Towards Sustainable Wireless Technologies: Automated Environmental Mapping Robotics

Abstract

The Automated Environment Mapping Robot (AEVMR) is an indoor mapping robot in mobile robotics. It combines hardware and software modules for navigation, data acquisition, transfer, and mapping. The setup includes a computer for 2D environment construction, a mobile robot collecting sensory data, and wireless communication for data exchange. The aim is to create a reliable architecture for self-driving robots. The paper introduces a multi-layered system for real-time monitoring and control, consisting of a primary layer on a Raspberry Pi 4 B+ managing communication and decision-making via ROS, a secondary layer on ATmega 328p for sensor data collection, and a remote layer using an API for data processing like point cloud generation, intersection detection, map creation, and vital monitoring. This system enables real-time control, historical data access, and efficient functions such as point cloud generation and map creation, enhancing its versatility for applications requiring monitoring and control.

Keyword: AEVMR (Automated environment mapping robot), Robot Operating System (ROS), Intelligent systems, Image Stitching, 3D environment, Sustainable Wireless Technologies, Environmental Mapping Solutions and Automated Sustainability

Authors

Akshat Singh
Vivekananda School of Information Technology
VIPS-TC
Delhi, India

Shubham Singh
Vivekananda School of Information Technology
VIPS-TC
Delhi, India

Harsh Pal
Vivekananda School of Information Technology
VIPS-TC
Delhi, India

Deepali Kamthania
Vivekananda School of Information Technology
VIPS-TC
Delhi, India

Nivedita Palia
Vivekananda School of Information Technology
VIPS-TC
Delhi, India

I. INTRODUCTION

Intelligent systems, exemplified by mobile robots, rely on a suite of capabilities including environment modeling, recognition, and planning to navigate and interact with their surroundings effectively, mirroring human cognitive processes. These systems are particularly invaluable in post-disaster scenarios, where they can assist in exploration and environmental assessment tasks [1]. Historically, in the early 1990s, the challenges of mobile robot navigation were addressed in separate silos, with distinct focus on environment mapping, localization, and trajectory planning [2,3]. Over time, it became evident that these problems are intricately linked, as the constructed map and the robot's pose are interdependent. Consequently, the approach evolved towards integrating mapping and localization, leading to methodologies such as Simultaneous Localization and Mapping (SLAM) or Concurrent Mapping and Localization (CML) [4,5]. Despite these advancements, deploying mobile robots in real-world settings presents a myriad of challenges. These include the need to adapt to constantly changing environments, which underscores the importance of robotic mapping. Additionally, the dynamic nature of scenes can cause visual recognition failures, posing a significant obstacle. Moreover, the unpredictable operating environment complicates robot planning processes [6]. To address these challenges, this paper introduces an integrated autonomous mobile robot system designed for exploring unknown environments. This system comprises three key components: map generation, path planning, and path tracking. By seamlessly integrating these functionalities, the proposed system aims to enhance the robot's ability to navigate and interact with its environment effectively, even in dynamic and unpredictable scenarios. The system navigates using ultrasonic/laser sensors, creating an occupancy grid to avoid obstacles and find optimal routes through genetic algorithms [7]. For effective use of agricultural robots in farms, accurate topological maps are crucial but often manually created. Heiwolt et al. [8] introduces an innovative method to automatically generate topological maps along crop rows using aerial images. The system's effectiveness is demonstrated through simulations and real farm aerial images, offering potential for improved agricultural robot deployment. Jamieson et al. [9] proposed an innovative approach to address the challenge of distributed semantic mapping by multiple robots in unfamiliar environments. In their solution, each robot learns autonomously using an unsupervised semantic model, complemented by a multiway matching algorithm. This algorithm facilitates the establishment of consistent matches between the learned semantic labels from different robots, thereby ensuring the creation of high-quality global maps with enhanced semantic understanding. Wang et al. [10] introduced a novel approach aimed at enriching robots' spatial awareness. Their method involves integrating Ultra-Wideband (UWB)-based distance sensing Internet of Things (IoT) devices into ordinary objects, effectively transforming the environment into a smart space with semantic labels. This integration enhances the robots' ability to navigate and operate within the environment. In practical terms, for mobile robots to effectively interact with the physical world, they must first create a virtual representation of their surroundings. Achieving this necessitates accurate localization, which is crucial for generating high-quality maps. Simultaneous Localization and Mapping (SLAM) techniques enable robots to observe their motion within unfamiliar environments while simultaneously constructing maps based on 3D points obtained through various sensors such as sonars, lasers, odometry, GPS, and video cameras [11]. The existing 3D scanning technology uses very expensive lasers thus they are not feasible [12]. In recent years due to their lack of portability, image-based 3D modeling has gained popularity. One of the most fundamental prerequisites for a totally autonomous mobile robot is robotic mapping.

These robots should be able to construct their own abstract structures, instead of using a pre-loaded map of the physical environment. The adaptive map is important for providing navigation information to the robot, allowing it to move around in the surroundings [13]. Deep networks with 3D convolutional operations, which are direct analogues of classical 2D operations, are used in traditional methods of 3D object generative modelling to learn volumetric predictions. However, when attempting to forecast 3D shapes, where information is concentrated mainly on the surfaces, these algorithms are computationally inefficient [14]. The 3D generative modeling framework demonstrates remarkable efficiency in producing object shapes represented as dense point clouds. By employing 2D convolutional operations, the framework can predict the 3D structure from various viewpoints while simultaneously integrating geometric reasoning through 2D projection optimization. This approach enables the reconstruction of 3D images with notable efficiency, emphasizing shape similarity and prediction density as key factors in the process [15]. The studies on city-scale outdoor scenes have shown promising results but they all lack a good amount of 3D perception, sense of material, and texture. Previously, researchers have worked in computer vision broadly, resulting in many method techniques and models for generating a scene from a set of multiple images, reconstruction of the 3D object using techniques like point cloud, and many more. In applications and navigation, it is difficult to build 3D maps of unknown environment. A low-cost Kinect sensor equipped on a mobile robot was used combining visual and depth information for dense point cloud alignment for capturing and processing the data [16]. Image stitching, also known as mosaicing, stands as a pivotal technology within the realms of computer vision and graphics. It addresses the intrinsic constraints posed by the field of view (FOV) in both images and videos, thereby enabling the creation of seamless and comprehensive visual representations. This transformative technology integrates multiple images or video frames to produce panoramic views or extended fields of vision, transcending the limitations of individual captures [17]. It combines two or more images of the same scene into one high resolution image called panoramic image. The stitching techniques extrapolates the regions of the panorama where there is information from a single image only [18]. The image stitching creates natural-looking mosaics free of artifacts due to relative camera motion, illumination changes, and optical aberrations [19]. A comprehensive method for fully automatic panoramic image stitching has been elucidated, leveraging an invariant feature-based strategy. This approach ensures robust matching of panoramic image sequences, even in the face of rotation, zoom, and changes in illumination across input images. By conceptualizing image stitching as a multi-image matching challenge, the method automatically identifies matching relationships between images. As a result, panoramas can be recognized within unordered datasets. Furthermore, employing multi-band blending techniques facilitates the generation of high-quality output panoramas, seamlessly integrating information from multiple source images [20]. In case of significant depth variation or object motion multiple registration have greater accuracy as it allows different depths to captured [21]. The work proposed previously does not extend to a full 3D scene while maintaining the original geometry using only visual inputs, having a versatile readable format. The research proposes a new mapping method and a general standard for a 3-dimensional environment to represent as data while maintaining the 3D geometry and structure of the scenes. The proposed work can be used in the areas of machine learning, artificial intelligence, 3D modeling, industrial application, and many more to understand better environment construction better.

This paper presents the design and implementation of a multi-layered system for real-time monitoring and control of a robot. The system is comprised of three layers: the primary layer, the secondary layer, and the remote layer. The primary layer, running on Raspberry Pi OS Lite on a Raspberry Pi 4 B+ device, manages tasks such as communication, task allocation, API calls, decision-making, and data collection and bundling using the Robot Operating System (ROS). The secondary layer, implemented on an ATmega 328p, collects sensor data and sends it to the primary layer for real-time processing. The remote layer uses an API to access and process data generated by the robot, including point cloud generation, intersection detection, map generation, and vital monitoring. The system allows for real-time monitoring and control of the robot and access to historical data. The communication bridge between the remote system and the robot is implemented using Python and the FastAPI web framework, which provides a secure and efficient way for the remote system to send commands to the robot and access archived data. Additionally, the API is protected by an encrypted key to ensuring that only authorized users can access the robot's data and control its functionality. The use of ROS in the primary layer allows for efficient communication and data sharing between different processes, improving the overall flexibility of the system. Furthermore, the Raspberry Pi 4 B+ device used in the primary layer is equipped with 4 GB of RAM and a Quad-Core 64-bit Broadcom 2711 Cortex A72 Processor, which optimizes performance by executing tasks in parallel and reducing response time. The use of ultrasonic sensors, IR proximity sensors, a gyroscope sensor, and an accelerometer sensor in the second layer allows for accurate and comprehensive data collection. The data collected is then parsed and archived in a database for later use. The Raspberry Pi continuously listens to the communication port byte and processes sensor data as soon as it receives it, which ensures that the robot responds quickly to sensor data and performs real-time processing.

II. PROPOSED FRAMEWORK

The proposed framework to enhance comprehension of 3D environments by preserving the geometry in a universally accessible format, while also achieving cost efficiency through reliance solely on visual inputs is shown in Fig. 1.

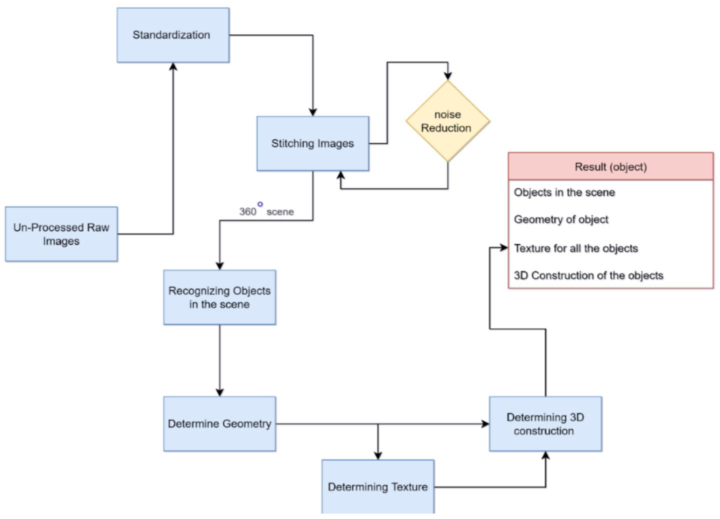

Figure 1: Proposed Framework of AVEMR

The true 3D environment is created by collecting multiple raw images of the scene and standardizing it in a standard form a 360° scene of the environment, using image stitching and reducing noise in the images like artifacts, unfilled gaps, non-satisfactory stitching of the images, and more. The process is continued until a satisfactory outcome is achieved. Further processing requires extracting data like recognizing an object, texture/material, and geometry of the objects. The fundamental process of image stitching involves working with a series of images captured from a consistent position. The camera is rotated around its optical center. The transformation between the second and first images is calculated.

The second image is translated to align and overlap it with the first image. The two images are combined by blending them to form a mosaic. In the case of additional images, proceed with the same process iteratively. When a satisfactory 360° image is achieved, then we try to detect all the objects in the image, edges, and more to understand the geometry, texture, and 3D construction of the image using their relevant techniques. By doing all this processing, all the data required to construct the 360° 3D image are created. The details of the steps involved in this process are as follows:

1. **Standardization:** Data collected from different sources can lead to conflict in further processing so a format is required in which all the data collected can be forged for better maintainability and readability by factoring images in the same file type and resizing images on the same scale

2. **Generating 360° Image:** After standardization, a simple 3D scene is created by stitching all the images together to create panorama, if you wrap the result in a spherical or a cylindrical wall from inside it will result in the mirage of a 360°scene.

3. **Object Recognition:** In this phase, we try to detect all the objects in a certain range using previous techniques. By finding objects in the images a 3D perception can be added to image in our next step.

4. **Determining Geometry:** In this phase, the geometry of the scene and objects in the scene are gathered. By measuring the distance between the object and the camera a point cloud is made to convert 2D scene to a 3D model/scene. The point cloud gives better depth. For accurate distance measurement, initial system calibration is necessary. Improved precision is achieved through the camera calibration, involving computation of both extrinsic parameters, used for world-to-camera frame conversion via rotation and translation matrices, and intrinsic parameters, encompassing internal camera aspects like focal length for pixel conversion.

5. **Determining 3D Structure and Texture:** In this part the 3D structure of the object in the scene is gathered using machine learning, the structure of the object using shadow is estimated, and more. The determination of the 3D structure in details and the texture and material in the scene helps to have better details of the environment and helps in better judgment.

III. METHODOLOGY

The robot system has been developed using a combination of various chips, technology, communications layer, and various sensors to manage various tasks such as communication, task allocation, API calls, decision-making, and sensory data collection. The robot has been trained in virtual reality using different indoor environments built using Blender and Unity to improve its performance and adaptability in real-world scenarios. The methodology used to develop and train the robot system involved several steps as follows:

1. The primary layer of the robot system has been developed using Raspberry Pi 4 and ROS (Robot Operating System) to manage various tasks such as communication, task allocation, API calls, decision-making, and sensory data collection.
2. Sensor data has been collected by connecting sensors such as gyroscopes, accelerometers, ultrasonic, and IR Proximity sensors to the robot. The data has been then sent to the Raspberry Pi for parsing and archiving for later use.
3. A communication bridge has been established between the robot and the remote system using Python and the FastAPI web framework. The API, built on top of FastAPI, provides a secure and efficient way for the remote system to send commands to the robot and access archived data for future processing and monitoring.
4. A training process in virtual reality has been implemented to improve the robot's performance and adaptability in real-world scenarios. This involved building different indoor environments using Blender and Unity, and using these environments to simulate real-world scenarios for the robot to learn from. The robot has been trained using these virtual environments to adapt to different lighting conditions, obstacles, and other factors that it may encounter in real-world scenarios.

In summary, this methodology has been effective in developing a robust and adaptable robot system that can perform well in real-world scenarios.

IV. ARCHITECTURE

The proposed system is made up of three layers: the primary layer, the secondary layer, and the remote layer. The primary layer, running on Raspberry Pi OS Lite on Raspberry Pi 4 B+, manages tasks such as communication, task allocation, API calls, decision-making, sensory data collection, and data bundling using ROS. The secondary layer, implemented on an ATmega 328p, collects sensor data and sends it to the primary layer for real-time processing. The remote layer uses an API to access and process data generated by the robot, including point cloud generation, intersection detection, map generation, and vital monitoring. The system allows for real-time monitoring and control of the robot and access to historical data.

1. **Primary Layer:** The primary layer is responsible for managing various tasks such as communication, task allocation, API calls, decision-making, sensory data collection, and data bundling. These tasks are executed and managed using Raspberry Pi OS Lite, a Linux-based headless operating system, on a Raspberry Pi 4 B+ device which is equipped with 4 GB of RAM, a Quad-Core 64-bit Broadcom 2711 Cortex A72 Processor. To optimize performance, the majority of tasks performed on the chip are executed in parallel, which helps to reduce response time. Additionally, sensors that are directly connected to the Raspberry Pi are managed through the use of ROS. This operating system works on a node-based system, where each node represents a separate process that communicates with other nodes. Each node is responsible for a specific function or task, such as reading sensor data, performing control or planning algorithms, or interacting with hardware devices. Nodes communicate with each other using a publish-subscribe model, which allows for efficient data sharing between different nodes. A node can publish data to a topic, and other nodes can subscribe to that topic to receive the data. This model allows nodes to communicate with each other without having to be aware of the specifics of how the other nodes are implemented, which improves the overall flexibility of the system. The Raspberry Pi is connected to the ATmega 328p using a USB serial interface and listens to the communication port byte. The sensor layer, implemented on the ATmega 328p, sends packets of data containing sensor readings to the Raspberry Pi. These packets contain values from a variety of sensors including a gyroscope sensor, an accelerometer sensor, an array of ultrasonic sensors, and an array of IR Proximity sensors. The data is parsed into a usable format by the Raspberry Pi and then archived in a database for later use.

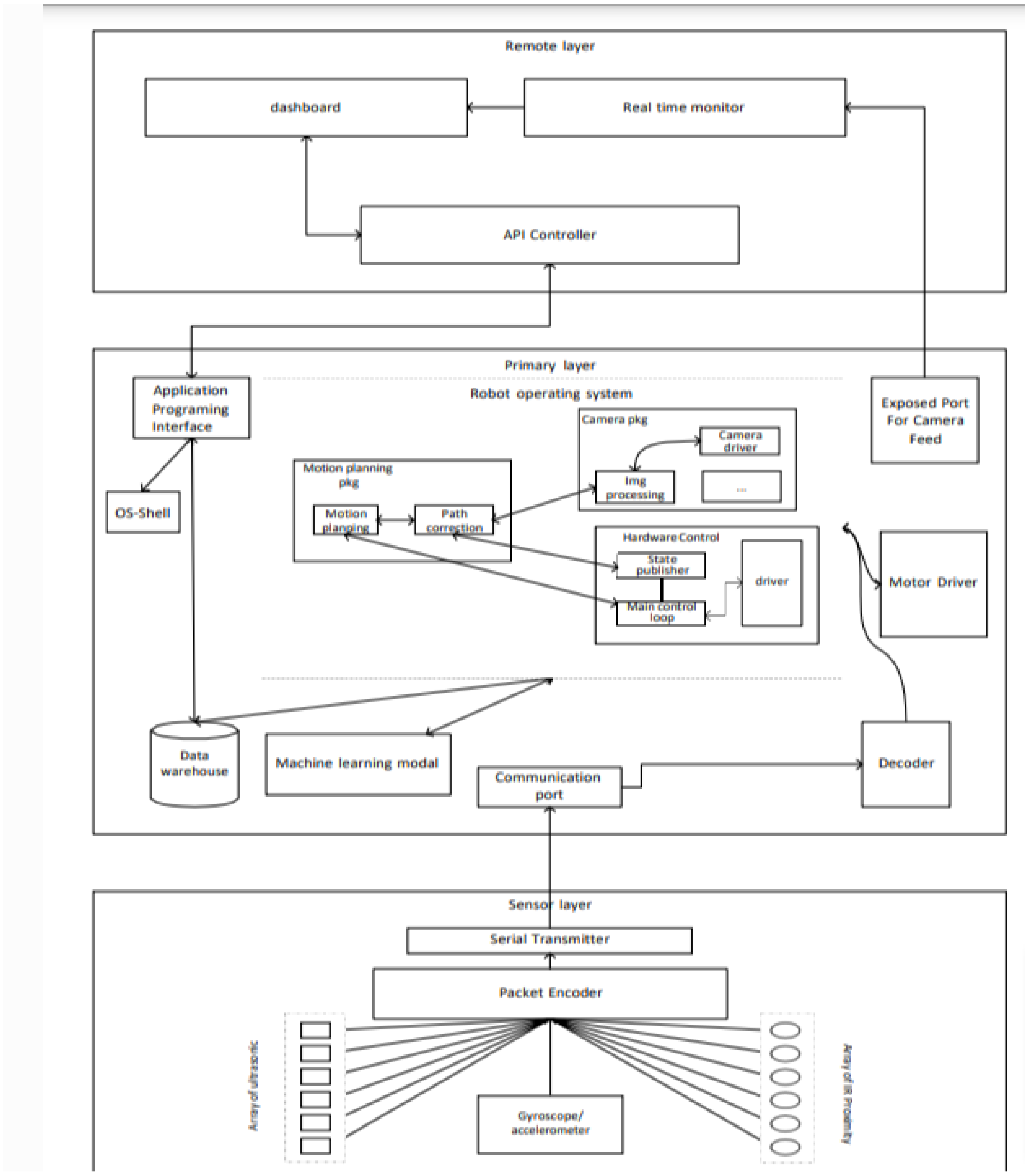

Figure 2: Details of Architecture

This setup allows for efficient communication between the sensor layer and the primary layer, enabling real-time data processing and decision-making. The Raspberry Pi continuously listens to the communication port byte and processes the sensor data as soon as it receives it, this ensures that the robot responds quickly to sensor data and performs real-time processing.

2. **Application Programming Interface (API):** The communication bridge between the remote system and the robot is implemented using Python and the FastAPI web framework. The API, which is built on top of FastAPI, provides a secure and efficient way for the remote system to send commands to the robot and access archived data from the robot for future processing and monitoring. The API runs on a dedicated thread to reduce response time and ensure real-time communication. Access to the API is protected by an encrypted key, which is required to be provided with each API request to ensure that only authorized users can access the robot's data and control its functionality. This added security feature helps to protect the robot and the data it generates from unauthorized access.

3. **Robot Operating System (RoS):** ROS (Robot Operating System) serves as an adaptable, open-source meta-operating system tailored specifically to the distinctive requirements of robotic systems. Offering a range of services, ROS facilitates hardware abstraction, precise control over low-level devices, seamless message-passing between processes, and efficient package management. At its core, the ROS runtime comprises a decentralized network of processes, each communicating through the ROS communication infrastructure. This infrastructure supports various modes of communication, including synchronous communication via services, asynchronous data streaming over topics, and centralized data storage on a parameter server. In the ROS ecosystem, nodes represent individual processes responsible for computation, offering modularity at a granular level. The ROS Master component handles name registration and lookup, while the parameter server serves as a centralized repository for storing data. Communication among nodes relies on messages, which are routed through a transport system featuring publish/subscribe semantics. Additionally, ROS supports services for request/reply interactions, defined by pairs of message structures: one for the request and one for the reply.

   - **Database:** Database is memory database Redis provides fast access, but can also persist all write to permanent storage to survive reboots and system failures. Redis provides three major advantages: it is versatile in memory data structures, enable building data infrastructure for real-time applications that require low latency and high throughput. Perfect for enhancing performance by storing database queries, handling intricate computations, managing API requests, and maintaining session states. The stream data type excels in facilitating rapid data ingestion, messaging, event sourcing, and delivering notifications. Each entry in the database stores all the necessary information about the state of the rover in a particular condition like timestamp, reference frame, image id, ultrasonic data, speed, and infrared data.

   - **Machine Learning Model:** A machine learning model has been trained using unsupervised learning techniques in a virtual environment to control the movement of a rover. The training process entailed gathering and preparing data from various sensors on the rover, such as a camera, ultrasonic sensors, IR Proximity sensors, gyroscopes, and speed, in real-time. This model can process the sensor data to make decisions on how to move the rover, making it able to navigate and operate autonomously without human intervention. The model can also be fine-tuned and retrained as needed to improve its performance over time. The visual input from the camera has been used to train the model for tasks such as object detection, obstacle avoidance, and localization. Ultrasonic sensor data was used to train the model to measure the distance to nearby objects, while IR Proximity sensor data was used to train the model to sense the ledges and pit holes in the floor. Gyroscope data has been used to train the model to measure the orientation and angular velocity of the rover. The speed has used as a parameter to train the model to control the movement of the rover. The trained model was then deployed on the rover to control its movement in real-time. The model processes sensor data and makes decisions on how to move the rover based on that data. The model can be fine-tuned and retrained as needed to improve its performance over time. This approach allows the rover to navigate and make decisions autonomously, without human intervention.

4. **Sensor Layer:** The RTOS layer of the system is responsible for communicating with all the sensors, except for the visual sensor (camera). This layer is implemented on an ATmega 328p, which provides faster response times compared to the Raspberry Pi. The primary function of this layer is to collect data from the sensors and send an encoded packet to the Raspberry Pi for further processing. The use of ATmega 328p, which is specifically designed for real-time applications, ensures that the system can respond quickly to sensor data and perform real-time processing. Three modules are being used for the data collection MPU6050 (three-axis gyroscope and accelerometer), Ultrasonic Array, and IR Array. The MPU6050 is a small, low-power, 6-axis motion tracking device that combines a 3-axis accelerometer and a 3-axis gyroscope. It is commonly used in applications such as robotics, drones, mobile devices, and gaming. The accelerometer measures linear acceleration in the x, y, and z axes, while the gyroscope measures angular velocity around the same axes. The MPU6050 can be used to determine the orientation, tilt, and movement of the device it is mounted on. The MPU6050 communicates with a microcontroller or processor via I2C or SPI communication protocols. It has a built-in Digital Motion Processor (DMP) which can be used to perform complex calculations such as sensor fusion, gesture recognition, and motion detection. The MPU6050 also features several power-saving modes and can operate at a wide range of temperatures. It is widely used in the field of robotics for attitude control, navigation, and localization. It's also used in mobile devices to track the device's position and orientation, and in drones for stabilization.

   - **Ultrasonic Sensor Array:** The sensor used is the HC-SR04 ultrasonic sensor to make an array of 6 ultrasonic sensors to collect distance data to avoid collision and obstacles in the paths. It works by emitting an ultrasonic sound wave and measuring the time it takes for the sound wave to bounce back to the sensor. Based on the time delay, the distance to an object can be calculated. The sensor consists of two main parts: a transmitter and a receiver. The transmitter sends out an ultrasonic wave and the receiver listens for the reflected wave. The distance to an object is calculated by measuring the time delay between the transmitted and received signals. The HC-SR04 can measure distances from 2cm to 400cm with an accuracy of 3mm. It can be easily interfaced with a microcontroller using the trigger and echo pins.

   - **IR Proximity Sensor Array:** An IR Proximity Sensor is a device that uses infrared technology to detect the presence of an object within a certain range. It works by emitting infrared radiation and measuring the amount of radiation that is reflected by the sensor. When an object comes within the sensing range of the sensor, it reflects infrared radiation to the sensor. The sensor then detects this reflected radiation and generates an output signal indicating the presence of an object. An array of IR Proximity sensors is being used to detect any ledges and potholes on the surface.

5. **Remote Layer:** The remote system in the network is responsible for processing data generated by the robot on demand using an API that is hosted on the primary layer. The API runs on a dedicated thread to ensure real-time data processing as well as access to archived data. The main task of the remote layer is to monitor the robot and send commands using the API. The API allows the remote system to request data from the robot on demand. The primary layer running on the robot processes the data and sends it back to the remote system for further processing. This allows for real-time monitoring and control of the robot, as well as the ability to access historical data for analysis and troubleshooting. The remote system can also access the robot's sensor data, status information, and historical logs to troubleshoot any issues that may occur during the robot's operation. Additionally, the remote system can also send commands to the robot for performing specific tasks, such as moving to a specific location, performing a specific action, or changing its operating parameters. This ability to send commands to the robot enables the remote system to control and manage the robot's operation remotely, making it a powerful tool for monitoring and controlling robots in various applications, such as industrial automation, logistics, and field services. It helps in point cloud generation, intersection detection, map generation and vital monitoring.

   - **Point Cloud Generation:** The system generates a point cloud from images captured by the robot's visual sensor (camera).
   - **Intersection Detection:** The system finds intersections in the point cloud to generate a 3D scene of the environment.
   - **Map Generation:** The system generates a map of the environment based on the point cloud and intersection data.
   - **Vital Monitoring:** The system monitors various vital parameters of the robot such as battery level, temperature, and other sensor data, to ensure the proper functioning of the robot.

6. **Detailed System Specification (Module Wise)**

   - **Module 1: Hardware**

     - The hardware module includes the physical components of the robot such as the ATmega-328p microcontroller, Raspberry Pi, ultrasonic sensors, IR proximity sensors, and motors.
     - The ATmega-328p microcontroller is responsible for controlling the robot's movements and sensor data collection. It is programmed in C/C++ and communicates with the Raspberry Pi through serial communication.
     - The Raspberry Pi acts as the main processor for the robot and is responsible for running the ROS (Robot Operating System) and Unity environments. It also communicates with the remote system through an API for real-time monitoring and control.
     - The ultrasonic sensors are used for spatial navigation and obstacle avoidance whilethe IR proximity sensors are used for detecting deformations in the surface.

- ➢ The motors are responsible for moving the robot's wheels and are controlled by theATmega-328p microcontroller.

- **Module 2: Software**

  - ➢ The software module includes the ROS and Unity environments, as well as the scripts and programs used for controlling the robot.
  - ➢ The ROS environment is used for controlling the robot's movements and sensor data collection. It also provides a framework for communication between the different components of the system.
  - ➢ The Unity environment is used for training the robot in virtual environments and simulating real-world scenarios. It also provides a platform for creating a 3Dvisualization of the robot's environment.
  - ➢ The scripts and programs used for controlling the robot are written in Python and C/C++. They are responsible for converting sensor data into actionable informationand controlling the robot's movements.

- **Module 3: Network**

  - ➢ The network module includes the communication protocols and infrastructure usedfor connecting the robot to the remote system.
  - ➢ The robot communicates with the remote system through an API using the HTTPprotocol. This allows for real-time monitoring and control of the robot.
  - ➢ The robot is also connected to the internet through WiFi or Ethernet, allowing forremote access and control.

- **Module 4: People**

  - ➢ The people module includes the users and operators of the system.
  - ➢ The users of the system include those who will be interacting with the robot, such as researchers and engineers. They will be responsible for controlling the robot andanalyzing the data collected by the sensors.
  - ➢ The operators of the system include those who will be responsible for maintaining and troubleshooting the robot. They will be responsible for ensuring that the robot is functioning correctly and making any necessary repairs or updates.

- **Module 5: Security**

  - ➢ The security module includes measures to protect the system from unauthorizedaccess and data breaches.
  - ➢ The system uses a secure login system for authentication and authorization ofusers.
  - ➢ Data is encrypted during transmission to protect against eavesdropping and tampering.
  - ➢ Regular backups are made to protect against data loss and disaster recovery.

- **Module 6: User Interface**

  - The user interface module includes the interface used for interacting with the robotand the remote system.
  - The interface has been thoughtfully crafted to ensure user-friendliness and straightforward navigation.
  - The interface provides real-time data visualization and control options for therobot.
  - The interface also provides access to historical data and reports for analysis anddecision making.

V. TRAINING AND TESTING

1. **Testing:** To test the performance of the proposed system, we used a simulation environment consisting of two scenes: a scene with multiple types of obstacles and a scene with a full track with areas turning 90 degrees and roads having elevated surfaces. The rover model was made in Solidworks and hexagonal in shape with sensors on each side making a total sensor count of 6. The rover can be controlled using "WASD" for basic movement and also includes two cameras, one for sensor viewing and the other for a 3D person view.

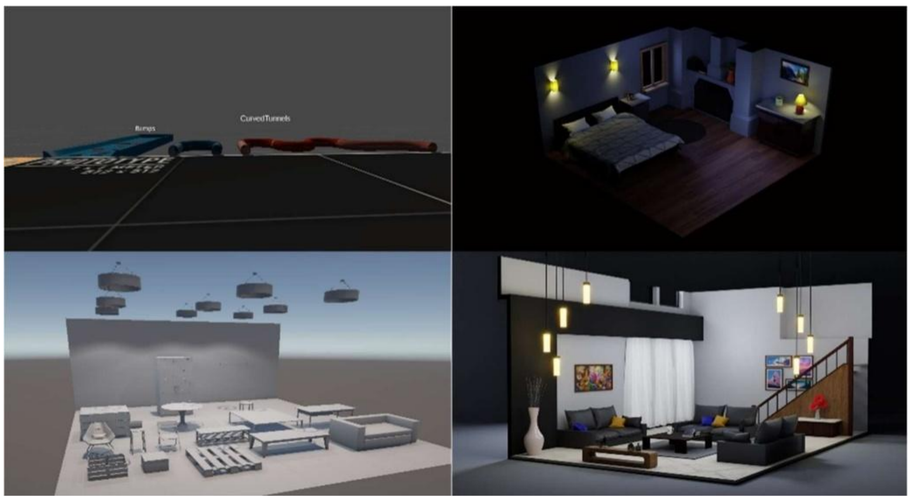

Figure 3: Simulation Screenshots

The rover was trained in virtual reality using the Unity game engine with the ML-Agents plugin, which allows for the use of machine-learning techniques to improve the performance and adaptability of the robot in real-world scenarios. The results of the training were recorded and analyzed using Tensorflow. Overall, the proposed system was able to provide a real-time, low-latency, and secure system with additional data redundancy.

2. **Training:** The rover's behavior is controlled using a script called "behavior parameters," which handles all input to the rover given by the machine. The "ray perception sensor 3D" script is also added to the rover, which detects objects in the environment using ray casting and provides information about the object that the ray hits. A "decision requester" script is also added, which controls the decisions made by the rover during each episode of the training. During the training, the rover is placed in the environment and is given specific tasks to complete, such as navigating through obstacles or following a specific path. The rover's actions are controlled by the machine, but a human can also intervene and take control of the simulation using heuristics, which bind the WASD keys to the motion of the rover.

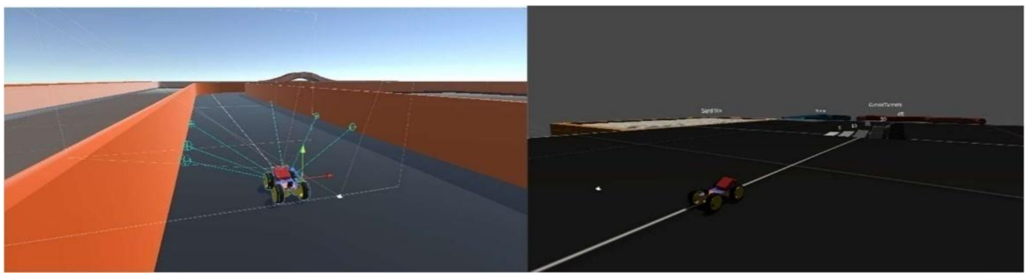

Figure 4: Training Screenshots

Rewards and punishments are given to the rover based on its actions and performance, and this information is recorded and used to adjust the rover's behavior.

$$Total\ Reward = scaled\left(weighted\big(AVG(DistanceFromWall) - rewards \times max(0, speed)\big)\right)$$

The training process can be monitored live using TensorFlow, which generates a graph representation of the rover's performance and progress. The training continues until the rover reaches a desired level of performance and is able to navigate the environment successfully.

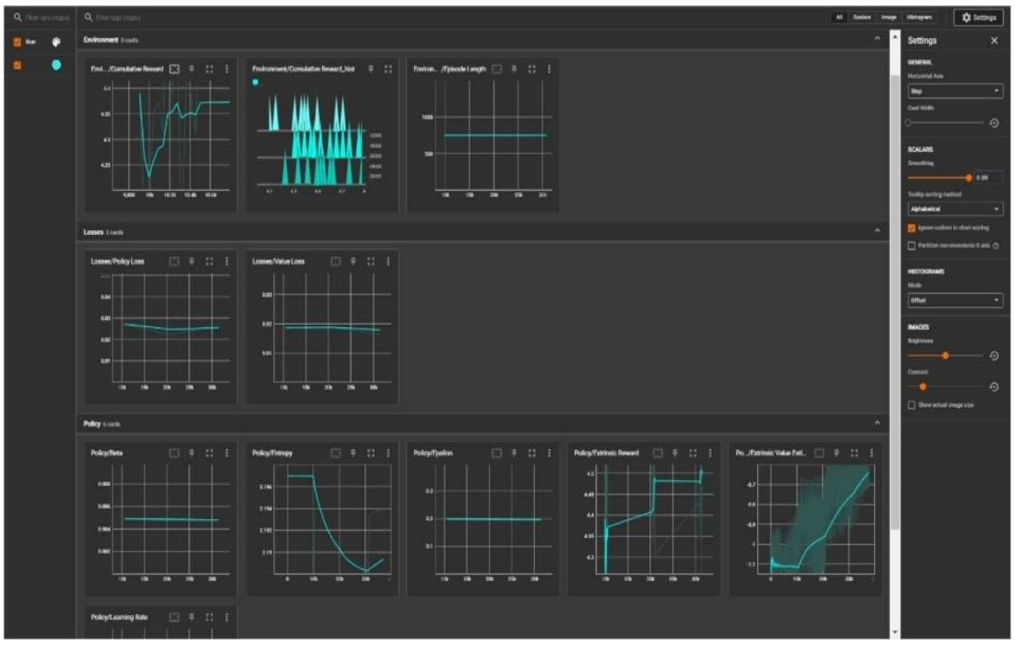

Figure 5: Tensorboard Screenshots

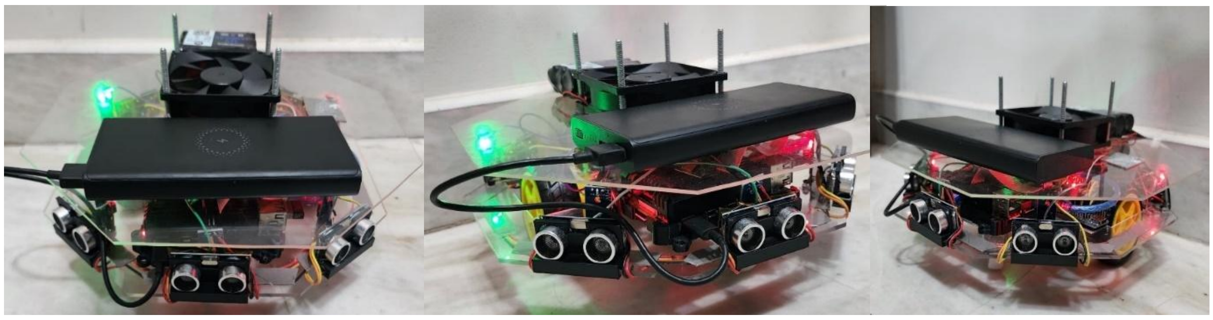

Figure 6: Model Front and Side Views

VI. APPLICATIONS

The proposed system can be applied in a variety of fields due to its ability to process sensor data in real time, generate point clouds, maps, and intersection detections, and access historical data.

1. **Automated Navigation:** The robot could use sensor data to navigate through an unknown environment and avoid obstacles. The system's ability to generate point clouds and maps can be used to create a 3D representation of the environment, which can be used for path planning and localization. The robot could also use the data from its sensors, such as ultrasonic and IR proximity sensors, to detect intersections and determine their location. This would allow the robot to navigate through an unknown environment without human intervention.

2. **Monitoring and Surveillance:** Another potential application of the system is in monitoring and surveillance. The system's ability to access historical data and monitor the robot's vitals in real-time makes it a valuable tool for surveillance and monitoring

applications. The robot could be used to monitor a specific area or object and alert a user of any suspicious activity. For example, the robot could be used to monitor a construction site and alert the supervisor if any workers are in danger or if any equipment is malfunctioning. This would increase safety and efficiency on the job site.

3. **Rescue Operations:** The system could also be used in search and rescue operations. The robot could use sensor data to navigate through debris and locate survivors in disaster-stricken areas. The robot could also be used to search for missing people in rugged terrain or other difficult-to-navigate environments. The robot's ability to generate point clouds, maps, and intersection detections would be particularly useful in these types of scenarios.

4. **Industrial Environments:** In industrial environments, the system could be used to automate tasks such as inspection, maintenance, and monitoring. The robot could use sensor data to navigate through an industrial environment, such as a factory or a power plant, and perform specific tasks. For example, the robot could be used to inspect machinery and equipment for wear and tear, check for leaks, and report any issues to the supervisor. This would increase efficiency and reduce the need for human inspection, which can be dangerous in some industrial environments.

5. **Agriculture:** In agriculture, the system could be used to automate tasks such as crop monitoring, soil analysis, and irrigation. The robot could use sensor data to navigate through a field and perform specific tasks, such as measuring the health of crops, analyzing the soil, and controlling irrigation systems. This would increase efficiency and reduce the need for human labor in the field. Additionally, the robot's ability to access historical data and monitor vitals in real-time would be particularly useful in agriculture as it can help farmers to make better decisions, increasing crop yield and reducing waste.

These are just a few examples of the potential applications of the proposed system. The versatility of the system means it can be applied in various fields depending on the specific requirements and capabilities. The system's ability to process sensor data in real time, generate point clouds, maps, and intersection detections, and access historical data makes it a valuable tool for a wide range of applications. The proposed system offers several advantages over traditional robotic systems. One of the main advantages is its ability to collect and process real-time sensor data, which allows for real-time monitoring and control of the robot. This allows for quick response to changes in the environment, improving the robot's ability to navigate and avoid obstacles. The system has ability to access historical data for analysis and troubleshooting. It stores data in a database, allowing for easy access and analysis of past performance. This can be useful for identifying and addressing any issues that may arise during operation. The system also provides a secure and efficient way for remote systems to access and control the robot. The API, which is built on top of FastAPI, is protected by an encrypted key, ensuring that only authorized users can access the robot's data and control its functionality. This added security feature helps to protect the robot and the data it generates from unauthorized access. Additionally, the robot was trained in virtual reality using different indoor environments, built using Blender and Unity, to improve its performance and adaptability in real-world scenarios. This allows the robot to be more flexible and adaptable in various environments, increasing its overall effectiveness.The system leverages ROS, an open-source meta-operating system designed specifically for robots. ROS enhances system flexibility by offering a suite of services including hardware abstraction, precise control over low-level devices, seamless message-passing between processes, and efficient package management. The proposed system addresses the issue of inefficient data management by using advanced technologies such as Raspberry Pi, ATmega-328p, ROS, and Unity to create a robot that can collect and process sensor data in real-time. This allows for efficient data management and the ability to easily modify the database.

VII. CONCLUSIONS

The proposed system architecture enables the development of robots for spatial navigation in various scenarios. The system's modularity and flexibility allow for easy modification to suit different use cases. By utilizing the Raspberry Pi 4 B+ and ROS in the primary layer, the system is able to provide real-time, low latency, and secure communication. Additionally, the use of an API and an encrypted key in the remote layer ensures that only authorized users have access to the robot's data and control its functionality. The implementation of a secondary layer, collecting sensor data, and a virtual reality training process in the system, allow for efficient data collection and robot adaptation to different environments. The system also provides additional data redundancy, ensuring that important information is not lost in case of system failure. Overall, the proposed architecture provides a reliable and efficient solution for spatial navigation in various scenarios.

REFERENCES

[1] Rosas, F.G., Hoeller, F., Schneider, F.E. (2022). Automated Environmental Mapping with Behavior Trees. In: Sgurev, V., Jotsov, V., Kacprzyk, J. (eds) Advances in Intelligent Systems Research and Innovation. Studies in Systems, Decision and Control, vol 379. Springer, Cham. https://doi.org/10.1007/978-3-030-78124-8_4
[2] Thrun, S., Burgard,W., Fox, D, Probabilistic Robotics; MIT Press Cambridge: Cambridge, UK, 2000.
[3] Davison, A.J. Real-time simultaneous localisation and mapping with a single camera. In Proceedings of the Ninth IEEE International Conference on Computer Vision, Nice, France, 14–17 October 2003; p. 1403.
[4] Durrant-Whyte, H.; Bailey, T. Simultaneous localization and mapping: Part I. IEEE Robot. Autom. Mag. 2006, 13, 99–110.
[5] Feder, H.J.S.; Leonard, J.J.; Smith, C.M. Adaptive mobile robot navigation and mapping. Int. J. Robot. Res. 1999, 18, 650–668.
[6] Sung-Hyeon Joo, Sumaira Manzoor, Yuri Goncalves Rocha, Sang-Hyeon Bae, Kwang-Hee Lee, Tae-Yong Kuc and Minsung Kim, Autonomous Navigation Framework for Intelligent Robots Based on a Semantic Environment Modeling, Applied Science, vol. 10, 3219, 2020; doi:10.3390/app10093219
[7] Karthikeyan U. Gunasekaran, Evan Krell, Alaa Sheta, Scott A. King, Map Generation and Path Planning for Autonomous Mobile Robot in Static Environments Using GA, The 8th International Conference On Computer Science and Information Technology, July 2018, DOI:10.1109/CSIT.2018.8486385
[8] Karoline Heiwolt, Willow Mandil, Grzegorz Cielniak and Marc Hanheide, Automated Topological Mapping for AgriculturalRobots, 3rd UK-RAS Conference for PhD Students & Early Career Researchers, Hosted virtually by University of Lincoln, April 2020.
[9] Stewart Jamieson, Kaveh Fathian ,Kasra Khosoussi, Jonathan P.How, Yogesh Girdhar, Multi-Robot Distributed Semantic Mapping in Unfamiliar Environments through Online Matching of Learned Representations, arXiv:2103.14805v1 [cs.RO] 27 Mar 2021
[10] Tianyi Wang,Ke Huo, Muzhi Han, Daniel McArthur, ZeAn, David Cappeleri, and Karthik Ramani, Autonomous Robotic Exploration and Mapping of Smart Indoor Environments With UWB-IoT Devices, 2020,Association for the Advancement of Artificial Intelligence
[11] Federico Bertolli Paolo Fiorini, Visual Slam – Mobile Robot LocalizationwWith Environment Mapping, IFAC Proceedings Volumes, Volume 39, Issue 15, Pages 286-291, 2006.

[12] Thrun, W. Burgard, and D. Fox. A real−time algorithm for mobile robot mapping with applications to multi−robot and 3d mapping. In Proc. IEEE Intl. Conf. on Robotics and Automation, volume 1, pages 321–328, 2000.
[13] Miguel Angelo de Abreu de Sousa and Andre Riyuiti Hirakawa, Robotic mapping and navigation in unknown environments using adaptive automata, Adaptive and Natural Computing Algorithms, C. Lemaître, C.A. Reyes, and J.A. Gonzalez (Eds.): IBERAMIA LNAI 3315, Springer-Verlag Berlin Heidelberg, pp. 562–571, 2004.
[14] Kun Qian, Xudong Ma, Fang Fang, Hong Yang, 3D Environmental Mapping of Mobile Robot Using a Low-cost Depth Camera, IEEE Conference on Mechanotronics and Automation, Japan, 2013.
[15] Chen-Hsuan Lin, Chen Kong, Simon Lucey, Learning Efficient Point Cloud Generation for Dense 3D Object Reconstruction, Computer Vision and Pattern Recognition, 2017.
[16] Kun Qian, Xudong Ma, Fang Fang, Hong Yang, 3D Environmental Mapping of Mobile Robot Using a Low-cost Depth Camera, Proceeding of 2013 IEEE International Conference on Mechatronics and Automation, August, Japan,2013
[17] LYU Wei, Zhong Zhou, CHEN Lang and Yi Zhou, A survey on image and video stitching, Virtual Reality & Intelligent Hardware 1(1):47, 2019.
[18] Ebtsam Ade, Mohammed Elmogy and Hazem M El-Bakry, Image Stitching based on Feature Extraction Techniques: A Survey, International Journal of Computer Applications 99(6):1-8, 2014.
[19] Chung-Ching Lin, Sharathchandra U. Pankanti, Karthikeyan Natesan Ramamurthy, and Aleksandr Y. Aravkin, Adaptive As-Natural-As-Possible Image Stitching, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1155--1163, 2015.
[20] Matthew Brown and David G. Lowe, Automatic Panoramic Image Stitching using Invariant Features.
[21] Charles Herrmann, Chen Wan, Richard Strong Bowen, Emil Keyder, Michael Krainin, Ce Liu, and Ramin Zabih, Robust image stitching with multiple registrations, European Conference on Computer Vision ECCV 2018, LNCS version.