ASEAN Engineering Journal

Full Paper

DEVELOPMENT AND PERFORMANCE ANALYSIS OF ORTHOGONAL SONAR ARRAY FOR AUTONOMOUS MOBILE ROBOT SLAM IMPLEMENTATION

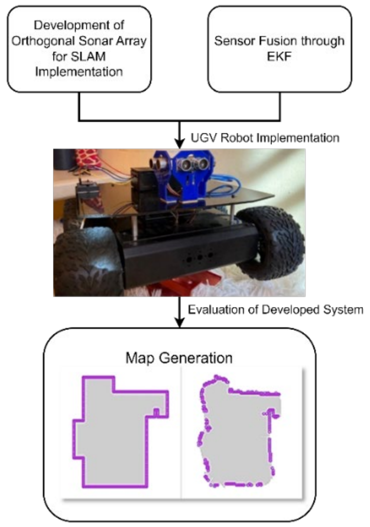
Timothy Scott Chu[a], Alvin Chua[a], John Anthony Jose[b], Edwin Sybingco[b], Candy Espulgar[c], Neil Justin Romblon[c], Emanuele Lindo Secco[d]

[a]Department of Mechanical Engineering, Gokongwei College of Engineering, De La Salle University, 2401 Taft Ave, Malate, Manila, 1004 Philippines
[b]Department of Electronics and Computer Engineering, Gokongwei College of Engineering, De La Salle University, 2401 Taft Ave, Manila, Philippines 1004
[c]Department of Software Technology, College of Computer Studies, De La Salle University, 2401 Taft Ave, Malate, Manila, 1004 Philippines
[d]Robotics Laboratory, School of Mathematics, Computer Science & Engineering, Liverpool Hope University, UK

Article history
Received
06 July 2023
Received in revised form
26 July 2024
Accepted
27 July 2024
Published online
30 November 2024

*Corresponding author
timothy.chu@dlsu.edu.ph

Graphical abstract


Development of Orthogonal Sonar Array for SLAM Implementation
Sensor Fusion through EKF
UGV Robot Implementation
Evaluation of Developed System
Map Generation

Abstract

Simultaneous Localization and Mapping (SLAM) research focuses on different techniques to develop efficient systems. Acoustic SLAM (aSLAM) is an alternative technique that is unrestricted from visual SLAM (vSLAM) limitations and is a cheaper than LiDAR SLAM. Nevertheless, current aSLAM implementations do usually require several units of ultrasonic range sensors which invalidate the advantages of aSLAM vs vSLAM. This study presents a novel aSLAM system where the number of ultrasonic range sensors is halved and combined with the possibility of varying the orientation angle between the sensors, providing a significant reduction of cost while preserving the performance. The paper presents an Orthogonal Sonar Array (OSA) setup of three sensors, which is a variation of the traditional aSLAM implementations (i.e. 6 sensors). This setup has been tested by generating a map representation of three experimental scenarios and comparing the results against a CAD model of the environment. The setup was able to successfully reconstruct the three environments with boundary accuracies of 72.91%, 77%, and 80.70% respectively. The least generated map has then been utilized as a reference to perform a path planning task and to validate the usability of the map generated from the OSA setup.

Keywords: AMR, aSLAM, Mobile Robots, SLAM, Simultaneous Localization and Mapping, Ultrasonic Range Sensors

1.0 INTRODUCTION

Technological innovations have fostered developments in robotic systems, including Autonomous Mobile Robots (AMRs). AMRs are mobile robots capable of traversing a defined space on their own or without direct input from humans. If the application space is undefined, AMRs employ the Simultaneous Localization and Mapping (SLAM) technique to generate a map of its environment and achieve autonomous operations.

SLAM allows a mobile robot to reconstruct its environment to a digital model and define its position through the use of appropriate sensors [1]. The type of sensor would depend on which SLAM technique is employed. Visual SLAM (vSLAM) utilizes monocular, RGB-D, or stereo cameras to generate pose estimates of the AMR and the obstacles to generate a model of the environment [2-4]; however, the vSLAM is sensitive to environmental lighting as highlighted in the study [5]. LiDAR SLAM is another method that utilizes laser beams (i.e. a LiDAR sensor), that has a higher resolution, longer range, and are insensitive to environment illumination, to execute SLAM [6]. However, LiDAR SLAM tends to be an expensive solution as shown in the studies [7-9]. Acoustic SLAM (aSLAM) makes use of microphone arrays to localize the sound waves transmitted by the source and generate a map [10]. Thus, the technique would function despite poor environmental lighting conditions but would often require several sensor units.

1.1 Background

Previous studies on a SLAM, such as [11-13], conducted SLAM experimentation that makes use of acoustic sensors: for example, in [11] the authors presented an autonomous terrestrial robot that possesses similar features to that of a bat using a set of bio-sonar sensors from the DJI Ronin gimbal. After performing the mapping sequence, the researchers found that the generated map was not very accurate, and a misleading factor was the presence of soft objects such as plants which caused errors in the estimation of the object borders. Similarly, papers [12-14] investigated the use of ultrasonic sensors for mapping applications and highlighted that the use of multiple ultrasonic sensors is necessary to compensate for the sensor's limited field of view (FOV). In addition, both studies highlighted the tradeoff between the map quality and the time of mapping. A better map quality often requires a more detailed mapping sequence, which, in turn, would take more time to be acquired.

An attempt to escape from this limit and produce a better map quality is to employ a sensor fusion strategy with 2 or more sensors, instead of being limited to only one. Studies have explored a sensor fusion SLAM implementation using several acoustic sensors combined with encoders, and Inertial Motion Units (IMU), such as in [15-16]. Both these studies were able to achieve a robust 2D map reconstruction while utilizing at least 6 ultrasonic range sensors. The motivation for this methodology is to address the FOV limitation of aSLAM while generating a working map in a reasonable time.

The common objective for the optimization of a SLAM system is clearly to generate a workable map with a balance between the cost and its performance, as evidenced in [15-16]. In this context, one SLAM technique of interest is the aSLAM, since it addresses the limitations of vSLAM and, at the same time, it relies on acoustic sensors which are generally cheaper than a LiDAR sensor. However, current studies on aSLAM often employ several acoustic sensors which partially invalidate these cost advantages. The study in [13] utilized a sonar ring controlled by an Intel NUC. The sonar ring is an assembly of six ultrasonic range finders in a hexagonal configuration. However, the paper highlights that engaging all six sensors causes interference which affects their data. Their findings highlighted that engaging 2 adjacent sensors is the primary cause of this problem hence they opted to engage one ultrasonic range finder at a time in a certain sequence that skips the adjacent sensor. From this observation, this paper aims to present the Orthogonal Sonar Array (OSA), a system that offers a set of advantages through a few acoustic sensors vs previous work [13,17], specifically addressing the interference concern that was highlighted. The paper also investigates the performance of the OSA through a 2D map reconstruction and then employs path planning based on this map.

2.0 THEORETICAL CONSIDERATIONS

2.1 Orthogonal Sonar Array (OSA)

The Orthogonal Sonar Array (OSA) refers to the positioning and orientation of multiple acoustic sensors. In previous work, authors explored the use of 3 ultrasonic range sensors which were oriented at 90° from each other as shown in Figure 1; other authors presented 6 units that were oriented at 60° as it is reported in [15-16].

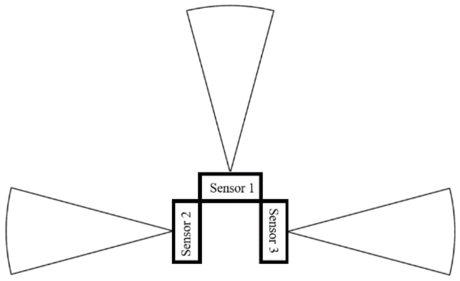


Figure 1. Orthogonal Sonar Array configuration with 3 sensors

The aforementioned articles on aSLAM made use of 6 ultrasonic range sensors - angled at 60° - to achieve a 360° FOV. Increasing the number of sensors would also increase the number of inputs of any algorithm, such as, for example, a sensor fusion algorithm. Therefore, rather than increasing the number of sensors, some authors leveraged the kinematics of the mobile robot and changed the device orientation to obtain the necessary data, as shown in Figure 2.

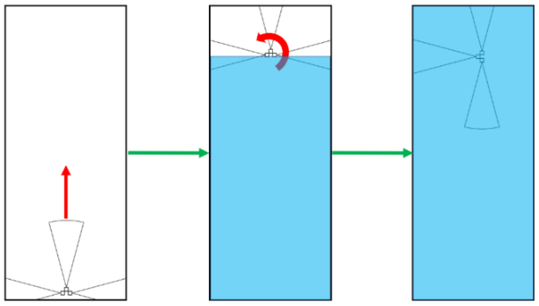


Figure 2 Data Gathering with AMR Motion. Blue Represents the area in which data has been collected.

2.2 Data Fusion Framework

For SLAM implementations to reconstruct a map, data must be obtained from multiple sensors, processed, and finally updated during the overall process. Figure 3 shows the framework of how the AMR collects and processes data to reconstruct its environment in a 2D map representation. The AMR navigates around its environment and as it navigates the sensor network – which is composed of the ultrasonic range sensors, IMU, and wheel encoders – it continually collects observations from its surroundings, which are then collated and processed using the Extended Kalman Filter (EKF), and finally recorded. A map estimation can be generated by stitching these data from the history of the observations by the AMR. The study employed a Raspberry Pi-Arduino Interfaced controller for the operation and data collection of the AMR, as shown in Figure 4.

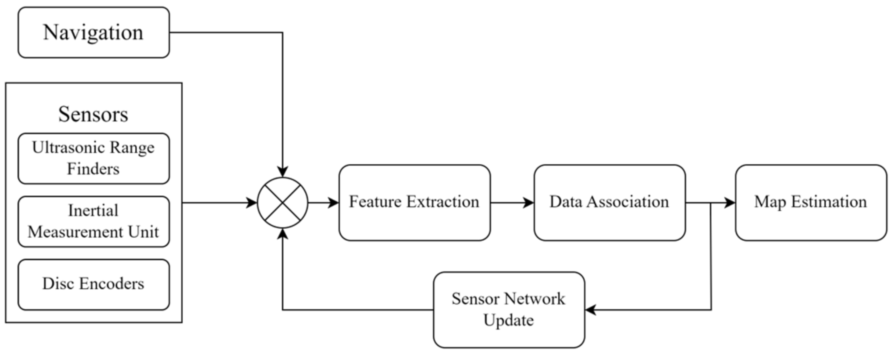


Figure 3 Data Fusion Framework, modified from [18]

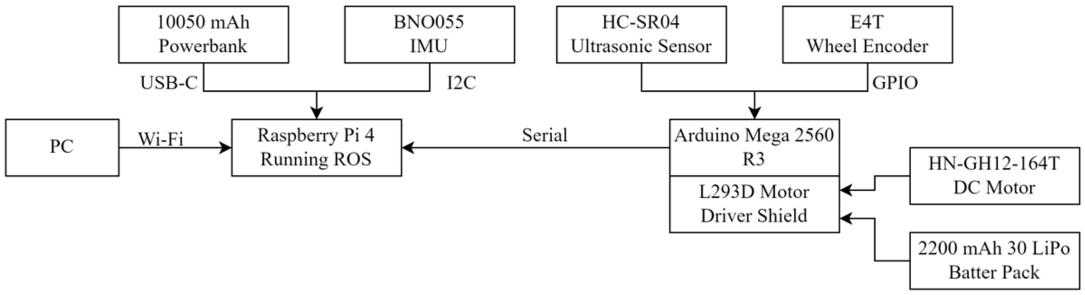


Figure 4 AMR Hardware Architecture

2.3 Robot Operating System

The study used a robotic infrastructure called Robot Operating System (ROS), specifically the ROS Melodic distro. The ROS architecture can seamlessly facilitate the communication of various electronics through nodes and topics. Nodes are executable files associated with sensor or robot functions that can publish or subscribe to a topic. A topic is a data-sharing space that contains information, called messages, from one node, that can be accessed by another node. The study utilized two primary packages, the robot_localization package, and the GMapping package. The former package collects odometry and IMU information to localize the robot's position in a defined space, while the latter utilizes laser range finders and odometry data to produce a 2-dimensional map. Given that the study makes use of Ultrasonic Range Finders, additional configuration was done to ensure compatibility. Another mapping package from ROS is hector_slam, however, this package makes use of Laser Scans to gather 3D data points for map generation and is more suitable for robotic systems that utilize lidar, rather than ultrasonic range finders. Some ROS functionalities such as rosbag and rviz were used to gather data and create a visualization of the SLAM implementation.

2.4 Extended Kalman Filter

The Kalman Filter (KF) is an iterative process to produce state estimates and data fusion for certain applications, such as map generation for this paper. It starts by initializing a state prediction estimate based on system-associated uncertainties and current variables followed by updating the predicted estimate by comparing it with the actual measured values. A key parameter for this algorithm is the Kalman Gain calculation which influences the weights on the prediction and measurement in updating the state estimate as shown in equation 1. The Kalman Gain is mathematically represented in equation 2.

$$x'_t = x_t + K(y - Hx_t) \quad (1)$$

$$K = \frac{P_t H}{H P_t H^T + R} \quad (2)$$

Where *x* is the state vector, *t* is the time step, *K* is the Kalman Gain, *P* is the covariance matrix, *H* is the measurement matrix, H^T is the transposed measurement matrix, and *R* is the measurement of the covariance matrix.

As the name suggests, EKF serves as an extension to the traditional Kalman Filter where it addresses the latter's limitation in handling linear systems. It has a similar process to traditional KF, but an additional process called *linearization* is introduced. Due to the multimodal nature of acoustic sensors, this study employed EKF with the inputs shown in Figure 3.

2.5 Mapping

The data obtained from the Data Fusion Framework are fed to the GMapping Algorithm for further processing to generate a 2D map representation of its environment. This algorithm is considered to be the efficient variation of the Rao-Blackwellized particle filter to generate map grids from laser range finder data [13]. The algorithm would always refer to the pose estimation from the odometry data of the wheel encoders and compounds the pose updates during operation represented in equation 3, where x represents the robot's pose and ω represents odometry data.

$$x'^{(i)}_t = x'^{(i)}_{t-1} + \omega_{t-1} \quad (3)$$

First, the peak values in the target distribution are used to simulate the proposed distribution shown in Equation 4 and the Normalization Parameter, η, is calculated through Equation 5.

$$x_t^{(i)} \sim p\left(x_t \middle| m_{1t-1}^{(i)}, x_{1t-1}^{(i)}, z_{1t}, u_{1t}\right) \tag{4}$$

$$\eta = \sum_{j=1}^{K} p(z_t | m_{t-1}^{(j)}, x_j) \tag{5}$$

The Normalization Parameter is calculated for the Gaussian EKF proposal, μ, for the collection and updating of the environmental data, which are presented in Equations 6 and 7.

$$\mu_t^{(i)} = \frac{1}{\eta} \sum_{j=1}^{K} x_j p(z_t | m_{t-1}^{(j)}, x_j) \tag{6}$$

$$\sum\nolimits_t^{(i)} = \frac{1}{\eta} \sum_{j=1}^{K} p(z_t | m_{t-1}^{(j)}, x_j)(x_j - \mu_t^{(i)})(x_j - \mu_i^{(i)})^T \tag{7}$$

where:

p	- Posterior Probability	z	- Environment Information
x_t	- Robot Trajectory	m	- Decomposed Map
u	- Odometer Reading		

2.6 Path Planning

Path planning is defined as the capacity of the AMR to generate an optimal path from point to point while avoiding potential static obstacles [19]. One algorithm for path planning is called the A* search algorithm or A* algorithm. This is a graph-search algorithm that is primarily employed on a grid. It is similar to Dijkstra's algorithm, but it is particularly efficient for single-node to single-node queries [20-21].

The process makes use of a heuristic function that estimates the cost to reach the goal node from the start node. The algorithm utilizes a priority system, where new nodes or lower-cost nodes are given priority or are examined first. This queue allows the algorithm to compute the shortest path in less time than Dijkstra's algorithm. Figure 5 shows the flowchart of the A* Algorithm.

3.0 METHODOLOGY

There are two main phases in the development of an AMR (as illustrated in Figure 6), namely:

- the set-up phase, where the robot itself is assembled, programmed, and calibrated; and
- the experimental phase, wherein the AMR underwent testing to evaluate its performance. The discussion of each subprocesses will be discussed in the succeeding subsections.

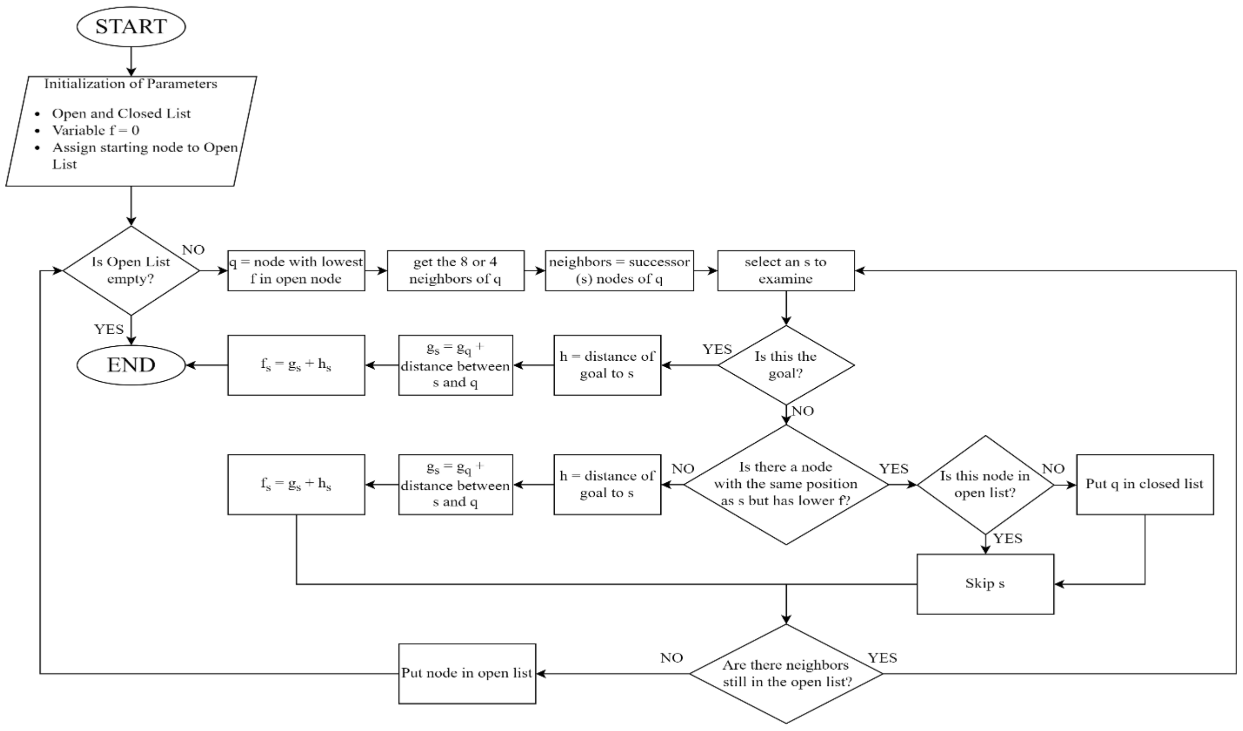

START
Initialization of Parameters
Open and Closed List
Variable f = 0
Assign starting node to Open List
Is Open List empty?
NO
YES
END
q = node with lowest f in open node
get the 8 or 4 neighbors of q
neighbors = successor (s) nodes of q
select an s to examine
Is this the goal?
YES
NO
h = distance of goal to s
g_s = g_q + distance between s and q
f_s = g_s + h_s
Is there a node with the same position as s but has lower f?
NO
YES
Is this node in open list?
NO
YES
Put q in closed list
Skip s
h = distance of goal to s
g_s = g_q + distance between s and q
f_s = g_s + h_s
Are there neighbors still in the open list?
NO
YES
Put node in open list

Figure 5. Flowchart of A* Algorithm

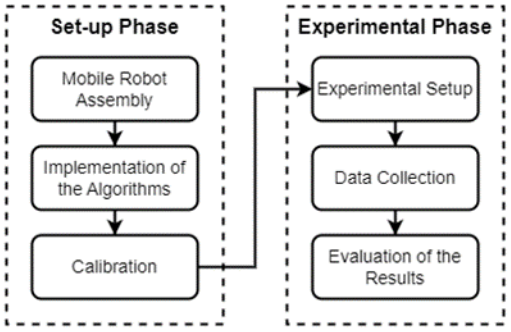


Figure 7. Methodological Framework with the set-up and experimental (or validation) phases

3.1 Mobile Robot Assembly

The proposed AMR (as illustrated in Figure 7) is a four-wheel-drive mobile robot that makes use of ultrasonic sensors, an IMU, and wheel encoders to perform SLAM.

The Lynxmotion A4WD1 is used as the main body of the robotic system. The microcontrollers are a Raspberry Pi 4b and an Arduino Mega that commands the other components, including 3 ultrasonic ranging sensors, 2-wheel encoders, an IMU, and two DC motors that actuate 2 motorized wheels.

The 3 ultrasonic ranging sensors are arranged using the OSA attached to the front side of the robot. The 2-wheel encoders, one for each side (i.e., left and right), are used to obtain rotational motion data. The IMU is attached to the baseplate for the determination of the robot's position. The wheel encoders, along with the IMU, are also used to gather odometry data. All sensors except for the IMU are connected to the Arduino Mega, while the IMU is connected to the Raspberry Pi as it requires a higher capacity for the processing of the complex data it collects.

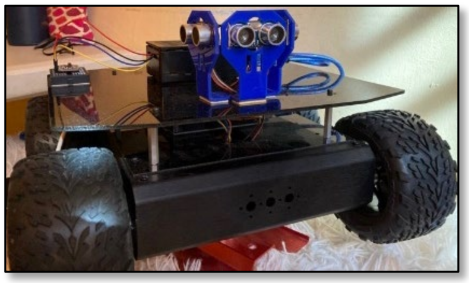


Figure 6. The Autonomous Mobile Robot

3.2 Calibration

Before the AMR system can be evaluated, the system has to be calibrated to ensure that minimal issues – from factors external to the aSLAM implementation – may arise. The system is calibrated in two stages (1) wheel encoder calibration and (2) ultrasonic range finder calibration.

To get an accurate measurement of the distance, which is indirectly measured through the 2 wheel encoders, 2 tests were performed: a manual calibration and an operational calibration. For both tests, a tape measure is secured onto the ground while the robot is to travel for a 1 m distance multiple times (Figure 8). The number of generated ticks for that distance is then recorded. Manual calibration involves pushing the robot physically, while operational calibration involves programmatically instructing the robot to travel for 1 m. After multiple trials, the researchers recorded an average of 8342.3 ticks per meter for the manual test, and an average of 8235.1 ticks per meter was recorded for the operational test. Finally, 8300 ticks per meter were set as a reference for a 1 m distance.

Figure 8. Wheel Encoder Test

The implementation of the ultrasonic sensors allows for pings detected at and above a certain distance to be ignored. A variety of ranges were tested to determine which maximum distance is optimal for the GMapping algorithm. Figure 9 shows the GMapping results of the different ultrasonic sensor ranges tested: it was noticed that a range below 0.5 m gives a less complete map since most of the obstacles were ignored due to the limited range. The two viable ranges to compare are 0.65 m and 1 m. To definitively decide on the maximum range for the ultrasonic sensors, it was decided to compare the boundary accuracy (a metric to be discussed in Section 3.6) between the 1 m and 0.65 m generated maps. It was found that the 0.65 m maximum threshold produced a higher boundary accuracy, at 88.02%, rather than the 1 m maximum threshold (72.41%). Therefore a 0.65 m range for the ultrasonic ranging sensors was adopted.

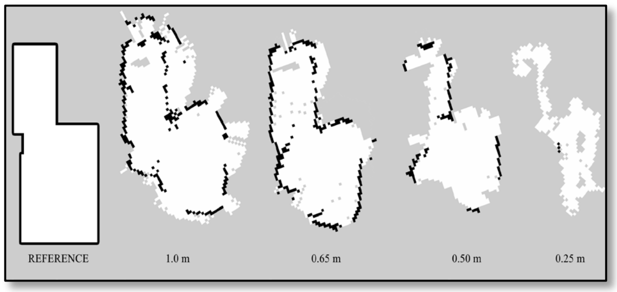


Figure 9. Mapping at Varying Ultrasonic Sensor Ranges

3.3 Implementation of Algorithms

The SLAM algorithm is designed in the Robot Operating System (ROS)'s GMapping algorithm, which is executed in the Raspberry Pi 4b. The IMU data are directly transmitted to the Raspberry Pi board, which is remotely and wirelessly connected to a computer, where the robot's command can be set.

On the other hand, the ultrasonic sensors and the wheel encoders are connected to the Arduino Mega board, which communicates these data to the main Raspberry Pi board. The

Arduino Mega is also responsible for controlling the 2 motors. Pathfinding instructions are finally produced by the Raspberry Pi board and then sent to the Arduino Mega board for execution.

3.4 Experimental Setup

The AMR is evaluated in terms of its mapping quality and navigation performance on its generated maps. In both experiments, the robot is extensively operated in three 10-20 square meter static environments. Three different layouts were tested to get an in-depth look at the robot's flexibility and performance.

The first environment (environment 1) is a large living room with an open space leading into a narrow hallway. This location was chosen to test whether the robot could map and navigate through a simple open area (Figure 10).

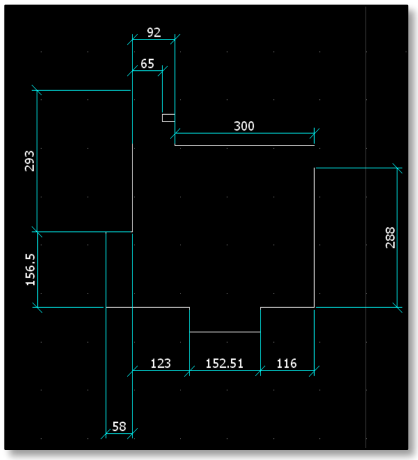


Figure 10. Floor Plan of Environment 1 (measurements are reported in cm)

Environment 2 features two rooms connected by a narrow 0.41 m hallway. An obstacle - a small coffee table - is also present in the middle of one of the rooms. The environment was selected to test how well the robot can map narrow environments and map the areas with some obstacles within (Figure 11).

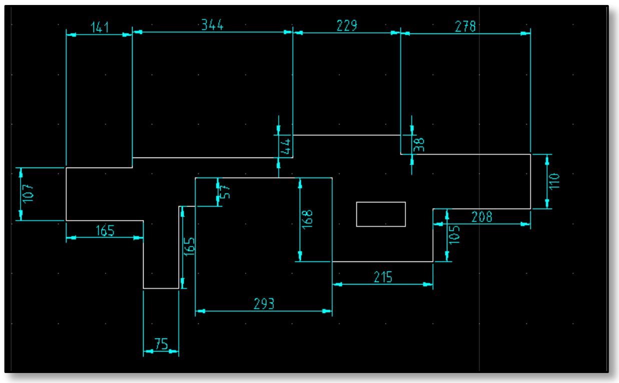


Figure 11. Floor Plan of Environment 2 (measurements are reported in cm)

Finally, Environment 3's main feature is an L-shaped hallway with no obstacles within. This environment was adopted to test how well the robot could map and navigate through areas with sharp corners and hallways. Figure 12 illustrates its floor plan.

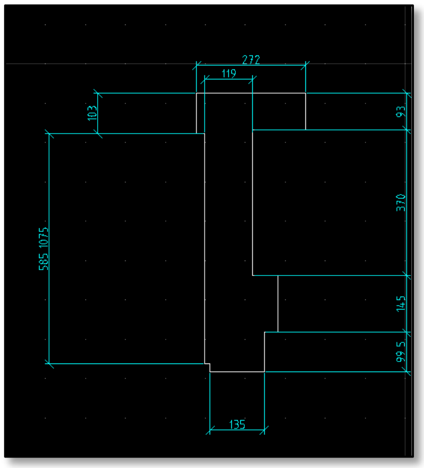


Figure 12. Floor Plan of Environment 3 (measurements are reported in cm)

3.5 Data Collection

As was previously mentioned, the robot is evaluated for its mapping quality and its navigation performance. The succeeding subsections detail how the data are collected for each criterion.

For mapping, the robot is left in the environment to automatically explore and map the area and its boundaries. At the beginning of each trial, the robot is placed at the bottom of the leftmost side of the environment. The robot's goal destination would then be the topmost portion of the rightmost side of the open space.

To explain and visualize how the experiment is conducted, Figure 13 shows a sample testing environment that has a dimension of 7 meter x 4 meter. The cell area is divided into 1 meter x 1 meter, representing the occupied and unoccupied spaces within the room. The AMR will initially start at the bottom left corner of the room (tile A in the figure) and must traverse towards the grid at the top right corner of the room (tile B) while avoiding the obstacles and taking the path with the shortest possible distance. If the AMR stops in a position whose center is close (or exactly positioned) to B, with minimal collision, the result of the trial is considered as successful. However, in the actual data gathering, the cell area of the applied grid map is set to be 0.05meter x 0.05 meter.

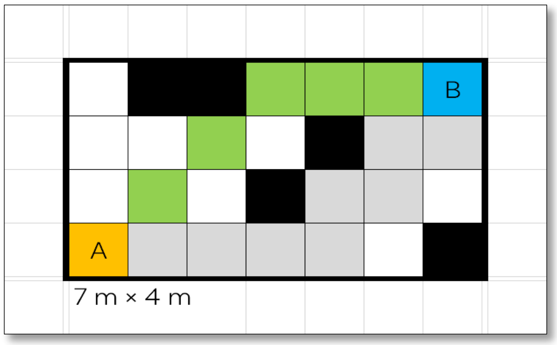


Figure 13 Sample Environment

To gather the data for evaluating the AMR system's mapping accuracy, the length of each wall was measured and then sketched on a CAD software for comparison vs the real map. The generated map from the GMapping algorithm was then imported into MATLAB as a matrix, where each cell could have the following values: -1 for unknown space, 0 for free space, and 100 for occupied space. Figure 14 shows the generated map, where the unknown space is reported in grey color, the free space is in white, and the occupied space is in red.

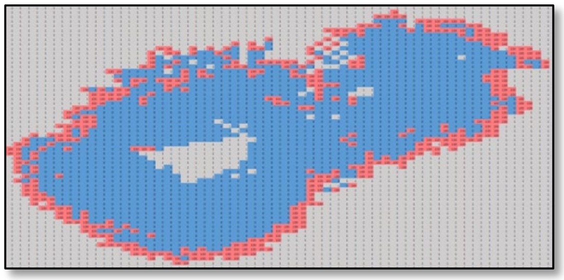


Figure 144. MATLAB Generated Map Visualization

To evaluate the AMR's navigation performance, the robot is instructed to traverse from point A to point B. The actual path traveled by the robot is then compared with a simulated path produced by the A* algorithm from MATLAB, on the generated map by the GMapping algorithm.

The position and distance traveled by the robot are obtained through the odometry data generated by the EKF sensor fusion of the wheel encoders and the IMU. The data generated by the EKF produced hundreds of samples per second. As such, the researchers opted to retrieve the x and y coordinates of the robot at each time interval.

3.6 Evaluation of Results

The metric used for evaluating mapping quality is boundary accuracy, which is presented in the Equation 6:

$$BoundAccuracy, \% = \left(1 - \frac{\#\,of\,cells_{occupied} \times 0.05\,{}^{m}/_{cell} - Boundary_{actual\,map}}{Boundary_{actual\,map}}\right)$$

The boundary of the actual map refers to the length of the perimeter of the actual map. The generated map's resolution is 0.05 m/cell, which means that a cell distance between 2 points is 0.05 m as well. This metric was used to numerically evaluate the mapping quality of the system.

For evaluating the navigation performance, it was measured the total distance travelled from point A to point B between the actual path traveled and the simulated path from the A* algorithm. The difference between the total distance of the two paths is then calculated.

4.0 RESULTS AND DISCUSSION

4.1 Mapping

The detected boundaries from the AMR's generated maps are compared against their corresponding environment maps. The boundary accuracies for all three testing environments are shown in Table 1.

Table 1. Mapping results for testing environments

Environment	Actual Boundary [m]	Generated Boundary [m]	Accuracy %
1	18.38	13.40	72.91
2	29.67	26.53	77.00
3	20.85	24.68	80.70

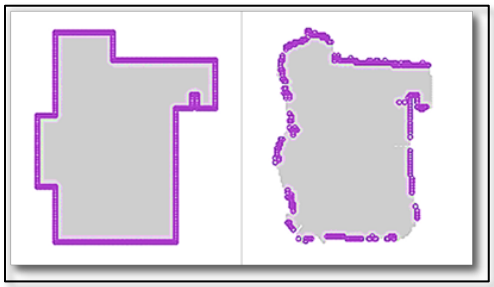


Figure 16. On the left and right panels, the measured and generated map

The AMR achieved an accuracy of 72.91% for the Environment 1. Based on the comparison between the actual map and the generated maps - shown in Figure 15 - it can be observed that the AMR failed to detect a noticeable portion of the environment's walls. This caused the generated map to possess reading errors, which would account for the low boundary accuracy.

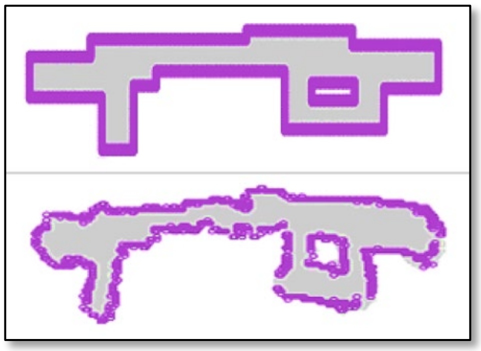


Figure 15. Measured (top) and generated (bottom)

For Environment 2, the AMR achieved an accuracy of 77%, which is slightly higher than that of Environment 1. However, it experienced the same issue of undetected obstacles similar to Environment 1. On top of that, wheel slippage caused parts of the map to be drawn askew. This can be seen in the map comparison shown in Figure 16. Certain trials in this environment also yielded some of the lowest boundary detection accuracies, which brought down the overall accuracy for the area. This could be further attributed to the tight space and the presence of large furniture in the area.

Finally, Environment 3 yielded an accuracy of 80.70%, which is the highest accuracy among the testing environments. Despite this, certain trials for this environment caused the AMR to overshoot the detected boundary by mapping parts of the surrounding area twice. This caused clustered points to accumulate in the generated map. This behavior can be possibly attributed to some wheel slippage, which causes the odometry to record false positives in terms of movement. This in turn causes the data fusion algorithm to record some duplicate values of the the same sonar data. Similarly to the tests in Environment 2, wheel slippage also caused parts of the area to be mapped askew, as shown in Figure 17.

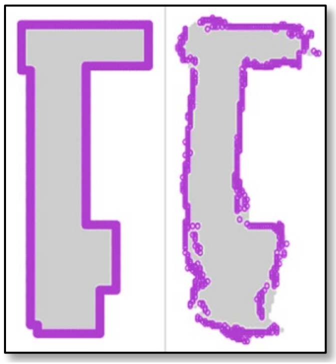


Figure 17. Measured (left) and generated (right)

4.2 Path Planning

To further validate the map usability, path planning was conducted on the generated map of Environment 1, which had the least boundary accuracy. The actual path travelled by the AMR was compared against a simulated ideal path trajectory for the area. These are shown in Figure 18.

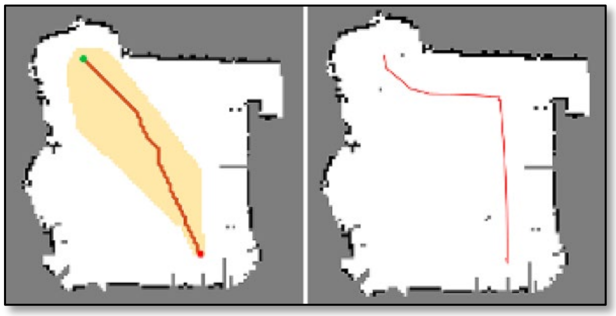


Figure 18. Simulated path trajectory (left), Actual path travelled by AMR (right)

The simulated path trajectory was computed to have a distance of 4.06 m. This is close to the actual distance travelled by the AMR, which was measured to be 5.18 m. Despite the AMR travelling approximately 1 m more than the simulated trajectory, it was able to navigate the generated map without any incident. This shows that the AMR's generated map is viable for basic navigation

4.3 Discussion

The results obtained from the experiments showed that the OSA setup for aSLAM was able to successfully reconstruct a 2D map representation of the three sample environments with at least 72.91% boundary accuracy. It is inferred that one can reduce the number of ultrasonic range sensors and maximize the AMR's mobility to obtain useful data points. This reduction in the number of sensor inputs into the algorithm may reduce the computational complexity and resource requirements of the system. During the experimentation, wheel slippage was observed which contributed to the reading inaccuracies by providing duplicate data or "skewing" data as seen in the test within Environments 2 and 3. This can be addressed by changing the wheels to attain better traction between the surface and the AMR. To further investigate the viability of the generated map, the map with the least boundary accuracy was used to perform path planning. Results presented in Figure 18 showed that the generated map could be properly used as a reference to run the A* Algorithm simulations as well as to conduct actual path planning.

5.0 CONCLUSION

In this study, aSLAM was highlighted to possess a strong potential in the context of SLAM implementation as it is not bounded by the typical limitations of vSLAM while providing a cheaper implementation than LiDAR SLAM.

Common aSLAM applications utilize several ultrasonic range sensors to capture a wider FOV, however, this paper presents the OSA for aSLAM implementation, where a few ultrasonic range sensors are adopted together with an AMR's kinematic mobility to obtain data points for the 2D map reconstruction. A set of 3 different environments was tested and results showed at least 72.91% boundary accuracy in terms of map reconstruction of these environments.

To further validate the map's usability, path planning was also performed with the least boundary accuracy. Results showed that applying the A* Algorithm allowed the AMR to traverse from a starting point to the goal. Moreover, the obtained results showed that the OSA is a viable setup for aSLAM implementations, suggesting that the proposed implementation can reduce the number of ultrasonic range sensors to 3. This finding allows for future works to investigate how the setup affects computational complexity and how it affects computational time and resources vs other methods [21]. Further testing can also be designed vs the mapping duration of the setup compared to the traditional aSLAM implementations.

Acknowledgement

The proponents of this research would like to extend its gratitude to University Research and Coordination (URCO), De La Salle University – Manila, Philippines providing resources which allowed the proponents to conduct the study.

Conflicts of Interest

The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper

References

[1] C. Cadena et al., 2016, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," in I*EEE Transactions on Robotics*, 32(6): 1309-1332. doi: 10.1109/TRO.2016.2624754.

[2] Y. Liu and J. Miura, 2021, "RDMO-SLAM: Real-Time Visual SLAM for Dynamic Environments Using Semantic Label Prediction With Optical Flow," in *IEEE Access,* 9: 106981-106997, doi: 10.1109/ACCESS.2021.3100426.

[3] A. Krishnan, S. Nayak, A. R. U. and S. Patilkulkarni, 2018, "Depth Camera based Autonomous Mobile Robot for Indoor Environments," *2018 4th International Conference for Convergence in Technology (I2CT),* Mangalore, India, 1-6, doi: 10.1109/I2CT42659.2018.9058246.

[4] J. P. M. Covolan, A. C. Sementille and S. R. R. Sanches, 2020, "A mapping of visual SLAM algorithms and their applications in augmented reality," *2020 22nd Symposium on Virtual and Augmented Reality (SVR), Porto de Galinhas, Brazil*, 20-29, doi: 10.1109/SVR51698.2020.00019.

[5] W. Yang and X. Zhai, 2019, "Contrast Limited Adaptive Histogram Equalization for an Advanced Stereo Visual SLAM System," *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), Guilin, China, 131-134, doi: 10.1109/CyberC.2019.00030.*

[6] T. H. Chan, H. Hesse and S. G. Ho, 2021, "LiDAR-Based 3D SLAM for Indoor Mapping," *2021 7th International Conference on Control, Automation and Robotics (ICCAR),* Singapore, 285-289, doi: 10.1109/ICCAR52225.2021.9463503.

[7] M. Liao, D. Wang and H. Yang, 2019 "Deploy Indoor 2D Laser SLAM on a Raspberry Pi-Based Mobile Robot," 2019 11th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), Hangzhou, China, ,7-10, doi: 10.1109/IHMSC.2019.10097.

[8] D. Shen, Y. Huang, Y. Wang and C. Zhao, 2018 "Research and Implementation of SLAM Based on LIDAR for Four-Wheeled Mobile Robot," *2018 IEEE International Conference of Intelligent Robotic and Control Engineering (IRCE),* Lanzhou, China, 19-23, doi: 10.1109/IRCE.2018.8492968.

[9] W. A. S. Norzam, H. F. Hawari, and K. Kamarudin, 2019. "Analysis of Mobile Robot Indoor mapping using GMAPPING based slam with different parameter," *IOP Conference Series: Materials Science and Engineering,* 705(1): 012037,

[10] C. Evers, A. H. Moore and P. A. Naylor, "Acoustic simultaneous localization and mapping (A-SLAM) of a moving microphone array and its surrounding speakers," 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, China, 2016, pp. 6-10, doi: 10.1109/ICASSP.2016.7471626.

[11] I. Eliakim, Z. Cohen, G. Kosa, and Y. Yovel, 2018. "A fully autonomous terrestrial bat-like acoustic robot," *PLOS Computational Biology,* 14(9). doi: 10.1371/journal.pcbi.1006406

[12] A. Burguera, Y. González, and G. Oliver, 2009 "Sonar sensor models and their application to mobile robot localization," *Sensors*, 9(12): 10217–10243. doi: 10.3390/s91210217

[13] P. Venkatesan, P. N. and R. Mohan, 2019, "Performing SLAM using Low-Cost Sensors for Autonomous Navigation in household environments," *2019 IEEE 5th International Conference for Convergence in Technology (I2CT), Bombay, India,* 1-5, doi: 10.1109/I2CT45611.2019.9033697.

[14] I. Dokmanić, L. Daudet and M. Vetterli, 2016 "From acoustic room reconstruction to slam," 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, China. 6345-6349, doi: 10.1109/ICASSP.2016.7472898.

[15] A. Bassiri, M. Asghari Oskoei, and A. Basiri, 2018. "Particle filter and finite impulse response filter fusion and Hector Slam to improve the performance of robot positioning," *Journal of Robotics,* 2018: 1–9

[16] Y. Dai and M. Zhao, 2021. "Grey Wolf Resampling-based Rao-blackwellized particle filter for mobile robot simultaneous localization and mapping," *Journal of Robotics,* 2021: 1–9, doi: 10.1155/2021/4978984

[17] S. Kuswadi, J. W. Santoso, M. Nasyir Tamara and M. Nuh, 2018 "Application SLAM and Path Planning using A-Star Algorithm for Mobile Robot in Indoor Disaster Area," 2018 International Electronics Symposium on Engineering Technology and Applications (IES-ETA), Bali, Indonesia, 270-274, doi: 10.1109/ELECSYM.2018.8615555.

[18] A. Urru, D. Piras and A. Palmas, 2019"Data Fusion algorithms to improve test range sensors accuracy and precision," *2019 International Conference on Range Technology (ICORT), Balasore, India*, 1-4, doi: 10.1109/ICORT46471.2019.9069667.

[19] M. J. M, R. Mathew, and S. S. Hiremath, 2019 "Reinforcement learning based approach for mobile robot navigation," 2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE),. doi: 10.1109/iccike47802.2019.9004256

[20] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, 2011. Introduction to autonomous mobile robots. Cambridge, MA: MIT Press,

[21] F. Peralta, M. Arzamendia, D. Gregor, D. G. Reina, and S. Toral, 2020. "A comparison of local path planning techniques of autonomous surface vehicles for monitoring applications: The Ypacarai Lake Case-Study," *Sensors,* 20(5): 1488, doi: 10.3390/s20051488.

[22] V. Germanos and E. L. Secco, 2016 "Formal Verification of Robotics Navigation Algorithms," *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES), Paris, France.* 177-180, doi: 10.1109/CSE-EUC-DCABES.2016.181.