Journal of Science and Technology – Smart Systems and Devices
Volume 36, Issue 1, January 2026, 001-008

An IoT System to Measure and Detect Foreign Object Debris (FOD) on Airport Runways Using AI and Computer Vision

***Hiep Nguyen-Hoang, Quyen Nguyen-Van, Vinh Tran-Quang*[*]**
Hanoi University of Science and Technology, Ha Noi, Vietnam
[*]Corresponding author email: vinh.tranquang1@hust.edu.vn

Abstract
Air transportation is one of the fastest and safest modes of transport today. However, ensuring its safety and efficiency requires effective management of various risk factors, particularly foreign object debris (FOD) on airport runways. FOD can cause severe damage to aircraft engines and structures, disrupt operations, and even lead to fatal accidents. This study presents a method for detecting, classifying, and estimating the size of FOD on airport runways. The system integrates YOLOv11, a calibrated camera, Canny Edge Detection, and a LiDAR sensor and deploy on a Jetson Nano for efficient processing. Experimental results demonstrate the ability of the system to accurately classify FOD and measure its size in real time. The system achieves an accuracy ranging from 70% to 100% within a 0 to 3 metre distance between the FOD and the camera. This contributes to more efficient FOD detection and collection using robots or rovers, thereby enhancing runway safety and reducing risks in aviation operations.

Keywords: Camera calibration, Canny Edge Detection, FOD, Jetson Nano, LiDAR, YOLOv11.

1. Introduction

Air transportation plays a crucial role in economic growth and in improving quality of life. It is a key driver of global trade, international tourism, and the interconnection of economies and cultures worldwide. With millions of flights operating annually, ensuring airport safety has become increasingly essential. One significant threat to flight safety is foreign object debris (FOD) found on airport runways. FOD refers to any object that does not belong on a runway and poses a risk to aircraft, including loose hardware, luggage fragment, birds, or even small tools. In extreme cases, FOD can lead to catastrophic incidents. The tragic crash of Air France Flight 4590 in 2000 is often cited as a key example that raised global awareness of the importance of detecting FOD [1].

Therefore, FOD detection is crucial to maintaining safety in critical areas, such as airport runways. The traditional method of FOD detection is limited by human factors such as visibility, distraction, and fatigue, which reduce its reliability. Recent advances have enabled automated FOD detection using technologies such as cameras [2], radar [3, 4], LiDAR [5]. Although these systems improve runway safety and airport efficiency, they require substantial investment and significant computational and hardware resources. Despite advances in FOD detection and alerting, most solutions still lack practical methods for debris collection or removal, limiting their real-world effectiveness.

Recent advances in computer vision and embedded systems, particularly deep learning, have shown great promise for automating object detection and classification. Specifically, the You Only Look Once (YOLO) architecture has gained popularity due to its balance of high detection speed and accuracy. In parallel, the Jetson Nano also shows strong performance when running YOLO models. With its NVIDIA GPU, it delivers significantly better real-time object detection performance compared to other embedded platforms such as the Raspberry Pi. Furthermore, camera calibration techniques make it possible to estimate real-world dimensions from 2D images [6], and LiDAR sensors add depth information to improve spatial awareness. By integrating these technologies, we can build a compact real-time system to detect FOD, classify its type, and estimate its size. This capability is crucial for assessing threat levels and choosing appropriate removal methods.

Therefore, in this study, we focus on developing a comprehensive system that integrates YOLOv11, Canny Edge Detection, camera calibration, and LiDAR sensors. All components are implemented on the Jetson Nano platform, leveraging its GPU for efficient edge computing. The primary objective of our work is not only to localize and classify FODs, but also to estimate their physical dimensions (length and width). These estimations remain accurate even when the objects are tilted or positioned obliquely with respect to the camera's viewpoint. This additional information is crucial for enabling a robotic arm to accurately, efficiently grasp and remove the detected objects.

p-ISSN 3093-3285
e-ISSN 3093-3315
https://doi.org/10.51316/jst.187.ssad.2026.36.1.1
Received: Jun 11, 2025; Revised: Jun 27, 2025;
Accepted: Jun 27, 2025; Online: Dec 20, 2025

2. Related Works

Traditionally, FOD collection has been performed by airport personnel through visual inspection and manual removal. This process requires closing the runway, is time consuming, expensive, and prone to error [7]. Preventive measures are also used, such as raising staff awareness, using FOD bins and signs, organizing FOD teams, and inspection walks [3]. Periodic sweeping with specialized equipment, such as the FOD Boss and FOD Sweep, is also carried out. However, despite these preventive measures, maintaining consistent and effective FOD control remains challenging [7]. To improve efficiency and minimize runway closure time, significant research has focused on automatic FOD detection systems with various sensor technologies.

Millimeter-wave radar is a common technology for FOD detection [3]. Radar signal processing methods include Constant False Alarm Rate (CFAR) techniques such as cell averaging CFAR (CA-CFAR), clutter map CFAR, and ordered statistics CFAR (OS-CFAR). A millimeter-wave radar-based FOD detection method has been proposed using the Iterative Adaptive Approach (IAA) [4], which combines interference suppression with exploiting scene sparsity. The challenges for radars include dealing with clutter and interference reflections. Radar systems are also sensitive to leakage signals and strong near-field reflections, which can reduce receiver sensitivity [4]. Their performance may be degraded by adverse weather conditions (e.g., rain or puddles), and the equipment can be expensive.

LiDAR technology has also been applied to FOD detection due to its ability to scan the environment, detect FOD, and generate high resolution 3D point clouds [5]. It performs well in various lighting conditions, including at night. However, its performance can degrade in adverse weather (e.g., rain, fog), and it may struggle with highly reflective or absorptive surfaces. LiDAR systems are also relatively expensive and power-consuming.

Camera systems are widely adopted for FOD detection due to their ability to capture detailed visual information. Early methods, such as Local Binary Patterns and Histogram of Oriented Gradients [2], were proposed, but they struggled with background variability.

Deep learning and computer vision advances have significantly improved object detection accuracy while reducing costs. Models such as Faster R-CNN, YOLO, Xception, and Focal Loss have proven effective in identifying small objects [2], whereas deep convolutional neural networks combined with transfer learning consistently outperform traditional feature-based approaches in material recognition. However, challenges remain, including the difficulty of distinguishing objects from background materials such as concrete and the limited availability of large, diverse annotated FOD datasets, both of which hinder classification accuracy and model reliability.

3. Proposed Solution

The proposed system is capable of localizing, detecting, classifying, and estimating the size of FOD. The modular and efficient design enables straightforward integration into rovers or robotic platforms. This system leverages computer vision and sensor fusion techniques to ensure accurate recognition and measurement of FOD on airport runways. Details of the operation of the system are illustrated in Fig. 1.

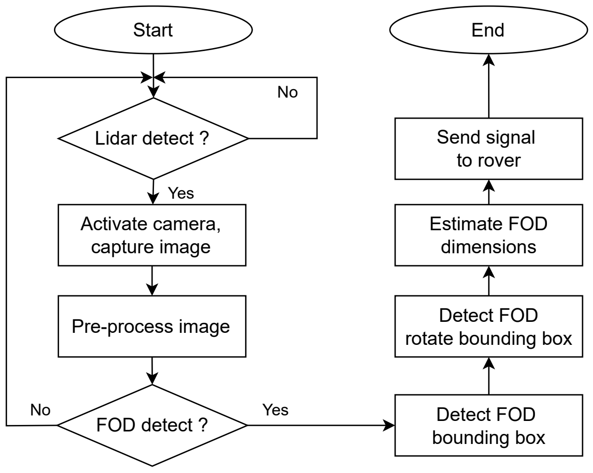


Fig. 1. System model for FOD detection, classification, and size estimation

The LiDAR system remains continuously active throughout the operation of the rover. Upon detecting a potential object, the rover stops, activating the onboard camera to capture an image. The image undergoes preprocessing, including distortion correction, before being fed into a YOLOv11 model fine-tuned on a custom FOD dataset for detection. If no FOD is identified, the system turns off the camera and resumes LiDAR monitoring, allowing the rover to continue moving. If FOD is detected, the system determines its bounding box, computing both standard and rotated versions for a more precise size estimation. Based on this rotated representation, the object's dimensions are calculated, and a signal is sent to the rover's control system to initiate FOD retrieval, using methods like magnetic extraction or robotic arm manipulation.

Before finalizing the system architecture, multiple experiments and comparisons of different algorithmic approaches were conducted, considering constraints such as computational power, energy efficiency, and compact design. Three concepts were evaluated: keeping LiDAR always active while triggering the camera upon detection, using only a continuously active camera without LiDAR, and implementing periodic stops for the rover to capture images.

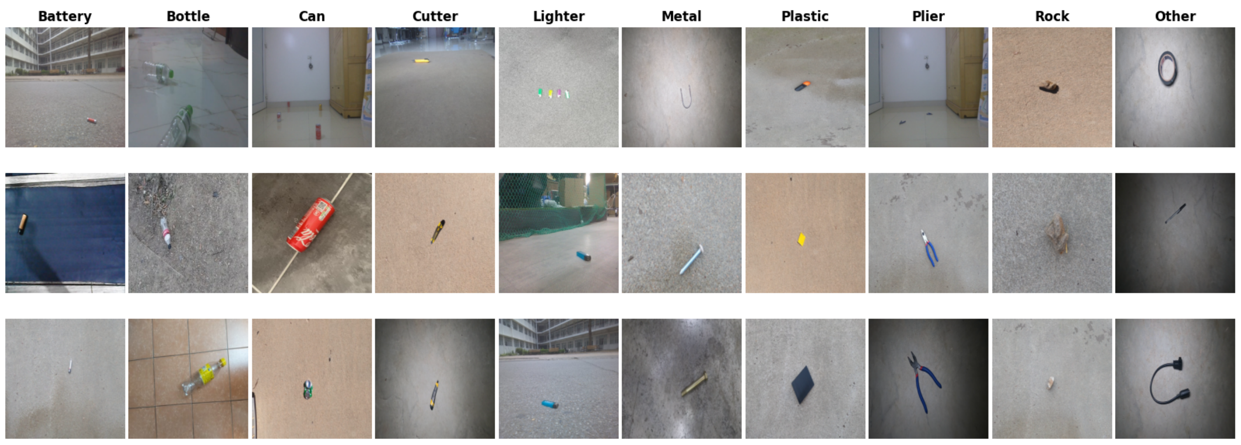


Fig. 2. Example images from the 10-class FOD dataset

To compare the energy consumption between different approaches, we used a power measurement circuit to record the overall power consumption. Specifically, the measurement was carried out by monitoring the power drawn through the adapter powering for Jetson Nano. The circuit includes a current transformer rated at Imax 100A for current sensing and is capable of measuring voltage in the range of AC 80–260V at 50/60Hz. The device measures the total power consumption delivered to the load and includes losses within the adapter. To further enhance the analysis and provide a more detailed breakdown, we additionally used the jtop command-line tool on the Jetson Nano platform. This tool allowed us to directly monitor the CPU power consumption and system load in real-time, making the comparison between different approaches more intuitive and transparent.

In the second approach, where the camera is constantly active and responsible for tasks such as AI-based object detection, rotation of the bounding box estimation, and size computation, the system required a significantly high power consumption. Specifically, the experimental results showed a peak power consumption of up to 10.21W when using the Jetson Nano platform. In contrast, with the first approach, in which only LiDAR is continuously active, the instantaneous peak power was significantly lower, around 2.16W.

The third approach, which periodically activates the camera, helps reduce power consumption but introduces operational limitations for the rover. Without LiDAR, distance estimation is based on alternative methods, such as the pixel-to-millimeter ratio on a static background, which produced low precision (40–60%) within a range of 0–3 meters. Furthermore, since FOD is not always aligned with the direction of the rover in the image, precise navigation and object collection become more challenging.

Incorporating a LiDAR sensor is crucial for determining the distance between the rover and the FOD. Improve navigation by detecting objects aligned with the path of the rover, allowing simple left- or right-turns followed by straight movement toward the target. Furthermore, LiDAR's high-accuracy distance measurements (95-100%) improve the camera size estimation precision, achieving 70–100% accuracy within 0–3 meters, significantly boosting the reliability and efficiency of the FOD collection process.

4. Implementation

4.1. FOD Data Collection, Training, and Evaluation

In deep learning, especially object detection, data quality and quantity are critical for success. No model performs effectively without sufficient high-quality data. After defining the problem and selecting an appropriate model, we collected relevant datasets for training and evaluation.

Data Collection Methodology: Images of FOD were self-collected in real-world settings using a Raspberry Pi Camera Module 2. Additional data sets were collected from online sources such as Roboflow, Kaggle, Google Images, and GitHub. Previously published scientific papers were also reviewed to identify usable datasets.

Data Requirements: We focused on capturing images of FOD on flat surfaces such as concrete floors under varying distances and conditions. For objects within 0–1 meters, images were sourced from online datasets and previous research. For the 1 to 3 meter range, the images were self-captured using a camera, including raw images, zoomed-in shots using the camera's zoom function, and augmented images generated by cropping to improve diversity.

Outcome: As a result, a dataset was constructed containing images of 10 different FOD classes, as shown in Fig. 2. To enhance its robustness, the dataset was expanded to a total of 23,284 images using various data augmentation techniques. This dataset was used to train YOLO-based deep learning models to detect FOD under varied conditions.

After refining and augmenting the data, we trained the object detection model using YOLOv11x with pre-trained weights as the baseline, then fine-tuned it on our custom dataset. The data was split into training (70%), validation (20%), and testing (10%), ensuring an unbiased final evaluation. The model achieved 96.8% mAP@0.5, 97.3% precision, and 94.3% recall, demonstrating high detection accuracy and effective localization in FOD classes.

4.2. Camera Calibration

Camera calibration is crucial to improving the accuracy of computer vision systems. By determining the intrinsic parameters and distortion coefficients and evaluating them using the mean reprojection error, we can effectively remove image distortions. This enhances the reliability of subsequent tasks such as 2D-to-3D coordinate transformation, measure object's size in real-world. The calibration results obtained from the Raspberry Pi Camera Module V2 can be reused for all future images captured by the same camera, ensuring consistent accuracy across applications.

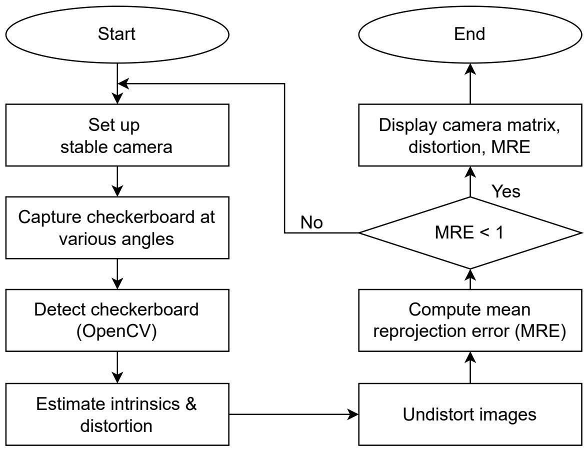

Start
Set up stable camera
Capture checkerboard at various angles
Detect checkerboard (OpenCV)
Estimate intrinsics & distortion
Undistort images
Compute mean reprojection error (MRE)
MRE < 1
No
Yes
Display camera matrix, distortion, MRE
End

Fig. 3. Camera calibration workflow using OpenCV

The camera calibration process, presented in Fig. 3, produces two key outputs:

Camera Matrix (Intrinsic Parameters): This 3×3 matrix contains intrinsic parameters unique to the camera. It is used to map 3D points in the world to 2D image coordinates [6].

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{1}$$

where f_x and f_y represent the focal lengths in the x and y directions, measured in pixels. Similarly, c_x and c_y denote the optical center, which is typically located near the center of the image.

Distortion Coefficients: These parameters compensate for nonlinear distortions introduced by the lens, addressing both radial and tangential distortion effects. By applying these corrections, an undistorted image can be obtained, better preserving real-world geometry [6].

$$\mathbf{D} = [k_1, k_2, p_1, p_2, k_3], \tag{2}$$

where k_1, k_2, k_3 represents the radial distortion, which accounts for the effects of barrel or pincushion distortion. Similarly, p_1 and p_2 correspond to tangential distortion, addressing any misalignment between the lens and the sensor.

4.3. Rotate Bounding Box Detection

Edge detection is essential in computer vision to extract structural details from images. Our system employs Canny Edge Detection to suppress noise and accurately identify object contours for size estimation. Each detected FOD is first cropped using its bounding box to minimize background interference. Then, Canny Edge Detection is applied to refine the boundary of the object. Finally, images are converted to grayscale (0–255 intensity) to reduce memory usage, improve processing speed, and ensure compatibility with (3) [8].

$$I_{\text{gray}} = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B. \tag{3}$$

Before applying edge detection algorithms such as Canny, image blurring serves as a crucial preprocessing step to reduce unwanted noise and enhance contour detection accuracy. To achieve this, a Gaussian blur is applied, which smooths the image by convolving it with a 2D Gaussian kernel [9].

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right), \tag{4}$$

$$I_{\text{blur}} = I_{\text{gray}} * G(x,y). \tag{5}$$

Here, x and y represent the relative coordinates of the pixel within the kernel. The standard deviation, σ, regulates the level of smoothing applied within the kernel region, with higher values of σ producing more pronounced blurring effects. The kernel matrix, defined as $(2k+1) \times (2k+1)$, determines the neighborhood of pixels surrounding each target pixel that are taken into account during convolution. Increasing the size of the kernel expands the spatial influence of the blur.

After converting the image to grayscale and applying Gaussian blur, pixels are categorized as strong edges, weak edges, or non-edges based on their grayscale intensity. This classification relies on the gradient, which measures the rate of intensity change at each pixel. The gradient is computed by convolving the image region I with the Sobel filter, where I represents a neighborhood around each pixel with a size defined by the kernel matrix $(2k+1) \times (2k+1)$ [9].

$$G_x = I * K_x,\ G_y = I * K_y, \tag{6}$$

$$|G| = \sqrt{G_x^2 + G_y^2}, \tag{7}$$

$$\theta(x,y) = \arctan\left(\frac{G_y}{G_x}\right). \tag{8}$$

After computing the magnitudes and directions of the gradient, neighboring pixels often exhibit high gradient values, forming thick edges. However, for precise edge detection, the algorithm applies non-maximum suppression, retaining only local maximum gradient magnitudes along $\theta(x,y)$ while discarding others. This is followed by double thresholding and edge tracking by hysteresis. Pixels with gradients above the high threshold are classified as strong edges (intensity 255), those between low and high thresholds are weak edges (kept only if connected to strong edges), and those below the low threshold are considered non-edges (intensity 0). These multi-stage processes ensure accurate edge detection while reducing noise and false positives.

To improve contour continuity, two fundamental morphological operations are applied, dilation and erosion [10]. Erosion eliminates isolated noise pixels and smooths boundaries by turning edge pixels into a background. In contrast, dilation reconnects broken edges and strengthens the prominence of the object in the image. For a precise object description, the contours are extracted from the binary edge map. Our approach focuses solely on external contours, disregarding nested ones, and applies approximation to retain essential shape-defining points. The extracted contours are sorted, and the small-area contours are discarded to remove noise. A rotated bounding box is then computed to better enclose tilted objects without distortion. The method considers each edge of the convex hull of an object as a potential base for a bounding rectangle. Four calipers are used, two parallel to the base and two orthogonal, to form the enclosed rectangle. The algorithm selects the rectangle of the smallest area in rotations, ensuring the most compact bounding box for rotated objects. Fig. 4 illustrates the full process.

4.4. 2D-to-3D Coordinate Transformation

In computer vision, the camera model defines the mathematical link between a 2D pixel coordinate and its corresponding 3D point in the real world. This transformation is essential for applications like 3D reconstruction, object localization, and size estimation. Given the common structure of commodity cameras, the pinhole camera model [11] is widely used to describe image formation. This section outlines the process of projecting pixel coordinates into 3D space using camera parameters and geometric constraints.

According to OpenCV conventions, the image coordinate system is defined on the 2D image plane with its origin at the top-left corner. The x-axis extends

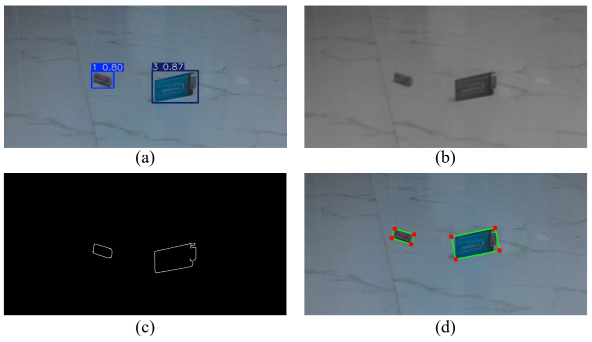


Fig. 4. These experiments illustrate the FOD detection process: (a) Bounding box detection, (b) Grayscale image, (c) Canny Edge Detection, and (d) Final rotate bounding box

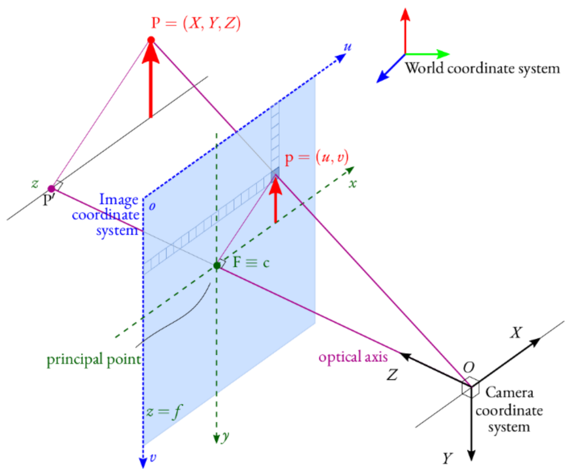


Fig. 5. Pinhole camera model

to the right, while the y-axis points downward, aligned with the pixel grid structure. The camera coordinate system is centered at the optical center O, with the z-axis extending forward into the scene, the x-axis directed to the right, and the y-axis downward. The world coordinate system represents global object positions and is typically linked to the camera coordinate system via extrinsic parameters (rotation and translation matrices). Due to image inversion in optical systems, the y-axis in the world coordinate system may appear flipped relative to the camera's y-axis. The components and their functions are illustrated in Fig. 5 [12].

Let $\mathbf{P} = \begin{bmatrix} X & Y & Z \end{bmatrix}^T$ be an arbitrary 3D point seen by a camera situated at the origin O in camera coordinate system, and $\mathbf{p} = \begin{bmatrix} u & v \end{bmatrix}^T$ be the image of $\mathbf{P}$ in the image coordinate system, the point $\mathbf{p}$ corresponds to the projection of a 3D point $\mathbf{P}$ onto the image plane. This projection is formed by the intersection of the light ray from $\mathbf{P}$ passing through the optical center O of the camera with the image plane.

Suppose that the projective plane is perpendicular to the z-axis of the camera coordinate system; its intersection with the optical axis occurs at the principal point, denoted as $\mathbf{F} = \begin{bmatrix} 0 & 0 & f \end{bmatrix}^T$ in the camera coordinate system or $\mathbf{c} = \begin{bmatrix} c_x & c_y \end{bmatrix}^T$ in the image coordinate system. Therefore, the relationship between *ouv* and *cxy* is given by [12]:

$$u = x + c_x, \quad v = y + c_y. \tag{9}$$

To convert to real-world units, we need the imager scale factors s_u, s_v, which define the number of pixels per unit (e.g. DPI or PPI). The projected object size with respect to the real size and distance from the camera (focal length f) gives:

$$x = s_u \frac{fX}{Z}, \quad y = s_v \frac{fY}{Z}. \tag{10}$$

From (10), we establish the relationship between a 3D point in the camera coordinate system and its corresponding projection in the image space. This transformation defines how spatial information is mapped onto the image plane, facilitating tasks such as object localization and scene reconstruction.

$$u = s_u \frac{fX}{Z} + c_x, \quad v = s_v \frac{fY}{Z} + c_y, \tag{11}$$

where $f_x = s_u f$ and $f_y = s_v f$. Together with c_x and c_y, these parameters form the Camera Matrix **K**, as defined in (1), which is derived through camera calibration.

In order to determine the actual size of the object, it is essential to retrieve the 3D coordinates in the reverse direction, which requires depth information, i.e., the Z coordinate, of each pixel. Based on (11), we obtain:

$$X = \frac{(u - c_x)Z}{f_x}, \quad Y = \frac{(v - c_y)Z}{f_y}. \tag{12}$$

Using similar triangles in Fig. 5, we get the following.

$$\frac{OP'}{OF} = \frac{OP}{Op} \Leftrightarrow \frac{Z}{f} = \frac{d}{\sqrt{f^2 + m^2}}, \tag{13}$$

and

$$\frac{Fp}{PP'} = \frac{OF}{OP'} \Leftrightarrow \frac{m}{n} = \frac{f}{Z}, \tag{14}$$

where the size m of the projection Fp relating to the size n of the real segment PP' is given by similar triangles. From (13) and (14), we have:

$$Z = \frac{df}{\sqrt{f^2 + f^2 n^2 / Z^2}} = \frac{d}{\sqrt{1 + \frac{X^2 + Y^2}{Z^2}}}. \tag{15}$$

Equation (12) gives:

$$\frac{X}{Z} = \frac{u - c_x}{f_x}, \quad \frac{Y}{Z} = \frac{v - c_y}{f_y}. \tag{16}$$

Substituting (16) into (15) gives:

$$Z = \frac{d}{\sqrt{\left(\frac{u - c_x}{f_x}\right)^2 + \left(\frac{v - c_y}{f_y}\right)^2 + 1}}, \tag{17}$$

where d represents the distance from the point of the object to the camera, obtained from the LiDAR measurements. Using the coordinates of the four vertices of the rotated bounding box, we transform these points into 3D space based on the given equation. The object's actual dimensions (length × width) are then calculated using Euclidean distances between the corresponding 3D points.

4.5. Implement on Jetson Nano

In modern IoT projects, the selection of the right hardware platform is crucial to efficiency, deployability, and long-term stability. Among the available options, the NVIDIA Jetson Nano stands out for embedded computing because of its balance of performance, power efficiency, and ease of integration. Its 128-core NVIDIA Maxwell GPU onboard enables parallel processing, benefiting deep learning, image processing, and computer vision tasks. However, deploying AI models with GPU acceleration using PyTorch, Torchvision, and NumPy presents software compatibility challenges. The Ultralytics YOLO library requires Python 3.7+, while OpenCV depends on Python 3.6+, yet Jetson Nano's official PyTorch wheel supports only JetPack 4.6, limited to Python 3.6, causing version conflicts.

To address compatibility issues, PyTorch must be built from source (available on GitHub) using Python 3.8, allowing seamless integration with modern AI libraries and GPU acceleration. The build process employs CMake with the Ninja build system for efficient compilation, while Clang is used as a compiler to enhance performance and compatibility. Furthermore, NumPy support is explicitly enabled (`USE_NUMPY=ON`) to ensure smooth numerical and AI-related computations.

5. Test Results and Analysis

5.1. LiDAR Detection Capability Evaluation

To evaluate the LiDAR's ability to detect FOD of various sizes, we conducted tests using objects with different dimensions, focusing on maximum detection range, sensitivity to object size, measurement accuracy, and performance under different lighting and surface conditions. The tests included two environments: an ideal setting with a concrete floor at night, minimizing ambient light interference and improving signal-to-noise ratio (SNR) for enhanced accuracy, and a challenging setting with smooth ceramic tile or transparent surfaces like acrylic or glass-like tiles, which could cause

Table 1. Measured object size and accuracy at different distances with camera field of view set at 60 cm height

Object		**Battery**		**Lighter**		**Cutter**		**Piler**	
		Length	Width	Length	Width	Length	Width	Length	Width
Actual size (cm)		5.0	1.5	8.3	2.6	15.2	4.0	12.8	8.0
100 cm	Measured size	5.1	1.3	8.3	2.3	15.2	4.0	12.9	6.1
	Accuracy (%)	**98.0**	**86.7**	**100**	**88.5**	**100**	**100**	**99. 2**	**76.3**
160 cm	Measured size	4.8	1.2	8.0	2.3	14.8	3.5	12.6	5.8
	Accuracy (%)	**96.0**	**80.0**	**96.4**	**88.5**	**97.4**	**87.5**	**98.4**	**72.5**
220 cm	Measured size	4.7	1.2	7.9	2.2	14.2	3.4	11.4	5.8
	Accuracy (%)	**94.0**	**80.0**	**95.1**	**84.6**	**93.4**	**85.0**	**89.1**	**72.5**
280 cm	Measured size	4.3	1.1	7.5	2.0	14.1	3.1	12.3	5.5
	Accuracy (%)	**86.0**	**73.3**	**90.4**	**76.9**	**92.8**	**77.5**	**96.1**	**68.8**

LiDAR beams to scatter or pass through, leading to weak or missing return signals. The performance of LiDAR in detecting FOD under two different conditions is illustrated in Fig. 6, highlighting the variations between different object sizes. Under low light conditions, the LiDAR exhibited significantly better detection performance than on glossy surfaces. As shown in Fig. 6b, for a 17×5 cm FOD, the maximum detection range on the glossy floor was limited to 925 cm, while in low light conditions it reached up to 1350 cm. Measurement accuracy decreased with increasing distance, with noticeable errors occurring beyond 10 m. In contrast, for objects within the range of 0-3 m, the LiDAR consistently maintained high accuracy and stable detection rates between 95% and 100%.

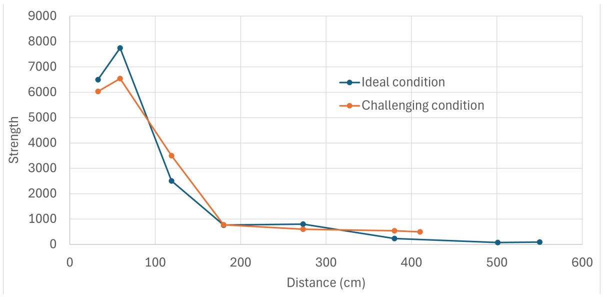
5.2. Experimental Validation of FOD Size Estimation

To evaluate the precision of the estimation of the size of the FOD, we conducted extensive field experiments in a controlled environment that replicates real-world airport conditions. Various FOD objects were placed at different distances from the camera, and the tests also considered variations in the camera's field of view by adjusting its height. A subset of randomly selected data, captured with the camera positioned at a height of 60 cm, is presented in Table 1.

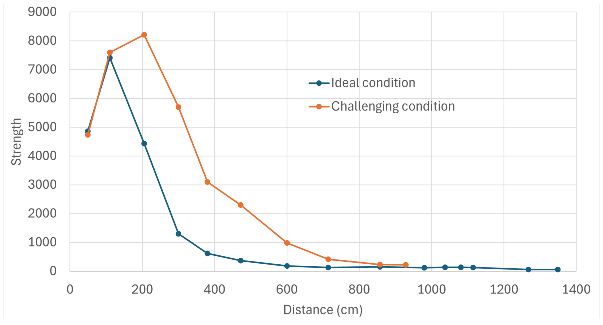
The test results indicate that the accuracy of the system decreases approximately linearly with increasing distance between the camera and the object. Regular, cubic-shaped objects (e.g., lighters, batteries, cutters) are detected and measured more precisely than irregular objects (e.g., pliers, stones). Furthermore, the length estimation achieves higher accuracy (80–100%) compared to the width (70–90%), largely influenced by factors such as the field of view of the camera, the position of the object, the quality of the boundary box, and the distance from the camera.

5.3. Testing GPU Performance on Jetson Nano

To assess the Jetson Nano's GPU efficiency, we conducted tests under two scenarios: continuous video processing and single-frame image inference. The

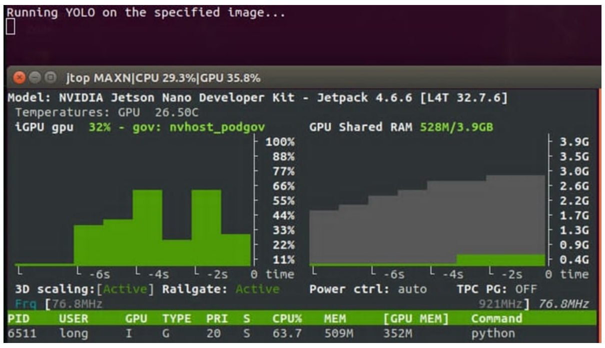


(a)



(b)

Fig. 6. LiDAR signal strength and distance under ideal and challenging conditions: (a) FOD size of 2×5 cm and (b) FOD size of 17×5 cm



```
Running YOLO on the specified image...

jtop MAXN|CPU 29.3%|GPU 35.8%
Model: NVIDIA Jetson Nano Developer Kit - Jetpack 4.6.6 [L4T 32.7.6]
 Temperatures: GPU  26.50C
 iGPU gpu  32% - gov: nvhost_podgov      GPU Shared RAM 528M/3.9GB
 3D scaling:[Active] Railgate: Active    Power ctrl: auto     TPC PG: OFF
 Frq [76.8MHz                                              921MHz] 76.8MHz
PID    USER      GPU  TYPE  PRI  S   CPU%   MEM     [GPU MEM]   Command
6511   long      I    G     20   S   63.7   509M    352M        python
```

Fig. 7. GPU usage evaluation on Jetson Nano during single-frame processing

results indicate that continuous video processing fully utilizes the GPU (close to 100%), while single-frame image inference uses only 50 to 70% of the GPU, depending on the complexity and resolution of the model. Fig. 7 illustrates the usage of GPU during single-image processing. The inference time for video processing using the GPU averages 935 ms per frame, which is 15–17 times faster than the CPU's average of 16.028 ms. However, for single-image processing, the GPU takes approximately 1 minute which offers little to no performance gain compared to the CPU.

6. Conclusion

This paper presents a solution for detecting, classifying and estimating the size of Foreign Object Debris (FOD) on airport runways. The system can be integrated with a robotic arm for automatic retrieval based on estimated position and size. Experiments confirm accurate classification and real-time size measurement, with LiDAR achieving a precision of 95 to 100% in distance estimation and a size estimation of 70 to 100% for objects within 0 to 3 meters. GPU acceleration enhances the video processing speed by 15 to 17 times compared to CPU-only execution, ensuring efficient real-time operation. The system is well-suited for IoT applications, including smart airport systems, and can be adapted for future developments.

The proposed system performs well in detecting, classifying, and estimating FOD size, but has some limitations. The camera field of view impacts the accuracy, with length measurements being more precise than width due to perspective. Although LiDAR provides accurate distance measurements beyond 3 meters, the camera struggles with small distant objects, leading to detection errors. Using higher-resolution cameras, optical zoom lenses, or multicamera setups can improve visibility and improve detection reliability.

References

[1] J. Shan, L. Miccinesi, A. Beni, L. Pagnini, A. Cioncolini, and M. Pieraccini, A review of foreign object debris detection on airport runways: sensors and algorithms, Remote Sensing, vol. 17, no. 2, pp. 225, 2025.

[2] J. Almeida, G. Cruz, D. Silva, and T. Oliveira, Application of deep learning to the detection of foreign object debris at Aerodromes' movement area. in VISIGRAPP (5: VISAPP), 2023, pp. 814–821.

[3] G. Mehdi and J. Miao, Millimeter wave FMCW radar for foreign object debris (FOD) detection at airport runways, in Proceedings of 2012 9th International Bhurban Conference on Applied Sciences & Technology (IBCAST), 2012, pp. 407–412.

[4] Y. Wan, X. Liang, X. Bu, and Y. Liu, FOD detection method based on iterative adaptive approach for millimeter-wave radar, Sensors, vol. 21, no. 4, pp. 1241, 2021.

[5] C. A. Amadi, K. Mbanisi, and W. J. Smit, An Introduction to the ros2_control framework using a low cost, Differential Drive Robot, Sept. 2024. https://doi.org/10.13140/RG.2.2.15748.54408.

[6] P. Todorov, Multi-camera Calibration, K. Ikeuchi, Springer International Publishing, 2021, pp. 825–825.

[7] S. Öztürk and A. E. Kuzucuoğlu, A multi-robot coordination approach for autonomous runway Foreign Object Debris (FOD) clearance, Robotics and Autonomous Systems, vol. 75, pp. 244–259, 2016.

[8] C. Saravanan, Color image to grayscale image conversion, in 2010 Second International Conference on Computer Engineering and Applications, 2010, pp. 196–199.

[9] S. A. Parah, J. A. Sheikh, J. A. Akhoon, N. A. Loan, and G. M. Bhat, Information hiding in edges: A high capacity information hiding technique using hybrid edge detection, Multimedia Tools and Applications, vol. 77, pp. 185–207, 2018.

[10] P. Soille, Morphological Image Analysis: Principles and Applications, 2nd ed., Springer-Verlag, 2003.

[11] P. Sturm, Pinhole camera model, Computer Vision: A Reference Guide, pp. 983–986, 2021.

[12] L. Hoang-An, Camera model: intrinsic parameters, July 30, 2018. [Online] Available: https://lhoangan.github.io/camera-params/. Accessed on: Mar. 12025.