *RandomSearchCV* incorporates cross-validation to evaluate parameter combinations, balancing thoroughness and efficiency in the search process.

4.3.1. Classical and deep learning methods

Among classical methods, models such as *linear regression*, *decision trees*, and *random forest* were explored. Decision trees are intuitive and applicable to both classification and regression tasks. However, these models are often sensitive to data, prone to overfitting, and can exhibit instability. Small variations in the data can lead to significantly different tree structures. To address these issues, the *Random Forest* method [82,90] was employed. It averages the results of multiple trees, reducing instability and enhancing precision. This method involves an ensemble of trees trained with randomly selected examples, producing a final output by averaging predictions to determine the optimal result.

Given the spatial nature of the data, models based on *deep learning*, particularly *Convolutional Neural Networks (CNNs)*, were explored. *CNNs* combine convolutional and pooling layers to extract features and reduce complexity [82,91]. This approach reduces computational load and improves robustness against spatial displacements [92].

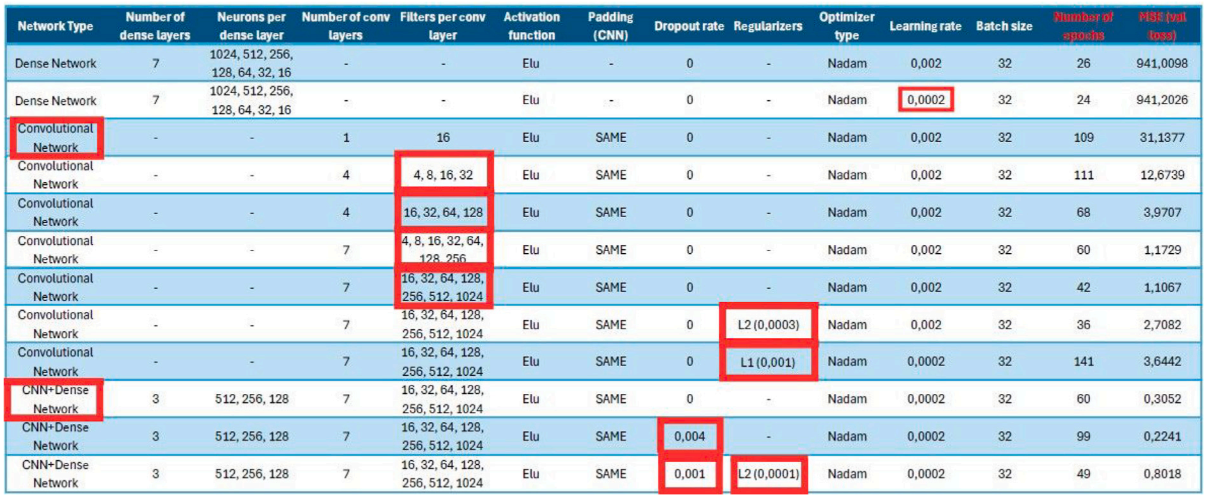
During the training of deep learning models, 20% of the training set was allocated for validation. This allowed for the tuning of hyperparameters such as *learning rate*, the number of layers and neurons, optimizer, *mini-batch* size, and regularization strategies. Fig. 12 illustrates the hyperparameter tuning process. Each model underwent at least five training iterations to evaluate thoroughly its performance, with the best-performing model from these sessions being selected.

Among the models evaluated, a neural network with 7 convolutional layers was selected. Each convolutional layer is followed by pooling layer to manage the network's complexity. The number of filters in the convolutional layers increases progressively, starting at 16 and doubling in subsequent layers: 32, 64, 128, 256, 512, and 1024 filters.

Next, a *flattening layer* is added to convert the output of the last convolutional layer into a one-dimensional vector. Subsequently, three dense layers are included, containing 512, 256, and 128 neurons, respectively, with *dropout layers* between them to reduce the risk of overfitting. Finally, an output layer with a single neuron is added to predict the volume value.

The comparative performance of the various models is summarized in Table 9.

To accommodate the memory constraints of embedded systems while preserving performance, a simplified version of the final model was developed with fewer layers. This version included 4 convolutional layers with 2, 4, 8, and 16 filters, respectively. Additionally, the model was evaluated using an alternative target value, *FreeSpace*, instead of the original *volTotal*, to assess performance under different configurations and explore the benefits of using a scaled target value. The optimized model is summarized in Table 10.



Network Type	Number of dense layers	Neurons per dense layer	Number of conv layers	Filters per conv layer	Activation function	Padding (CNN)	Dropout rate	Regularizers	Optimizer type	Learning rate	Batch size	Number of epochs	MSE (val loss)
Dense Network	7	1024, 512, 256, 128, 64, 32, 16	-	-	Elu	-	0	-	Nadam	0,002	32	26	941,0098
Dense Network	7	1024, 512, 256, 128, 64, 32, 16	-	-	Elu	-	0	-	Nadam	0,0002	32	24	941,2026
Convolutional Network	-	-	1	16	Elu	SAME	0	-	Nadam	0,002	32	109	31,1377
Convolutional Network	-	-	4	4, 8, 16, 32	Elu	SAME	0	-	Nadam	0,002	32	111	12,6739
Convolutional Network	-	-	4	16, 32, 64, 128	Elu	SAME	0	-	Nadam	0,002	32	68	3,9707
Convolutional Network	-	-	7	4, 8, 16, 32, 64, 128, 256	Elu	SAME	0	-	Nadam	0,002	32	60	1,1729
Convolutional Network	-	-	7	16, 32, 64, 128, 256, 512, 1024	Elu	SAME	0	-	Nadam	0,002	32	42	1,1067
Convolutional Network	-	-	7	16, 32, 64, 128, 256, 512, 1024	Elu	SAME	0	L2 (0,0003)	Nadam	0,002	32	36	2,7082
Convolutional Network	-	-	7	16, 32, 64, 128, 256, 512, 1024	Elu	SAME	0	L1 (0,001)	Nadam	0,0002	32	141	3,6442
CNN+Dense Network	3	512, 256, 128	7	16, 32, 64, 128, 256, 512, 1024	Elu	SAME	0	-	Nadam	0,0002	32	60	0,3052
CNN+Dense Network	3	512, 256, 128	7	16, 32, 64, 128, 256, 512, 1024	Elu	SAME	0,004	-	Nadam	0,0002	32	99	0,2241
CNN+Dense Network	3	512, 256, 128	7	16, 32, 64, 128, 256, 512, 1024	Elu	SAME	0,001	L2 (0,0001)	Nadam	0,0002	32	49	0,8018

Fig. 12. Hyperparameter search across different deep learning models. Adjustments to hyperparameters compared to the previous trained model are highlighted in *red*. Ultimately, the penultimate hyperparameter combination was selected, as it achieved the lowest final MSE. This model integrates convolutional and fully connected layers, striking an optimal balance between performance and computational efficiency.

Table 9
Performance comparison of trained neural network models.

Model	Error	Size
Linear regression	High (> 50%)	Very small (10 s of KB)
Decision tree regressor	Moderate (< 20%)	Very small (10 s of KB)
Random forest regressor	Very low (< 5%)	Large (MB range)
Fully connected network	High (> 200%)	Small (100 s of KB)
Convolutional network	Low (< 10%)	Very small (10 s of KB)
CNN + Fully connected	Very low (< 5%)	Large (MB range)

Table 10
Details of the optimized CNN model.

Layer (Type)	Output shape	Parameters
Conv1D (conv1d)	(None, 128, 2)	8
MaxPooling1D (max_pooling1d)	(None, 64, 2)	0
Conv1D (conv1d_1)	(None, 64, 4)	28
MaxPooling1D (max_pooling1d_1)	(None, 32, 4)	0
Conv1D (conv1d_2)	(None, 32, 8)	104
MaxPooling1D (max_pooling1d_2)	(None, 16, 8)	0
Conv1D (conv1d_3)	(None, 16, 16)	400
MaxPooling1D (max_pooling1d_3)	(None, 8, 16)	0
Flatten (flatten)	(None, 128)	0
Dense (dense)	(None, 1)	129
Total parameters		**669 (2.61 KB)**

The selection of the final convolutional neural network (CNN) model was based on a trade-off between performance and implementation constraints. Initially, deeper CNNs with up to 7 convolutional layers and dense layers were tested. These models reached Mean Squared Errors (MSE) as low as 0.22 dm^3, but their memory usage exceeded the capacity of the target embedded platform (ESP32-S3).

The 4-layer model was selected after evaluating multiple combinations of convolutional layers and filters. While simpler networks failed to capture spatial depth patterns, and deeper networks exceeded the network limitations, the 4-layer CNN offered a balance between computational cost and performance. It demonstrated good convergence behavior and consistent performance across multiple load configurations, including unseen combinations.

Despite the compact size of the optimized network (669 parameters), the learning problem remains highly structured due to the fixed sensing geometry and the constrained operational domain of the dataset. The input consists of two 8×8 TOF depth maps (128 distance values) acquired from fixed viewpoints within a container of constant dimensions. Under these conditions, the model is not required to reconstruct a full 3D shape of the load, but to estimate the unoccupied space from sparse depth patterns that recurrently encode occupancy cues (e.g., near-wall returns, depth discontinuities, and shadowed regions) within the same container geometry. Therefore, the task is better interpreted as a statistical inference problem under partial observability, rather than explicit occlusion reconstruction.

From a modeling standpoint, a small CNN is appropriate because convolution and pooling exploit weight sharing and local structure, allowing the network to extract robust spatial features with far fewer parameters than fully-connected alternatives. This constrained capacity also acts as an implicit regularizer, reducing overfitting and improving stability for embedded deployment. Regarding measurement uncertainty, the multizone TOF sensors exhibit distance-dependent ranging errors on the order of 5%–6% at typical operating distances, which impose a non-negligible physical lower bound on volumetric accuracy (on the order of a few dm^3 for a $\sim$182 dm^3 container, depending on range and surface conditions). However, the dominant source of error in high-occupancy regimes is not random ranging noise alone, but occlusions and unobserved regions arising from sparse, line-of-sight TOF sampling.

To anchor this lower bound with an order-of-magnitude estimate, each sensor provides an 8×8 depth grid (64 zones) over a container footprint of approximately 9.70×4.05 dm^2, i.e., about 0.61 dm^2 per zone. A representative 5%–6% depth error at meter-scale ranges corresponds to centimeter-scale perturbations in the measured depth, which translates into dm^3-scale uncertainty when propagated over the container footprint. This analytical estimate isolates the fundamental hardware-limited error floor. While secondary effects such as multipath or occlusions are addressed via neural inference, this derivation establishes the physical precision boundary inherent to the sensing modality under the container's footprint.

Although the architecture was tailored for embedded inference, comparisons were also made with more complex models during early experimentation. These included:

- Fully-connected MLPs with up to 7 hidden layers - poor spatial feature capture and prone to overfitting.
- Pre-trained CNNs (e.g., VGG16, MobileNet) - discarded due to input incompatibility (arrays of 1×128 values) and excessive memory usage.

The chosen architecture thus provides an efficient trade-off: lightweight enough for microcontroller deployment and accurate enough for real-world generalization, making it suitable for embedded logistics systems.

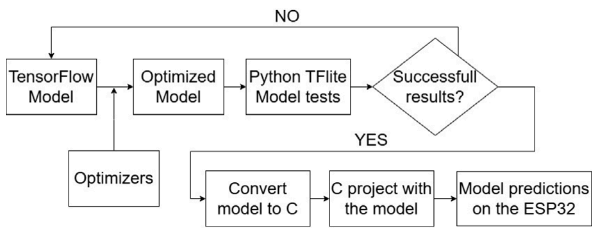


Fig. 13. Procedure for microcontroller deployment.

Once the best model was selected and before deployment on the microcontroller, the inference tests were conducted using this optimal model.

4.4. Model optimization and firmware implementation

After completing these inference tests and confirming that the results were free of significant errors, the optimized model was refined further and transformed into a compact version suitable for microcontroller deployment. TensorFlow Lite was used for this purpose, ensuring compatibility with the TensorFlow neural network programming library. Fig. 13 illustrates the workflow for model optimization and deployment on embedded systems.

Algorithm 3: Deployment on the microcontroller.

```
Input: Keys $s$, $t$, $e$, TFlite model.
Output: Model prediction.
Setup: Variables, Timer, USB serial (UART/CDC), and I2C protocol initialization.
       Initialize and configure sensors. Create memory space for the model, initialize
       resolver, build interpreter, allocate memory for tensors and create model input
       and output.
while 1 do
    if TimerInterrupt == true then
        Read sensors data; /*1 row, 128 values                    */
        Scale data; /*Dataset mean and std.dev                    */
        Model input ← scaled data;
        Perform inference;
        Show prediction;
    end
end
```

To correctly utilize the model in C, the memory size reserved for storing the model on the microcontroller had to be defined. Additionally, it was crucial to have a thorough understanding of the model's input and output dimensions to prevent issues related to dimensionality or prediction errors. The C program followed the structure outlined in Algorithm 3. This program was implemented with a *timerInterrupt* period of 40 ms achieving 25 Hz refresh rate; enough for real-time occupancy monitoring.

5. Experimental results

After training different models for volume estimation, results were obtained for two main approaches: machine learning and deep learning. Below, the results of each approach are presented, along with the final performance of the optimized model on a microcontroller.

5.1. Classical machine learning models training results

Initially, several machine learning models were evaluated using the *sklearn* library: *LinearRegression*, *DecisionTreeRegressor*, and *RandomForestRegressor*. The initial training results for each model are detailed in Table 11.

The linear model exhibited a higher error compared to the other models, indicating limited predictive capability, which was also reflected in its lower R^2 score. The decision tree produced near-zero error on the evaluated split, which is indicative of memorization (or potential split sensitivity) in highly structured input–output settings. The random forest also achieved very strong scores; however, these results should be interpreted as *dataset-dependent* and as a reference baseline within the present experimental protocol. These scores establish a benchmark for the predictive potential of each model family within the current dataset. While these results identify peak accuracy levels, the definitive selection for deployment is governed by the operational constraints of the target hardware, which are addressed in the subsequent comparative analysis of the final inference model.

Table 11
Results obtained after training classical machine learning models, with RMSE, MAE and R^2 calculated over five iterations for each model.

Model	RMSE (dm^3)	MAE (dm^3)	R^2
LinearRegression	10.9614	7.5013	0.9343
DecisionTreeRegressor	2.1968e−13	9.8369e−14	1.0
RandomForestRegressor	0.4687	0.1747	0.9998

Table 12
Cross-validation for the linear regression and decision tree regression.

Model	Scores (RMSE per fold) (dm^3)	Mean (dm^3)	Std. dev (dm^3)	R^2
Linear Regression	10.947, 10.867, 10.856, 11.064, 11.014, 10.9988, 10.880, 11.041, 11.055, 11.038	10.976	0.0777	0.9341
Decision Tree Regression	2.514, 2.734, 2.426, 2.526, 2.498, 2.5635, 2.779, 2.727, 2.892, 2.668	2.6328	0.1414	0.9962

The extremely small error reported for the decision tree (on the order of 10^{-13} dm^3) should not be interpreted as physically meaningful accuracy. It is an artefact of evaluating a high-capacity memorization model on a particular split of a highly structured dataset, where the tree can reproduce the training mapping almost exactly [93].

Thus, applying cross-validation was crucial to assess robustness for the models where it was computationally feasible (linear regression and the decision tree). The cross-validation outcomes are presented in Table 12.

The cross-validation results revealed that the decision tree was overfitting: the validation errors in each fold (RMSE, reported as *Scores*) were orders of magnitude higher than the near-zero error observed on the initial split, indicating strong split sensitivity, although they remained low in absolute terms. This confirms that the decision tree performance in Table 11 was not robust under resampling.

Although the random forest is typically less prone to overfitting than a single decision tree due to bagging and feature subsampling, it may still overfit under structured datasets such as the one considered here. For this reason, the random forest results are reported as a strong *reference baseline* for contextual benchmarking, rather than as evidence of a generally optimal model. Moreover, the memory footprint and computational cost of the random forest make it unsuitable for deployment on the target ESP32 platform, which motivates the selection of the compact CNN as the final embedded solution. Accordingly, Table 11 is used to contextualize achievable error levels on the present dataset, while the CNN is selected as the deployable model under the embedded constraints.

To rule out information leakage, the classical models (and the deep learning models) were trained using only the 128 TOF distance values as inputs. All case identifiers and volume-related fields were excluded from the feature matrix and used only to construct the regression targets. All preprocessing operations (imputation/scaling) are fitted on training data only and then applied to the held-out test set. In addition, the inference splits are defined at the case level (Section 3 and Table 8), so samples from the same case are never shared across training and testing.

Table 13
Deep learning model training results comparing the utilized target value and optimization performance.

Model	Target value	MSE (Train)	MSE (Validation)
CNN + Fully connected	volTotal (dm^3)	1.1903	1.0160
CNN + Fully connected	FreeSpace (%)	0.0014	0.0013
CNN (Optimized - 7 layers)	volTotal (dm^3)	8.1495	8.2923
CNN (Optimized - 7 layers)	FreeSpace (%)	0.0019	0.0018
CNN (Optimized - 4 layers)	FreeSpace (%)	0.0018	0.0018

In addition to the reported classical ML baselines, non-ML geometric heuristics commonly used in the literature – such as voxelization-based volume reconstruction or top-plane/heightfield integration – were not evaluated in this study. Under the present sensing configuration (two fixed 8 × 8 multizone TOF sensors and severe line-of-sight occlusions), such methods are dominated by partial observability and tend to introduce systematic bias, particularly in high-occupancy regimes, unless additional viewpoints or higher spatial resolution are available. Consequently, a fair comparison against these geometric approaches would require a different sensing configuration and is left as future work.

5.2. Deep learning training results

Several deep learning architectures were explored. Purely dense networks were tested under different hyperparameter configurations [94, 95] but yielded unsatisfactory performance, with errors above 900 dm^3. In contrast, convolutional neural networks (CNNs) achieved a significant improvement, reducing the error to below 10 dm^3. The selected CNN architecture combined convolutional and dense layers, but its initial size (±2.7 million parameters, several MB) was impractical for microcontroller deployment. Therefore, an optimized lightweight version was developed, containing only 669 parameters and a memory footprint of a few KB, while maintaining comparable performance. This reduced model demonstrates high potential for implementation on resource-constrained devices.

Table 13 presents the final training results, comparing the performance using the 'volTotal' target value in dm^3 and the 'FreeSpace' target value in percentages.

5.3. Inference tests results

Using the structure of the optimal 4-layer convolutional model, the network's input data was customized by creating specific splits based on the cases selected for the inference test set.

To evaluate the test set results, performance was assessed using standard regression metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), the coefficient of determination (R^2), and Pearson correlation, which quantifies the linear relationship between predicted and true values. Since the target variable (FreeSpace) is modeled in normalized form (0–1), MAE and RMSE are also reported in physical units (dm^3) by scaling with the container volume ($V_c = 182.28$ dm^3), i.e., $Error(\text{dm}^3) = Error(0{-}1) \times V_c$.

Table 14 summarizes the aggregate results for both inference stress tests. Both scenarios achieve $R^2 > 0.92$ and Pearson correlation above 0.96, indicating strong linear agreement between predicted and true *FreeSpace* at the dataset level. Note that these R^2 values are computed under case-level inference stress tests, and are therefore not directly comparable to the classical-model scores in Table 11 and Table 12, which were obtained under a different evaluation protocol. In physical units, the mean absolute error is approximately 8.1 (dm^3) (about 4.4% of the container volume), providing a direct interpretation of the observed error magnitude in volume estimation. Although a Random Forest regressor achieves slightly lower absolute error on this dataset, its memory footprint and computational cost make it unsuitable for embedded deployment. The optimized CNN therefore represents a more appropriate trade-off between performance and deployability under the ESP32-S3 constraints.

Fig. 14 complements the aggregate metrics by reporting the MAE per case in dm^3. The error is not uniform across cases: most configurations remain within a low-to-moderate error regime, while a limited number of cases exhibit substantially larger deviations. In the *unseen boxes* stress test Fig. 14a, the largest errors concentrate in a few configurations (e.g., case 24) and in several high-occupancy arrangements, where occlusions and shadow regions reduce the effective sensing coverage from the two TOF viewpoints. Importantly, the model does not "reconstruct" occluded geometry, it provides a statistical estimate of *FreeSpace* under partial observability, learned from the distribution of training configurations. In the *selected cases* stress test Fig. 14b, the error profile is more distributed across cases, with fewer extreme outliers, indicating that occupancy alone does not fully explain the deviations and that the dominant factor is the specific spatial arrangement and the resulting shadow regions. Errors are reported per-case in dm^3 (rather than relative percentage errors) to avoid interpretability artefacts when *FreeSpace* is small in near-full regimes, where percentage-based errors can appear inflated or misleading. Therefore, the reported MAE should be interpreted as a combination of intrinsic sensing uncertainty (few dm^3 scale) and occlusion/observability effects that dominate near-full regimes.

In near-full configurations, a moderate absolute deviation in the estimated free volume can correspond to a large relative discrepancy with respect to the remaining capacity. For this reason, the analysis focuses on absolute errors (dm^3) and explicitly reports worst-case behavior in high-occupancy regimes.

To further examine the highest-packing-density regime, Table 15 reports the MAE (in dm^3) for cases 57–63 in both inference stress tests. While errors increase for several of these cases, the ranking differs between stress tests, supporting that occlusion patterns induced by the arrangement drive the largest deviations.

In this near-full subset (57–63), the worst-case MAE reaches approximately 19.8 dm^3 (case 63 in the unseen-boxes stress test), i.e, about 10.9% of the container capacity (182.282 dm^3). This quantifies the practical limit of the current two-viewpoint configuration under high packing density, where shadow regions dominate the error.

This behavior is most relevant operationally because high-occupancy conditions coincide with the most critical decisions about whether additional cargo can be accepted. Therefore, near-full estimates should be treated conservatively and are identified as a priority target for future improvements (e.g., additional viewpoints or sensing modalities).

5.4. Microcontroller implementation results

The optimized model was simplified and adapted for deployment on a microcontroller using TensorFlow Lite. The first step focused on ensuring that the simplified model preserved comparable occupancy-estimation performance.

Subsequently, the model was converted into a binary format compatible with C for deployment on the microcontroller. Table 16 presents some real-time results obtained after implementing the model on the microcontroller.

The results obtained on the microcontroller were excellent, meeting the expectations established during Python testing. For instance, when using only box 4, the expected value for FreeSpace was 0.92 (92%), and the microcontroller's results closely matched this value within an acceptable error margin. This demonstrates the model's effectiveness in performing accurate inferences. Similarly, when testing other box combinations with higher occupation, the precision remained high: in cases with expected values of 73.37%, 38.67%, and 26.15%, the microcontroller-based model produced results very close to these values, with minimal errors. These findings indicate that the model performs well without overfitting.

Table 14
Comparison of inference metrics for each test scenario. MAE and RMSE are reported both on the normalized target (0–1) and in physical units (dm^3), using $Error(dm^3) = Error(0-1) \times 182.282$ dm^3.

Inference scenario	MSE (0–1)	MAE (0–1)	MAE (dm^3)	RMSE (0–1)	RMSE (dm^3)	R^2 score	Pearson
Inference capability of unseen boxes	0.0044	0.0448	8.1652	0.0661	12.0415	0.9292	0.9682
Inference capability of selected cases	0.0045	0.0444	8.0891	0.0672	12.2453	0.9233	0.9642

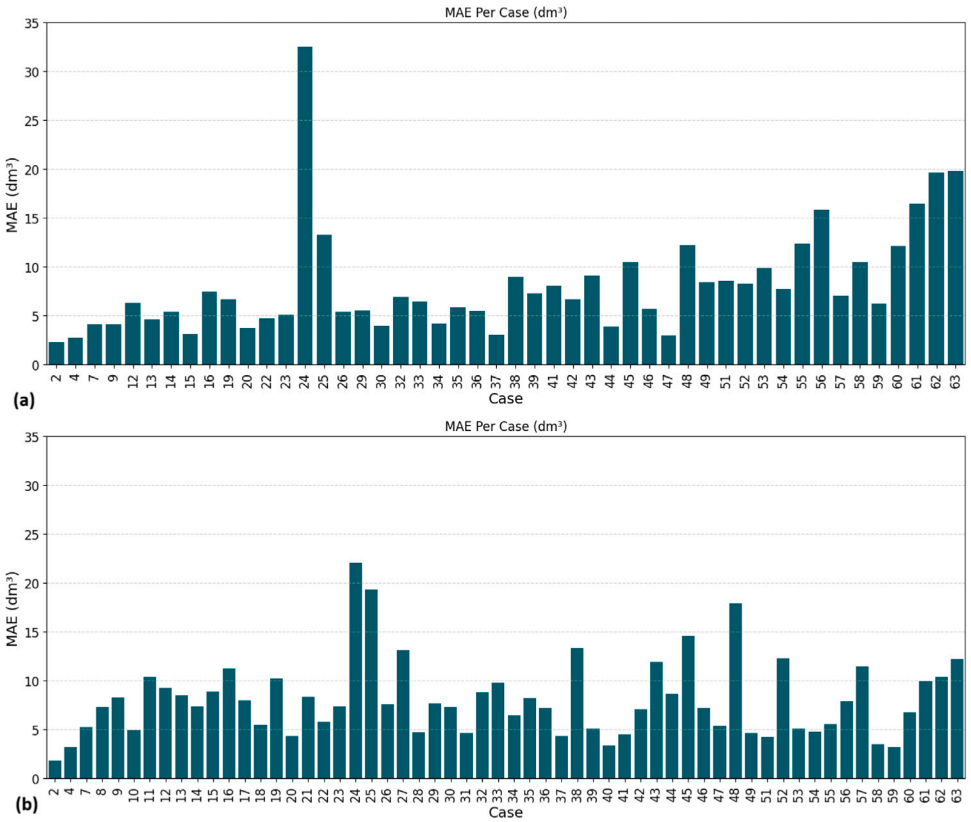


Fig. 14. Mean absolute error per case (MAE in dm^3) for the two inference stress tests: (a) *unseen boxes* (training set excluding boxes 2 and 4; evaluation on configurations containing those boxes), and (b) *selected cases* (training set limited to cases 1, 3, 5, 6 and their full combination, case 50; evaluation on the remaining cases). Each bar represents the MAE computed across all measurements for the corresponding case.

Table 15
MAE per case (dm^3) for the high-occupancy configurations (cases 57–63) in both inference stress tests.

Case	Unseen boxes	Selected cases
57	7.098	11.466
58	10.487	3.492
59	6.220	3.192
60	12.122	6.801
61	16.511	9.974
62	19.654	10.406
63	19.834	12.245

Table 16
Real-time prediction results using the microcontroller. Predictions (%) represent the mean of four samples taken. The reported inference latency corresponds to the CNN forward pass executed on the ESP32-S3 using int8 precision.

Test case	Target value (%)	Predictions (%)
With box 4 (Case 4)	92.003	92.14
Without boxes 2, 6 (Case 48)	73.368	71.57
Without box 5 (Case 58)	38.667	36.24
With all boxes (Case 63)	26.151	27.96
Average inference latency (ESP32-S3, int8): 20 ms (± 5 ms)		

To assess the consistency between the PC-based inference and its deployment on the microcontroller using TensorFlow Lite, the original dataset was utilized. A test set was preprocessed in a similar manner and sent to the microcontroller via serial communication for inference on a set of 1000 samples. The ESP32 returned the results through the same USB serial (UART/CDC), which were then compared with the target values.

Subsequently, predictions made in Python were compared with those obtained from the microcontroller to ensure that performance using TensorFlow Lite remained consistent on the ESP32. The differences between the two predictions were calculated, and the mean of these differences was used to determine the average error across the 1000 dataset samples. Table 17 presents the results for 10 samples from the test set.

Table 17
Comparison of prediction results for the first ten test samples using TensorFlow Lite on ESP32 firmware and Python software.

Target value (%)	Firmware pred. (%)	Python pred. (%)	Abs. Difference (%)
37.553	37.003	37.003452	0.000452
31.350	31.958	31.957573	0.000427
77.001	68.194	68.194079	0.000079
78.879	78.252	78.252113	0.000113
39.586	41.417	41.417181	0.000181
39.347	40.946	40.945733	0.000267
87.484	87.993	87.993401	0.000401
54.349	55.636	55.635929	0.000071
76.082	61.280	61.280239	0.000239
84.079	87.310	87.309605	0.000395

The results indicated a minor error after subtraction, likely due to the decimal precision used when obtaining measurements with the firmware. In Table 17, the firmware was configured to make predictions with three decimal places of precision.

By increasing the number of decimals in the firmware's predictions, the error decreased, confirming that the difference was solely attributable to the decimal precision used during inference. The average error in both cases was below one per thousand, indicating that the microcontroller's behavior was equivalent to that of Python. With three decimals, the mean difference is 2.4867×10^{-4}%, while using four decimals reduced it to 2.6746×10^{-5}%.

In addition to prediction accuracy, the execution performance of the deployed model was evaluated on the target hardware. The inference latency of the 4-layer CNN was measured on the ESP32-S3 by timing the TensorFlow Lite int8 forward pass, achieving an average of 20 ms (standard deviation 5 ms) over 1000 consecutive runs. The system was operated with the VL53L7CX configured at 60 Hz, corresponding to an acquisition period of approximately 16.7 ms per frame. Under this configuration, the effective update rate is determined by the combined sensing and inference cycle rather than by either component alone. Nevertheless, the resulting throughput remains sufficient for real-time occupancy monitoring within the proposed embedded system.

6. Discussion and limitations

6.1. Occlusions and irregular cargo

As mentioned in previous sections, one of the key challenges in cargo volume estimation is handling occlusions and hidden voids. Since TOF sensors capture only the visible top surface of the stacked cargo, they cannot directly observe internal cavities or voids beneath the surface. Therefore, when irregularly shaped objects create hidden empty spaces, the system can *overestimate the occupied volume* (equivalently, *underestimate FreeSpace*), because the unobserved voids are interpreted as filled.

The results indicate that the neural network can partially *mitigate* occlusion-related errors by exploiting recurrent depth-pattern cues (e.g., discontinuities, near-wall returns, and shadowed regions) that correlate with *FreeSpace* in the training distribution. Importantly, this behavior should not be interpreted as explicit geometric reconstruction of occluded volumes, but as a *dataset-dependent statistical prior* learned from the specific set of cuboidal objects, container geometry, and packing patterns considered in this study.

As a consequence, significant domain shifts (e.g., irregular or deformable objects, different packing statistics, or in-transit reconfigurations) may reduce performance and introduce systematic bias. The reported results therefore support generalization within the structured, quasi-static cuboidal regime considered in this study (including unseen box combinations), whereas extensions to unstructured or dynamic cargo are not yet validated. The physics-based simulations in Appendix provide a rigorous geometric sensitivity analysis. These results quantify the expected impact of heterogeneous shapes on the sensing layout, serving as a validated computational baseline for the system's extension beyond the current experimental domain. Further validation would require additional measurements in the target regime and, if needed, corresponding updates to the sensing layout and training data.

In addition to total volume, the percentage of free space was also tested as an alternative output target. Although linearly related to volume, this normalized formulation can facilitate training stability, similarly to input normalization. In practice, both targets yielded comparable predictive performance.

The current system establishes a high-performance baseline for structured cuboidal cargo. Evaluation under heterogeneous geometries is addressed as a targeted extension of this operational domain, with initial results provided in Appendix to define the parameters for future specialized retraining. Consequently, the validation of performance under asymmetrical cargo, deformable items, and dynamic conditions is identified as the next operational phase, requiring dedicated in-situ measurements to calibrate the model for these specific logistics regimes.

Irregular cargo introduces additional sources of uncertainty that are not fully represented by structured box configurations. First, non-uniform contact points and unpredictable stacking can create viewpoint-dependent occlusions and shadow regions for the two fixed TOF viewpoints, which may bias the estimator toward *underestimating FreeSpace* (i.e., *overestimating occupied volume*) when relevant portions of the load are unobserved. Second, irregular objects often lead to packing inefficiencies and internal voids, so the *effective* space consumption can exceed the object's physical volume. Importantly, this effect is aligned with the definition of *FreeSpace* as *usable* empty capacity: when voids and geometric dead space reduce practical loading capacity, a lower *FreeSpace* estimate may still reflect the operational reality, even if it differs from the nominal physical volume of the items.

Since high-occupancy conditions amplify partial-observability effects (shadow regions and arrangement-dependent occlusions), it is reasonable to expect that estimation errors could increase further when dealing with irregularly shaped cargo. The presence of voids and complex geometries may introduce additional discrepancies in the volume estimation process, which should be addressed in the future work through dataset expansion and sensor fusion techniques.

Operationally, the estimator is intended to support capacity checks rather than fine-grained packing layout planning. Since decision risk is asymmetric, practical deployment should prioritize avoiding false acceptance when the available space is marginal. Accordingly, when the predicted *FreeSpace* approaches a near-full threshold, the estimate is best used as a conservative decision rule (e.g., a "near-full" flag) rather than relying on the exact dm^3 value. This interpretation is consistent with an error structure dominated by arrangement-dependent partial observability, under which small test-dependent biases can occur. This preserves real-time usability for routing and acceptance decisions while favoring safe rejection over false acceptance.

Beyond line-of-sight occlusions, the spatial resolution of the sensing setup imposes a structural observability limit. Each VL53L7CX provides an 8×8 multizone depth grid. With two sensors, the system samples the scene through 128 range measurements per frame. For a container of 182.282 dm^3, this constitutes a coarse spatial discretization, so fine geometric details (e.g., small gaps between parcels, narrow voids, or local overhangs) may remain unobserved irrespective of the learning model. Under such partial and low-resolution observations, the network does not perform explicit 3D reconstruction of hidden geometry. Instead, it estimates *FreeSpace* by exploiting statistical regularities of depth patterns within the training distribution. Consequently, the ability to account for occluded voids may degrade under domain shifts (different cargo shapes, packing statistics, or sensor placement). Improving baseline observability would require either higher-resolution multizone TOF sensing (e.g., VL53L5CX-class devices [96]) and/or increased sensor density with multi-sensor fusion, at the cost of higher data rates, power consumption, and computational requirements. Beyond compact multizone sensors, TOF imaging devices such as the Melexis MLX5027 can provide richer depth information, but a subtantially higher cost; for instance, its evaluation board is listed at approximately

Table 18
Geometric line-of-sight coverage analysis of the proposed sensing layout under idealized assumptions for different cargo volumes. Coverage values are conceptual and do not represent validated measurement accuracy.

	Vehicle	Dimensions (dm)	Cargo vol. (dm^3)	Line-of-sight covered volume (dm^3)	Geometric coverage (%)
Cargo bikes	Cubical box	8 × 8 × 8	512.00	484.32	94.6
	Elongated box	8 × 6 × 8	384.00	362.82	94.5
Vans	Small van	16 × 14.3 × 12	2745.60	2657.99	96.8
	Large van	38 × 17 × 18.8	12,144.80	12,144.80	100.0

Coverage values are purely geometric and must be experimentally re-established for each container geometry and sensor layout. Range limits of the VL53L7CX (up to 350 cm per zone) are not considered.

USD 500–600 [97]. At the opposite end of the spectrum, low-cost TOF modules such as MaixSense A010 are available at around USD 25–35 (e.g., USD 26.20 in a representative listing) [98], but adopting such alternatives may entail different performance trade-offs and integration constraints.

6.2. Cargo dynamics and robustness

The total volume of cargo remains constant during transport; however, minor shifts in object placement can affect TOF sensor readings and thus the estimated volume. Training included diverse placement patterns (Section 6.4) to promote generalization across spatial arrangements, which is expected to enhance robustness under real-world dynamic conditions. The dataset used in this study already incorporated multiple randomized stacking configurations of the same cargo items, which indirectly replicates displacement effects and helped the neural network learn to generalize across different spatial arrangements. This capability is expected to enhance robustness under real-world dynamic conditions.

Nevertheless, the present work focused on controlled static scenarios to establish a performance baseline. Future work will include dynamic testing during transport (e.g., on a moving cargo bike) and environmental exposure tests. Inertial sensing may also be explored to improve reliability under motion. Therefore, the reported performance should be interpreted as feasibility under static indoor conditions. Deployment-oriented validation will require on-vehicle experiments (cargo bike in motion) to assess robustness under vibration, motion-induced shifts, and environmental exposure (including outdoor illumination variability).

In addition to technical field testing, deployment-oriented evaluation should include pilot trials on operational cargo bikes involving professional couriers. Such trials would allow assessment of installation practicality, workflow integration, usability, and perceived utility of the system under real delivery routines. While courier input was incorporated at the requirements-definition stage through a focus group (Section 2.2), user-centered validation of the deployed prototype was outside the scope of the present study and is therefore identified as future work.

6.3. System adaptability and scalability

Table 18 summarizes a geometric line-of-sight coverage analysis of the sensing layout when hypothetically adapted to different cargo volumes. The system was designed and experimentally validated only for the container geometry used in this study, and in its current form it is most reliable for low-to-moderate occupancy scenarios involving structured cuboidal cargo. Accordingly, the values reported in Table 18 provide a geometric feasibility framework for platform adaptation. These figures establish the theoretical line-of-sight upper bounds for each geometry, serving as a blueprint for the sensor density and layout redesign required to maintain accuracy in larger-scale deployments. In particular, the analysis does not account for sensor range limits: the VL53L7CX is specified up to 350 cm per zone under favorable conditions, and maximum ranging capability depends on reflectance and ambient light. Consequently, for larger platforms (e.g., trucks and 20/40 ft containers), a two-sensor configuration would not be sufficient to cover the full volume at operational distances. Transferring the approach to other vehicle types would therefore require re-designing sensor placement and density (or selecting sensors with longer ranging capability), followed by dataset acquisition and model re-validation for each specific geometry.

Beyond the reference cargo-bike container, the approach is intended as an on-device occupancy state estimation module that may support higher-level logistics decisions. As potential application targets, the approach could be adapted to contexts such as unmanned parcel lockers, robotic delivery units, smart warehouses, and fleet vehicles. In these settings, reporting free space (rather than its spatial distribution) supports capacity acceptance checks, dynamic task allocation and rerouting, exception detection when observed occupancy deviates from the plan, and fleet-level utilization KPIs (Key Performance Indicators). For workflows that already track parcel dimensions upstream, the system's estimates act as an independent feedback signal to validate planned loading and detect divergence, rather than to compute stacking layouts. Additionally, in shipping containers, trucks, and vans, the system provides occupancy feedback to support load-distribution policies and improve utilization tracking.

The system's design is inherently scalable at the fleet level because the volume estimation is performed locally on each unit using a lightweight embedded model, and fleet expansion does not increase server-side computational load. As a result, increasing the number of instrumented vehicles does not increase the computational burden on any central server, and scalability is decoupled from vehicle-specific sensing adaptations that may be required for different geometries. A fleet management platform would only aggregate and visualize per-vehicle outputs, without executing the estimation pipeline itself. Consequently, scalability depends primarily on the communication and coordination infrastructure rather than on the core estimation process, making the system well suited for large-scale deployment in urban logistics fleets.

6.4. Dataset design considerations

As described in Section 3.1, box placements were varied to approximate a uniform spatial distribution, using pseudo-random positioning for fewer or smaller boxes and semi-structured variation for larger or more numerous boxes. This strategy was essential to ensure representative coverage of possible configurations and to provide the neural network with sufficient variability for robust generalization. While full statistical uniformity was not formally validated, the large number of unique positions (60 per case × 63 cases) across different box combinations ensured high positional variability, which helped the neural network generalize across diverse scenarios. Future work could explore automated or algorithm-driven positioning to improve repeatability and quantifiable uniformity.

At the same time, selecting an adequate number of boxes to encompass various size ranges was essential for representative measurements. A larger variety of boxes enriched the dataset, increasing the chances of achieving better results. Additionally, determining the number of samples to take at each position was an important consideration.

Although it might be assumed that collecting multiple measurements from the same position could lead to overfitting, this was not necessarily the case as long as an appropriate number of data samples were maintained per position. Sensor readings inherently exhibit a margin of error due to measurement tolerances and environmental noise. Consequently, the emphasis was placed on diversifying box placement over increasing the number of samples for each combination. Diverse box positions were crucial for ensuring representative data and a comprehensive view of each combination.

In this study, cubic and rectangular objects were used to establish a controlled baseline before considering more complex geometries. This cuboidal focus is a *case-study constraint* motivated by the operational context reported in the focus group, where a large share of last-mile shipments arrives in rectangular cartons/boxes. Accordingly, the validation is not intended as a general statement about all urban logistics cargo; rather, it provides a controlled starting point before extending the evaluation to heterogeneous geometries. Extending the dataset to include irregular shapes is a natural next step to stress-test generalization under occlusions and non-cuboidal profiles, and to validate whether performance trends observed in controlled conditions hold in more realistic packing scenarios. As an intermediate step, the dataset can be expanded using synthetic generation, and the most representative configurations can then be validated through physical tests. Exploratory simulations using an external synthetic test set (no retraining) are reported in Appendix.

The dataset used in this work is based on manual placements of real packages, and therefore does not include externally measured 6-DOF (Degrees of Freedom) ground-truth poses (position and orientation) for each box. Only box identities and volumes are available as ground-truth targets, while placement variability is introduced through manual re-arrangement. The inclusion of pose annotations via external tracking (e.g., fiducials or motion capture systems) are identified as future work, subject to institutional and privacy or operational constraints. The dataset will be made publicly available in accordance with FAIR principles upon publication of the article.

6.5. Technical contributions

While TOF sensors have been used in various depth-sensing applications, this study presented an innovative approach by integrating TOF data with deep learning techniques to enhance volume estimation accuracy in cargo logistics. Unlike conventional methods that rely on predefined geometric assumptions or direct volumetric calculations, our approach leveraged advanced neural network-based processing to improve measurement precision and adaptability.

A key innovation of this study was the neural network-based compensation for occlusions. Traditional TOF-based measurement systems often struggle with occlusions, particularly in cargo environments where objects may partially block sensor readings. By training a Convolutional Neural Network (CNN) on diverse stacking configurations, our system was capable of inferring hidden volumes based on learned spatial patterns. This capability addressed one of the major limitations of raw TOF sensor data, significantly improving estimation accuracy in controlled structured cargo scenarios.

Another important aspect is the adaptive volumetric estimation through learned feature extraction. Instead of relying on manually defined volume estimation rules, the model autonomously learns relevant spatial patterns, eliminating the need for explicit hand-engineered features or geometric assumptions. This allows it to generalize more effectively across different cargo arrangements, making it more robust in handling variations in cargo stacking and distribution.

Furthermore, this study demonstrates the optimized deployment of AI-driven volume estimation in resource-constrained environments. Unlike LiDAR-based solutions, which require significant computation power and are often expensive, our system proves that high-accuracy volume estimation can be achieved on embedded microcontrollers such as the ESP32-S3. This enables not only on-device estimation but also the implementation of basic decision rules, for example in automated lockers where a parcel can be accepted only if the measured free space exceeds a predefined threshold.

7. Conclusion

The chosen system design involves using *Time-of-Flight (TOF) sensors* in combination with *Artificial Neural Networks (ANNs)* due to their advantages in accuracy, energy efficiency, and cost. TOF sensors generate high-resolution distance matrices with moderate energy consumption, ideal for autonomous systems such as cargo bikes. Combined with artificial neural networks, data is efficiently processed and analyzed, supporting accurate volume estimation. This combination also reduces processing complexity on the device, adapting to the limitations of a microcontroller environment. By integrating this system into cargo bike operations, logistics companies can automate load occupancy estimation, support capacity acceptance checks before dispatch, and detect deviations between planned and observed loading. This approach not only improves the overall efficiency of last-mile delivery but also enhances package management in distribution centers, supporting more informed allocation of cargo across multiple delivery vehicles.

The results demonstrate that the use of TOF sensors is highly effective for this type of application. Despite potential error margins in sensor measurements, the creation of a robust dataset and a proper training of the neural network led to high accuracy. The end-to-end process – from data collection and visualization to sensor adjustment and generating a sufficiently large dataset – proved effective in training a simple yet precise network.

This study shows that the proposed volume estimation system is applicable to cargo containers similar to those found in commercial cargo bikes. A comparison with industry-standard cargo bike models shows that the sensor configuration and methodology can be easily adapted to existing logistics operations.

The proposed system offers a cost-effective alternative to high-end volume estimation solutions. By utilizing low-cost TOF sensors and an embedded microcontroller, the implementation remains affordable for urban logistics applications, making it viable for real-world deployment.

While the proposed system demonstrated high accuracy in structured cargo configurations, a key limitation of this study is the absence of irregularly shaped objects in the dataset. Such cases introduce additional challenges due to variable contact points, unpredictable stacking, and non-uniform volume distributions, which often result in hidden voids and occlusion-driven estimation errors. The model partially compensates for occlusions by learning statistical patterns from the training data, but residual errors persist in scenarios with complex geometries.

To address these limitations, future work will focus on expanding the dataset beyond structured cuboidal cargo to include irregular and deformable shapes, initially through synthetic data generation in simulation environments and subsequently through physical experiments with real parcels. In particular, Appendix introduces a Blender-based pipeline to generate TOF-like depth grids and reports an exploratory evaluation on synthetic structured and irregular objects without retraining, which is intended only as a proxy to study geometry-induced effects and the expected sim-to-real gap. Robust deployment, however, will require dedicated acquisition of real data with irregular parcels under realistic packing conditions, followed by retraining and re-validation for each target cargo regime. Furthermore, integrating sensor-fusion techniques (e.g., combining TOF with RGB-D cameras or LiDAR) and more advanced learning strategies will be explored to improve robustness under occlusions and broaden applicability. In addition, future work will address cargo displacement scenarios, where IMUs (Inertial Measurement Units) could be incorporated to detect shifts and compensate for motion-induced variations in TOF readings, thereby enabling deployment-oriented testing under real transport conditions.

Improving baseline observability would require either higher-resolution multizone TOF sensors (e.g., VL53L5CX-class devices, Melexis MLX5027 or MaixSense A010), or increased sensor density with multi-sensor fusion, at the cost of higher data rates, power consumption, and computational requirements. These options are identified as potential extensions rather than capabilities of the present prototype.

Alternative sensor configurations should also be investigated, such as positioning the TOF sensors at the upper corners instead of the central edges. Such configurations may affect coverage uniformity and overall system robustness, potentially mitigating occlusion issues and improving total volume coverage, and therefore deserve further exploration in future studies.

Regarding the neural network design, it was concluded that convolutional neural networks (CNNs) were the most effective architecture, learning patterns through their filters more efficiently than dense layers. While a highly accurate model was achieved, an optimized and simplified network with fewer layers was preferred to fit the memory constraints of embedded systems without compromising performance.

Post-training, the model demonstrated excellent inference capabilities in test cases, validating the robustness of the dataset. Even when certain boxes or combinations are missing, the model was able to predict their volumes within an acceptable margin of error.

Finally, deploying the model on the microcontroller demonstrated the practicality of running a high-accuracy, small-sized model on embedded systems using TensorFlow Lite. The results showed errors below 10%, underscoring the viability of an optimized model for hardware limited resources while maintaining outstanding performance.

These results not only validate the proposed approach but also open the door to broader applications in real-time load monitoring and occupancy state estimation in resource-constrained environments, paving the way for further advancements in urban autonomous mobility applications as a reliable and scalable solution.

CRediT authorship contribution statement

Diego Remón: Writing – review & editing, Writing – original draft, Software, Formal analysis, Data curation. **Alberto Gascón:** Writing – review & editing, Validation, Supervision, Methodology, Investigation, Conceptualization. **Álvaro Marco:** Writing – review & editing, Writing – original draft, Supervision, Conceptualization. **Teresa Blanco:** Writing – review & editing, Visualization, Methodology, Investigation, Funding acquisition, Conceptualization. **Roberto Casas:** Writing – review & editing, Validation, Supervision, Methodology, Investigation, Funding acquisition, Conceptualization.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used ChatGPT in order to improve language readability. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Grant information

This research was funded by Spanish Ministry of Science and Innovation (Grant Number: 419 PID2020-116011RB-C22), by Spanish Ministry of Industry and Tourism (Grant Number: 2023/DIH_01/000355), and by Government of Aragon (Grant Number: 420 T27_23R).

Table A.19
Summary of model performance on synthetic datasets generated from structured cuboidal packages (no retraining).

Dataset	Samples	MAE (dm^3)	RMSE (dm^3)	R^2
Real test sets	115,200–139,200	~8.1	~12.1	~0.92
Synthetic (boxes)	5834	9.02	11.997	0.27

Appendix. Synthetic data simulations and exploratory analysis

This appendix reports a set of exploratory physics-based simulations conducted to assess the feasibility of extending the experimental dataset beyond the original set of six cuboidal packages. The objective of these simulations is not to validate generalization performance nor to replace real-world measurements, but to investigate whether synthetic data can reproduce error magnitudes comparable to those observed experimentally and to characterize the expected sim-to-real gap.

A.1. Simulation setup

A physics-based simulation environment was implemented using Blender to generate synthetic depth measurements consistent with the geometry of the prototype container and the nominal field-of-view configuration of the two TOF sensors. The container dimensions and sensor placement matched those described in Section 2. Synthetic distance maps were generated by ray-casting from the virtual sensor viewpoints, producing 8 × 8 depth matrices per sensor, consistent with the resolution of the *VL53L7CX* devices.

To approximate sensor uncertainty, a multiplicative perturbation of ±5% was applied to the simulated distance values. This perturbation is intended as a coarse approximation of range uncertainty and does not model sensor-specific artefacts such as invalid returns, multipath effects, surface reflectance variability, zone aggregation behavior, or non-Gaussian noise characteristics. As a result, the simulations capture geometric constraints but only partially reproduce the physical measurement process of the real sensor.

A.2. Simulation with structured cuboidal packages

In a first exploratory study, the simulator was used to generate synthetic arrangements of structured cuboidal packages derived from the original box set. The largest box was intentionally excluded from this synthetic dataset, as its dimensions made unconstrained random placement under realistic packing conditions impractical within the current simulation framework. The trained model was evaluated directly on this synthetic dataset without retraining.

Across 5834 simulated samples, errors ranged approximately between 6 and 13 dm^3, yielding a global R^2 of 0.27 Table A.19. This lower R^2 is partly explained by the reduced variance of the synthetic target distribution (largest box excluded) and by the sensitivity of R^2 to the occupied-volume range. Although the correlation is significantly lower than in the real-world test sets, the absolute error magnitudes are of the same order as those observed experimentally. This result suggests that the simulator captures relevant geometric and sensing constraints affecting volumetric estimation, while also highlighting a substantial sim-to-real gap attributable to unmodeled sensor artefacts and environmental effects. Fig. A.15 reports the per case MAE in dm^3 for the synthetic test set.

A.3. Simulation with irregular objects

A second exploratory study extended the simulation to include irregular object geometries, such as cylinders and polyhedral meshes, in order to probe the behavior of the trained model outside the structured cuboidal domain. Again, the model was evaluated without retraining.

As summarized in Table A.20, the irregular-object synthetic test set yields a global R^2 of 0.86 with a MAE of 6.18 dm^3. To characterize

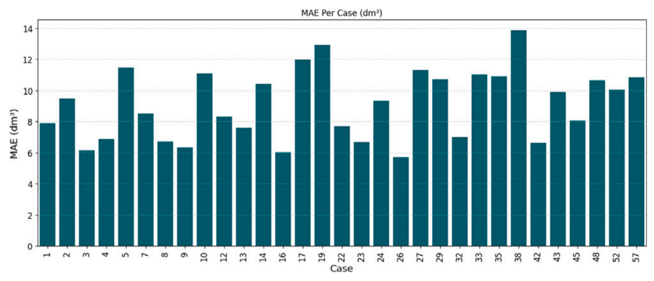


Fig. A.15. Mean absolute error per case (MAE in dm^3) obtained on the synthetic test set generated from structured cuboidal packages. The model was evaluated without retraining. Each bar corresponds to one box-combination case in the simulated dataset, illustrating the distribution and magnitude of errors under synthetic sensing conditions.

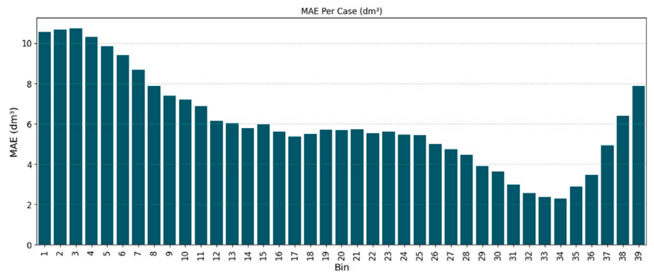


Fig. A.16. Mean absolute error per bin (MAE in dm^3) on the synthetic test set generated from irregular objects. The model was evaluated without retraining. Each bar corresponds to a volume bin of width ΔV = 1.82282 dm^3, illustrating how the error evolves as occupancy increases under synthetic sensing conditions.

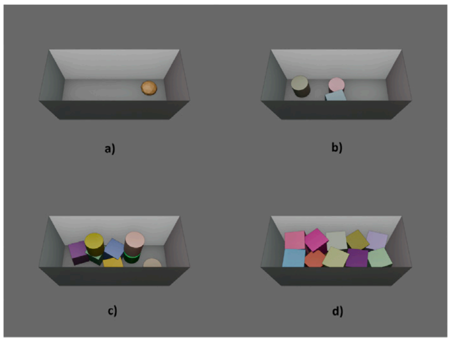


Fig. A.17. Representative examples of synthetic simulation configurations generated in Blender for the exploratory studies. (a) Low-occupancy scenario with a single irregular object; (b) intermediate occupancy with a small set of mixed geometries; (c) higher occupancy with increased geometric complexity and partial occlusions; (d) higher-occupancy configuration illustrating packing- and occlusion-dominated regimes. These examples qualitatively illustrate the diversity of object geometries and occupancy levels considered in the physics-based simulations.

Table A.20
Summary of model performance on synthetic datasets generated from irregular object simulations (no retraining).

Dataset	Samples	MAE (dm^3)	RMSE (dm^3)	R^2
Real test sets	115,200–139,200	~8.1	~12.1	~0.92
Synthetic (irregular objects)	27,300	6.18	8.82	0.86

how the error evolves with occupancy, results are reported using *bins* rather than discrete *cases*. Each bin corresponds to a target range of total load volume with width ΔV = 1.82282 dm^3 (i.e., bin 1 spans $[0, \Delta V]$, bin 2 spans $(\Delta V, 2\Delta V)$, and so forth up to bin 39). Fig. A.16 reports the MAE per bin in dm^3, revealing a consistent pattern across occupancy regimes: higher errors at low occupancy, reduced errors at intermediate occupancy, and increasing geometric ambiguity induced by irregular shapes, and increasing errors at higher occupancy, where occlusions become more frequent. The occupancy range in this experiment is limited to bins 1–39 (0–71.08998 dm^3). Higher-volume bins were not included because unconstrained random filling of the container with irregular objects frequently leads to severe inter-object occlusions and packing deadlocks in the current simulation pipeline, preventing the generation of stable near-full configurations without introducing additional constraints. Achieving higher-occupancy synthetic regimes would therefore require a more structured placement strategy (e.g., layered or rule-based packing) and corresponding modifications to the simulation script. Finally, although the global R^2 is higher in this synthetic setting, these results should not be interpreted as evidence of full generalization to irregular real cargo. Rather, they illustrate how geometric diversity and occupancy interacts with sparse TOF measurements under simplified sensing assumptions.

Representative examples of the simulated configurations used in this last study are shown in Fig. A.17, illustrating different occupancy regimes and object geometries considered in the synthetic experiments.

Data availability

Data will be available here: Remón, D., Gascón Roche, A., Marco, A., Blanco, T., & Casas, R. (2026). A smart container for real-time load occupancy estimation using embedded neural inference (dataset) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.18614817.

References

[1] G. Ostojić, S. Stevan, K. Dragan, T. Branislav, T. Srdjan, in: B. Lalic, D. Gracanin, Z. Anisic (Eds.), Proceedings on 25th International Joint Conference on Industrial Engineering and Operations Management, IJCIEOM, Springer International Publishing, Cham, 2020, pp. 483–491.
[2] Raleigh, What Is an Electric Cargo Bike?.
[3] Cycling-News, What is a cargo bike? Everything you need to know, 2023.
[4] T. Cherniavska, N. Tanklevska, B. Cherniavskyi, Implementation of European innovations in last-mile logistics: opportunities for adapting best practices to foster the sustainable development of e-commerce, 2025.
[5] S.N. Biehl, H. Kaufmann, A. Schönemann, T. Melz, W. Kaal, W.G. Kolodziej, Cargo bikes-sustainable means of transportation, in: Resource Efficient Vehicles Conference-Rev2021KTH, Royal Institute of Technology, Sweden, 2021.
[6] P.R. Dimitri Marincek, V. Lurkin, Cargo bikes for personal transport: A user segmentation based on motivations for use, Int. J. Sustain. Transp. (2024) 1–14.
[7] A. Thomas, Electric bicycles and cargo bikes—Tools for parents to keep on biking in auto-centric communities? Findings from a US metropolitan area, Int. J. Sustain. Transp. 16 (7) (2022) 637–646.
[8] Mordor Intelligence, Germany courier, express, and parcel (CEP) market size & share analysis - growth trends & forecasts (2025 - 2030), 2025.
[9] Mordor Intelligence, Switzerland e-bike market size & share analysis - growth trends & forecasts up to 2029, 2025.
[10] S. Gosavi, Cargo bicycle market report 2025 (global edition), 8, 2025, Cognitive Market Resarch.
[11] Mordor Intelligence, Europe E-cargo bike market size & share analysis - growth trends and forecast (2026 - 2031), 2025.
[12] L. Mantecchini, F.P. Nanni Costa, V. Rizzello, Last mile urban freight distribution: A modelling framework to estimate E-cargo bike freight attraction demand share, Futur. Transp. 5 (1) (2025).
[13] D. Marincek, P. Rérat, V. Lurkin, Cargo bikes for personal transport: A user segmentation based on motivations for use, Int. J. Sustain. Transp. 18 (9) (2024) 751–764.
[14] L. Cameli, M. Pazzini, R. Ceriani, V. Vignali, A. Simone, C. Lantieri, Integrating environmental sensing into cargo bikes for pollution-aware logistics in last-mile deliveries, Sensors 25 (15) (2025).
[15] M. Bissel, S. Becker, Can cargo bikes compete with cars? Cargo bike sharing users rate cargo bikes superior on most motives – Especially if they reduced car ownership, Transp. Res. Part F: Traffic Psychol. Behav. 101 (2024) 218–235.

[16] B. Kin, W. Ploos van Amstel, R. Fransen, Light electric freight vehicles–beyond the hype?, tNO / HAN / hogeschool Van Amsterdam, 2024.

[17] I. Chatziioannou, E. Bakogiannis, C. Karolemeas, E. Kourmpa, K. Papadaki, T. Vlastos, Urban environment's contributory factors for the adoption of cargo bike usage: A systematic literature review, Futur. Transp. 4 (1) (2024) 92–106.

[18] J. Gruber, M. Plener, L. Damer, I. Dubernet, Car or cargo bike? determinants for the use of a small vehicle type in urban logistics: A stated preference survey among commercial transport operators, Transp. Res. Rec. 2678 (11) (2024) 1548–1561, http://dx.doi.org/10.1177/03611981241245676.

[19] A. Galkin, L. Švadlenka, R. Vrba, L.K. de Oliveira, Evaluation of cargo bike program for parcel deliveries in a medium-sized city, Transp. Res. Part D: Transp. Environ. 140 (2025) 104609.

[20] F.A. Malik, R. Egan, C.M. Dowling, B. Caulfield, Factors influencing e-cargo bike mode choice for small businesses, Renew. Sustain. Energy Rev. 178 (2023) 113253.

[21] S. Narayanan, C. Antoniou, Electric cargo cycles - A comprehensive review, Transp. Policy 116 (2022) 278–303.

[22] Pedalsure, A guide to cargo bikes: All you need to know, 2022.

[23] C. Kiefer, F. Behrendt, Smart e-bike monitoring system: real-time open source and open hardware GPS assistance and sensor data for electrically-assisted bicycles, IET Intell. Transp. Syst. 10 (2016) 79–88.

[24] V. Naumov, M. Pawluś, Identifying the optimal packing and routing to improve last-mile delivery using cargo bicycles, Energies 14 (2021).

[25] D. Carracedo, H. Mostofi, Electric cargo bikes in urban areas: A new mobility option for private transportation, Transp. Res. Interdiscip. Perspect. 16 (2022) 100705.

[26] A.S. C. Dou, A. Tumino, The limitations of electric cargo bikes – a systematic literature review, in: Proceedings of the 27th International Symposium on Logistics, ISL 2023, 2023.

[27] J. Torres-Solis, T.H. Falk, T. Chau, A review of indoor localization technologies: towards navigational assistance for topographical disorientation, in: F.J.V. Molina (Ed.), Ambient Intelligence, IntechOpen, London, 2010.

[28] L.M. Pires, T. Alves, M. Vassaramo, V. Fialho, Design and development of a high-accuracy IoT system for real-time load and space monitoring in shipping containers, Designs 9 (2) (2025).

[29] T. Maus, N. Zengeler, D. Sänger, T. Glasmachers, Volume determination challenges in waste sorting facilities: Observations and strategies, Sensors 24 (7) (2024).

[30] B. Andò, S. Baglio, R. Crispino, V. Marletta, An introduction to indoor localization techniques. Case of study: A multi-trilateration-based localization system with user–environment interaction feature, Appl. Sci. 11 (16) (2021).

[31] D. F. Albuquerque, J. M. N. Vieira, C. A. C. Bastos, P. J. S. G. Ferreira, Ultrasonic ofdm pulse for beacon identification and distance measurement in reverberant environments, in: Proceedings of the 1st International Conference on Pervasive and Embedded Computing and Communication Systems, PECCS, INSTICC, SciTePress, 2011, pp. 124–132.

[32] Z. Tao, Y. Wang, W.K. Ao, F. Zhou, D. Zhang, Y.-Q. Ni, Evaluation of dynamic responses in steel lever of leveling valve during metro train operations, Measurement 256 (2025) 118198.

[33] Y. Wu, J. Zhou, S. Wu, H. Yu, A deep learning-based framework for impact load identification using strain data on composite plates, Measurement 256 (2025) 118497.

[34] D.-Z. Dang, B.-Y. Su, Y.-W. Wang, W.K. Ao, Y.-Q. Ni, A pencil lead break-triggered, adversarial autoencoder-based approach for rapid and robust rail damage detection, Eng. Appl. Artif. Intell. 150 (2025) 110637.

[35] W.K. Ao, Q.-C. Tang, A. Pavic, Decentralised H robust control of MTMDs for mitigating vibration of a slender MDOF floor configuration, Thin-Walled Struct. 203 (2024) 112226.

[36] F. Yang, Y. Luo, L. Du, Y. Zhang, S. Xie, Study on interpretability of artificial neural network models for dynamic load identification, Measurement 251 (2025) 117210.

[37] T. Peng, Z. Zhang, F. Chen, D. Zeng, Dimension measurement and key point detection of boxes through laser-triangulation and deep learning-based techniques, Appl. Sci. 10 (2020).

[38] P. Artaso, G. López-Nicolás, Volume estimation of merchandise using multiple range cameras, Measurement 89 (2016) 223–238.

[39] J. Bierende, J. Braun, P. Costa, J. Lima, A.I. Pereira, Volume estimation of an indoor space with LiDAR scanner, in: A.I. Pereira, A. Košir, F.P. Fernandes, M.F. Pacheco, J.P. Teixeira, R.P. Lopes (Eds.), Optimization, Learning Algorithms and Applications, Springer International Publishing, Cham, 2022, pp. 78–92.

[40] A. Prayitno, M. Shiddiq, D.S. Arief, V.V. Dasta, R.H. Fitra, 3D imaging using cross line laser for box volume estimation, AIP Conf. Proc. 2221 (2020) 060004.

[41] M. Keskin, D. Saribas, Container level measurement methods and testing an ultrasonic level sensor, Int. J. Automot. Eng. Technol. (2017).

[42] P. Mohindru, Development of liquid level measurement technology: A review, Flow Meas. Instrum. 89 (2023) 102295.

[43] J. Xu, Y. Lu, E. Olaniyi, L. Harvey, Online volume measurement of sweetpotatoes by A LiDAR-based machine vision system, J. Food Eng. 361 (2024) 111725.

[44] N. Ladplee, A. Pimpin, W. Srituravanich, N. Damrongplasit, Volumetric measurement of rectangular parcel box using LiDAR depth camera for dimensioning and 3D bin packing applications, in: 2022 IEEE International Conference on Consumer Electronics-Asia, ICCE-Asia, 2022, pp. 1–4.

[45] E. Arnold, M. Dianati, R. de Temple, S. Fallah, Cooperative perception for 3D object detection in driving scenarios using infrastructure sensors, IEEE Trans. Intell. Transp. Syst. 23 (2022) 1852–1864.

[46] X. Dai, J. Hu, H. Zhang, A. Shitu, C. Luo, A. Osman, S. Sfarra, Y. Duan, Multi-task faster R-CNN for nighttime pedestrian detection and distance estimation, Infrared Phys. Technol. 115 (2021) 103694.

[47] Tao Peng, Zhijiang Zhang, Yingjie Song, F. Chen, D. Zeng, Portable system for box volume measurement based on line-structured light vision and deep learning, Sensors 19 (2019) 3921.

[48] L. Hamad, M.A. Khan, A. Mohamed, Object depth and size estimation using stereo-vision and integration with SLAM, IEEE Sensors Lett. 8 (4) (2024) 1–4.

[49] A. Pagliai, M. Ammoniaci, D. Sarri, R. Lisci, R. Perria, M. Vieri, M.E.M. D'Arcangelo, P. Storchi, S.-P. Kartsiotis, Comparison of aerial and ground 3D point clouds for canopy size assessment in precision viticulture, Remote. Sens. 14 (5) (2022).

[50] C. Anqi, N. Xiuhua, Z. Long, Principal component Analysis(PCA) normal estimation based on TOF sensor, ICIIBMS, in: 2023 8th International Conference on Intelligent Informatics and Biomedical Sciences, vol. 8, 2023, pp. 291–294.

[51] F. Piron, D. Morrison, M.R. Yuce, J.-M. Redouté, A review of single-photon avalanche diode time-of-flight imaging sensor arrays, IEEE Sensors J. 21 (11) (2021) 12654–12666.

[52] J. Medrano, F. Yumbla, G. Lee, J. Yi, M. Kim, E. Auh, J. Park, I. Oh, N. Pico, H. Moon, Box segmentation, position and size estimation for robotic box handling applications, in: 2022 19th International Conference on Ubiquitous Robots, UR, 2022, pp. 194–199.

[53] G. Omotara, S.M.A. Tousi, J. Decker, D. Brake, G.N. DeSouza, High-throughput and accurate 3D scanning of cattle using time-of-flight sensors and deep learning, Sensors 24 (16) (2024).

[54] H.X. Huynh, B.H. Lam, H.V.C. Le, T.T. Le, N. Duong-Trung, Design of an IoT ultrasonic-vision based system for automatic fruit sorting utilizing size and color, Internet Things 25 (2024) 101017.

[55] F. Aliew, An approach for precise distance measuring using ultrasonic sensors, 2022.

[56] L. Angrisani, A. Baccigalupi, R.S. Lo Moriello, Ultrasonic-based distance measurement through discrete extended Kalman filter, in: Kalman Filter Recent Advances and Applications, 2009.

[57] L. Ye, J. Liu, J. Zhang, J. Ju, Y. Wang, A novel size-aware local contrast measure for tiny infrared target detection, IEEE Geosci. Remote. Sens. Lett. 22 (2025) 1–5.

[58] S. Sun, B. Mo, J. Xu, D. Li, J. Zhao, S. Han, Multi-YOLOv8: An infrared moving small object detection model based on YOLOv8 for air vehicle, Neurocomputing 588 (2024) 127685.

[59] B. Dellen, I.A. Rojas Jofre, Volume measurement with a consumer depth camera based on structured infrared light, in: Proceedings of the 16th Catalan Conference on Artificial Intelligence, Poster Session, 2013, pp. 1–10.

[60] T. Zhang, J. Niu, S. Liu, T. Pan, B.B. Gupta, Three-dimensional measurement using structured light based on deep learning, Comput. Syst. Sci. Eng. 36 (2021) 1.

[61] S. Liu, Y. Zhang, T. Shao, M. Yuan, Research on 3D measurement model by line structure light vision, 2018.

[62] H. Zhang, Y. Zhang, Q. Feng, K. Zhang, Research on unknown space target pose estimation method based on point cloud, IEEE Access 12 (2024) 149381–149390.

[63] Ouster, Introducing the L2X chip: 2X the processing power, 2X the data output to power ouster's most reliable and rugged sensors, 2021.

[64] A. Afzalaghaeinaeini, J. Seo, D. Lee, H. Lee, Design of dust-filtering algorithms for LiDAR sensors using intensity and range information in off-road vehicles, Sensors 22 (11) (2022).

[65] Renke, Ultrasonic sensor work, used, limitations and FAQs, 2024.

[66] Sensor One-Stop, A comparative analysis of infrared and laser PM sensors, 2024.

[67] P.P. Shinde, S. Shah, A review of machine learning and deep learning applications, in: 2018 Fourth International Conference on Computing Communication Control and Automation, ICCUBEA, 2018, pp. 1–6.

[68] T. Peng, Z. Zhang, F. Chen, D. Zeng, Dimension measurement and key point detection of boxes through laser-triangulation and deep learning-based techniques, 2020.

[69] Sensolus, Asset visibility. Made simple, 2025.

[70] Velco, Creating value for the e-bike industry with IoT solutions, 2025.

[71] Frotcom, The intelligent fleet management for your company, 2025.

[72] R. Ready, Complete fleet management, at your fingertips, 2025.

[73] Teltonika, Fleet telematics, 2025.

[74] Didcom, Connecting innovation to empower fleet management, 2025.

[75] Uplogic Technologies, From manual to automated: Guide to optimizing your courier delivery, 2023.

[76] B.M. Mohsen, AI-driven optimization of urban logistics in smart cities: Integrating autonomous vehicles and IoT for efficient delivery systems, Sustainability 16 (24) (2024).

[77] K.M. Hussain, Revolutionizing route optimization systems with artificial intelligence for a smarter, sustainable logistics ecosystem, Int. J. Comput. Sci. Mob. Comput. 14 (2) (2025).

[78] J. Mas Burrel, M.T. Blanco Bascuas, Diseño y desarrollo de un sistema producto-servicio para la monitorización de flotas de vehículos de reparto de última milla y la medición del volumen de carga, 2025.

[79] STMicroelectronics, Time-of-flight (ToF) 8x8 multizone ranging sensor with 90 degrees FoV, 2024.

[80] UnitElectronics, ESP32-S3-WROOM-1, 2024.

[81] N. D.Q., Spherical coordinates.

[82] A. Geron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, second ed. O'Reilly Media, Inc. 2019.

[83] Scikit-learn, Scikit-learn. Machine learning in Python, 2024.

[84] S. Pölsterl, Scikit-survival: A library for time-to-event analysis built on top of scikit-learn, J. Mach. Learn. Res. 21 (212) (2020) 1–6.

[85] TensorFlow, TensorFlow. An end-to-end platform for machine learning, 2024.

[86] B. Pang, E. Nijkamp, Y.N. Wu, Deep learning with TensorFlow: A review, J. Educ. Behav. Stat. 45 (2) (2020) 227–248.

[87] M. Greenacre, P.J.F. Groenen, T. Hastie, A.I. D'Enza, A. Markos, E. Tuzhilina, Principal component analysis, Nat. Rev. Methods Prim. 2 (2022).

[88] B.M.S. Hasan, A.M. Abdulazeez, A review of principal component analysis algorithm for dimensionality reduction, J. Soft Comput. Data Min. 2 (1) (2021) 20–30.

[89] T.O. Hodson, Root mean square error (RMSE) or mean absolute error (MAE): When to use them or not, Geosci. Model. Dev. Discuss. 2022 (2022) 1–10.

[90] V. Rodriguez-Galiano, M. Sanchez-Castillo, M. Chica-Olmo, M. Chica-Rivas, Machine learning predictive models for mineral prospectivity: An evaluation of neural networks, random forest, regression trees and support vector machines, Ore Geol. Rev. 71 (2015) 804–818.

[91] Z. Li, F. Liu, W. Yang, S. Peng, J. Zhou, A survey of convolutional neural networks: Analysis, applications, and prospects, IEEE Trans. Neural Netw. Learn. Syst. 33 (12) (2022) 6999–7019.

[92] Y. Cui, F. Zhou, J. Wang, X. Liu, Y. Lin, S. Belongie, Kernel pooling for convolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2921–2930.

[93] T. Hastie, R. Tibshirani, J. Friedman, et al., The elements of statistical learning, 2009.

[94] A. Srivastava, B.S. Rawat, G. Singh, V. Bhatnagar, P.K. Saini, S.A. Dhondiyal, A review of optimization algorithms for training neural networks, in: 2023 International Conference on Sustainable Emerging Innovations in Engineering and Technology, ICSEIET, 2023, pp. 886–890.

[95] R. Abdulkadirov, P. Lyakhov, N. Nagornov, Survey of optimization algorithms in modern neural networks, Mathematics 11 (11) (2023).

[96] STMicroelectronics, VL53l5cx: Time-of-Flight multizone ranging sensor (Data brief), 2021.

[97] Melexis Innovation With Heart, MLX75123BA time-of-flight companion chip, 2019.

[98] A. Enterprises, MaixSense A010 distance measurement sensor module (Product listing), 2025.