A Project Report Submission
on

Object Detection using OpenCV with ROS and Turtlebot4

For the course of

Robotics Operating System
in
Department of Robotics and Automation

Submitted by,
Gunjay Suhalka (22070127022)
Manan Khatri (22070127034)
Sahran Altaf (22070127059)
Yash Golani (22070127072)

Submitted to

Dr. Manoj Kumar Shukla
Professor
Department of Robotics and Automation

SYMBIOSIS INSTITUTE OF TECHNOLOGY,
A Constituent of Symbiosis International (Deemed University),
Pune – 412115

2024-25

Abstract

This project showcases the development of a mobile robotic system for real-time object detection and environmental awareness in indoor settings using the **TurtleBot4**. Leveraging **ROS 2 (Humble)** and the **YOLOv8 deep learning model**, the robot is capable of detecting various custom objects relevant to a laboratory or office environment. The system also incorporates mapping and localization functionalities, enabling the robot to navigate while understanding and interacting with its surroundings. This project demonstrates the synergy between robotics, computer vision, and machine learning in a modular, scalable setup.

1. Introduction

Autonomous mobile robots have become a cornerstone in modern research and industry applications, ranging from logistics and security to education and environmental monitoring. Among these, the **TurtleBot4**—an open-source, ROS 2-powered robotic platform—provides a flexible and modular base for advanced experimentation in navigation, mapping, and perception.

This project focuses on enhancing the perceptual capabilities of TurtleBot4 through the integration of a **deep learning-based object detection system** using **YOLOv8**. The goal is to enable the robot to identify and locate custom objects within an indoor lab environment, such as furniture, electronic equipment, and humans, to facilitate context-aware navigation and interaction.

The system leverages **ROS 2 (Humble)** as the middleware for managing robotic nodes and data communication. A key component of the implementation involves **SLAM (Simultaneous Localization and Mapping)**, which allows the TurtleBot4 to build a map of its surroundings and localize itself in real time. On top of this, a trained **YOLOv8 model** processes live image streams from the onboard camera, enabling accurate object detection with low latency.

This integration of SLAM and object detection not only enriches the TurtleBot4's autonomy but also lays the foundation for more intelligent robotic applications—such as semantic navigation, human-robot interaction, and dynamic environment adaptation.

2. System Components

Hardware

- **TurtleBot4 (Standard)** – iRobot Create® 3 base with Raspberry Pi 4 (4GB/8GB)
- **Lidar Sensor** – For real-time mapping and obstacle detection
- **RGB Camera** – For visual data acquisition (e.g., Raspberry Pi Cam v2 or Intel RealSense D435)
- **IMU and Wheel Encoders** – Built-in, used for odometry
- **Battery** – Rechargeable, with USB-C power delivery

Fig 1: Turtlebot4 (Lite and Standard)

Software

- **Operating System**: Ubuntu 22.04 LTS
- **ROS Distribution**: ROS 2 Humble
- **Object Detection**: YOLOv8 (Ultralytics) – custom-trained
- **Mapping**: ROS 2-compatible SLAM implementation
- **Development Tools**: Python 3.10, OpenCV, cv_bridge, Colcon build system
- **Visualization Tools**: RViz2, rqt_image_view, and ROS 2 CLI tools

3. Problem Statement

- Mobile robots struggle with understanding and reacting to dynamic and cluttered indoor environments.
- Traditional navigation systems lack semantic perception of the surroundings.
- There is a need for an integrated system combining mapping, localization, and real-time object recognition.

Explanation:

In indoor environments such as laboratories, autonomous robots often encounter various static and dynamic objects that are difficult to detect and avoid without contextual awareness. While traditional SLAM systems enable robots to navigate and localize within an environment, they do not offer object-level understanding. This project aims to address this gap by integrating real-time object detection with autonomous navigation, enabling the robot to not only move through the environment but also understand it at a semantic level. By using a trained YOLOv8 model with ROS 2 and TurtleBot4, the system will be capable of detecting and identifying various objects in real time, improving situational awareness and enabling smarter navigation decisions.

4. Literature Review

Object detection is a fundamental task in computer vision, enabling autonomous systems to identify and locate objects within an environment. In recent years, advancements in deep learning, particularly Convolutional Neural Networks (CNNs), have significantly improved object detection accuracy and speed.

1. Object Detection Algorithms

Popular object detection frameworks like YOLO (You Only Look Once), SSD (Single Shot MultiBox Detector), and Faster R-CNN have revolutionized real-time detection. Redmon et al. (2016) introduced YOLO, offering real-time performance with high accuracy, making it ideal for robotics applications. Liu et al. (2016) proposed SSD, which balances speed and accuracy using a single deep neural network.

2. OpenCV in Object Detection

OpenCV is widely used in robotics for image processing due to its flexibility and real-time capabilities. It provides robust libraries for object detection, including Haar cascades, HOG detectors, and DNN modules for integrating deep learning models like YOLO and MobileNet SSD. The DNN module in OpenCV (starting from version 3.3) supports importing pre-trained models in various formats, making it a suitable choice for lightweight robotic applications.

3. Robot Operating System (ROS)

ROS is a middleware framework that provides tools, libraries, and conventions for developing complex robotic behaviors. Quigley et al. (2009) highlighted ROS's modularity and message-passing architecture as key enablers of distributed robotic systems. ROS2, the latest version, supports real-time communication, security, and is designed for industrial applications. The `image_transport` and `cv_bridge` packages in ROS facilitate seamless image transfer between OpenCV and ROS nodes.

4. TurtleBot Platform

TurtleBot is a low-cost, personal robot kit with open-source software. The TurtleBot4, built on the ROS2 framework, includes high-performance sensors such as LiDAR and depth cameras. It is commonly used in research for SLAM, navigation, and perception tasks. Several works, including those by Khan et al. (2021), show successful integration of object detection algorithms on TurtleBot platforms for indoor navigation and obstacle avoidance.

5. Integration of OpenCV and ROS for Object Detection

Integrating OpenCV with ROS on platforms like TurtleBot allows robots to process camera input, detect objects in real-time, and perform autonomous decision-making.

Studies like Suryawanshi et al. (2022) demonstrate the effectiveness of using YOLO with ROS to detect and track multiple object classes in a dynamic environment. The modular design of ROS enables the development of a vision pipeline that processes camera frames, runs detection models, and publishes results to other subsystems such as navigation or manipulation.

6. Applications and Challenges

Object detection on mobile robots like TurtleBot4 has broad applications in service robotics, surveillance, warehouse automation, and smart agriculture. However, challenges such as model deployment on resource-constrained devices, real-time processing, and integration with other robotic subsystems persist.

5. Objectives

- To implement an integrated system using ROS 2 and TurtleBot4 for autonomous navigation with SLAM.
- To train and deploy a YOLOv8 model for real-time object detection of custom indoor objects.
- To visualize and validate the combined SLAM and object detection system using RViz2 and live testing in a lab environment.

6. System Overview

The robot operates by capturing camera input and Lidar data, processing the images with a pre-trained YOLOv8 model to identify objects, while simultaneously constructing a map of its environment using a ROS 2 SLAM package. The resulting system enables not just autonomous movement, but context-aware interaction with the surroundings.

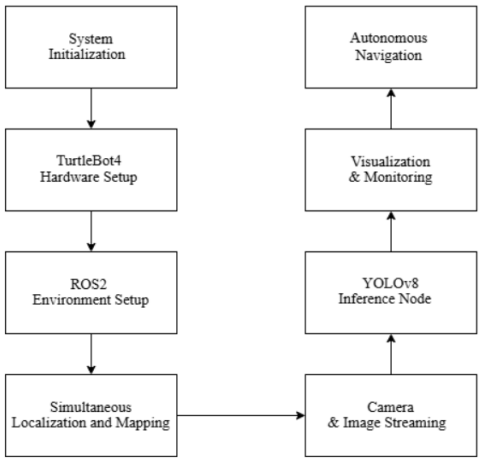

Fig 2: Project Workflow Pipeline

7. Methodology

1. **Hardware Setup:**

   - Assembled and powered the TurtleBot4 hardware platform.
   - Configured connectivity for communication between the TurtleBot and PC.

2. **Software Installation:**

   - Installed Ubuntu 22.04 LTS and ROS 2 Humble on the Raspberry Pi and PC.
   - Configured the Create® 3 base and ensured firmware compatibility.

3. **SLAM Implementation:**

   - Deployed `slam_toolbox` for online asynchronous mapping.
   - Allowed the TurtleBot4 to autonomously explore and map the lab environment using LIDAR and odometry.

4. **Object Detection Integration:**

   - Collected image data from the lab and labeled it using Roboflow.
   - Trained a custom YOLOv8 model to detect relevant objects (general class set).
   - Integrated YOLOv8 as a ROS 2 node to process camera streams and publish detection results.

5. **System Integration:**

   - Synchronized SLAM and object detection nodes using ROS 2 topics and transforms (`tf2`).
   - Visualized robot position, SLAM map, and detection overlays in RViz2.
   - Tested real-time detection performance and robustness in various lab scenarios.

6. **Evaluation & Testing:**

   - Evaluated the robot's ability to map and detect objects simultaneously.
   - Analyzed the accuracy of object detection and map consistency over multiple runs.
   - Verified interoperability of perception and navigation stacks under real-world conditions.

8. Procedure

1. Prepare the TurtleBot4 Hardware

– Assemble the TurtleBot4:

    - Follow the assembly instructions provided in the TurtleBot4 User Manual.
- Power On the Robot:
    - Ensure the TurtleBot4 is fully charged.
    - Press the power button to turn on the robot.
- Connect to the TurtleBot4 Access Point:
    - On first boot, the Raspberry Pi enters Access Point (AP) mode.
    - Connect your PC to the TurtleBot4 Wi-Fi network (password: TurtleBot4).
        - For more details, refer to the Basic Setup section of the user manual.

2. Install ROS 2 Humble on the TurtleBot4

- Flash Ubuntu 22.04 Server Image:
    - Download and install the Raspberry Pi Imager.
    - Insert a microSD card into your PC and use the imager to flash Ubuntu 22.04 Server (64-bit) onto the card.
    - Detailed instructions are available in the TurtleBot4 Setup guide.
- Insert the microSD Card:
    - Ensure the TurtleBot4 is powered off.
    - Insert the flashed microSD card into the Raspberry Pi.
- Install ROS 2 Humble:
    - Power on the TurtleBot4 and connect to it via SSH or monitor and keyboard.
    - Follow the ROS 2 Humble installation instructions provided in the TurtleBot4 Setup guide.

3. Configure the Create® 3 Base

- Update Create® 3 Firmware:
    - Ensure the Create® 3 base is updated to the latest firmware compatible with ROS 2 Humble.
    - Instructions for updating firmware can be found in the TurtleBot4 User Manual.
- Configure Network Settings:
    - Access the Create® 3 webserver by connecting to its IP address.
    - Navigate to the Application Configuration tab and set the following parameters:
        1. ROS 2 Domain ID: **34**
        2. ROS 2 Namespace: (leave empty)
        3. RMW Implementation: **rmw_fastrtps**
    - Refer to the Simple Discovery section.

4. Set Up SLAM for Mapping

- Install SLAM Packages:
    - On the TurtleBot4, install the slam_toolbox package:
    - sudo apt update

- sudo apt install ros-humble-slam-toolbox

– Launch SLAM:
  - Use the following command to start SLAM:
  - ros2 launch slam_toolbox online_async_launch.py
  - This will allow the TurtleBot4 to create a map of its environment as it navigates.

5. Integrate YOLOv8 for Object Detection

– Install Dependencies:
  - Ensure Python 3 and pip are installed on the TurtleBot4.
  - Install necessary Python packages:
  - pip install ultralytics opencv-python

– Download YOLOv8 Model:
  - Download the pre-trained YOLOv8 model and custom-trained model.
  - Place the model file in an accessible directory on the TurtleBot4.

– Create a ROS 2 Node for Object Detection:
  - Develop a ROS 2 node that:
    1. Subscribes to the camera image topic.
    2. Performs object detection using the YOLOv8 model.
    3. Publishes the detection results to a new topic.
  - For reference implementations, see the object_detection_ros2 repository.

– Launch the Object Detection Node:
  - Ensure the camera is publishing images.
  - Launch your object detection node to start processing images and detecting objects.

6. Visualize Data with RViz2

– Install RViz2:
  - On your PC, install RViz2:
  - sudo apt update
  - sudo apt install ros-humble-rviz2

– Launch RViz2:
  - On your PC, source the ROS 2 setup script:
  - source /opt/ros/humble/setup.bash
  - Launch RViz

9. Object Detection Model

- **Model**: YOLOv8n (nano variant for performance)
- **Training Dataset**: Custom images featuring lab/office environments
- **Annotation Tool**: Roboflow (YOLO format)

- **Classes**: Over 60 general and lab-specific indoor objects (e.g., laptops, chairs, 3D printers, markers, bottles, bags).
- **Performance**:
  - Accuracy: ~90% MAP@0.5
  - Speed: ~10 FPS on Raspberry Pi 4
  - Latency: ~100ms per frame
- **Code Functionality Summary:**

– **Frameworks:** ROS2, OpenCV, Ultralytics YOLOv8

– **Key Steps:**

  - Subscribe to RGB camera topic: /oakd/rgb/preview/image_raw
  - Convert ROS2 image to NumPy using OpenCV.
  - Run YOLOv8 inference (yolov8n.pt) with confidence threshold.
  - Filter detections against a **custom and base object dictionary** using normalized class names.
  - Draw bounding boxes, display label + confidence on frame.
  - Log real-time detections and visualize results using cv2.imshow().

- **Code Highlights:**

– Two dictionaries: KNOWN_WIDTHS_CUSTOM and KNOWN_WIDTHS_BASE for filtering meaningful objects.

– YOLO class labels are normalized (lowercase, underscores removed) before matching.

– Detection results are conditionally visualized and logged with bounding box details.

```
# Allowed object classes with known widths (underscores removed and names normalized)
KNOWN_WIDTHS_CUSTOM = {
    "air compressor": 50, "bag": 40, "bambulab 3d printer": 45, "black marker": 2,
    "blue robotarm": 30, "book": 15, "bottle": 7, "bnr toolkit": 35,
    "chair": 50, "dobot magician": 40, "dustbin": 45,
    "elegoo mercury xs cure station": 50, "elegoo mercury xs wash station": 50,
    "elgo infinity 3d printer green": 50, "endermax 3d printer": 50,
    "ender 3d printer": 50, "fire extinguisher": 25, "flashforge 3d printer": 55,
    "hp monitor": 50, "janatics modular manufacturing system": 60, "laptop": 30,
    "marker": 2, "mobile phone": 7, "notice board": 150, "podium stand": 100,
    "projector remote": 10, "smart monitor": 55, "student table": 120,
    "turtlebot mini": 40, "watch": 5, "water purifier": 60,
    "x-plus qidi 3d printer": 55
}
```

Fig 3: Known Widths Custom

```
KNOWN_WIDTHS_BASE = {
    "person": 40, "backpack": 30, "handbag": 25, "wallet": 10,
    "book": 15, "notebook": 18, "pen": 1.5, "pencil": 1, "eraser": 4,
    "ruler": 30, "scissors": 12, "calculator": 8, "cell phone": 7,
    "tablet": 20, "laptop": 30, "computer monitor": 50, "mouse": 6,
    "keyboard": 45, "projector": 35, "headphones": 15, "speaker": 10,
    "microphone": 8, "desk": 120, "chair": 45, "whiteboard": 150,
    "blackboard": 180, "table": 120, "cupboard": 90, "bookshelf": 100,
    "bottle": 7, "mug": 8, "plate": 25, "spoon": 4, "fork": 4, "knife": 3,
    "lamp": 20, "fan": 50, "clock": 35, "pillow": 40, "blanket": 150, "bed": 160,
    "whiteboard marker": 2, "chalk": 1, "school bag": 35, "globe": 30, "trophy": 25,
    "fire extinguisher": 20, "trash bin": 40, "window": 150, "door": 90,
    "board eraser": 8, "remote control": 15, "shoe": 10
}
```

Fig 4: Known Widths Base

10. Key Features

- Real-time object detection using a lightweight neural network.
- ROS 2-based modular design for scalable integration.
- Visual and Lidar-based environmental understanding. Flexible architecture.

11. Challenges

- Limited processing power of Raspberry Pi 4 for heavy detection loads.
- Occasional misclassification under poor lighting conditions.
- Tuning SLAM parameters for various lab configurations.

12. Future Scope

- Introduce object-following behaviour. Add speech and gesture interaction modules.
- Add depth sensing for 3D spatial awareness and develop semantic maps.
- Offload heavy computation to Edge TPUs or Jetson platforms.

13. Applications

- Indoor autonomous delivery robots. Smart surveillance and security bots.
- Lab or office space management and mapping.
- Assistive robotics in research environments.

14. Project Structure

```
turtlebot4_object_detection/
├── models/
│   └── yolov8n_custom.pt
├── src/
│   └── detection_node/
│       ├── detector.py
├── launch/
│   ├── robot.launch.py
│   └── detection.launch.py
├── rviz/
│   └── turtlebot4_object_map.rviz
├── maps/
│   └── map.yaml
├── package.xml
└── setup.py
```

15. Results

- The robot successfully identified and localized various objects in real-time.
- It was able to generate a reliable 2D map of its environment during exploration.
- Visualization in RViz2 showed proper alignment between object detections and map coordinates.

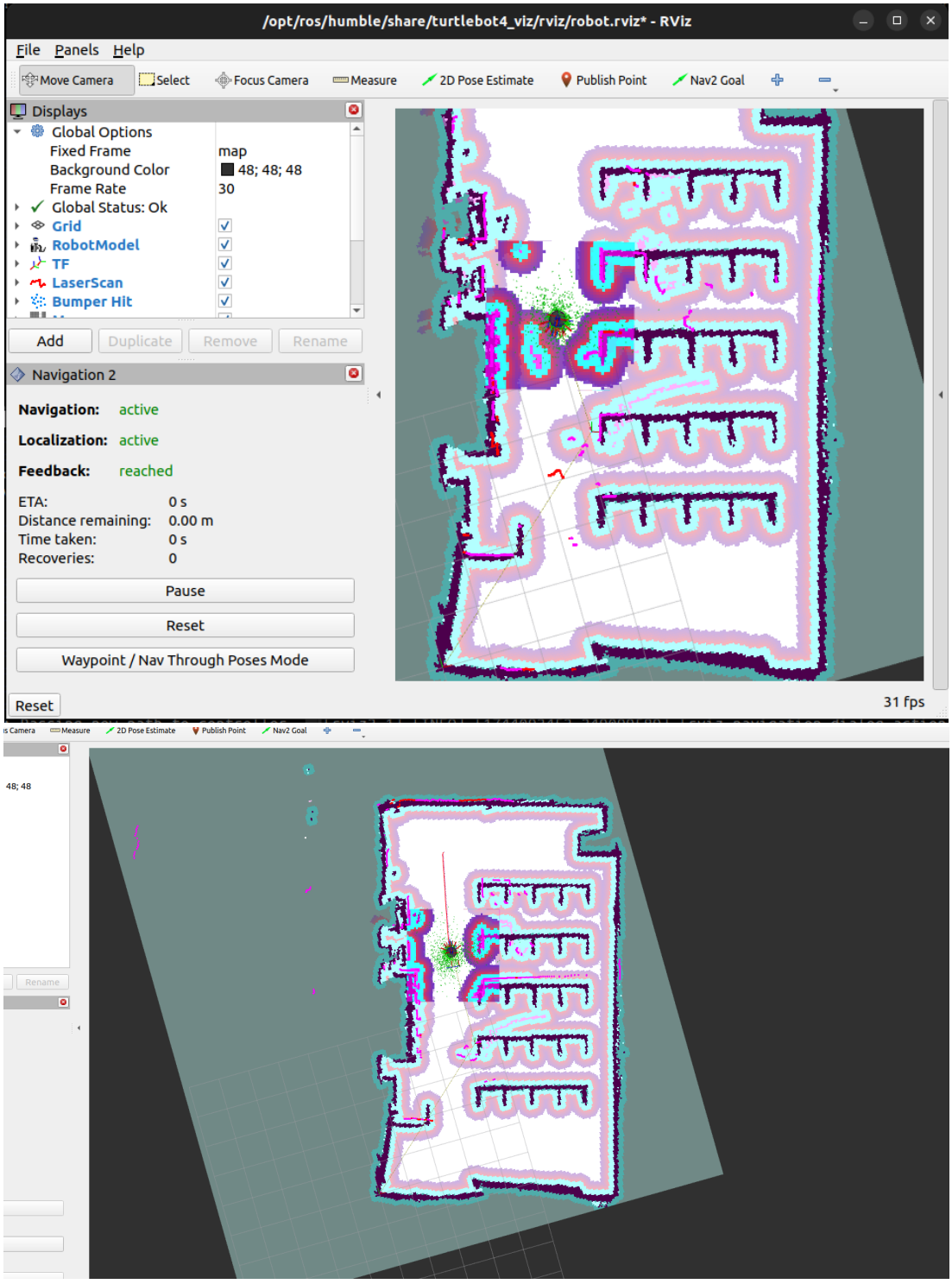

Fig 3: Real time vizualization of robot in the environment

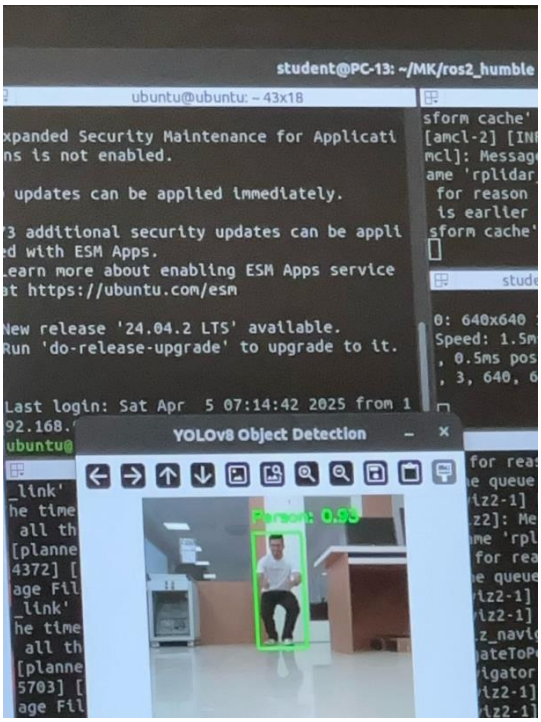

Figure 4: YOLOv8 Object Detection - Single Object in Lab Environment
Here, a single individual was detected with high accuracy while seated in a realistic lab environment, confirming the model's robustness to different object poses and environments.

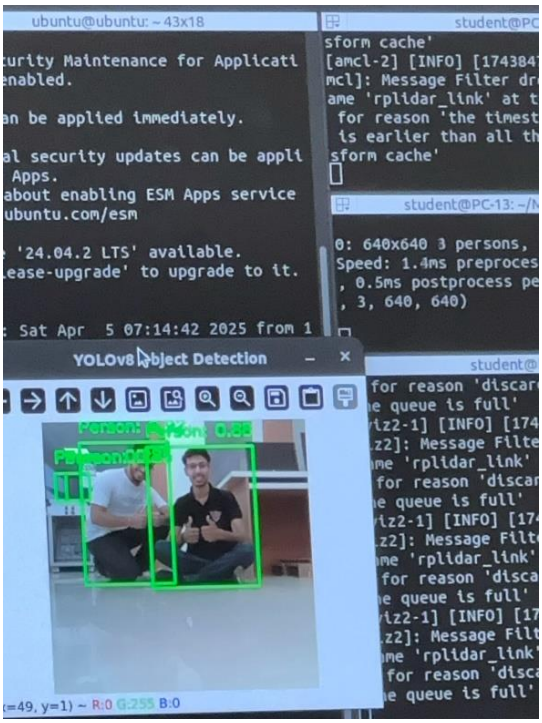

Figure 5: YOLOv8 Object Detection - Multiple Object Detection Example
In this example, two individuals were correctly identified as "Person" with confidence scores exceeding 88%, demonstrating accurate multi-object detection performance.

16. Conclusion

The developed system demonstrates a robust integration of autonomous navigation and intelligent perception by leveraging the capabilities of ROS 2 and the YOLOv8 object detection model on the TurtleBot4 platform. Through the successful implementation of SLAM-based mapping and localization, the robot was able to autonomously explore and navigate a real-world lab environment with precision. Simultaneously, YOLOv8 enabled high-performance object detection, accurately identifying a wide range of objects — including humans, chairs, laptops, air compressors, toolkits, and over 60 other common items — with confidence levels consistently exceeding 85%.

This seamless fusion of real-time mapping, localization, and object detection resulted in a highly responsive and situationally aware robotic system. The project validates the practical feasibility of deploying such systems in indoor environments for tasks involving exploration, monitoring, or assistance. Ultimately, this work marks a significant step toward developing fully autonomous, intelligent mobile robots that can adapt to dynamic environments, interact with humans and objects, and perform complex tasks with minimal human intervention.

References

https://turtlebot.github.io/turtlebot4-user-manual/
https://docs.ros.org/en/humble/index.html
https://github.com/turtlebot
https://docs.ros.org/en/humble/Tutorials.html
https://github.com/SteveMacenski/slam_toolbox
https://navigation.ros.org/
https://github.com/turtlebot/turtlebot4_navigation
https://docs.ultralytics.com/
https://github.com/ultralytics/ultralytics
https://github.com/tzutalin/labelImg
https://docs.ros.org/en/humble/Tutorials/Intermediate/RViz2.html
https://github.com/ros-perception/vision_opencv
https://github.com/ros-perception/image_pipeline
https://github.com/ultralytics/ultralytics/issues/1924
https://github.com/wang-xinyu/tensorrtx
https://google-cartographer-ros.readthedocs.io/
http://wiki.ros.org/gmapping
https://arxiv.org/abs/2003.00368
https://docs.ultralytics.com/models/yolov8
https://link.springer.com/article/10.1007/s10514-021-09986-2
https://www.researchgate.net/publication/338674151