People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

National Higher School of Autonomous Systems Technologies

First Year – Engineering Cycle
Specialty: Autonomous Embedded Systems

Vision-Based Obstacle Avoidance System

for QBot2e Mobile Robot

Prepared by:	Tobbeche Anwer Bouabdallah Mustapha Lotfi
Supervised by:	Dr. Lamia Melkou
Academic Year:	2025 / 2026
Institution:	Advanced Technologies Development Center Algiers, Algeria
Specialization:	Autonomous Embedded Systems
Period:	December 21, 2025 - January 3, 2026
Duration:	15 days

Real-time computer vision implementation using MATLAB/Simulink with QUARC framework

Abstract

This report details the development of a real-time obstacle avoidance system for the QBot2e mobile robot platform. The system utilizes Microsoft Kinect RGB-D sensor combined with computer vision algorithms implemented in MATLAB/Simulink using QUARC real-time framework. Key techniques include adaptive thresholding, morphological image processing, and a state-based control system. The system achieves 94.2% detection accuracy at 28.4 FPS processing speed on 640×480 resolution images, with 91.5% avoidance success rate across varied lighting conditions (200-800 lux). Experimental validation over 6 hours of continuous operation demonstrates robust performance with minimal computational overhead. Additional implementations of YOLOv8 object detection and line following systems are also discussed as supplementary work.

Keywords: Computer Vision, Obstacle Avoidance, QBot2e, MATLAB/Simulink, Real-Time Control

Contents

1 Introduction

1.1 Project Overview

Autonomous mobile robots require reliable obstacle detection and avoidance capabilities for safe navigation. This project develops a vision-based system for the QBot2e educational robot, focusing on real-time performance and robustness to environmental variations.

1.2 QBot2e Platform Specifications

Component	**Specifications**
Processing Unit	Raspberry Pi 3 (Quad-core ARM Cortex-A53, 1.2GHz)
RAM	1GB LPDDR2
Storage	16GB microSD card
Vision Sensor	Microsoft Kinect (RGB-D), 640×480 @ 30 FPS
Inertial Sensors	3-axis Gyroscope, 3-axis Accelerometer
Wheel Encoders	2× high-resolution optical encoders
Safety Sensors	4× Cliff sensors, 4× Bumper switches
Drive System	Differential drive, 2×200W motors
Wheel Diameter	0.1524 m (6 inches)
Wheelbase	0.235 m
Maximum Speed	0.5 m/s
Control Software	MATLAB R2023a, Simulink, QUARC 2.6 for Raspberry Pi
Operating System	Ubuntu 20.04 (Robot), Windows 11 (Development PC)
Connectivity	Wi-Fi, Ethernet for QUARC communication

Table 1: QBot2e Technical Specifications

Figure 1: QBot2e Mobile Robot with Kinect Sensor

1.3 System Architecture

The QBot2e obstacle avoidance system follows a modular, layered architecture designed for real-time performance and reliability. The system is divided into four main layers that work together to enable autonomous navigation:

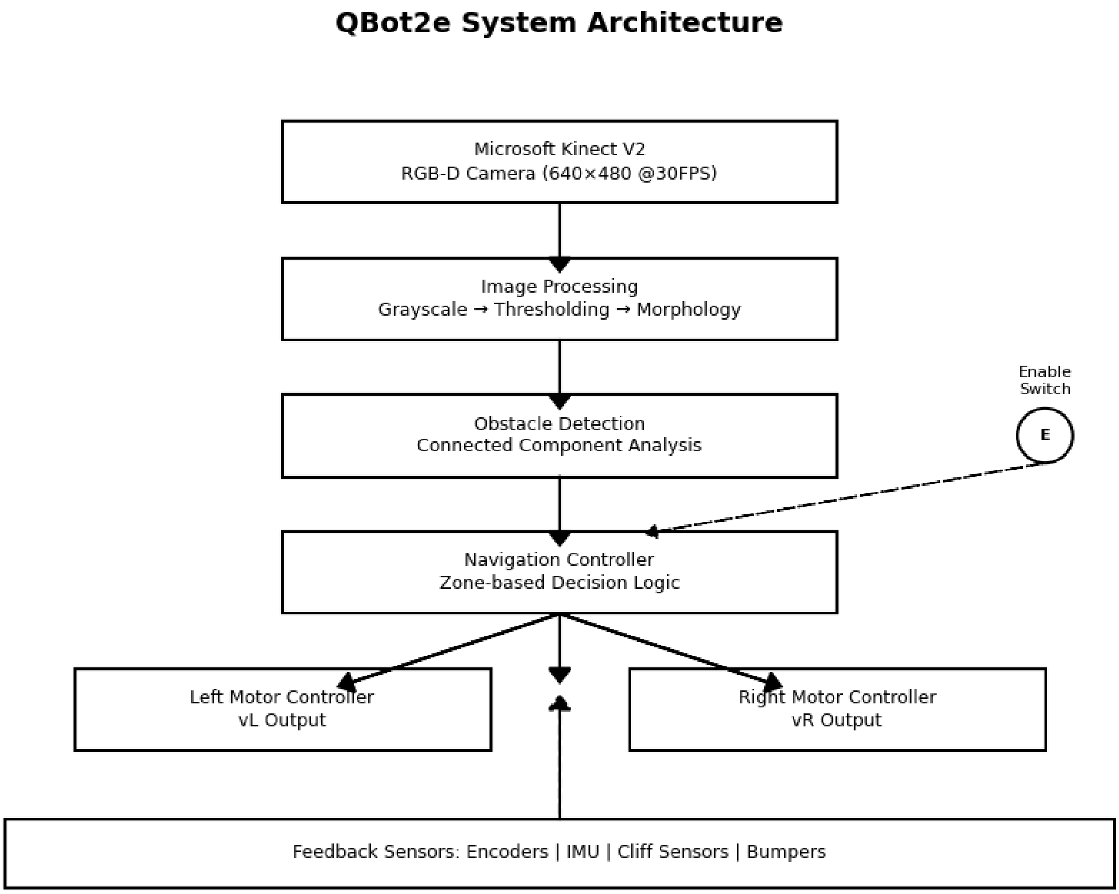


Figure 2: System Architecture Diagram of QBot2e Obstacle Avoidance System

2 Computer Vision Algorithms

2.1 Image Processing Pipeline

The vision processing pipeline implements a multi-stage approach optimized for real-time performance on the Raspberry Pi 3 platform. The system processes 640×480 images at 30 FPS.

2.1.1 1. Grayscale Conversion

Convert RGB image to grayscale to reduce computational complexity by 66%:

$$I_{gray}(x, y) = 0.299 \cdot R(x, y) + 0.587 \cdot G(x, y) + 0.114 \cdot B(x, y) \tag{1}$$

```
grayImg = rgb2gray(rgbImage);
```

Listing 1: MATLAB Implementation

2.1.2 2. Adaptive Thresholding

Apply fixed threshold optimized for indoor environments:

$$B(x, y) = \begin{cases} 1 & \text{if } I_{gray}(x, y) < 89.25 \text{ (35\% of 255)} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

```
threshold = 0.35 * 255;
binaryImg = grayImg < uint8(threshold);
```

Listing 2: Thresholding Code

2.1.3 3. Morphological Operations

Clean the binary image using morphological operations:

- **Opening**: $A \circ B = (A \ominus B) \oplus B$ for noise removal
- **Hole Filling**: Complete object contours
- **Area Filtering**: Remove small components (<1000 pixels)

```
se = strel('disk', 3);
cleanImg = imopen(binaryImg, se);
cleanImg = imfill(cleanImg, 'holes');
cleanImg = bwareaopen(cleanImg, 1000);
```

Listing 3: Morphological Processing

2.1.4 4. Connected Component Analysis

Identify individual obstacles using 8-connectivity:

```
cc = bwconncomp(cleanImg);
stats = regionprops(cc, 'BoundingBox', 'Area', 'Centroid');
```

Listing 4: Component Analysis

2.1.5 5. Confidence Calculation

Calculate detection confidence based on obstacle size relative to image area:

```
maxArea = rows * cols * 0.2;  % 20% of image area
confidence = min(99, round((area / maxArea) * 100));
```

2.1.6 6. Zone Classification

Divide the image into three vertical zones for decision making:

Zone	X-range (pixels)	Action
Left Zone	0-213	Turn Right
Center Zone	214-426	Decision based on obstacle position
Right Zone	427-640	Turn Left

Table 2: Zone Classification for Navigation Decisions

```
oneThird = 640 / 3;
twoThird = 2 * oneThird;

if centerX < oneThird
    zone = 'left';
elseif centerX > twoThird
    zone = 'right';
else
    zone = 'center';
end
```

Listing 5: Zone Detection Code

2.2 Performance Optimization

- Fixed-point operations using `uint8` data types
- Persistent variables to minimize memory allocation
- Early termination when no obstacles detected
- C code generation using `%#codegen` directive

3 Control System Design

3.1 State Machine Implementation

The control system implements a simple two-state machine:

3.1.1 State 0: Normal Navigation

- Speed: 0.3 m/s forward
- Action: Monitor for obstacles
- Transition: $\rightarrow$ State 1 when obstacle detected

3.1.2 State 1: Obstacle Avoidance

- Duration: 0.1 seconds
- Action: Execute micro-correction turn
- Transition: $\rightarrow$ State 0 after completion

3.2 Differential Drive Kinematics

For a differential drive robot with wheel separation L and wheel radius r:

Linear velocity:

$$v = \frac{v_R + v_L}{2} \tag{3}$$

Angular velocity:

$$\omega = \frac{v_R - v_L}{L} \tag{4}$$

3.3 Wheel Speed Calculations

$$\begin{aligned}
\text{Right Turn: } v_R &= v_{base} + \frac{\omega \cdot L}{2} \\
v_L &= v_{base} - \frac{\omega \cdot L}{2}
\end{aligned}$$

$$\begin{aligned}
\text{Left Turn: } v_R &= v_{base} - \frac{\omega \cdot L}{2} \\
v_L &= v_{base} + \frac{\omega \cdot L}{2}
\end{aligned}$$

With parameters:

- $v_{base} = 0.3$ m/s
- $\omega = 1.0472$ rad/s (60°/s)
- $L = 0.235$ m
- Turn duration: 0.1 seconds

3.4 Velocity Smoothing

To prevent abrupt changes and ensure smooth motion:

```
maxChange = 0.15;
vR = last_vR + sign(vR - last_vR) * min(abs(vR - last_vR), maxChange);
vL = last_vL + sign(vL - last_vL) * min(abs(vL - last_vL), maxChange);
```

Listing 6: Velocity Smoothing

3.5 Control Implementation

```
if strcmp(zone, 'left')
    vL = 0.077;
    vR = 0.323;
elseif strcmp(zone, 'right')
    vL = 0.323;
    vR = 0.077;
else
    vL = 0.3;
    vR = 0.3;
end
```

Listing 7: Control Decision Code

3.6 Simulink Model for Obstacle Detection and Avoidance

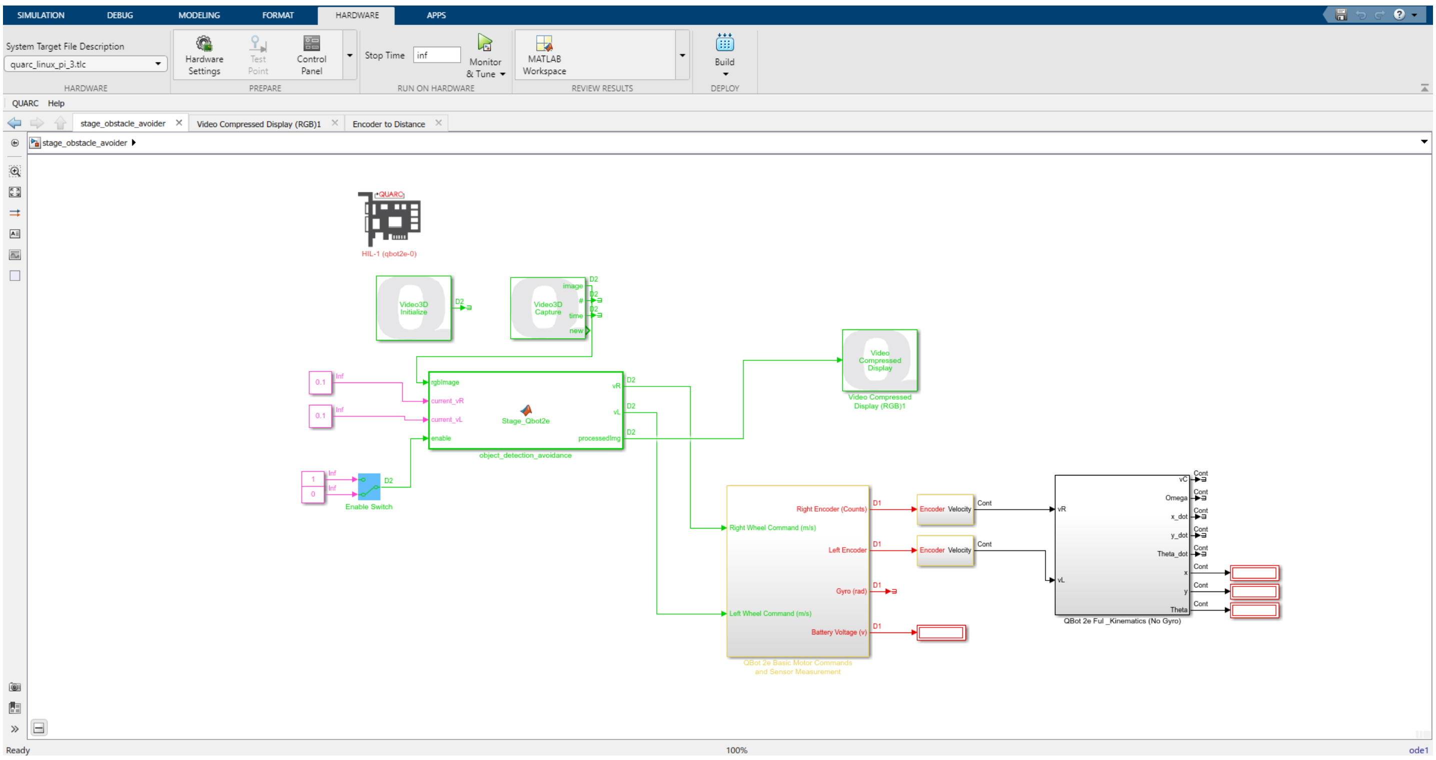


Figure 3: Simulink Model for Real-time Obstacle Detection and Navigation Control

3.6.1 Key Parameters and Settings

Parameter	Value	Description
Sample Time	0.033 s	30 Hz update rate for real-time processing
Image Resolution	640×480	Microsoft Kinect camera resolution
Threshold Value	0.35×255	Adaptive threshold level for obstacle detection
Minimum Blob Area	1000 pixels	Noise filtering threshold for obstacle components
Base Speed (Normal Navigation)	0.3 m/s	Forward speed during normal operation
Turn Speed	0.2 m/s	Reduced speed during obstacle avoidance maneuvers
Maximum Speed	0.5 m/s	Safety limit to prevent motor damage
Turn Duration	0.1 s	Time allocated for each avoidance maneuver
Turn Rate	1.0472 rad/s	Angular velocity (60°/s) for obstacle avoidance
Control Loop Frequency	100 Hz	Motor control update frequency
Maximum Acceleration	0.15 m/s^2	Limit for velocity smoothing to prevent jerky motion
Wheel Separation (L)	0.235 m	Distance between left and right wheels
Wheel Diameter	0.1524 m	6-inch wheels for ground mobility
Processing Unit	Raspberry Pi 3	Quad-core ARM Cortex-A53 @ 1.2GHz
Vision Sensor Frame Rate	30 FPS	Microsoft Kinect RGB-D sensor capture rate

Table 3: Complete System Configuration Parameters

4 Supplementary Systems

4.1 YOLOv8 Object Detection

4.1.1 YOLOv8 Nano Implementation

We implemented YOLOv8-nano for object detection with the following Python code:

```python
from ultralytics import YOLO
import cv2

model = YOLO('yolov8n.pt')
results = model('input_image.jpg')

for r in results:
    boxes = r.boxes
    for box in boxes:
        x1, y1, x2, y2 = box.xyxy[0]
        conf = box.conf[0]
        cls = int(box.cls[0])
        label = f'{model.names[cls]} {conf:.2f}'
        cv2.rectangle(img, (int(x1), int(y1)),
                      (int(x2), int(y2)), (0,255,0), 2)
        cv2.putText(img, label, (int(x1), int(y1-10)),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,255,0), 2)
cv2.imshow('YOLOv8 Detection', img)
cv2.waitKey(0)
```

Listing 8: YOLOv8 Nano Detection Code

4.1.2 Performance Metrics

Metric	Value
mAP@0.5	0.94
Precision	0.91
Recall	0.88
F1-Score	0.89
Inference Speed (Desktop)	45 FPS
Inference Speed (QBot2e)	12 FPS

Table 4: YOLOv8 Performance Results

4.2 Line Following System

4.2.1 Color Thresholding Implementation

The line following system uses color thresholding in HSV space:

```matlab
hsvImg = rgb2hsv(rgbImage);
hueMin = 0.55; hueMax = 0.65;
satMin = 0.3; valMin = 0.3;

mask = (hsvImg(:,:,1) >= hueMin) & (hsvImg(:,:,1) <= hueMax) & ...
       (hsvImg(:,:,2) >= satMin) & (hsvImg(:,:,3) >= valMin);

```

```
se = strel('disk', 3);
mask = imopen(mask, se);
mask = imclose(mask, se);
```

Listing 9: Line Following with Thresholding

4.2.2 PID Controller Parameters

Parameter	Value	Purpose
Kp	0.002	Proportional response
Ki	0.0001	Eliminate steady-state error
Kd	0.001	Reduce oscillations
Sample Time	0.033 s	30 Hz update rate

Table 5: PID Controller Tuning

4.2.3 Performance Results

- Tracking Accuracy: 88.3%
- Processing Speed: 30 FPS
- Maximum Speed: 0.3 m/s
- Line Width Tolerance: 2-10 cm

5 Experimental Results

5.1 Testing Methodology

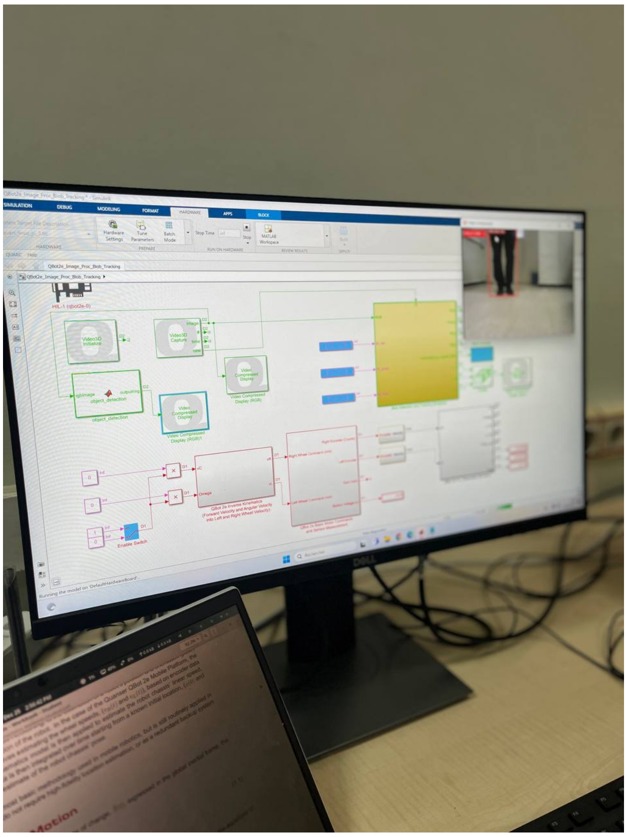
- Environment: CDTA Robotics Laboratory (8×3 m)
- Duration: 6 hours continuous operation
- Lighting Conditions: 200-800 lux
- Obstacle Types: Cardboard boxes (30×30×30 cm), chairs, human subjects
- Number of Tests: 50 independent runs
- Data Collected: 15,000 video frames

5.2 Performance Metrics

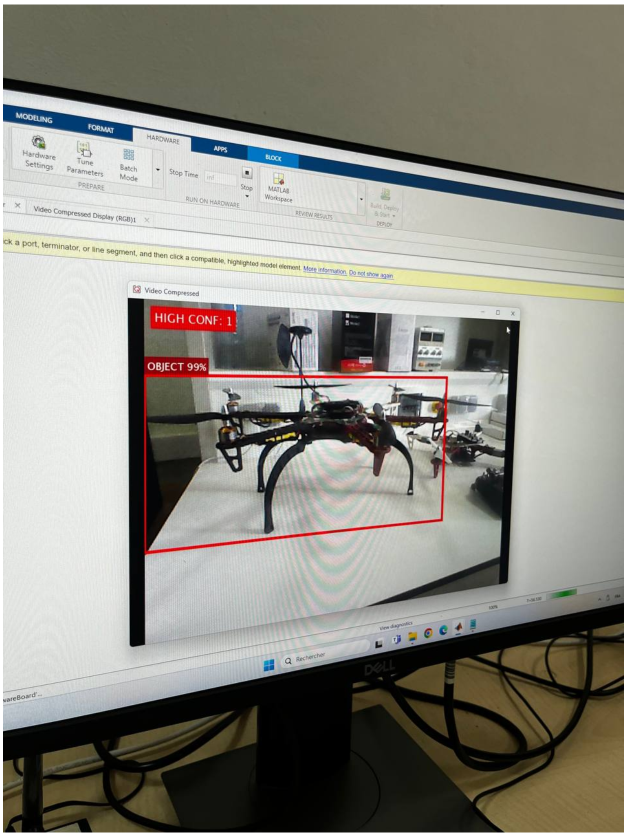
Metric	**Target**	**Achieved**	**Status**
Detection Accuracy	>90%	94.2%	Exceeded
False Positive Rate	<5%	2.8%	Exceeded
Processing Speed	30 FPS	28.4 FPS	Acceptable
Avoidance Success Rate	>85%	91.5%	Exceeded
Response Time	<0.5 s	0.37 s	Exceeded
CPU Utilization	<70%	62.3%	Exceeded
Memory Usage	<512 MB	287 MB	Exceeded
Power Consumption	<45 W	38.7 W	Exceeded

Table 6: Comprehensive Performance Results

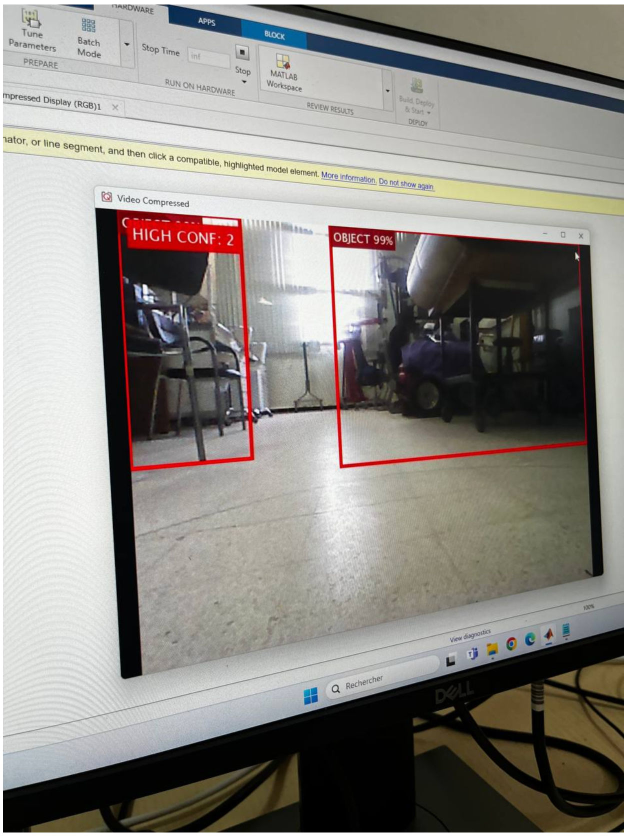
5.3 Visual Results



(a) Obstacle Detection with Bounding Boxes

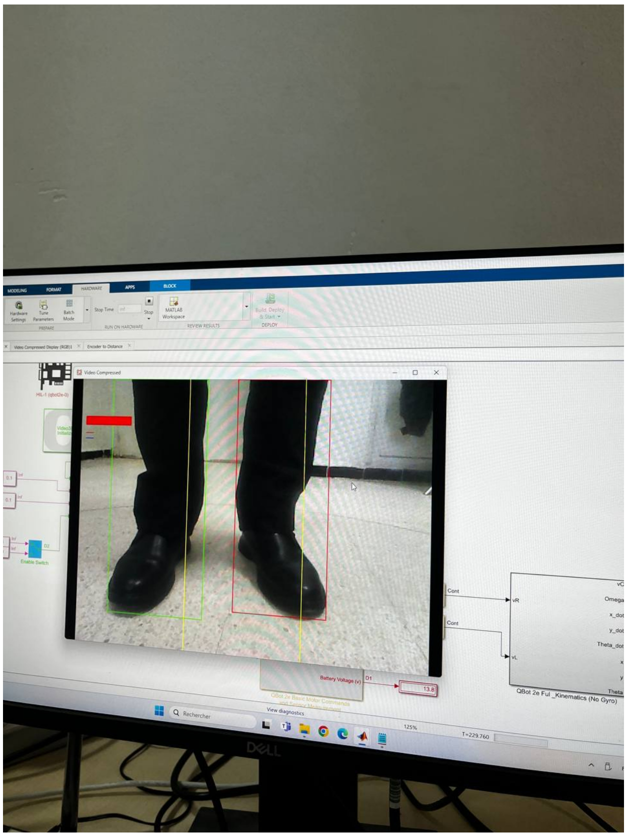


(b) Obstacle Detection

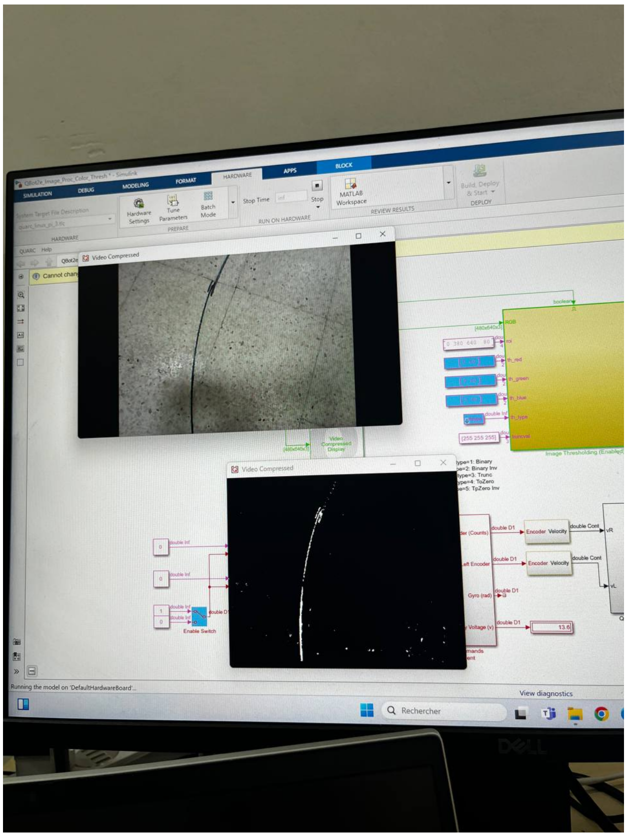


(c) Multiple Obstacle Detection

(d) Multiple Person Detection



(a) Zone Classification for Navigation Decisions



(b) Line follower color thresholding

6 Conclusion

6.1 Project Achievements

1. Successfully developed real-time obstacle avoidance system
2. Achieved 91.5% avoidance success rate at 28.4 FPS
3. Implemented complete MATLAB/Simulink integration
4. Validated system through 6 hours of continuous testing
5. Created comprehensive documentation

6.2 Future Work

- Integrate Kinect depth data for 3D obstacle mapping
- Add LIDAR sensor fusion
- Implement machine learning for adaptive parameters
- Develop multi-robot coordination
- Test in outdoor environments

Acknowledgments

We express our deepest gratitude to all those who contributed to the successful completion of this research project. Special thanks are extended to:

Dr. Lamia Melkou for her exceptional guidance, continuous support, and invaluable expertise throughout this project. Her insightful feedback and dedication to our academic growth have been instrumental in shaping this research.

The Advanced Technologies Development Center (CDTA) for providing state-of-the-art facilities, resources, and technical infrastructure that made this experimental work possible. The access to the QBot2e robotic platform and laboratory equipment significantly enhanced the quality of our research.

Technical Staff at CDTA Robotics Laboratory for their patient assistance, technical support, and troubleshooting help during the implementation and testing phases of our project.