# An Introduction to TurtleBot4 Systems for Object Detection

Allan Paiz

## 1 Introduction

This paper is meant to briefly introduce and document the available methods for object detection via the TurtleBot4 (TB4). This paper covers short introductions to the hardware, methods used, mathematical foundations, and recommended software architecture for its use. There are likely pieces and concepts I may have missed or information that is not necessary.

## 2 TurtleBot 4

The TurtleBot4 (TB4) uses an iRobot Create 3 mobile base, a Raspberry Pi 4B onboard computer, a Spatial AI stereo (OAK-D) camera, and 2D LiDAR (RPLIDAR-A1). TurtleBot4's `turtlebot4_bringup` package contains the launch and configuration files that run the robot's software; this paper is interested in the **OAK-D** and **RPLIDAR** nodes [1]. It uses a Create 3 as the base platform, and builds on it with the TB4 shell and UI boards on a Raspberry Pi 4B that runs the TB4 software. On top of the UI board sits the RPLIDAR A1M8 360 degree lidar, and an OAK-D camera.


| TurtleBot 4 Standard | | TurtleBot 4 Lite |
|---|---|---|
| 341 x 339 x 351 mm | **Dimensions** | 341 x 339 x 192 mm |
| 3.9 kg | **Weight** | 3.3 kg |
| 0.31 m/s | **Max. Speed** | 0.31 m/s |
| 9 kg - Default<br>15 kg - Custom Configuration | **Max. Payload** | 9 kg - Default<br>15 kg - Custom Configuration |
| 2.5 - 4.0 hrs (load dependent) | **Operating Time** | 2.5 - 4.0 hrs (load dependent) |
| OAK-D-PRO | **Camera** | OAK-D-LITE |
| RPLIDAR-A1 | **LiDAR** | RPLIDAR-A1 |
| Yes | **Accessible Power & USB Ports** | No |
| Yes | **OLED Display** | No |
| Yes | **Mounting Plate** | No |
| ROS 2 | **Software** | ROS 2 |
| Raspberry Pi 4B (4 GB) | **Computer** | Raspberry Pi 4B (4 GB) |

Figure 1: TurtleBot4 User Manual [1]

## 2.1 RPLIDAR

The RPLIDAR A1M8 is a 360-degree laser range scanner with a 0.15 – 12 m (≈ 0.5 – 39.4 ft) range. It generates a 2D planar scan of the robot's surroundings. It has a configurable scan rate from 2–10 Hz, a sample rate up to 8,000 samples per second, and angular resolution at or below 1 degree [2]. The TB4 manual states that USB 2.0 is sufficient for the RPLIDAR. Its range measurements are strong for detecting nearby obstacles at the scan plane, but weak for semantic classification because the scan is a single horizontal slice.

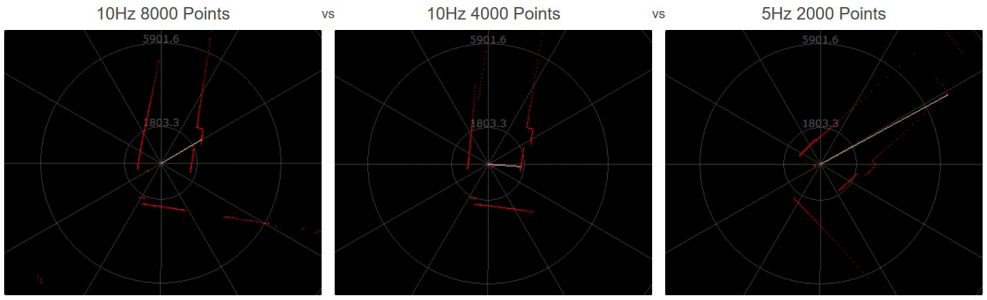


Figure 2: SLAMTEC RPLIDAR-A1 overview, comparison under different conditions [2]

## 2.2 OAK-D Camera

The standard model uses an OAK-D Pro, while the Lite model uses an OAK-D Lite [3]. The camera from Luxonis uses a 4K IMX214 color sensor along with a pair of OV7251/OV9282 (Lite/Pro) stereo sensors to produce high-quality color and depth images. The Pro sensors add an IR laser dot projector and an IR illumination LED (see Figure 3); this creates higher-quality depth images and performs better in low light [4]. The onboard Myriad X VPU allows the camera to run computer vision applications, object tracking, and AI models [5].

Figure 3: Luxonis Docs IR Illumination [4]

## 2.3 Create 3

The Create 3 base documentation lists bumper, cliff, wheel drop, infrared proximity, odometry, and IMU sensing. This could be used as a near-field safety and fault-detection layer [6, 7] to trigger emergency stopping, low-speed recovery, and validation of nearby obstacles that the camera or LiDAR missed.

# 3 Methods

## 3.1 Two-stage detectors

Two-stage detectors rely on a sequential workflow that mimics how a human might analyze a scene. The detector first proposes object regions and then proceeds to classification and refinement [8]. Two-stage detectors separate tasks but they tend to be heavier detectors.

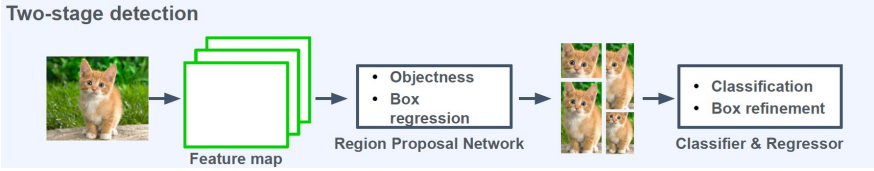


Figure 4: Infos Gene-cotrading Two Stage Object Detection

## 3.2 One-stage detectors and YOLO

One-stage models analyze the entire image in a single pass, simultaneously predicting bounding box coordinates and class probabilities from input pixels [8]. Figure 5 shows bounding boxes and class probabilities in three selected frames from an output video using the model `YOLOv8`.


Figure 5: Sample frames from video output using YOLOv8 on our TB4s

The current model `YOLOv26` introduces native end-to-end inference, a lighter detection head, an updated training recipe, and task-specific heads for detection, segmentation, pose estimation, classification, and oriented detection [9].

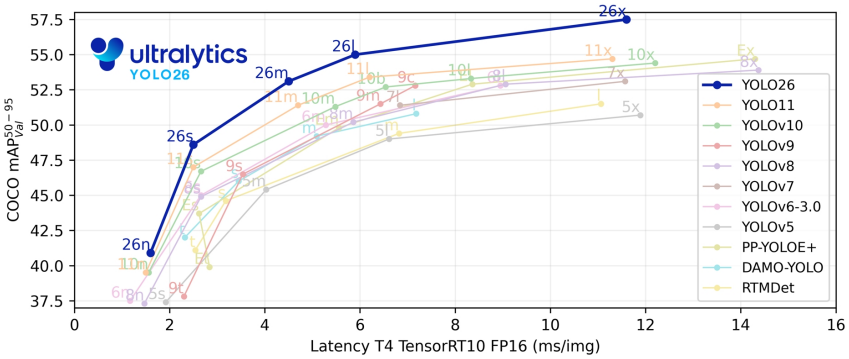


Figure 6: Ultralytics Docs/Models/YOLOv26 Overview [9]

### 3.3 Transformer detectors

A Real-Time Detection Transformer (RT-DETR) is a cutting-edge end-to-end object detector that maintains high accuracy [10]. For a TB4, this model is stronger as a research baseline if the project goal is comparing detection formulations, studying open-vocabulary detection, or using external compute.

### 3.4 LiDAR

The available 2D LiDAR detection should be used for obstacle detection or tracking. The RPLIDAR A1 outputs sampling data continuously; each sample point contains the information shown in Figure 7. LiDAR methods could be used to produce trackable obstacle states for dynamic navigation.

| Data Type | Unit | Description |
|---|---|---|
| Distance | mm | Current measured distance value between the rotating core of the RPLIDAR A1 and the sampling point |
| Heading | degree | Current heading angle of the measurement |
| Quality | level | Quality of the measurement |
| Start Flag | (Boolean) | Flag of a new scan |

Figure 7: SLAMTEC A1M8 Datasheet [2]

DROW showed that convolutional neural networks can be applied to 2D range data for wheelchairs and walkers, and the authors released a ROS node and dataset [11]. A 2026 paper encodes three consecutive 2D LiDAR scans as RGB channels for YOLOv8n input and reports high accuracy and low latency [12].

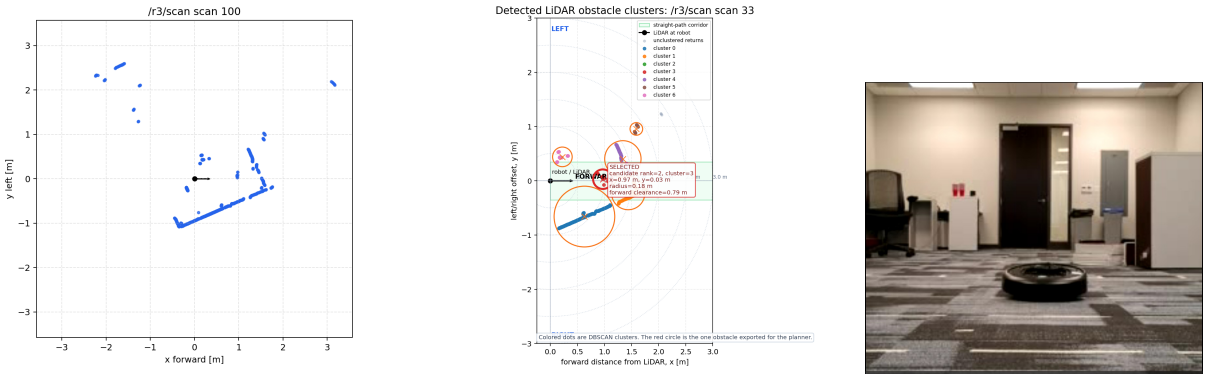


Figure 8: IBMRobo Lab Example (left) raw LiDAR probe, (middle) scan plot with object detection, and (right) video screenshot

# 4 Mathematical fundamentals

## 4.1 Camera projection

Camera detection begins with projective geometry.

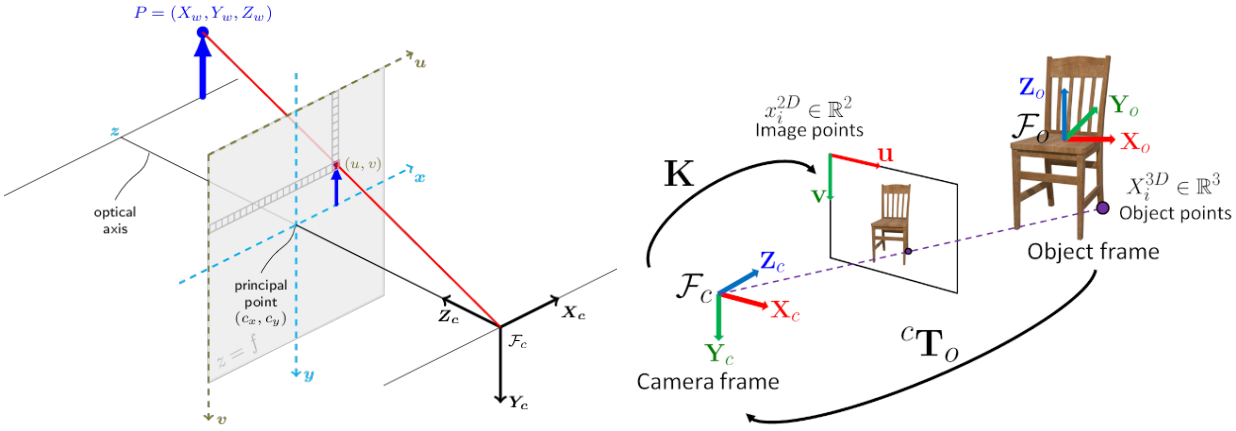


Figure 9: OpenCV Docs, (left) Pinhole camera model and (right) Perspective projection from object to camera frame [13]

A 3D point $P_w = [X_w, Y_w, Z_w, 1]^T$ in the world frame is transformed into the camera frame by extrinsic parameters $[R|t]$, then projected into image coordinates by the intrinsic matrix $K$:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R|t] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}. \tag{1}$$

The intrinsic matrix contains focal lengths and the principal point:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

In normalized form,

$$u = f_x \frac{X_c}{Z_c} + c_x, \qquad v = f_y \frac{Y_c}{Z_c} + c_y. \tag{3}$$

Calibration estimates the intrinsic and extrinsic parameters needed to map between 3D world coordinates and 2D image coordinates. The intrinsic parameters encode optical center and focal length, while extrinsic parameters encode camera pose [14, 13]. This model is required whenever a 2D bounding box must be converted into a robot-frame obstacle estimate.

## 4.2 Stereo depth

The OAK-D stereo pair estimates depth from disparity. If the same point appears at different horizontal image coordinates in the left and right rectified stereo images, disparity $d$ is the pixel shift between the views. Depth is approximately

$$Z = \frac{fB}{d}, \tag{4}$$

where $f$ is the focal length in pixels and $B$ is the stereo baseline. Stereo depth from DepthAI is disparity-based and provides modes and filters [15].

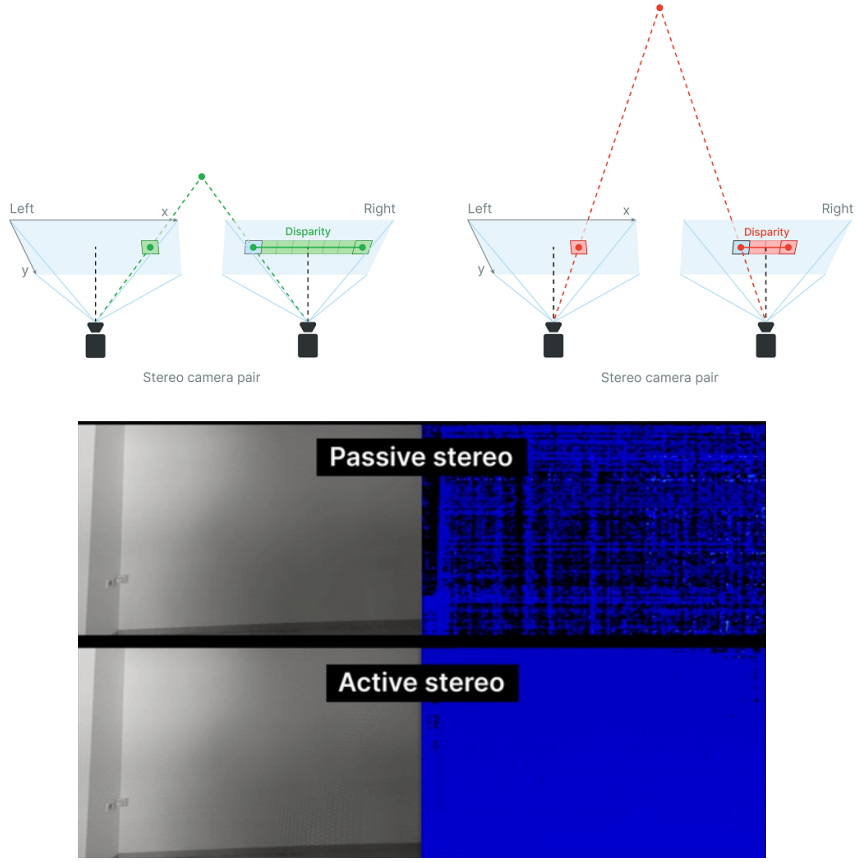


Figure 10: Luxonis Docs, (top) Stereo Depth Basics and (bottom) Fixing noisy depth [15]

Small disparity errors produce large depth errors at long range because $Z$ is inversely proportional to $d$. Close objects are easier to localize because disparity is large. Long-range objects, textureless objects, transparent surfaces, and motion blur are harder. The controller should therefore receive depth uncertainty, not only a point estimate.

### 4.3 LiDAR scan geometry

The RPLIDAR provides polar range measurements:

$$z_i = (r_i, \theta_i). \tag{5}$$

Each valid beam maps to a point in the LiDAR frame:

$$x_i = r_i \cos\theta_i, \qquad y_i = r_i \sin\theta_i. \tag{6}$$

A 2D scan can be segmented by grouping adjacent beams with small range discontinuities. A cluster can then be converted into geometric features: centroid, width, arc length, curvature, closest point, and radial velocity if tracked across scans. Classical 2D LiDAR obstacle detection is mainly geometry and tracking [11, 16].

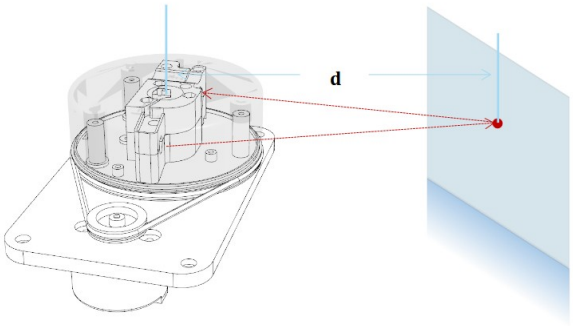


Figure 11: SLAMTEC A1M8 Datasheet, The RPLIDAR A1 Working Schematic, Fig 1-2. [2]

### 4.4 State estimation and tracking

Frame detections are noisy, a controller would need tracks. A standard obstacle track can use a constant-velocity state:

$$x_j = [p_x, p_y, v_x, v_y]^T, \tag{7}$$

with prediction model

$$x_{j,k+1} = Ax_{j,k} + w_k, \tag{8}$$

and measurement model

$$y_{j,k} = Hx_{j,k} + v_k. \tag{9}$$

The estimator can be a Kalman filter or an extended Kalman filter when the measurement model is nonlinear. Data association can be handled by nearest-neighbor matching, Hungarian assignment, or track confidence logic. A Kalman filter with covariance inflation could be used for the safety-margin formulation. [18]

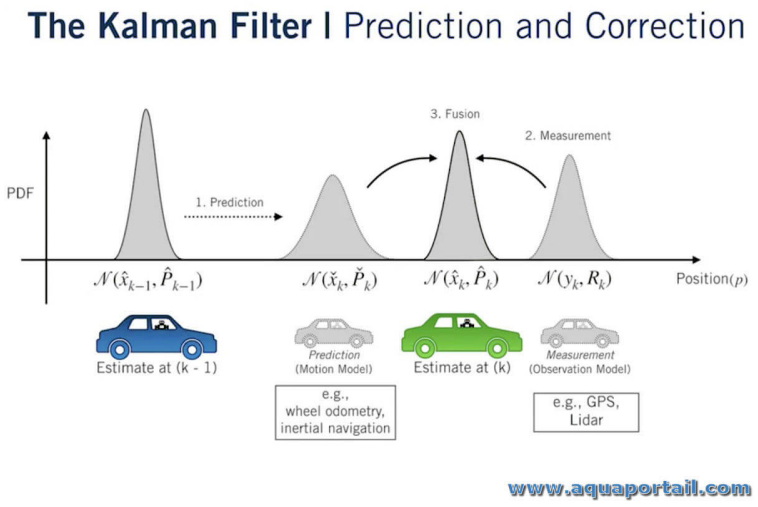


Figure 12: AquaPortail, The Kalman Filter [19]

# 5 Software architecture

ChatGPT 5.5 recommends an architecture based on a ROS 2 graph with separate nodes for sensing, detection, tracking, fusion, planning, control, and safety [20]. When the OAK-D is used as the semantic detector, image inference should run as close to the sensor as possible, preferably using DepthAI's pipeline. ROS 2 can then orchestrate message passing, filtering, tracking, logging, and controller integration. The RPLIDAR will feed tracked obstacle states into the planning and event-triggered control layer and act as the geometric safety layer.

## 5.1 Data pipeline

The perception-control data pipeline should be timestamped and frame-consistent. An example pipeline provided by ChatGPT 5.5 can be described in seven stages.

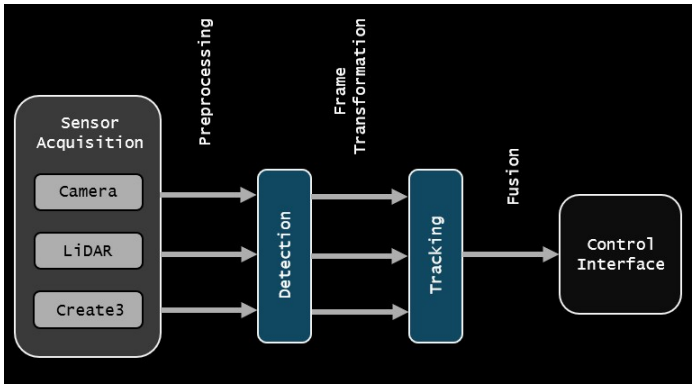


Figure 13: Example Pipeline [20]

**Sensor acquisition.** The OAK-D publishes RGB images, stereo depth, and spatial detections. The RPLIDAR publishes planar range scans. The Create 3 publishes odometry, IMU, hazard, and proximity information. Every message must preserve acquisition timestamp and processing timestamp.

**Preprocessing.** Preprocessing must match the detector input; stereo depth requires confidence filtering and clipping to the robot's useful indoor range. LiDAR scans require invalid range removal, angular filtering, and conversion from polar beams to Cartesian points.

**Detection.** The camera detector outputs boxes, labels, and confidences. A DepthAI spatial detection graph can attach approximate 3D coordinates to each detection by combining YOLO output with stereo depth. The LiDAR detector should produce geometric clusters.

**Frame transformation.** All detections must be transformed into a common frame. The usual chain is `sensor_frame -> base_link -> odom -> map`. The (camera/LiDAR)-to-base transforms are fixed, while the base-to-(odom/map) transform comes from odometry and localization. Incorrect transforms produce false obstacle locations.

**Tracking.** Tracking associates detections over time and estimates position, velocity, and covariance. Camera detections provide semantic class and approximate 3D position. LiDAR clusters provide planar range and local obstacle contour. Create 3 hazards provide close range events. The tracker should publish a single tracked-obstacle-array message with obstacle ID, state, covariance, class, confidence, footprint, and timestamp.

**Fusion.** For a TB4, fusion can enhance the performance of 3D object detection by utilizing complementary information between depth-aware LiDAR points and semantically rich images [21]. Camera detections supply class and stereo position, while LiDAR supplies planar occupancy, and both are combined in a tracker.

**Control interface.** The planner and controller should subscribe to tracked obstacles, not raw detections. They should receive the latest obstacle state estimate and its uncertainty.

# 6 Conclusion

This report introduces the TB4 sensing stack and the perception methods for object detection.

## 6.1 Lab Tests

I used Codex/GPT-5.5 to generate scripts to test some of these methods, as seen in Figure 5 and Figure 7. I was able to produce video and lidar scans and, use open source and known algorithms and methods.

Using the existing obstacle avoidance controller, Figure 14 shows the lidar scans and hardcoded obstacle selection on the left, and the right plot shows the execution and trajectory comparisons. Notice the TB4 stops and bounces, this is because the generated system currently has a bug during the transform phase, which miscalculated where the obstacle actually is.

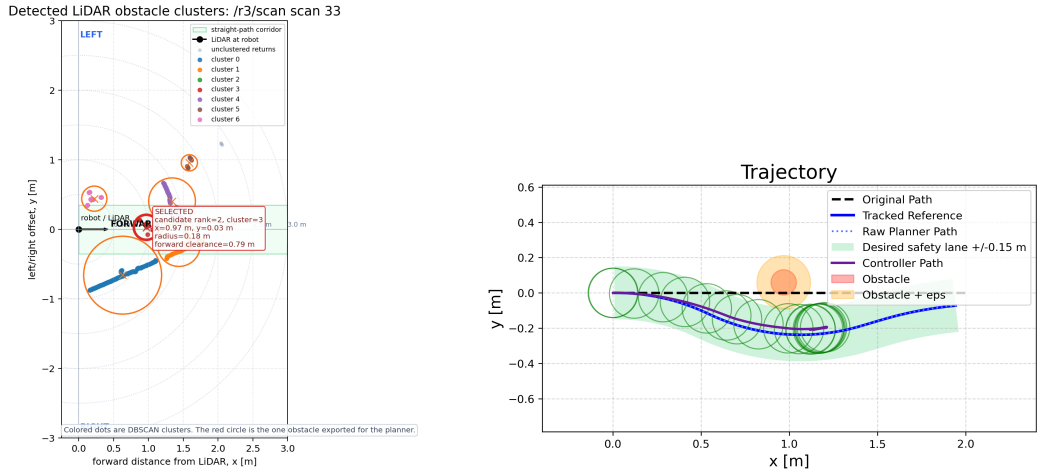


Figure 14: Lab Test (left) scan plot with object detection and (right) real time trajectory plot

# A Use of AI

I used ChatGPT-5.5 (OpenAI) as an assistive tool for drafting, research, and other queries. I reviewed and/or edited all AI-assisted outputs.

# References


[1] Clearpath Robotics, "TurtleBot4 Robot," TurtleBot4 User Manual. [Online]. Available: https://turtlebot.github.io/turtlebot4-user-manual/software/turtlebot4_robot.html

[2] SLAMTEC, "RPLIDAR-A1 Laser Range Scanner." [Online]. Available: https://www.slamtec.com/en/lidar/a1

[3] Clearpath Robotics, "Features," TurtleBot4 User Manual. [Online]. Available: https://turtlebot.github.io/turtlebot4-user-manual/overview/features.html

[4] Luxonis, "OAK-D Pro," Luxonis Documentation. [Online]. Available: https://docs.luxonis.com/hardware/products/OAK-D%20Pro

[5] Luxonis, "OAK-D-Lite Datasheet." [Online]. Available: https://www.mouser.com/catalog/specsheets/Luxonis_4-13-2022_OAK-D-Lite_AF%20Datasheet%204-12-22.pdf

[6] iRobot Education, "Hazards," Create 3 Docs. [Online]. Available: https://iroboteducation.github.io/create3_docs/api/hazards/

[7] iRobot Education, "Create 3 API Documentation." [Online]. Available: https://iroboteducation.github.io/create3_docs/api/

[8] Ultralytics, "Glossary." [Online]. Available: https://www.ultralytics.com/glossary/

[9] Ultralytics, "Ultralytics YOLO Docs." [Online]. Available: https://docs.ultralytics.com/

[10] N. Carion et al., "End-to-End Object Detection with Transformers," arXiv:2005.12872, 2020. [Online]. Available: https://arxiv.org/abs/2005.12872

[11] D. Beyer et al., "Deep Person Detection in Two-Dimensional Range Data," arXiv:1603.02636, 2016. [Online]. Available: https://arxiv.org/abs/1603.02636

[12] S. Chen et al., "Real-Time 2D LiDAR Object Detection Using Three-Frame RGB Scan Encoding," arXiv:2602.02167, 2026. [Online]. Available: https://arxiv.org/abs/2602.02167

[13] OpenCV, "Camera Calibration and 3D Reconstruction." [Online]. Available: https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html

[14] MathWorks, "What Is Camera Calibration?" [Online]. Available: https://www.mathworks.com/help/vision/ug/camera-calibration.html

[15] Luxonis, "StereoDepth," DepthAI Documentation. [Online]. Available: https://docs.luxonis.com/software-v3/depthai/depthai-components/nodes/stereo_depth

[16] D. D. D. Droeschel, T. Linder, and K. O. Arras, "DR-SPAAM: A Spatial-Attention and Auto-regressive Model for Person Detection in 2D Range Data," arXiv:2004.14079, 2020. [Online]. Available: https://arxiv.org/abs/2004.14079

[17] Wikipedia contributors, "Object detection," *Wikipedia, The Free Encyclopedia*. [Online]. Available: https://en.wikipedia.org/wiki/Object_detection. Accessed: July 8, 2026.

[18] Wikipedia contributors, "Kalman filter," *Wikipedia, The Free Encyclopedia*. [Online]. Available: https://en.wikipedia.org/wiki/Kalman_filter. Accessed: July 8, 2026.

[19] AquaPortail, "Filtre de Kalman : définition et explications," *AquaPortail*. [Online]. Available: https://www.aquaportail.com/dictionnaire/definition/5274/filtre-de-kalman. Accessed: July 8, 2026.

[20] OpenAI, "GPT-5.5." [Online]. Available: https://openai.com

[21] Y. Chen et al., "VoxelNextFusion: A Simple, Unified and Effective Voxel Fusion Framework for Multi-Modal 3D Object Detection," arXiv:2401.02702, 2024. [Online]. Available: https://arxiv.org/abs/2401.02702

[22] "Object Detection for TurtleBot4 Sensors: Framing the Problem for a Control-Centered Project," GPT-research PDF output, uploaded PDF, 2026.