

Register number:113323106005

Name: Arjun Rathinam R

Naan mudhalvan id: aut113323eca03

Department: II YEAR-ECE-A

AUTONOMOUS VEHICLES AND ROBOTICS SYSTEM

Objective:

This phase focuses on enhancing system performance in autonomous navigation, robotics coordination, sensor fusion, and real-time AI decision-making. Key goals include improving object detection accuracy, optimizing route planning, refining robotic actuation, and ensuring system reliability in dynamic environments.

1. Autonomous Navigation and AI Model Performance

Overview:

The AI model used for autonomous decision-making will be enhanced using advanced training data sets from varied environments (urban, rural, obstacle-heavy zones).

Performance Improvements:

- ✓ Enhanced Detection: Retrained convolutional neural networks (CNNs) for higher precision in object detection.
- ✓ Path Optimization: Improved Dijkstra and A* algorithms for smoother and more efficient pathfinding.

Outcome:

Improved vehicle response to obstacles, traffic signs, and unpredictable elements. Enhanced safety, route efficiency, and real-time adaptability.

2. Robotics Actuation and Control System Optimization

Overview:

The robotic arm/system in the vehicle will be optimized for improved responsiveness, load handling, and adaptive movement in real-world tasks.

Key Enhancements:

- ✓ PID Tuning: Recalibrated control loops for smoother motion and accurate feedback.
- ✓ Real-time Kinematics: Enhanced inverse kinematics calculations for precision in object manipulation.

Outcome:

Smoother, faster, and more accurate robot arm movement in dynamic tasks like loading/unloading or object sorting.

3. Sensor Integration and Fusion

Overview:

Sensor data from LiDAR, cameras, GPS, and IMUs will be synchronized and fused for better situational awareness.

Key Enhancements:

- ✓ Kalman Filtering: Applied for improved positional tracking.
- ✓ Sensor Fusion Algorithms: Improved data accuracy using multi-sensor fusion techniques.

Outcome:

More accurate environment mapping and vehicle localization even in GPS-denied areas.

4. Data Security and System Reliability

Overview:

Ensures that system data (route, control signals, sensor feedback) is securely transmitted and stored. Also enhances system reliability against failures.

Key Enhancements:

- ✓ Secure Communication: End-to-end encryption for vehicle-to-server and inter-robot communication.
- ✓ Fail-Safe Mechanisms: Emergency protocols and watchdog systems implemented.

Outcome:

System remains functional and secure under operational stress, maintaining integrity and preventing unauthorized access.

5. Performance Testing and Metrics Collection

Overview:

Testing under controlled and semi-real conditions to evaluate responsiveness, obstacle avoidance, fuel/power efficiency, and coordination in fleets.

Implementation:

- ✓ Simulated Environments: Tested with ROS and Gazebo simulation for scalability and unpredictability.
- ✓ Field Testing: Live testing in mixed-terrain areas.
- ✓ Metrics Logged: Response latency, object detection accuracy, path deviation, and fuel consumption.

Outcome:

The system exhibits robust performance, with reduced errors and adaptive learning for continuous improvement.

Key Challenges in Phase 4

1. Dynamic Obstacle Handling

- ✓ Challenge: Managing unpredictable human or vehicular obstacles.
- ✓ Solution: Use of real-time object tracking and predictive movement modeling.

2. Inter-System Coordination

- ✓ Challenge: Synchronizing multiple robotic units or vehicles.
- ✓ Solution: Implementation of decentralized communication protocols.

3. Power Management

- ✓ Challenge: High energy consumption due to sensors and processing units.
- ✓ Solution: Optimization of active modules and use of low-power processing during idle states.

Outcomes of Phase 4

- ✓ Improved Navigation & Safety: Accurate, efficient route planning and collision avoidance.
- ✓ Advanced Robotics Functionality: Real-time interaction with dynamic environments.
- ✓ Reliable Sensor Data Fusion: Higher situational awareness and vehicle stability.
- ✓ Secure & Scalable Framework: Ready for real-world deployment with secure, fault-tolerant architecture.

Next Steps for Finalization

In the final phase, the system will be evaluated in extended real-world scenarios, user feedback will be collected, and the AI model will be fine-tuned. Deployment-ready documentation and compliance testing will also be completed.

Source code:

```
import cv2

def get_frame():
    cap = cv2.VideoCapture(0)
    while True:
        ret, frame = cap.read()
        if not ret:
            continue
        yield frame
```

```
import cv2
import numpy as np

def detect_line(frame):
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    lower = np.array([20, 100, 100])
    upper = np.array([30, 255, 255])
    mask = cv2.inRange(hsv, lower, upper)

    contours, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    if contours:
        largest = max(contours, key=cv2.contourArea)
        x, y, w, h = cv2.boundingRect(largest)
        center = x + w // 2
        return center, mask
    return None, mask
```

```
import random

def get_distance():
    return random.randint(5, 100) # distance in cm
```

```

def control_logic(line_pos, distance):
    if distance < 20:
        return "STOP - Obstacle Ahead"

    if line_pos is None:
        return "Searching for line..."

    if line_pos < 200:
        return "Turn Left"
    elif line_pos > 440:
        return "Turn Right"
    else:
        return "Move Forward"

```

```

import cv2
from vision import detect_line
from sensors import get_distance
from controller import control_logic

cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        continue

    line_pos, mask = detect_line(frame)
    distance = get_distance()
    decision = control_logic(line_pos, distance)

    print(f"Distance: {distance} cm | Line: {line_pos} | Action: {decision}")
    cv2.putText(frame, decision, (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2)

    cv2.imshow("Camera", frame)
    cv2.imshow("Line Detection", mask)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()

```



Output:

Distance: 56 cm | Line: 320 | Action: Move Forward
Distance: 43 cm | Line: 510 | Action: Turn Right
Distance: 12 cm | Line: 310 | Action: STOP - Obstacle Ahead
Distance: 67 cm | Line: 180 | Action: Turn Left
Distance: 85 cm | Line: None | Action: Searching for line...

Metric tables:



Autonomous Vehicles and Robotics — Metrics Table

Category	Metric	Unit / Description	Goal / Ideal Value
 Perception	Object Detection Accuracy	% of correctly detected objects	> 95%
	Lane Detection Accuracy	% accuracy in identifying road/lane markings	> 90%
	Sensor Fusion Latency	Time to combine data from multiple sensors	< 100 ms
	Frame Processing Rate	Frames per second (FPS) processed by the system	> 30 FPS
 Localization	Positioning Error	Average deviation from true position (e.g., GPS error)	< 10 cm
	SLAM Accuracy (Robotics)	Error in mapping & location estimation	< 5%

🔴 Decision Making	Path Planning Success Rate	% of successful navigation attempts	> 98%
	Obstacle Avoidance Rate	% of correctly avoided obstacles	> 95%
	Reaction Time	Time taken to react to a new obstacle	< 300 ms
⚙️ Control	Actuator Response Time	Delay between command and motor/sensor action	< 50 ms
	Steering Accuracy	Deviation from planned path in degrees	< 2°
🟢 Efficiency	Energy Consumption	Watts per kilometer / hour (for electric or mobile robots)	As low as possible (e.g., < 500W)
	CPU/GPU Utilization	System resource load during operation	< 70%