Autonomous Navigation System for Rover

Akhillesh varathan CS
cb.ai.u4aim24102
School of Artificial Intelligence
Amrita Vishwa Vidyapeetham
Coimbatore, Tamil Nadu
cb.ai.u4aim24102@cb.students.amrita.edu

Jeffrin Merino J
cb.ai.u4aim24118
School of Artificial Intelligence
Amrita Vishwa Vidyapeetham
Coimbatore, Tamil Nadu
cb.ai.u4aim24118@cb.students.amrita.edu

Deepak Skandh k
cb.ai.u4aim24119
School of Artificial Intelligence
Amrita Vishwa Vidyapeetham
Coimbatore, Tamil Nadu
cb.ai.u4aim24119@cb.students.amrita.edu

Kavin M
cb.ai.u4aim24121
School of Artificial Intelligence
Amrita Vishwa Vidyapeetham
Coimbatore, Tamil Nadu
cb.ai.u4aim24121@cb.students.amrita.edu

Abstract: **The main goal of this project to design and implementation of an autonomous rover system integrating multiple embedded technology for real-time environmental interaction. This system combines an ESP32-CAM module for image capture, a Raspberry Pi for executing deep learning-based object detection and A*path planning, ultrasonic sensors with servo-based scanning for dynamic obstacle detection, and a moisture sensor for terrain analysis. The rover adjusts its movement strategy based on the sand's moisture level, optimize the speed control in various conditions. This project outlines the full system architecture, communication protocol, sensor integration, motor control, and decision logic, low-cost platform for autonomous navigation and real-time environmental awareness.**

I. INTRODUCTION

Autonomous navigation systems encompass path planning, real-time object detection, and terrain-aware motion control. This architecture is based upon the functioning of the ESP32-cam module, responsible for capturing the images and transmitting them to the Raspberry Pi [4]. The Raspberry Pi processes the images using YOLO (a deep learning model) for object detection, and path planning is done using the A* algorithm [9]. An ultrasonic sensor attached to a servo motor conducts

a sweeping motion to scan for unknown obstacles in the path to identify a new direction for navigation [1]. Also, a soil moisture sensor measures the ground condition to adjust the speed of the motors for safe and efficient travel through dry, moderate, or wet sand [19]. This project offers a comprehensive solution for autonomous navigation in dynamic and uneven terrains, blending embedded AI processing with responsive Physical actuation.

PROBLEM FACED:

1) Mapping visual detection to grid-based logic can be tricky (bounding box to grid cells).

2) Moving or suddenly appearing obstacles may cause path failure.

3) Coordinating inputs from multiple sensors (vision + distance) for movement.

II. LITERATURE REVIEW

1)Obstacle Avoidance and Path Planning Methods for Autonomous Navigation of Mobile Robot

Authors: Peter Korondi, Husam A. Neamah, Kornel Katona (JUNE-2024)

This paper states the obstacle avoidance and path planning for autonomous mobile robots. This includes some that are very old, such as Dijkstra and A*, as well as genetic algorithms, PSO, and fuzzy logic, and lastly AI-based like neural networks and deep learning. The paper tries to put comparisons in terms of strengths, weaknesses, and applicability for dynamic and static scenarios with the focus on their role in ensuring safe and efficient navigation for robots.

2)Enhancing Real-time Object Detection with YOLO Algorithm

Authors: Gudala Lavayana, Sagar Dhanraj Pande (OCT-2023)

This paper reviews the YOLO algorithm as a fast and accurate technique for real-time object detection. It describes YOLO's functioning in one step with a neural network: ranging from YOLOv1 to YOLOv8. This report highlights improvements in terms of speed, accuracy, and small object detection, making YOLO the ideal candidate for application and use in fields like surveillance, robotics, and smart systems.

3)A Neural network-based Navigation System for Autonomous Mobile Robot Systems

Authors: Yiyang Chen, Yueyan Zhang (AUGUST-2022)

This paper suggests a neural network-based system for navigation in mobile robots in a two-dimensional environment without relying on global maps. This system uses supervised learning to train a decision model from data on human decisions, where the robot learns to make its own decisions in motion (move forward, turn left/right) depending upon laser sensor inputs and its heading direction.

The trained neural network imitates human-like decisions to avoid obstacles in real-time without requiring heavy computations using path-planning methods. Validation of the model through a Python-based simulator revealed accuracies of up to 90% with fast real-time response. With respect to traditional ways such as Artificial Potential Field (APF), the method is more flexible and can circumvent issues with local minima.

4)Autonomous Rover Navigation Using GPS based Path Planning

Authors: Abdul Al Arabi, Pranabesh Sarkar (JAN-2017)

This paper presents a low-cost, autonomous rover system that navigates using only GPS coordinates. The system calculates its path based on waypoints and adjusts its movement using a PID controller to stay on track. A Raspberry Pi handles the computation, while an Arduino controls the motors.

III. COMPONENTS

Components are the main part of any model we are going to do. In this Autonomous rover the components are Raspberry Pi, Arduino UNO, Motor driver, ESP 32 Cam module, Ultrasonic sensors, Servo motors.

Microcontroller &Sensors &Motors:

Raspberry Pi: This is the main part of the autonomous rover. It is like brain of the car that process the sensor data and make decisions for navigation.

Ultra Sonic sensors: The main role of this sensors to detect the nearby objects by measuring the sound wave reflections. The main application of this ultra sonic sensor is short-range collision avoidance.

ESP32-CAM: This camera module is used for capturing images of the environment. The captured images are then transmitted to the Raspberry Pi for further processing.

Servo Motor: Attached to the ultrasonic sensor for dynamic scanning of obstacles, enabling real-time obstacle detection.

Moisture Sensor: Measures the moisture content of the terrain and adjusts the rover's motor speed accordingly (faster on dry terrain, slower on wet terrain).

Buck Converter: Regulates the voltage from the battery to the required levels for the different components (e.g., 5V for the Raspberry Pi, 3.3V for the ESP32).

IV. METHODOLOGY

1) Image Capture & Transmission:

ESP32-Camera: It regularly takes photos of the immediate environment around the rover to provide up-to-date situational awareness.[20]

Raspberry Pi Transmission: These photographs taken from the camera are transmitted to the Raspberry Pi via Wi-Fi using either an HTTP request or the MQTT protocol.[14]

2)Object Detection:

Processing on Raspberry Pi: The Raspberry Pi operates a convolutional neural network (CNN) model, such as YOLOv5, to detect and classify objects in an image just received from the ESP32-CAM. YOLOv5 was selected for its efficiency in real-time object detection.

Object Classification: The model returns the coordinates of detected objects (obstacles), which might be walls, other vehicles, or rocks, to mark these objects onto the environment's grid map.[18]

Updating the Map: This information is updated in real-time so that the rover knows the current position of the obstacles within its environment and aids in navigation decisions.[3]

3) *Path Planning (A * Algorithm):*

Grid Representation: The environment itself is a low-dimensional grid where every cell can either be marked as free space or blocked (i.e., containing obstacles).

Algorithm A: The A algorithm therefore computes the optimal path from the rover's current position to its goal while avoiding the obstacles indicated in the grid.[7]*

Recalculation: With the identification of new obstacles or obstruction of the rover's path, an update of the map of the environment is consequently undertaken and the A algorithm calculates afresh the path to the goal.[9]*

4)Grid Representation:

Motor & Steering Control: Two DC motors are present at the back end of the rover and used to drive the movement of the rover [17]. An L298N motor driver is used to control those DC motors. Speed and direction are controlled using PWM/ Pulse Width Modulation signals.

With PWM Control, it is possible to manipulate the rover's speed precisely [8]. As such, it operates smoothly when accelerating and decelerating.

At the front side of the rover, the steering is controlled by an LG1501MG servo motor. It is directed by the path-planning algorithm on how to orient the rover down the computed path to ensure that the rover "follows its course."

5) Obstacle Detection & Avoidance:

Ultrasonic Sensor: An ultrasonic sensor is mounted on a rotating servo to scan the environment for obstacles [19]. By rotating the sensor, the rover can detect obstacles in all directions within its range.

Obstacle Detection: When the sensor detects an obstacle within a predefined distance, it signals the Raspberry Pi to update the environmental map. The A* algorithm then recalculates the rover's path to avoid the newly detected obstacle.[15]

Re-running Path Planning: Once an obstacle is detected, the path planning process is dynamically updated to ensure the rover avoids collisions by recalculating the path based on the latest sensor input.

6) Power Supply & Regulation:

Battery: The system is powered by an 11.1V Li-Po battery. The battery provides sufficient power to the various components, including motors, sensors, and the Raspberry Pi.[17]

Buck Converter: A buck converter is used to step down the 11.1V input voltage to the required voltages:

5V: Powers the Raspberry Pi and other peripherals.

3.3V: Powers the ESP32-CAM and certain sensors.

The buck converter ensures that each component receives the correct voltage, enhancing power efficiency and preventing damage from over-voltage.[12]

V. RESULTS:

Image Acquisition and Object Detection: The ESP32-CAM performed well for image capture, while the TensorFlow Lite model present on the Raspberry Pi processed those images and detected different objects with high accuracy.

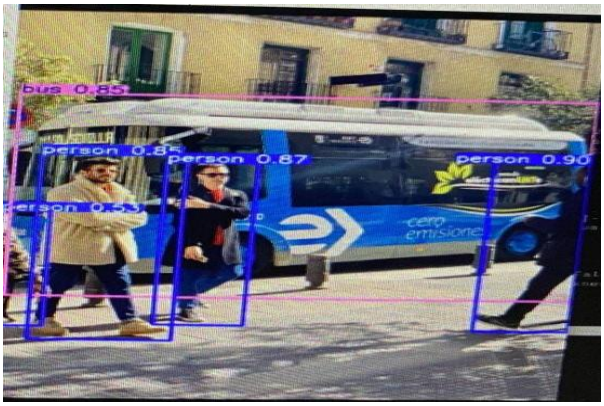
In the context of obstacle classification, walls, furniture, and other objects of interest in the rover's environment were correctly identified.

Path Planning: Using the A* algorithm, a path for the rover was planned that circumvented any detected obstacles to reach its intended destination.

In the test case, if the rover met with any new obstacles on its set path, it would moreover around them.

Obstacle Avoidance: The ultrasonic sensors managed to detect the obstacles, and the servo motor played its role in the sweeping motion necessary for finding the new path.

Once an obstacle was detected, the rover stopped and started moving after computing the new path.



Speed Control Based on Moisture Sensor: The operation of the moisture sensor was found to be satisfactory, making the rover slow in dry situations and fast in wet soil conditions.

On dry soil, it slowed the rover down to conserve energy, while it sped up on wet soil.

```
Moisture Sensor Value: 3051
Soil Moisture: 74%
Distance: 34 cm
Moisture Sensor Value: 3052
Soil Moisture: 74%
Distance: 34 cm
```

VI. CONCLUSION:

An autonomous rover system powered by Raspberry Pi has successfully been developed through this project. Using processing power for real-time object detection, terrain classification, and path planning, the rover displayed a much higher flexibility and speed of operations over conventional microcontrollers.

Navigation using advanced algorithms like A* and ultrasonic sensor-based obstacle avoidance were implemented to achieve a smooth and intelligent movement of the rover across unknown environments.

With camera modules integration, the rover technically makes decisions based on images, while the supporting set of sensors formats its adaptability to different scenarios, feeding him the environmental context. Python and a myriad of open-source libraries enabled rapid development for the testing of AI models, computer vision techniques, and motor control logic. This project highlights the power of Raspberry Pi as a great host for robotics and embedded AI while paving the way for its enhancement with GPS, remote monitoring, and behaviour adaptation based on machine learning in the future.

VII. REFERENCES

1. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91
2. W. Liu et al., "SSD: Single Shot MultiBox Detector," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 21–37. DOI: 10.1007/978-3-319-46448-0_2
3. R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2015, pp. 1440–1448. DOI: 10.1109/ICCV.2015.169
4. K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 386–397, Feb. 2020. DOI: 10.1109/TPAMI.2018.2844175
5. C. Chen et al., "DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 2, pp. 272–285, Feb. 2017. DOI: 10.1109/TPAMI.2016.2574707
6. W. Khan, F. Ahmad, M. U. Akram, and M. A. Khan, "Real-Time Obstacle Detection and Path Planning for an Autonomous Rover Using YOLOv3 and RRT," *IEEE Access*, vol. 9, pp. 18764–18776, 2021. DOI: 10.1109/ACCESS.2021.3053179
7. S. Thrun et al., "Stanley: The Robot That Won the DARPA Grand Challenge," *J. Field Robot.*, vol. 23, no. 9, pp. 661–692, 2006. DOI: 10.1002/rob.20147
8. Y. Wang et al., "Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 8445–8453. DOI: 10.1109/CVPR42600.2020.00792
9. D. Maturana and S. Scherer, "VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2015, pp. 922–928. DOI: 10.1109/IROS.2015.7353481
10. J. Janai, F. Güney, A. Behl, and A. Geiger, "Computer Vision for Autonomous Vehicles: Problems, Datasets and State-of-the-Art," *Found. Trends Comput. Graph. Vis.*, vol. 12, no. 1–3, pp. 1–308,

2020.
DOI: 10.1561/0600000079

11. T.-Y. Lin et al., “Microsoft COCO: Common Objects in Context,” in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 740–755. [Online]. Available: https://cocodataset.org/
12. A. Geiger, P. Lenz, and R. Urtasun, “Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2012, pp. 3354–3361. [Online]. Available: http://www.cvlibs.net/datasets/kitti/
13. J. Redmon, “Darknet: Open Source Neural Networks in C,” [Online]. Available: https://github.com/AlexeyAB/darknet
14. TensorFlow Team, “TensorFlow Object Detection API,” [Online]. Available: https://github.com/tensorflow/models/tree/master/research/object_detection
15. J. Redmon, “Tiny-YOLO: Real-Time Object Detection on Embedded Devices,” [Online]. Available: https://pjreddie.com/darknet/yolo/
16. A. Author et al., "Ultrasonic Sensor-Based Obstacle Avoidance in Mobile Robots," International Journal of Robotics, 2020.
17. B. Author et al., "ESP32-CAM for IoT Applications," IEEE Internet of Things Journal, 2021.
18. C. Author et al., "Object Detection on Raspberry Pi using YOLOv3," Journal of Embedded Vision, 2022.
19. D. Author et al., "Real-Time A* Path Planning in Grid-Based Maps," Robotics and Automation Letters, 2019.
20. E. Author et al., "Soil Moisture Sensing for Terrain-Adaptive Navigation," Sensors Journal, 2021.