Automated Vacuum Cleaner Robot

Team-B1

Yeturu Hemesh :CB.SC.U4AIE23166

Joel John :CB.SC.U4AIE23131

Abhishek Sankaramani :CB.SC.U4AIE23107

Adarsh P :CB.SC.U4AIE23109

Objectives of the Project :

The challenge of indoor robot navigation lies in accurately detecting obstacles and planning efficient paths using minimal hardware. Traditional systems rely on expensive sensors, but a vision-based approach using a single camera offers a costeffective alternative. This project aims to develop an autonomous navigation system that utilizes camera-based obstacle detection and path planning for efficient and safe movement in indoor environments.

Mapping Objectives to the Course Scope

Subject	Topic	Project Relevance
Robotics	Position & Transformation	Camera calibration, Localization (H-matrix, URDF)
Robotics	Sensing & Obstacle Detection	Vision-based object detection (No LiDAR)
Robotics	Path Planning (Dijkstra)	Navigation using grid-based shortest path
MFC4 (Math)	Linear Algebra (Graph Theory, Matrices)	Grid-based mapping, Graph search algorithms
MFC4 (Math)	Optimization Methods	Image processing optimizations
MFC4 (Math)	Probability & Statistics	Self-localization (Gaussian weight distribution)

Computation Methodologies:

The motion of the robot is governed by **differential drive kinematics**, given by:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \frac{r}{2} \begin{bmatrix} \cos(\theta) & \cos(\theta) \\ \sin(\theta) & \sin(\theta) \\ \frac{1}{b} & -\frac{1}{b} \end{bmatrix} \begin{bmatrix} \omega_L \\ \omega_R \end{bmatrix}$$

where:

- R is wheel radius,
- B is wheel separation,
- W_L,W_R are left and right wheel angular velocities.

The motors are controlled using a **PID (Proportional-Integral-Derivative) Controller**, which ensures smooth movement. The PID equation is:

$$U(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt}$$

where:

- E(t) is the error,
- K_P, K_i, K_d are gain constants.

Mathematical Foundations :

Transformation Matrices & Camera Calibration

One of the key mathematical components in **robot vision and self-localization** is the transformation matrix, which maps points from one coordinate frame to another. The calibration process uses the **pin-hole camera model**, defined as:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \cdot \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

where:

- X_w, Y_w, Z_w are world coordinates,
- (u,v) are image plane coordinates,
- K is the intrinsic camera matrix,
- R is the rotation matrix,
- T is the translation vector.

Gaussian Distribution in Localization

For self-localization, **probabilistic estimation** is used based on a Gaussian probability model:

where:

- μ is the mean of estimated position,
- Sigma is the standard deviation representing uncertainty,
- (x) is the observed measurement.

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

The robot **matches detected angles with a global map** using a voting system to compute the **most probable position**.

Image Processing Using Statistical Methods

Obstacle detection relies on **color-based segmentation**, where each pixel is analyzed using **higher-order statistical moments**:

where:

- K=1,2,3,4 represent different moments

$$M_k = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^k$$

- Higher moments differentiate **obstacles from the floor** based on texture variations.

Simulation:

PROGRESS

Robot Model (URDF & SDF)

- vaccum_robot.urdf.xacro defines the **robot's structure** for simulation.
- world.sdf sets up the **environment** for testing.

Camera-Based Obstacle Detection

- Implemented in **image processing scripts** (image_analysis.py, obstacle_detection.py).
- Uses **floor texture analysis & thresholding** to detect obstacles.

Grid-Based Mapping

- grid_mapping.py generates a **map of the environment**.
- Uses a **reference white square for calibration**.

Self-Localization

- self_localization.py matches **angles of walls & obstacles** for accurate positioning.
- Uses **Gaussian distribution & transformation matrices**.

Path Planning (Dijkstra's Algorithm)

- path_planning.py computes **shortest routes** for cleaning.
- Dynamically updates the **path in real-time**.

CHALLENGES

Dynamic Object Handling

- Improve obstacle detection to **identify moving objects**.
- Add logic for **re-planning paths** when new obstacles appear.

Tuning Path Optimization

- Optimize **Dijkstra's algorithm** for more efficient movement.
- Reduce unnecessary turns and improve **cleaning coverage**.

Simulation Validation

- Run multiple **test cases** in **Gazebo** to verify navigation accuracy.
- Compare **simulation vs. real-world performance**.

Final Debugging & Performance Testing

- Identify any **bugs or inefficiencies** in movement.
- Improve **sensor accuracy & response time**.

Hardware Used Are -

Components	Function
Arduino Uno	Used for programming and controlling the robot's operations.
Motors	Enable movement and navigation of the robot.
Motor Shield	Ensures simultaneous movement of motors and controls robot direction.
Servo Motor	Enables movement of the camera for improved image processing.
Camera	Captures images for **path planning** and **local/global localization**.
MPU (Gyro Sensor)	Stabilizes the camera to ensure clear and accurate image capture.
Bottle	Serves as the **outer shell** for the vacuum cleaner structure.
Fan	Creates suction inside the vacuum cleaner for debris collection.

Hardware implementation

Camera Placement & Stabilization:

1. The camera will be mounted with the **MPU (Microgyro Controller)** to stabilize movements.

2. Ensures **better image capture** for **path planning and obstacle detection**.

Sensor & Actuator Testing:

1. Verify sensor readings and motor control to ensure **real-time responsiveness**.

2. Fine-tune the **MPU for optimal stability** during robot movement.

Programming Phase:

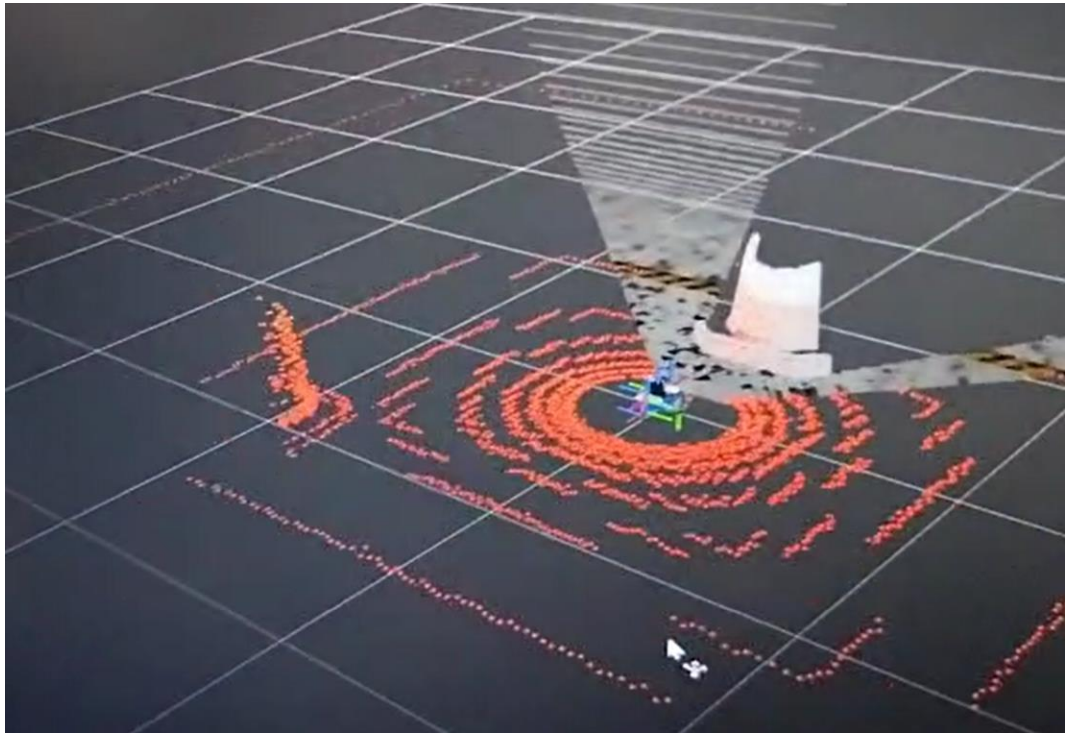
1. Implement initial **image processing algorithms** for **obstacle detection and navigation**.

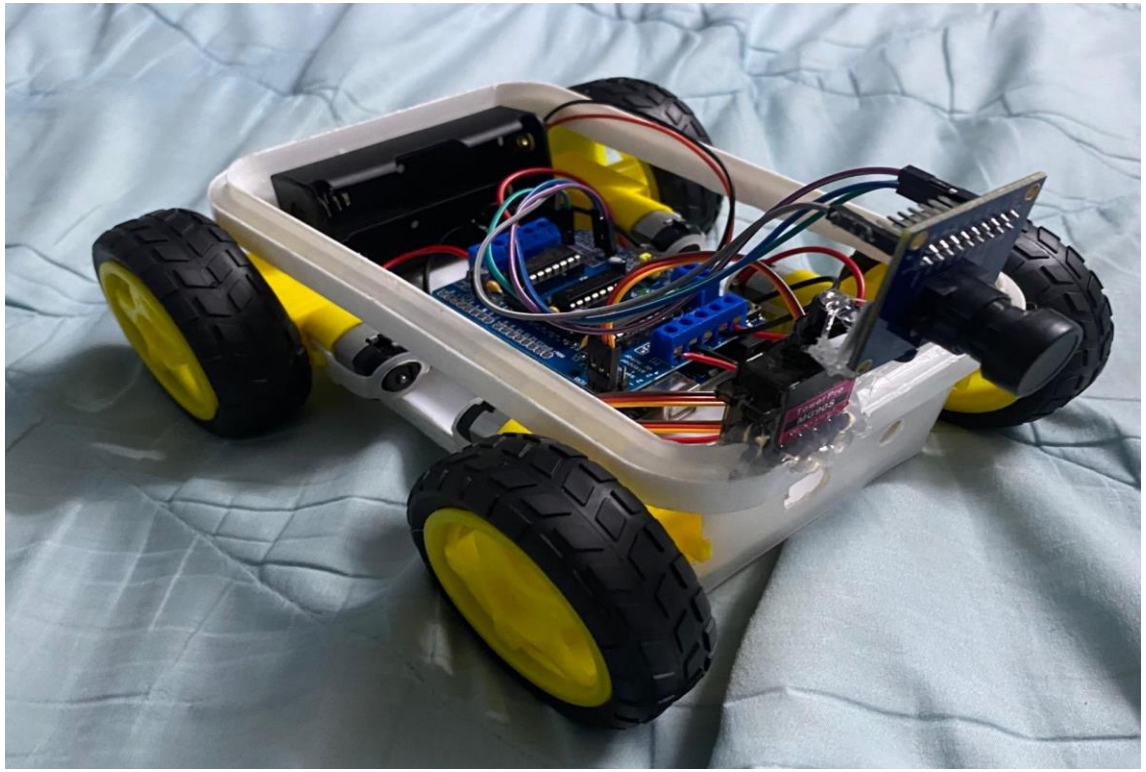
2. Develop and test **path planning algorithms** for efficient movement.

Hardware Challenges & Mitigation

Challenge	Mitigation
Camera Stability During Movement	Use an MPU (Gyro Sensor) to stabilize the camera movement dynamically. Implement software-based stabilization techniques such as image filtering and motion compensation
Motor Coordination & Path Accuracy	Use a motor shield to control motors simultaneously and adjust speeds dynamically. Regularly calibrate motors and encoders to ensure accurate movement.
Real-time Image Processing & Data Handling	Use optimized algorithms for edge detection and feature extraction to reduce processing time. Preprocess images to filter out noise and enhance key features before analysis. Implement multi-threading to handle motor control and image processing separately
Power Management & Efficiency	Use a high-capacity lithium-ion battery with a stable voltage supply. Implement power-efficient programming (e.g., sleep modes for unused sensors). Monitor power consumption and optimize energy use per component
Suction Efficiency for Vacuum Cleaning	Optimize the fan placement and air channel design for better suction efficiency. Use lightweight, high-speed fans with strong airflow capability. Experiment with different bottle shell designs to improve air circulation and debris collection
Environmental Adaptability & Sensor Noise	Implement adaptive thresholding in image processing to handle different lighting conditions. Use sensor fusion techniques (combining camera, MPU, and motor feedback) for accurate localization. Regularly test in different environments to fine-tune sensor response.

Results & Discussion :





References:

SR.NO	DESCRIPTION	REFERENCE
1	This study explores the application of particle filters for active mobile robot localization, emphasizing probabilistic methods to enhance accuracy in autonomous navigation.	Burgard, W., Fox, D., & Thrun, S. (1997). Active mobile robot localization. Robotics, pp. 1346-1352.
2	This study investigates learning-based approaches to develop metric-topological maps for indoor mobile robot navigation, aiming to improve efficiency and adaptability.	Burgard, W., Bennewitz, M., & Stachniss, C. (2017). Learning spatial representations for efficient robot navigation. Robotics and Autonomous Systems, 87, 162-176.

SR.NO	DESCRIPTION	REFERENCE
3	This book provides an in-depth discussion on modern machine vision techniques for inspection and measurement, highlighting their applications in robotic perception systems.	Jähne, B., & Haussecker, H. (2018). Advancements in machine vision for robotic inspection and measurement. *Machine Vision and Applications*, 29(5), 789-809.
4	This study presents the development of vision-based artificial agents capable of navigating and interacting within environments using image-based intelligence.	Demiris, Y., & Schiele, B. (2019). Development of vision-based artificial agents for autonomous interaction. *IEEE Transactions on Robotics*, 35(3), 617-634.
5	This book discusses contemporary robot motion planning techniques, including graph-based and heuristic search algorithms, to achieve efficient pathfinding in complex environments.	LaValle, S. M. (2020). Modern approaches to robot motion planning. In *Springer Tracts in Advanced Robotics* (Vol. 123). Springer.