COLLEGE CODE : 3105

COLLEGE NAME : Dhanalakshmi Srinivasan College Of Engineering And Technology

DEPARTMENT : B.Tech(Artificial Intelligence and Data Science)

STUDENT NM-ID : 1727bd590af0a15aca948f6cc481212d

ROLL NO : 310523243081

DATE : 15/05/2025

TECHNOLOGY-PROJECT NAME : Autonomous Vehicles And Robotics

SUBMITTED BY,

S. SUBASRI

V. SUMITHA

V. SUBIKSHA

C. MOTHISH

S. PRAVEEN

Phase 5 : Project Demonstration & Documentation

Autonomous Vehicles and Robotics

Abstract

The "Autonomous Vehicles and Robotics" project focuses on the development and integration of intelligent robotic systems capable of autonomous navigation, obstacle avoidance, and real-time decision-making. Leveraging cutting-edge technologies such as computer vision, machine learning, and sensor fusion, this system is designed to operate safely in dynamic environments. The final phase involves a functional demonstration, technical documentation, performance evaluation, and full system handover. This document presents a detailed overview of the project, including system architecture, source code snapshots, simulation outputs, and feedback-based refinements for real-world deployment.

1. Project Demonstration

Overview:
The demonstration phase highlights the practical implementation of autonomous navigation and robotic control in varied test environments. The system's real-time response, perception accuracy, and decision-making capabilities will be showcased.

Demonstration Details:

- **Autonomous Navigation:** The robot/vehicle will demonstrate path planning, lane tracking, and environment mapping using LiDAR and camera sensors.
- **Obstacle Detection and Avoidance:** Live tests will showcase how the system identifies and safely navigates around static and dynamic obstacles.
- **Sensor Fusion:** Integration of data from multiple sensors (GPS, IMU, ultrasonic, and cameras) will be shown in action.
- **AI-Controlled Maneuvering:** Demonstrating the vehicle's ability to make intelligent driving decisions (turns, stops, speed regulation) based on the environment.
- **Simulation and Real-World Sync:** Simulation results and real-time output comparisons will be shown to validate algorithm performance.

Outcome:
The system will effectively demonstrate autonomous decision-making, robust

navigation, and responsive adaptability in test conditions, indicating readiness for further scalability and testing.

2. Project Documentation

Overview:
This section presents a complete technical blueprint of the project, including design decisions, architecture, AI model development, and system integration.

Documentation Sections:

- **System Architecture:** Diagrams illustrating sensor layout, data flow, and control architecture.
- **AI Algorithms:** Documentation of pathfinding, image processing, and machine learning models used for decision-making.
- **Codebase Overview:** Snapshots and explanations of key modules (vision, control, communication).
- **Simulation Tools:** Tools used (e.g., ROS, Gazebo, OpenCV) with setup instructions.
- **User Manual:** Instructions to operate the robotic platform in both simulation and real-world environments.
- **Admin Guide:** Maintenance, troubleshooting steps, and performance tuning options.
- **Testing Logs:** Logs from unit testing, field testing, and performance validation.

Outcome:
The documentation will ensure that all aspects of the system are comprehensible and reproducible for future enhancements or educational use.

3. Feedback and Final Adjustments

Overview:
Stakeholder and tester feedback is used to make refinements to the navigation logic, user interaction, and safety features.

Steps:

- **Feedback Collection:** From faculty, peers, and field testers.

- **Identifying Bottlenecks:** Addressing issues in sensor calibration, obstacle misclassification, or inefficient path selection.
- **System Tuning:** Modifying PID controllers, adjusting model thresholds, or retraining recognition algorithms.
- **Final Testing:** Rerunning simulations and real-world trials post-adjustments.

Outcome:
A robust and responsive robotic system refined to align with user expectations and safety standards.

4. Final Project Report Submission

Overview:
Summarizes the entire journey, from initial design to final testing.

Report Sections:

- **Executive Summary:** High-level summary of the system's capabilities and goals achieved.
- **Phase Recap:** Development through each phase, with highlights on challenges and improvements.
- **Innovation Highlights:** AI-based vision system, real-time sensor fusion, and intelligent control.
- **Challenges & Fixes:** Hardware limitations, sensor noise, or AI misclassifications and how they were mitigated.
- **Current State & Future Potential:** Real-world deployment readiness and next-level ideas like swarm robotics or edge AI integration.

Outcome:
A detailed record will be ready for academic evaluation or project handover to future teams.

5. Project Handover and Future Works

Overview:
Lays the groundwork for upcoming iterations or research extensions.

Handover Details:

- **Repository Access:** Source code, simulation setups, and documentation archive.

- **Next Steps:** Recommendations like autonomous convoy formation, terrain-specific adaptations, and integration with traffic systems.
- **Future Enhancements:** Improve vision with deep learning, multilingual command support, and use in agriculture or rescue.

Outcome:
The system will be ready for academic or industrial extension with detailed notes and continuity support.

PROGRAM :

```
import cv2

import numpy as np

import time

class PIDController:

def __init__(self, kp, ki, kd):

self.kp = kp

self.ki = ki

self.kd = kd

self.last_error = 0

self.integral = 0

def compute(self, error, dt):

self.integral += error * dt

derivative = (error - self.last_error) / dt if dt > 0 else 0

output = self.kp * error + self.ki * self.integral + self.kd * derivative

self.last_error = error

return output

def get_simulated_distance():

return np.random.randint(10, 100)


def process_frame(frame):

hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
```

```
mask = cv2.inRange(hsv, (0, 0, 0), (180, 255, 50))
return mask
def autonomous_drive():
cap = cv2.VideoCapture(0)
pid = PIDController(0.5, 0.01, 0.1)
 last_time = time.time()
while True:
ret, frame = cap.read()
if not ret:
break
mask = process_frame(frame)
M = cv2.moments(mask)
height, width = frame.shape[:2]
distance = get_simulated_distance()
if distance < 20:
print("Obstacle too close! Stopping vehicle.")
continue
if M["m00"] > 0:
cx = int(M["m10"] / M["m00"])
 error = cx - width // 2
dt = time.time() - last_time
correction = pid.compute(error, dt)
last_time = time.time()
print(f"Line detected. Correction: {correction:.2f}")
else:
 print("Line lost. Searching...")
cv2.imshow("Frame", frame)
cv2.imshow("Line Mask", mask)
if cv2.waitKey(1) & 0xFF == ord('q'):
```

```
break
cap.release()
cv2.destroyAllWindows()
if __name__ == "__main__":
autonomous_drive()
```