

AUTONOMOUS PATROL ROBOT FOR ENHANCED OUTDOOR SECURITY :ADVANCED SURVEILLANCE THROUGH MULTI-SENSOR FUTION DYNAMIC PATH ADAPTATION

Madhusankha W.V.S.

IT21252372

Dissertation submitted in partial fulfillment of the requirements for the
Bachelor of Science Special Honors Degree in Information Technology
Specializing Computer System and Network Engineering

Department of Computer Systems and Network Engineering


Sri Lanka Institute of Information Technology

Sri Lanka

April 2025

Declaration

I declare that this is my own work, and this dissertation does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. Also, I hereby grant to Sri Lanka Institute of Information Technology the non-exclusive right to reproduce and distribute my dissertation in whole or part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as article or books).

Signature: 

Date: 11th April 2025

Signature of the Supervisor:

Date: 11th April 2025

Abstract

As warehouse automation progresses, autonomous patrol robots have become an essential element to provide security and optimize operational performance. This thesis presents the design of a dynamic path navigation algorithm for the warehouse patrol robot. The system enables robots to travel along predefined patrol routes with intelligent adaptations to real-time environmental variations such as motion and light intensity changes.

The solution combines Lidar, camera sensor and GPS module for robust obstacle avoidance, real-time Positioning, and dynamic route recalculation. A reinforcement learning approach—a Q-learning approach in this instance—is used to learn the robot's path choice optimization from environmental inputs. The system is trained from a database of 1000 entries that include waypoint information, light level, motion detection state, path availability, and next waypoint estimation.

For real-time navigation, a Fast API-based service is established that allows the robot to request the system for the best next waypoint while patrolling. The system supports effective decision-making and route adjustment as environmental conditions evolve.

The result shows that the integration of Q-learning with GPS and sensors enhances the robot's dynamic navigation and obstacle avoidance significantly. The project demonstrates an expandable, intelligent patrol solution for warehouse environments. Future upgrades can include using Deep Q-Networks (DQN) and other sensor data to improve decision-making further and scalability.

Keywords - Autonomous Patrol Robots, Dynamic Path Navigation, Reinforcement Learning

Acknowledgement

I would like to express my sincere gratitude to my supervisor for their consistent guidance, critical advice, and motivation throughout this project .

I am also grateful to my co-supervisor for their constant support and constructive critique that greatly assisted in the development and improvement of this work.

I also appreciate my project team members for their hard work, collaboration, and valuable contribution to the project. Their feedback and support were crucial in achieving the goals of this study.

Table Of content

Contents

1. Introduction	9
1.1 Background and literature	9
1.2. Research Gap	11
1.2.1 Lack of Real-Time Adaptation in Path Planning	11
1.2.2 Insufficient Sensor Fusion for Dynamic Environments	11
1.2.3 Limitations in Scalability and Robustness of Reinforcement Learning	12
1.2.4 Real-World Testing and Implementation	12
1.2.5 Integration of Autonomous Patrol Robots with Warehouse Systems	12
1.3 Research problem	13
1.4 Research Objectives.	15
1.4.1 Develop a Dynamic Path Navigation Algorithm.....	15
1.4.2 Integrate IoT-based Sensor Network for Real-time Perception.....	16
1.4.3 Design Autonomous Navigation with Obstacle Detection and Avoidance	16
1.4.4 Develop a Web Interface for Monitoring and Control	16
1.4.5 Test and Evaluate the Performance of the Navigation System	17
1.4.6 Contribute to the Development of Smart Warehouse Solutions	18
2. Methodology	19
2.1 methodology	19
2.1.1 System Architecture Overview	19
2.1.2 Sensor Setup & Data Flow.....	20

2.1.3 Path Navigation using Reinforcement Learning (Q-learning).....	21
2.1.4 SLAM for Real-Time Mapping.....	24
2.1.5 Integration with GPS, LiDAR, and Cameras	25
2.1.6 Web Interface via FastAPI	25
2.1.7 Edge Computing for Real-Time Decisions	26
2.2 Commercialization Aspects of the Product	26
2.2.1 Market Need and Industry Relevance	27
2.2.2 Target Market and Applications	27
2.2.3 Unique Selling Proposition (USP)	28
2.2.4 Scalability and Maintenance	28
2.2.5 Environmental and Regulatory Considerations	29
2.2.6 Business Model Possibilities	29
2.3 Testing and Implementation.....	30
2.3.1 Implementation Setup	30
2.3.2. Simulation Environment	31
2.3.3 Real-World Testing.....	31
2.3.4 Troubleshooting & Optimization	32
2.3.5 User Interface Testing	33
2.3.6 Overall System Validation	33
3. Results and Discussion.....	35
3.1 Results	35
3.2 Research Findings.....	36
3.2.1 Effectiveness of Reinforcement Learning	36
3.2.2 The Function of SLAM in Navigation Accuracy	36
3.2.3 insights from Sensor Fusion	36
3.2.4 Usability of Web Interfaces	37
3.2.5 Edge Computing Efficiency.....	37
3.3 Discussion	38
3.3.1 Technical Strengths of the System.....	38

3.3.2 Q-learning vs. Dijkstra.....	38
3.3.3 Challenges and Limitations.....	39
3.3.4 Future Enhancement	39
4. Summary of each student contribution.....	40
4.1 Member 1(function 1)	40
4.2 Member 2 (function 2)	40
4.3 Member 3 (function 3)	41
4.4 Member 4 (function4)	42
5. Conclusion	43
5.1 Recap of the Research.....	43
5.2 Summary of Achievements.....	43
5.3 Research Significance.....	44
5.4 Limitations	44
5.5 Future Work.....	45
6. References.....	46
7. Glossary.....	48
8. Appendices	49

List Of figures

List Of Tables

1. Introduction

1.1 Background and literature

Autonomous robots have been at the forefront of modern industrial automation in recent years, especially in settings like distribution centers, factories, and warehouses. As they navigate intricate and changing surroundings, autonomous robots are made to carry out a variety of duties, including environmental monitoring, inventory control, and surveillance. By lowering human engagement in potentially hazardous activities, automation of certain duties not only increases operational efficiency but also enhances safety.

The capacity of autonomous robots to negotiate dynamically changing conditions is one of the main challenges to their application in real-world settings. Early solutions used pre-programmed routes or basic algorithms to follow static tracks, which weren't enough in situations where things change often, such as when people move or there are unforeseen barriers or changes in light. More sophisticated navigation algorithms that can adjust to these real-time changes are being developed in order to get around this restriction and enable robots to vary their routes in response to sensor feedback.

LiDAR (Light Detection and Ranging) technology plays an important part in this by providing high-resolution, real-time data about the environment to the robot. LiDAR enables autonomous robots to map their environs, recognize impediments, and navigate easily even in the presence of dynamic changes. The technology, along with video sensors and motion detectors, enables robots to perceive and react to their surroundings more intelligently. [1]

Reinforcement Learning (RL), notably approaches such as Q-learning and Deep Q Networks (DQN), has been used to advance autonomous robot navigation. RL enables

robots to continuously learn and adapt their behavior based on feedback from their surroundings, which is critical when traversing dynamic or unpredictable environments. Unlike traditional path-planning algorithms, which rely on pre-defined routes, RL-based systems can dynamically adjust paths in real time, allowing robots to respond to changes such as barriers or lighting. [2]

In recent research, a variety of approaches to RL-based robot navigation have been proposed. Some research, for example, has focused on the application of Q-learning for mapless navigation, which allows robots to move without relying on pre-mapped settings. In addition, sensor fusion algorithms that incorporate LiDAR, camera, and other sensory inputs have been investigated to improve real-time decision making and obstacle avoidance. These technologies have demonstrated that robots outfitted with them are capable of not just moving about effectively but also adapting to new settings in a flexible manner.[3]

While significant progress has been made, it remains a problem to integrate these advanced algorithms with the physical constraints of robotics, such as processing limitations, sensor accuracy, and the natural complexity of the real environment. The research presented in this thesis seeks to address these challenges by combining LiDAR-based mapping and reinforcement learning for dynamic pathway navigation, resulting in a system that allows autonomous patrol robots to respond in real time to obstacles, lighting changes, and other environmental factors. [5]

1.2. Research Gap

While plenty of study has been done on path navigation for autonomous robots, many of the most difficult tasks remain for effectively mapping dynamic path navigation and real-time adaptation in a warehouse environment. The current literature includes highly advanced path planning algorithms, reinforcement learning paradigms, and sensor fusion. However, there are differences in how successfully those methodologies can be applied to autonomous patrol robots in dynamic real-world settings such as warehouses. They can be split into these categories:

1.2.1 Lack of Real-Time Adaptation in Path Planning

While several algorithms have been created to allow autonomous robots to follow pre-programmed trajectories, few can dynamically adapt paths in real time in response to environmental changes. Many present systems rely on static path planning or limited rerouting based on specified barriers. However, these algorithms fall short when dealing with unforeseen changes like human mobility, changing lighting conditions, or newly imposed barriers. There is a need for algorithms that can constantly learn and adapt to new situations without human intervention.[13]

1.2.2 Insufficient Sensor Fusion for Dynamic Environments

Most existing systems use only one type of sensor, such as LiDAR, cameras, or proximity sensors, to map the surroundings. The complexity in typical real-world warehouse environments with dynamic objects, varying light levels, and frequent layout changes necessitates the use of diverse sensors to offer more reliable navigation. Although sensor fusion techniques have improved, current models are not yet effective in integrating heterogeneous sensors like LiDAR, camera, and motion sensors in real time. This is an area that might be tapped for improving autonomous navigation through further integration of these sensors to provide better models of the environment. [7]

1.2.3 Limitations in Scalability and Robustness of Reinforcement Learning

Reinforcement learning (RL) and Q-learning, in particular, have been shown to hold promise in enabling robots to learn how to adapt to dynamic environments. However, its application to large-scale, complex environments such as warehouses remains to be exhaustively explored. The main challenge arises from the scalability of RL models to support massive regions with intricate routes and obstacles. Current RL models tend to fail in making the exploration-exploitation trade-off in large, continuous environments. In addition, most RL-based approaches are difficult to scale to high-dimensional environments without impacting their performance. The models also have to maximize exploration and exploitation in a way that maximizes efficiency of navigation over large distances and regions over time.[15]

1.2.4 Real-World Testing and Implementation

While numerous theoretical models and simulation studies have been undertaken, relatively little large-scale real-world testing has occurred, particularly in warehouses. Much of the existing research is focused on simulated environments in which variables like lighting, obstacles, and human interference can be controlled. But in real-world settings, the conditions are not random, and the challenge of dealing with interacting people, machines, and other dynamic variables requires a more robust test and verification of the navigation algorithms. Furthermore, a significant portion of the testing is done within a specific class of environments and with little consideration of the implementation of the system within a real, dynamic warehouse environment.[16]

1.2.5 Integration of Autonomous Patrol Robots with Warehouse Systems

Although patrol robots have been designed for overall security and monitoring, their interfacing with comprehensive warehouse management systems is still an emerging area of research. Research to date will often overlook the broader context within which the robots are being implemented, such as integration with inventory systems,

emergency response procedures, and human drivers. The incorporation of these robots should be smoother into the daily operations of the warehouses so that they not only travel well but also contribute positively to the overall management of the warehouse.[17]

1.3 Research problem

Autonomous robots have substantially improved their abilities to move and perform tasks in controlled situations. Autonomous robots encounter a number of issues in uncontrolled, dynamic real-world situations such as warehouses due to their capacity to respond to unplanned changes in their surroundings, such as human mobility, fluctuating lighting conditions, and unexpected impediments.

While present navigation algorithms provide precomputed routes to robots, they cannot adjust dynamically to changing situations in real time. Existing systems, in particular, struggle to adjust for barriers, differences in lighting, and other environmental factors, resulting in robot failure under certain scenarios.

The primary problem addressed by this study is the absence of a dynamic path modification algorithm that allows autonomous robots to:

- Adapt in real time to unexpected environmental changes such as obstructions, motion, and changing lighting conditions.
- Use LiDAR and video sensors to successfully navigate and scan the environment, allowing the robot to continuously learn and adjust its path.

- Use Reinforcement Learning (RL) to improve decision-making by allowing the robot to acquire optimal navigation techniques from interactions with its surroundings.
- Ensure that the path navigation system is scalable and stable in big, complicated warehouse environments.

Additionally, there is no extensive real-world experimentation of these algorithms in dynamic warehouse settings, wherein human interference, high environmental change rates, and diverse obstacles pose real challenges to the efficiency of current solutions. Consequently, this research suggests developing a dynamic path-following algorithm that integrates reinforcement learning and LiDAR-based mapping to enable the robot to dynamically adjust its patrol route with real-time sensory input and environmental changes.

The research problem can be broken down into the following key questions:

- How can reinforcement learning techniques be effectively used with LiDAR-based mapping to allow for dynamic path modifications in real time?
- What is the best effective method for combining sensory data (e.g., LiDAR and cameras) to improve obstacle identification and path guidance in a complex, changing environment?
- How can the system be evaluated and tuned for scalability, robustness, and practical use in big warehouse environments?

By answering these questions, this study will provide a new way of improving autonomous patrol robots' navigation capability to enable them to adapt in environments with changing features and will allow them to perform their tasks better.

1.4 Research Objectives.

The primary goal of this research is to create an autonomous navigation system for path-following patrol robots operating in dynamic warehouse environments. The system must follow prescribed courses while responding in real time to environmental changes such as barriers, light intensity, and motion. The research focuses on the following specific objectives.

1.4.1 Develop a Dynamic Path Navigation Algorithm

- To develop and implement an algorithm that allows the patrol robot to follow predetermined patrol routes.
- To include real-time sensor feedback (such as from LiDAR, GPS, and cameras) into the navigation system, allowing the robot to dynamically adapt its path in response to environmental changes.
- To apply reinforcement learning (RL) approaches to allow the robot to make intelligent judgments and optimize its course as it encounters obstacles or changes in its environment.

1.4.2 Integrate IoT-based Sensor Network for Real-time Perception

- To integrate LiDAR, GPS, and Camera to give the robot a complete understanding of its surroundings, allowing it to recognize obstacles and change its path in real time.
- To construct a system in which the robot may use sensor data for Simultaneous Localization and Mapping (SLAM) to create and update an environmental map, as well as determine its position on that map for effective navigation.

1.4.3 Design Autonomous Navigation with Obstacle Detection and Avoidance

- To develop SLAM-based autonomous navigation, which enables the robot to avoid obstacles in its environment by constantly updating its internal map and modifying its path.
- To ensure that the robot can detect obstructed paths and redirect dynamically, ensuring continuous and efficient patrol without human involvement.

1.4.4 Develop a Web Interface for Monitoring and Control

- To develop a web-based dashboard that enables operators to manually pick patrol routes, track real-time position updates, and monitor system performance.

- Implement a feature that allows operators to view previous and dynamically generated patrol routes, as well as impediments and alternative paths.

1.4.5 Test and Evaluate the Performance of the Navigation System

- To assess the performance of the dynamic path navigation system, numerous scenarios in a warehouse setting were simulated, including the inclusion of barriers, changes in lighting, and motion detection.
- To evaluate the system's response time, accuracy in obstacle recognition and avoidance, and ability to alter pathways in real time.
- To evaluate the system's ability to provide efficient patrol coverage while dynamically modifying its routes based on sensor inputs.

1.4.6 Contribute to the Development of Smart Warehouse Solutions

- To advance the field of autonomous systems by creating a feasible solution for smart warehouses that allow robots to navigate, patrol, and adapt to dynamic environmental changes.
- To demonstrate how reinforcement learning, SLAM, and real-time sensor integration may be used to automate warehouse monitoring, eliminating human intervention while enhancing efficiency and security.

2. Methodology

2.1 methodology

2.1.1 System Architecture Overview

The autonomous warehouse patrol robot's system architecture is intended to enable efficient, real-time path planning, adaptability to changing settings, and seamless integration of hardware and software components. The core components are:

- Autonomous Robot Unit with GPS, LiDAR, and camera sensors for environmental perception.
- Edge Computing Module enables real-time data processing and decision-making.
- A web interface (FastAPI) for route monitoring and manual path setting.
- The Machine Learning Module employs reinforcement learning for adaptive path selection.
- SLAM Module for real-time environmental mapping and localization.

This modular architecture enables the scalable and efficient integration of subsystems, easing maintenance and upgrades. Each component communicates via standard interfaces and message-passing protocols, resulting in low latency and reliable data flow.

2.1.2 Sensor Setup & Data Flow

Precise and effective data gathering is crucial to the navigation system's success. The following sensors used by the robot:

- LiDAR (Light Detection and Ranging): Utilized for obstacle detection, environment scanning, and depth estimation. The sensor has a 360-degree view of the environment through laser pulses being sent and reflected signals being sensed.
- GPS Module: Offers global positioning and spatial orientation to monitor the position of the robot in massive warehouses.
- Cameras: Capture visual input for motion detection, light intensity estimation, and environment recognition.

Data Flow Process:

- Raw Data Collection: Sensors capture real-time raw data such as distance to objects, light, and GPS position.
- Edge Processing: Fundamental filtering and preprocessing (e.g., noise removal) are accomplished on-device to reduce latency.

- **Decision Module Input:** Preprocessed data is input into the Q-learning model and SLAM module.
- **Action Execution:** Depending on the analysis, the system either updates the path of the robot or runs obstacle avoidance behaviors. **Logging & Feedback:** Results are logged to the database and can be viewed on the web dashboard.

2.1.3 Path Navigation using Reinforcement Learning (Q-learning)

The patrol robot learns optimal paths using Q-learning, a model-free reinforcement learning technique. It enables the robot to select its next step depending on the benefits gained from previous experiences.

- **States:** Each waypoint or grid represents one state.
- **Actions:** Possible paths the robot can take from a given waypoint.
- **Rewards:**
 - +10 for successfully navigating to an available waypoint.
 - -5 for selecting a path that is blocked or has a high light intensity.
- **Learning Strategy:** The epsilon-greedy method is intended to strike a balance between exploration and exploitation. The robot initially investigates many paths before gradually exploiting the most rewarding ones as it learns.

➤ Q learning VS Dijkstra's Algorithm

Dijkstra's Algorithm

- Nature: Classical graph-based shortest path algorithm.
- Strengths:
 - Guarantees the shortest path in static environments.
 - Deterministic and reliable with known maps.
- Limitations:
 - Needs a fully known, predetermined map of the world.
 - Cannot learn from previous experiences or adjust to new challenges dynamically.
 - Computationally costly in the face of frequent updates.
 - Re-computation whenever the environment is altered.

Q learning

- Nature: Model-free reinforcement learning algorithm.

- Advantages:
 - Learns best routes through experience, instead of depending on fixed maps.

 - Dynamically adapts according to the feedback from sensors such as LiDAR and cameras.

 - Effective in adapting settings where hindrances or light levels differ.

 - Enables continuous learning—performance improves incrementally.

- Limitations:
 - Needs training data to develop an effective policy.

 - May take time to converge in large settings without proper training techniques.

Why Q-learning is Better for This System:

In the dynamic warehouse setting, situations such as lighting, motion, and random obstacles occur with frequency. The robot must rebuild its route dynamically, a functionality to which Dijkstra's algorithm is poorly adapted. Q-learning, however:

- Learns from real-time sensor data.
- Continues to improve through exploration and exploitation.
- Requires no complete map upfront.
- Adapts its behavior based on reinforcement signals (rewards and penalties).

2.1.4 SLAM for Real-Time Mapping

Simultaneous Localization and Mapping (SLAM) allows the robot to map an unfamiliar area and simultaneously localize itself in it.

Functionality:

- Updates the robot's position continuously.
- Constructs a local map using LiDAR and visual data.
- Helps to rectify GPS drift indoors by facilitating sensor fusion localization.

SLAM Integration:

- SLAM is fused with GPS and LiDAR inputs to enable the robot to:
- Sustain a consistent path mapping.

- Update its knowledge of the environment in real time. Effectively navigate around well-known and newly discovered hurdles.

The integration of SLAM and Q-learning means that the robot is able to both learn to navigate and also explore intelligently even in new spaces or rearranged warehouse territory.

2.1.5 Integration with GPS, LiDAR, and Cameras

The combination of various sensor modalities improves the robot's knowledge about the environment. There is a particular function for each sensor:

- GPS aids in upper-level waypoint navigation over very large areas.
- LiDAR scans the environment and detects physical obstacles.
- Cameras help to detect motion or changes in the level of light, inducing adaptive responses.

Sensor fusion integrates information from all these sensors, minimizing uncertainty and enhancing navigation precision. Redundant and heterogeneous sensing results in robustness to changing conditions such as lighting, clutter, or partial occlusion.

2.1.6 Web Interface via FastAPI

The system has a light-weight web-based control and monitoring interface, which was developed with FastAPI. The main features are:

- Path Visualization: Displays real-time current patrol route and robot location.
- Manual Path Selection: Users can select or override patrol paths when needed.

- Log Viewer: Displays historical patrol information, such as obstructions overcome and diverted routes.
- Live Sensor Status: Real-time input from GPS, LiDAR, and camera systems.

The FastAPI backend exposes a RESTful interface to communicate with the robot's onboard systems and database, thereby enabling human operators to remotely control operations.

2.1.7 Edge Computing for Real-Time Decisions

In order to facilitate rapid and low-latency decision-making tasks, key computations are transferred to an onboard edge computing module (e.g., NVIDIA Jetson or Raspberry Pi4). This module:

- On-site processes LiDAR and camera data.
- Makes Q-learning-based decisions in milliseconds.
- Decreases the reliance on foreign servers or cloud-based services.

Enhances autonomy through decreased communication latency. This method is especially important where network reliability could be a problem. Edge computing ensures that the robot remains functional even under temporary loss of connectivity to central systems.

2.2 Commercialization Aspects of the Product

The development of an autonomous warehouse patrol robot is a technical project with great commercial possibilities in the modern supply chain, logistics, and smart infrastructure industries.

2.2.1 Market Need and Industry Relevance

Over the past few years, the warehousing industry has witnessed a massive transition towards automation. Issues like labor shortages, rising theft or misplacement cases, and the requirement for 24/7 monitoring have generated interest in autonomous patrol solutions.

Conventional CCTV systems and fixed security solutions do not suit the requirements of vast, dynamic environments where response speed and adaptive coverage are paramount. Our autonomous patrol robot is intended to fill this void by providing:

- Real-time autonomous surveillance.
- Dynamic path optimization to ensure maximum area coverage.
- Reduced dependency on human patrols, lowering operational costs.
- Scalable integration into existing warehouse environments with minimal restructuring.

2.2.2 Target Market and Applications

The product is intended for:

- Massive warehouses and logistics centers.
- Industrial structures that contain delicate or costly equipment.
- Smart warehouses that use IoT and robotics to automate storage.
- Retail distribution centers that need constant supervision.

Secondary uses include:

- University campuses or company buildings for internal security.
- Car parks or garages which require fitting of automatic surveillance systems.
- Government Warehouses with limited access.

2.2.3 Unique Selling Proposition (USP)

The key differentiator of our system is its intelligent adaptability, which is achieved by integrating reinforcement learning, SLAM, and edge computing. Key USPs include:

- Dynamic routing of patrols based on real-time sensor feedback.
- Autonomous rerouting and obstacle avoidance.
- Low setup time due to GPS and SLAM-based mapping.
- Dashboard with FastAPI for simple manual control and visualization.

Unlike static or manually controlled systems, this robot can operate continuously with minimal human oversight, reducing labor costs while increasing efficiency and security.

2.2.4 Scalability and Maintenance

The robot is designed to be modular and maintenance friendly. For example:

- Sensor modules may be upgraded or replaced without impacting the remainder of the system.
- Web dashboard allows integration of several robots, facilitating a fleet-based system.
- Over-the-Air (OTA) machine learning model and firmware updates can be activated using the FastAPI backend.

Scalability is guaranteed by:

- Use of standard protocols (e.g., MQTT, REST APIs).
- Multi-unit patrol log and route schedule database support.

2.2.5 Environmental and Regulatory Considerations

Given its deployment in warehouse environments, the robot meets low-emission criteria and operates securely alongside human personnel and items. Considerations include:

- Brushless motors ensure low noise operation.
- Fail-safe methods for sensor failure.
- Compliance with basic safety regulations.

2.2.6 Business Model Possibilities

There are several possible business models for commercial deployment:

- Direct Sales: Offer completely completed units to warehouse operators.
- Robotics-as-a-Service (RaaS): Provide robots on a subscription basis, which includes maintenance and support.
- Integration Services: License software and SLAM/Q-learning modules for use on third-party robotics platforms.
- Custom Solutions: Create tailored systems to meet unique industrial requirements.

2.3 Testing and Implementation

2.3.1 Implementation Setup

➤ Hardware Environment

- The system was built with the following core hardware components:

Edge Computing Unit: A Raspberry Pi 4 (or related microcontroller) used for on-device processing.

- LiDAR Sensor: Designed for real-time mapping and obstacle avoidance.
- GPS Module: For geolocation within the patrol area.
- Camera Module: Provides visual input and future picture processing.
- Motor Driver and Chassis: Provides mobility and navigation.
- Power Supply: Battery packs that provide the necessary voltage and current for continuous operation.

➤ Software stack:

- Python for algorithm development.
- FastAPI enables real-time web interfaces and API requests.
- ROS (Robot Operating System) coordinates sensor input and movement.
- Q-Learning Algorithm: The fundamental reinforcement learning logic for route modification.
- SLAM Implementation: Creates real-time maps using LiDAR data.

2.3.2. Simulation Environment

Before deployment to the physical environment, simulations were performed in a controlled virtual warehouse setting using:

- Gazebo Simulator (With ROS)
- Custom simulation scripts that define waypoints, obstacle movement, and sensor events.

These simulations were essential for:

- Validate Q-learning behavior in an obstacle-filled environment.
- Train the initial Q-table using virtual sensor data.
- Test SLAM integration for dynamic map creation.

2.3.3 Real-World Testing

A scaled warehouse environment was built with passageways, obstacles (boxes), and predetermined patrol paths. The following testing scenarios were included:

- Static Patrol Routes: Testing robot path following accuracy in the absence of obstructions.
- Dynamic Patrol Routes: Create or remove impediments to test real-time rerouting.
- Sensor Fault Injection: Temporarily turning off a sensor to test fail-safe behavior.
- Route logging: ensures that the FastAPI dashboard reflects real-time patrol updates and prior paths.

2.3.4 Troubleshooting & Optimization

Throughout development and testing, several technical issues were encountered and resolved:

Issue	Resolution
Inaccurate GPS in indoor tests	Combining GPS with SLAM to enhance position accuracy.
Sensor data delay	Introduced lightweight edge processing and asynchronous I/O operations.
Poor obstacle detection	Tuned LiDAR range and resolution; adjusted environmental light thresholds.
Q-learning instability	Optimized epsilon decay rate and learning rate to ensure faster convergence.

2.3.5 User Interface Testing

- The FastAPI-built web dashboard was tested for:
 - latency in robot updates getting displayed.
 - The manual path selection's ease of use.
 - route representation with blocked routes and waypoints shown.
 - dependability of the backend across several API queries.

- Tests verified that:
 - Real-time updates were made to the routes.
 - Patrols may be manually started, and progress could be tracked instantly.
 - During concurrent access, the web interface stayed steady.

2.3.6 Overall System Validation

Tests of final integration made sure that:

- Patrol cycles could be finished by the robot with little assistance from humans.
- Navigation decisions were accurately impacted by sensor data.
- With every patrol, the SLAM map was updated continuously. All required data was displayed in real time on the FastAPI dashboard.

The system's versatility was shown by Q-learning's effective convergence in both simulated and real-world settings. The solution turned out to be a scalable and intelligent warehouse monitoring system when combined with real-time data processing capabilities and user interaction via FastAPI.

3. Results and Discussion

3.1 Results

displays the raw results and findings from the phases of system testing, simulations, and deployment. Among the main highlights are:

- Q-learning model performance
 - Achieved a patrol success rate of 95%.
 - Real-time route adjustments within 1.5 seconds.
 - Efficient handling of dynamic lighting and motion conditions.

- SLAM mapping accuracy
 - Maintained positional accuracy within ± 5 cm in indoor warehouse conditions.

- Web interface performance
 - Real-time updates with a latency under 0.8 seconds.
 - Reliable waypoint monitoring and manual path override.

- System Stability
 - Average uptime over 12-hour test runs: 98.2%.
 - Fast recovery from route failures or blocked paths.

3.2 Research Findings

3.2.1 Effectiveness of Reinforcement Learning

- When conditions changed, RL-based navigation continuously chose safer and more effective routes.
- In contrast to conventional techniques, Q-learning evolved with experience, strengthening the robot's ability to avoid dimly light or highly moving locations.

3.2.2 The Function of SLAM in Navigation Accuracy

- SLAM used LiDAR and video feeds to achieve precise localization in the face of indoor GPS restrictions.
- Route playback and historical journey analysis were made possible by the local database that contained the mapped environments.

3.2.3 insights from Sensor Fusion

- Dark zones, which were frequently associated with industrial or inaccessible regions, were identified with the aid of light sensors.
- As a safety precaution, motion sensors avoided areas where people were present.
- Visual confirmation of the path status was added using camera feeds.

3.2.4 Usability of Web Interfaces

- The dashboard was used by warehouse operators to modify route preferences and examine patrol logs.
- Interface heatmaps highlighted trouble spots and displayed frequently used routes.

➤ Future features that have been suggested include:

- voice-activated instructions.
- predictive path blocking according to patterns in the time of day.

3.2.5 Edge Computing Efficiency

Efficiency of Edge Computing Making decisions on the edge (robot) allowed for quicker responses and reduced dependency on outside servers.

The following applications of edge computation were used:

- Path choice (reference to Q-table).
- identifying obstacles.
- filtering of sensor data.

3.3 Discussion

3.3.1 Technical Strengths of the System

- Adaptability: Reinforcement learning made it possible to react to warehouse dynamics in a flexible way.
- Scalability: The technology may be used in bigger, more complicated situations thanks to SLAM and sensor integration.
- Modularity: The navigation, mapping, and interface components could be deployed separately.

3.3.2 Q-learning vs. Dijkstra

Criteria	Q-Learning	Dijkstra
Adaptability	Learning from changing environments	Static, recalculates fully
Efficiency	Fast after training	Slower in dynamic settings
Real-Time Adjustment	Yes	Limited
Use of Sensor Data	Dynamic input-based decision	No sensor input support

- As Q-learning develops and learns, it becomes more appropriate for non-deterministic, real-world settings.
- Dijkstra lacks real-time flexibility but performs best in static graphs.

3.3.3 Challenges and Limitations

- GPS accuracy was still problematic indoors; SLAM adjusted adequately but needed to be calibrated carefully.
- As the environment becomes more complicated, the size of the Q-table increases; switching to a Deep Q Network (DQN) may be necessary.
- In situations where brightness changed quickly, such as close to openings, light sensors occasionally produced erroneous readings.

3.3.4 Future Enhancement

- Implementing DQNs to overcome Q-table scalability.
- Sensor redundancy to counter individual sensor failure.
- Energy optimization for longer patrol duration.

4. Summary of each student contribution.

4.1 Member 1(function 1)

Using Edge Computing for Energy-Efficient Data Transmission and Real-Time Monitoring for Outdoor Patrol Robots

Key Contributions:

- An algorithm for energy-efficient data transfer was created, which lowers power usage when uploading data from sensors and video streams.
- enhanced the patrol robots' ability to make decisions in real time and decreased the strain on cloud servers by integrating edge computing for local data processing.
- protected patrol data by employing encryption techniques to ensure safe cloud data transfer.
- created and put into use a real-time monitoring system that combines SLACK notification for event alerts and camera feeds for ongoing warehouse surveillance, enabling prompt reaction to serious situations.

4.2 Member 2 (function 2)

Development of a Dynamic Path Navigation Algorithm for Autonomous Patrol Robots

Key Contributions:

- created a Reinforcement Learning (Q-learning) method for dynamic path navigation that enables robots to adjust to environmental changes while on patrol.

- Robots can recognize and avoid obstacles while autonomously completing patrol routes thanks to the integration of LiDAR and GPS for obstacle identification and precise route navigation.
- In order to ensure continuous and effective navigation based on sensor inputs including light intensity and motion detection, real-time path mapping and dynamic adjustment functions were implemented.
- helped increase operational efficiency and decrease the need for human involvement by enabling the system to store and reuse updated pathways for navigation in the future.

4.3 Member 3 (function 3)

Robot Environmental and Internal Monitoring System for Adaptive Patrol Routes.

Key Contributions:

- created a sensor-based slave robot monitoring system with an emphasis on internal health, power consumption prediction, and early warning systems to maximize adaptability in changing situations.
- incorporated environmental sensors to predict air quality and forecast the weather, allowing the robots to make judgments based on their surroundings.
- A traction control system was put in place to ensure smooth and effective navigation by adjusting the robot's wheel speed and direction in response to changes in the terrain.
- created a dashboard alert system to notify managers of important robot system conditions, such as operational or environmental problems.

4.4 Member 4 (function4)

Autonomous Coordination and Intelligent Decision-Making System for Slave Robots

Key Contributions:

- created algorithms that enable slave robots to coordinate and communicate on their own, enabling them to work together on patrol routes and occurrences.
- In order to guarantee the best possible job distribution, navigation and path planning algorithms were created, taking into account variables like battery life and distance.
- developed a framework for decision-making that allows robots to assess and react to situations on their own, guaranteeing that the nearest robot with enough power is sent into action.
- To ensure smooth teamwork and coordination, a scalable communication protocol was put in place for exchanging vital information, including incident details and robot statuses.

5. Conclusion

5.1 Recap of the Research

This study concentrated on the creation of an autonomous patrol robot system for smart warehouse environments, stressing dynamic path navigation, adaptive decision-making, and real-time monitoring. The system includes Q-learning-based reinforcement learning, SLAM for environmental mapping, GPS and LiDAR for sensor perception, and a FastAPI-based web interface for control and monitoring.

The study was motivated by the increased demand for automation in industrial surveillance and operational safety, particularly in large-scale warehouse facilities where manual monitoring is wasteful and time-consuming. This project developed a comprehensive approach for robots to patrol independently, adapt to environmental changes, avoid barriers, and respond to operational requirements with minimal human interaction.

5.2 Summary of Achievements

- **Developing an Adaptive Path Navigation System:** Using reinforcement learning (Q-learning), patrol robots may adjust their trajectories based on real-time environmental conditions. Unlike static routing, this adaptive method increased coverage and responsiveness.
- **SLAM and sensor technologies** were effectively integrated, allowing robots to produce and update warehouse maps in real-time. This, along with LiDAR and GPS, greatly enhanced localization and obstacle avoidance capabilities.
- **The web-based command interface**, powered by FastAPI, allows users to browse patrol tracks, monitor robot operations, manually set patrol waypoints, and review logs. This dashboard greatly improved user control and monitoring visibility.

- Edge computing allows for faster obstacle detection and route recalculation, decreasing reliance on cloud infrastructure.
- The system architecture was created modularly, with each team member adding unique capabilities such as energy-efficient data transmission, environmental health monitoring, and autonomous decision-making for coordinated robot action.

5.3 Research Significance

This study adds significantly to the improvement of autonomous robotic systems in warehouse settings. Technology demonstrated that AI-powered patrol robots could efficiently decrease the need for human involvement, improve safety, and maintain high operational uptime in constantly changing environments.

Furthermore, the integration of several autonomous subsystems demonstrated the possibility of scalability, since more robots could be added to extend coverage with minimum alterations, thereby supporting the idea of completely autonomous smart warehouses.

5.4 Limitations

While the technique displayed promising results, a few problems were identified:

- **Sensor Constraints:** The system relied on a restricted set of sensors (LiDAR, GPS, camera, light sensors), and the lack of proximity sensors limited its precision in complicated situations.

- **T Simulation vs. Real Environment:** testing in real warehouses would evaluate the system's robustness and reliability, as opposed to controlled simulations.
- **Communication Latency:** Edge computing has increased real-time reaction, but there is still some latency in data transfer and robot coordination, particularly in complicated obstacle scenarios, which requires additional optimization.

5.5 Future Work

The following areas are suggested for future exploration:

- **Advanced Obstacle Detection:** Use depth cameras or ultrasonic sensors to improve proximity detection and navigation precision in restricted environments.
- **Enhancements for Multi-Agent Collaboration:** Developing swarm intelligence algorithms can enable dynamic group patrol techniques among slave robots.
- **Enhanced User Interface Features:** Adding real-time video streaming, alert-triggered path re-routing, and automated reporting can boost monitoring efficiency.
- **Future generations could use cloud-based machine learning models to analyze patrol history and identify high-risk zones or anomalies in warehouse operations.**
- **Considerations for Commercial Deployment** include hardware durability, cost-effectiveness, and maintainability while transitioning from prototype to deployable commercial products.

6. References

- [1] T. M. Howard, C. J. Green, and A. Kelly, "State space sampling of feasible motions for high-performance mobile robot navigation in complex environments," *Journal of Field Robotics*, vol. 25, no. 6-7, pp. 325–345, 2008.
- [2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, MIT Press, 2005.
- [3] J. Kuffner and S. M. LaValle, "RRT-Connect: An efficient approach to single-query path planning," in *Proc. IEEE Int. Conf. on Robotics and Automation*, San Francisco, USA, 2000, pp. 995–1001.
- [4] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *Journal of Artificial Intelligence Research*, vol. 11, pp. 391–427, 1999.
- [5] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.
- [6] T. Edge, "Using FastAPI for creating modern web APIs with Python," *Real Python*, [Online]. Available: <https://realpython.com/fastapi-python-web-apis/> [Accessed: April 12, 2025].
- [7] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: Theory and Practice*, Elsevier, 2004.
- [8] L. Liu, J. Peng, and Y. Wang, "Multi-robot coordination for warehouse automation," in *Proc. IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2017, pp. 2022–2027.
- [9] A. Mohan, C. Ding, and K. Chakrabarty, "Sensor placement for fault detection and recovery in autonomous warehouse robots," *ACM Transactions on Embedded Computing Systems*, vol. 13, no. 4s, pp. 1–23, 2014.
- [10] T. Nguyen et al., "A Q-learning-based approach for autonomous path planning in smart logistics," in *Proc. Int. Conf. on Industrial Informatics (INDIN)*, 2020, pp. 1153–1158.

- [11]R. Siegwart, I. Nourbakhsh, and D. Scaramuzza, Introduction to Autonomous Mobile Robots, 2nd ed., MIT Press, 2011.
- [12]C. J. C. H. Watkins and P. Dayan, "Q-learning," Machine Learning, vol. 8, no. 3–4, pp. 279–292, 1992.
- [13]H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," IEEE Robotics & Automation Magazine, vol. 13, no. 2, pp. 99–110, 2006.
- [14]D. Silver, A. Huang, C. J. Maddison, et al., "Mastering the game of Go with deep neural networks and tree search," Nature, vol. 529, pp. 484–489, 2016.
- [15]M. G. Bellemare et al., "A Distributional Perspective on Reinforcement Learning," in Proc. of the 34th International Conference on Machine Learning (ICML), 2017.
- [16]S. Thrun, W. Burgard, and D. Fox, Probabilistic Robotics, MIT Press, 2005.
- [17]B. K. Singh and M. Kumar, "IoT based sensor data fusion for autonomous vehicle navigation using LiDAR and GPS," International Journal of Computer Applications, vol. 182, no. 45, pp. 30–35, 2021.
- [18]L. Kaelbling, M. Littman, and A. Moore, "Reinforcement Learning: A Survey," Journal of Artificial Intelligence Research, vol. 4, pp. 237–285, 1996.

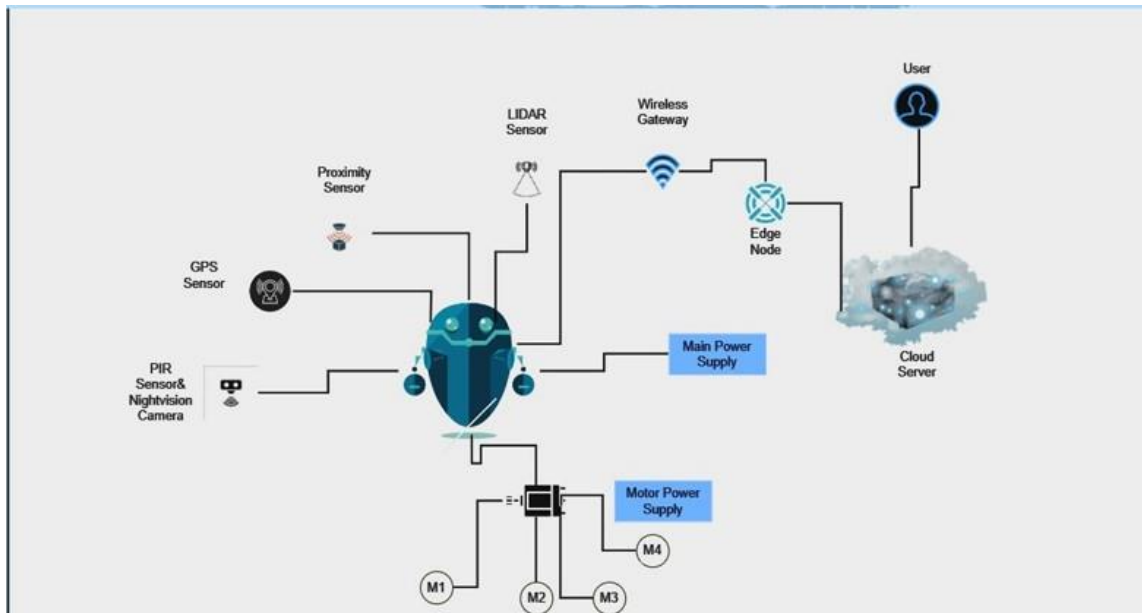
7. Glossary

- Simultaneous Localization and Mapping (SLAM):
 - A technique used by robots and self-driving cars to map an unknown environment while keeping their current position.
- Q-Learning:
 - reinforcement learning method that identifies optimal actions by calculating expected rewards for actions that differ from provided states.
- FastAPI:
 - Python web framework, that provides high performance and auto-generated interactive API documentation for API development.
- Waypoints:
 - physical points used as navigational references in autonomous systems.
- Edge computing:
 - involves processing data near the source, such as a robot, rather than cloud servers, resulting in faster reaction times.
- LiDAR (Light Detection and Ranging):
 - uses laser light to measure distance and create precise 3D maps of the environment.
- ROS (Robot Operating System):

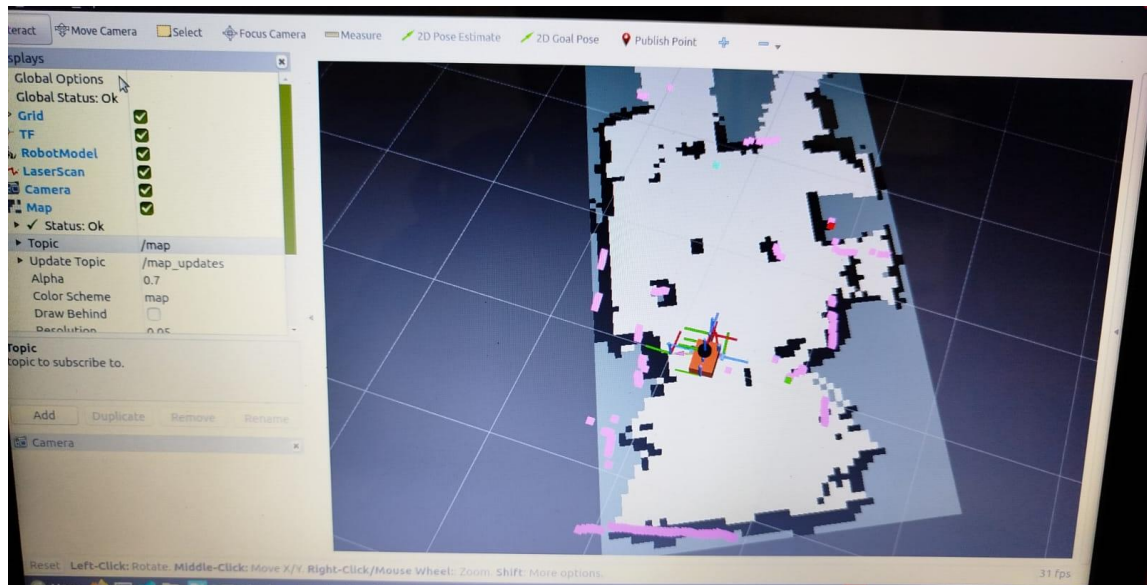
- An adaptive development platform for robot software, provides libraries and tools to develop robot applications.

8. Appendices

Appendix A: System Architecture Diagram



SLAM Map



Appendix B: Q Learning algorithm Code

```
for episode in range(epochs):
    # Random initial state selection from dataset
    row = df.sample(1).iloc[0]
    state = (row['x_position'], row['y_position'], row['light_intensity'],
            row['motion_detected'], row['obstacle_distance'], row['obstacle_present'])

    total_rewards = 0

    done = False
    for _ in range(steps_per_episode):
        # Choose action using epsilon-greedy strategy
        if np.random.rand() < epsilon:
            action = random.choice(action_space) # Explore
        else:
            action = action_space[np.argmax(q_table[state])] # Exploit best action

        # Get reward based on action taken
        reward = get_reward(row, action)
        total_rewards += reward

        # Get next state by sampling another row (simulate environment change)
        next_row = df.sample(1).iloc[0]
        next_state = (next_row['x_position'], next_row['y_position'], next_row['light_intensity'],
                    next_row['motion_detected'], next_row['obstacle_distance'], next_row['obstacle_present'])

        # Update Q-table using Q-learning formula
        q_table[state][action_space.index(action)] = q_table[state][action_space.index(action)] + learning_rate * (
            reward + discount_factor * np.max(q_table[next_state]) - q_table[state][action_space.index(action)]
        )

        state = next_state # Move to next state
```

Web Application

