







# A Case Study in Building 2D Maps with Robots

Theodor-Radu Grumeza<sup>1</sup><sup>a</sup>, Thomas-Andrei Lazăr<sup>1</sup><sup>b</sup>, Isabela Drămnesc<sup>1</sup><sup>c</sup>, Gabor Kusper<sup>2</sup><sup>d</sup>,  
Konstantinos Papadopoulos<sup>3</sup><sup>e</sup>, Nikolaos Fachantidis<sup>3</sup><sup>f</sup> and Ioannis Lefkos<sup>3</sup><sup>g</sup>

<sup>1</sup>Department of Computer Science, West University of Timisoara, Romania

<sup>2</sup>Department of Computer Science, Eszterhazy Karoly Catholic University, Eger, Hungary

<sup>3</sup>Laboratory of Informatics and Robotics Applications in Education and Society (LIREs), University of Macedonia, Thessaloniki, Greece

{theodor.grumeza, thomas.lazar, isabela.drannesc}@e-uvv.ro,

kusper.g

m.edu.gr

**Keywords:** 2D Maps Generation, Noise Reduction, Clustering, LiDAR, Sonar, Agilex, Scout Mini, Pepper Robot.

**Abstract:** In this paper, the authors are doing experimental work on generating 2D maps using the Agilex Scout Mini to utilize Pepper as an autonomous robot for guiding individuals within their university. This necessity arises from the lack of environmental data required for Pepper's navigation. Accurate and detailed maps are important for Pepper to orient itself effectively and provide reliable guidance. This process involves equipping Pepper to explore and document the university's physical layout, enabling autonomous movement and precise assistance for people. Key considerations include determining potential issues when using the two robots, the Scout with LiDAR and Pepper with Sonar, for map generation. Selecting an appropriate algorithm for noise reduction in the mapping points is a key feature for ensuring high-quality maps.

## 1 INTRODUCTION

The importance of 2D robotic mapping (Thrun, 2003) lies in the necessity for robots to understand their environment to perform their functions effectively. A well-constructed 2D map allows robots to navigate, make decisions, and interact with their surroundings more efficiently.


2D maps can be categorized into two main types: absolute and relative. Absolute maps consist of Points of Interest (POIs) with known GPS coordinates, accurate to a certain degree. In the case of relative maps, POIs do not have specific GPS coordinates. Instead, the map provides the relative distances and directions, i.e., vectors between POIs. This type of mapping is useful when GPS data is unavailable, like in the case of indoor navigation.


To create an absolute map, a GPS receiver with Real-Time Kinematic (RTK) corrections is required,


which allows an accuracy within 3-4 centimeters. For instance, in agricultural robotics (Blackmore et al., 2005), having a spatial database with the GPS coordinates of the fields to be cultivated is crucial. Such a database contains multiple layers, each with different types of data (e.g., parcel information, soil humidity, and soil quality), all linked to GPS coordinates.


The Pepper robot is an excellent social robot released by Aldebaran Robotics that can be used in our universities to interact with people and to assist and guide people indoors (Suddrey et al., 2018). Making Pepper autonomous is a big challenge due to the hardware and software limitations. There are several aspects to take into consideration:


- Can the robots be used to guide someone as an assistant to get from one room to another?
- How can Pepper avoid fixed obstacles and mobile obstacles (e.g., humans) that occur randomly along the way?
- How will Pepper know the environment (the map of the building in order to find the way)?
- Can Pepper be used as a scanning device for creating 2D maps of the building?
- Can a third-party device (Agilex Scout Mini) be used for mapping the ground floor of a building?


<sup>a</sup> <https://orcid.org/0009-0008-7709-9885>


<sup>b</sup> <https://orcid.org/0009-0005-4745-2260>

<sup>c</sup> <https://orcid.org/0000-0003-4686-2864>

<sup>d</sup> <https://orcid.org/0000-0001-6969-1629>

<sup>e</sup> <https://orcid.org/0009-0008-4999-1518>

<sup>f</sup> <https://orcid.org/0000-0002-8838-8091>

<sup>g</sup> <https://orcid.org/0000-0002-1895-4588>

## 2 STATE OF THE ART

There are a few fields of robotics which intersect with our research: Indoor localization, Machine vision, and Robotic mapping, especially 2D, sensor-based, and limited hardware robotic mapping.

In the case of indoor localization, one can rely on WiFi signal strength. By knowing the coordinates of WiFi routers and using a pre-made heat map, the position of a robot based on signal strength can be estimated. This method involves creating a heat map of the WiFi signal strengths in different locations and then using this map to triangulate the robot's position. Recent advancements in this area include the use of machine learning algorithms to enhance localization accuracy by addressing the signal fluctuation problems caused by dynamic environmental changes and shadowing effects (Salamah et al., 2016; Singh et al., 2021; Ayyalasomayajula et al., 2020; Arvai, 2021).

Other solutions rely on RFID tags, either active or passive, or beacons. These methods rely on signal triangulation and distance estimation techniques. RFID-based localization often involves measuring the time of flight (TOF) or received signal strength (RSS) to determine the distance between the robot and the tags or beacons. Such systems have shown promising results in enhancing the precision and reliability of indoor localization (Zhu and Xu, 2019; Arvai and Homolya, 2019; Arvai and Homolya, 2020).

A state-of-the-art solution combines these approaches (Kia et al., 2022).

In all these cases, filters like the Kalman filter (Kalman, 1960) are employed to better estimate the robot's position by accounting for statistical noise and other inaccuracies. This filtering process helps to refine the robot's localization, improving overall accuracy (Ivanov et al., 2018).

Robotic mapping involves various techniques for creating 2D maps, each with its advantages and limitations. There are various methods for creating 2D maps.

LiDAR is one of the most prevalent technologies used in robotic mapping due to its high accuracy and ability to provide detailed environmental data. It works by emitting laser beams and measuring the time it takes for the reflections to return, creating precise distance measurements. This method is highly effective in generating detailed and accurate maps, particularly in static and structured environments (Zhang and Singh, 2014).

Sonar technology uses sound waves to detect and map surroundings. While it is less accurate than LiDAR, it is more cost-effective and can perform well in certain conditions, such as underwater or in envi-

ronments with a lot of dust or smoke where optical sensors might struggle (Leonard and Durrant-Whyte, 1991). Since it is less accurate, is less suitable for detailed indoor mapping. Infrared (IR) sensors measure distances by emitting infrared light and measuring the reflection. They are commonly used for short-range detection and are effective in low-light conditions. IR sensors are often used with other sensors to enhance the overall mapping accuracy (Benet et al., 2002).

Previous studies on map generation show that generating maps with Pepper robots involves integrating sensors like LiDAR, cameras, and Inertial Measurement Units (IMUs) (Mostofi et al., 2014) to create accurate indoor maps. Up to our knowledge there is no study on generating maps with Agilex Scout Mini robot. To import maps into the Pepper robot for autonomous navigation, one can follow the steps based on integrating ROS (Robot Operating System) (Macenski et al., 2022) with Pepper's NAOqi framework as described in the papers (Suddrey et al., 2018).

Our approach differs because the process involves using Pepper's Sonar sensor and an Agilex Scout Mini robot with a LiDAR sensor. Initially, a Python script using the NAOqi framework allows Pepper to explore and map its environment. Due to inconsistent Sonar results, LiDAR from the Scout Mini is utilized for higher resolution mapping, facilitated by the Robot Operating System (ROS). Therefore, instead of using ROS on Pepper, the authors are using a collaborative robot, Scout Mini to run ROS.

## 3 THE PROBLEM AND APPROACH

### 3.1 Description of the Problem

To utilize Pepper as an autonomous robot for guiding individuals within our university, it is tackled the challenge of generating 2D maps with robots. The necessity for creating these maps arises due to the insufficient data available about the environment in which the robot is required to navigate. Accurate and detailed maps are essential for Pepper to orient itself and provide reliable guidance effectively. The process involves equipping Pepper with the capability to explore and document the physical layout of the university, ensuring that the robot can autonomously move and assist people with precise directions.

For the generation of the maps, one needs to take into consideration that this work does not plan to evaluate the SLAM methods like the ones presented in the previous section, but it aims to examine the following issues:

- Is there a problem when generating the 2D maps by having two robots: one equipped with LiDAR and one equipped with Sonar?
- Which sensor is more reliable and can get more data to be processed?
- Which algorithm can be used in order to obtain a good noise reduction for the points that compose the map?

### 3.2 Approach

In order to enable Pepper to autonomously move between rooms, the authors developed a method using 2D maps created with its Sonar sensor. This involved programming the robot to explore its environment, map it, and visualize the results through the NAOqi framework, which supports navigation and motion control.

The Python script connects to Pepper via its IP address and port, utilizing the 'ALNavigation' and 'ALMotion' services. After waking up the robot and setting safety distances to prevent collisions, it explores within a 15m radius, collecting environmental data to create and save a map. The robot then returns to its starting point, demonstrating its ability to use the map for navigation.

For visualizing, the data was processed with numpy and PIL (Umesh, 2012), by generating an image representation of the explored area. This effective autonomous mapping and navigation process highlighted the capabilities of the NAOqi framework.

Due to inconsistent mapping with the Sonar sensor, we also explored using the Agilex Scout Mini's LiDAR sensor for improved resolution. A Python script utilizing the Robot Operating System (ROS) was developed to collect and synchronize LiDAR and odometry data. The script subscribes to the '/scan' and '/odom' topics, allowing us to associate laser scan measurements with the robot's position.

Data is processed in real time, converting odometry from quaternion to Euler angles for accurate orientation. Buffered data is stored in unique CSV files to prevent memory overload and facilitate organized retrieval.

To ensure clarity in the generated maps, we employed advanced data handling and noise reduction techniques. Data was processed in manageable chunks, with  $x$  and  $y$  coordinates computed from range and angle measurements. The DBSCAN algorithm (Ester et al., 1996) was implemented to reduce noise by identifying valid data clusters and filtering out inaccuracies caused by environmental interferences.

**Data Preparation.** Points with invalid data, such as NaN or infinite values, are removed to maintain the dataset's integrity. This step ensures that only meaningful data is processed further.

**Clustering.** The DBSCAN algorithm is applied to the cleaned data in order to identify and filter out noise. Points classified as noise are discarded, leaving only the significant data points that contribute to the accurate mapping. This method is effective in handling complex and arbitrarily shaped clusters, making it ideal for Sonar and LiDAR data.

**Data Visualization.** After processing and filtering the data, the significant points are concatenated into a single dataset. The final step involves visualizing these points using a scatter plot, providing a clear and accurate representation of the 2D map. This visualization aids in interpreting and analyzing the spatial data, making it easier to identify key features and patterns.

**Importance of Noise Reduction in Sonar and LiDAR Scanning.** Noise reduction is critical in Sonar and LiDAR scanning for several reasons. It enhances the accuracy of the final map by removing erroneous readings that can distort the data. By filtering out noise, the map becomes clearer and more interpretable, allowing for better analysis. Efficient noise reduction algorithms like DBSCAN also ensure robust performance, enabling the processing of large datasets without compromising on speed or accuracy. Additionally, noise reduction compensates for environmental factors and sensor limitations, ensuring that the mapped data reflects the true spatial characteristics of the scanned area.

## 4 EXPERIMENTS WITH ALDEBARAN ROBOTICS PEPPER ROBOT

In this study, the Sonar sensor mounted on the Aldebaran Robotics Pepper robot was employed to scan a part from the ground floor of the university.

This sensor collected data as float values. The points were defined by their  $x$  and  $y$  coordinates, which denoted the robot's position within the environment.

Key libraries including 'qi' for robot-specific operations, 'numpy' for numerical computations, 'PIL.Image' for image processing,

‘sklearn.cluster.DBSCAN’ for clustering, ‘pandas’ for data manipulation, and ‘matplotlib.pyplot’ for plotting were used into the implementation of the method which helped us to create the maps.

The robot was activated, and safety distances were set to ensure secure navigation. Orthogonal and tangential safety distances were configured to 0.03 meters and 0.02 meters, respectively. The robot explored the environment within a 15-meter radius, capturing spatial data. This exploration data was saved for subsequent analysis.

Localization was initiated to allow the robot to navigate within the explored map. The robot returned to its initial position for consistency in data acquisition. Localization was then terminated.

The metrical map was retrieved, providing pixel values representing the environment. These pixel values were converted into an image for visualization using this formulae:

$$metric\_x = x * result\_map[0] + origin\_offset[0] \quad (1)$$

$$metric\_y = y * result\_map[0] + origin\_offset[1], \quad (2)$$

where  $x$  is the relative position of the robot into the environment,  $result\_map[0]$  is the resolution of the map in meters per pixel, and  $origin\_offset$  is the metrical offset of the pixel (0,0) of the map. Free space within the map was identified based on pixel intensity thresholds. Points representing free space were extracted and converted into metric coordinates using the map’s resolution and origin offset.

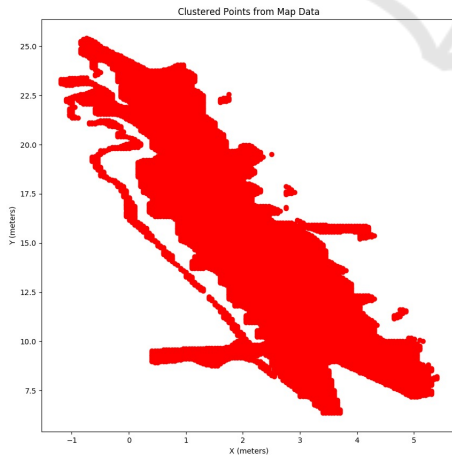


Figure 1: Sonar 2D map created with Pepper robot.

The DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm (Oliveira and Marçal, 2023) was applied to the extracted points. This clustering technique identified groups of points representing free space while filtering out noise. The algorithm’s parameters included an epsilon value of

0.3 meters and a minimum sample size of 10 points per cluster. Clustered points were stored in a data frame and visualized using scatter plots. This provided a clear representation of navigable areas within the mapped environment.

```
db = DBSCAN(eps=0.3, min_samples=100).fit(points)
labels = db.labels_
filtered_points = points[labels != -1]
```

Figure 2: The DBSCAN point filtering in Python.

## 5 EXPERIMENTS WITH AGILEX SCOUT MINI

In this study, the YDLiDAR G2 was employed [www.ydlidar.com/dowfile.html?cid=1&type=1](http://www.ydlidar.com/dowfile.html?cid=1&type=1), a 2D LiDAR sensor mounted on the Agilex Scout Mini, to scan the ground floor of the university.

This sensor collected up to 1512 float values per frame, with each value representing a distinct point. These points were defined by their  $x$  and  $y$  coordinates, which denote the robot’s position within the environment. The collected data points included several key parameters:

- *Range*: this value indicates the distance between the robot and surrounding objects (e.g., walls, doors, and static objects);
- *Theta*: this parameter represents the robot’s future position, considering its continuous movement during scanning;
- *Angle*: this is used to determine the angle at which the LiDAR sensor’s laser beam is reflected from various environmental points.

Throughout the scanning process, 3,763,368 points were recorded, each represented as a float value. This comprehensive data collection provided a detailed map of the university’s ground floor layout and static features.

To create the 2D maps, all the gathered points need to be converted into Cartesian coordinates using the following two formulae (Shim et al., 2005):

$$LiDAR\_x = x + ranges * np.cos(theta + angles) \quad (3)$$

$$LiDAR\_y = y + ranges * np.sin(theta + angles) \quad (4)$$

Formula (3) calculates the  $x$ -coordinates of the detected points in the map frame, using the range and angle to determine the position of each point relative to the sensor’s position and orientation. Formula (4), similarly, calculates the  $y$ -coordinates of the detected points.

During the initial scanning of the ground floor using the LiDAR on the Scout Mini as shown in Figure 3, numerous points were erroneously mapped outside the building due to the presence of many windows and glass doors. These external points generated unexpected noise in the map, potentially causing issues for the robot’s navigation. Since these points were included as part of the map, the robot may attempt to navigate along these nonexistent paths, leading to potential navigation errors.

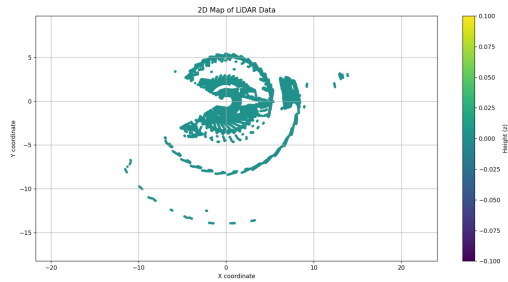


Figure 3: The first map generated by Scout.

Once the data is ingested, the LiDAR points are calculated by transforming polar coordinates (range and angle) into Cartesian coordinates ( $x$  and  $y$ ) relative to the robot’s position and orientation. This transformation uses the robot’s position, orientation, measured ranges, and angles to compute the  $x$  and  $y$  coordinates of the LiDAR points. These calculations are crucial as they convert the raw LiDAR data into a format suitable for further processing and visualization.

For the first scanning, only the  $x$  and  $y$  coordinates were considered, where the robot’s next position was not calculated, and it was unclear whether the robot was stationary or was moving continuously. After gathering 85.354 total points for the whole map of the ground floor, it needs to be taken into account all the values ( $x$ ,  $y$ , *Range*, *Theta*, and *Angle*) such that it will create a readable map as shown in Figure 4. Here, the total number of points in the dataset of 3,763,368 points was used, and the map was generated correctly (in the sense that they look similar to the real maps) but with a lot of noise.

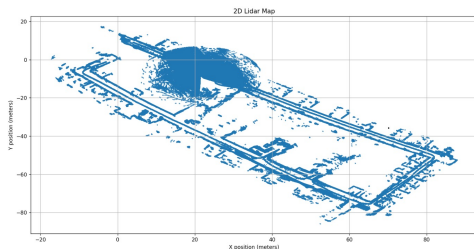


Figure 4: The LiDAR map without noise reduction.

The proposed method for generating a 2D LiDAR map with noise reduction involves several key steps, each one essential for processing and visualizing large-scale LiDAR datasets efficiently. Initially, the data is ingested in manageable chunks of 10,000 points. This chunk processing approach ensures efficient memory usage and facilitates the handling of extensive datasets, enabling the processing of LiDAR data without overwhelming the system’s resources (Perafan-Lopez et al., 2022).

To address noise in the LiDAR data, the method employs the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm (Oliveira and Marçal, 2023). DBSCAN is a clustering algorithm that identifies clusters in data by looking for areas of high point density. It requires two parameters:  $\epsilon$  (*eps*), which is the *radius* for neighborhood points, and *minPts*, which is the minimum number of points to form a dense region. The algorithm starts by selecting an unvisited point and checking its neighbourhood within  $\epsilon$ . If the point has at least *minPts* neighbours, it becomes a core point and starts forming a cluster by iteratively adding neighbouring core points and their neighbours. Points that do not meet this criterion are labelled as noise, and in Figure 2 is presented a code snippet, which is responsible for filtering the points that are out of the cluster. DBSCAN can find clusters of arbitrary shape and handle noise, making it suitable for spatial data like LiDAR maps.

The filtered points from each chunk are aggregated into a single dataset, which is then visualized using a scatter plot. This visualization step creates a 2D LiDAR map, with points plotted to represent the scanned environment. By adjusting the transparency of the points, the map provides a clear and detailed representation of the environment, highlighting the density and distribution of the LiDAR points. This method’s combination of efficient data processing, noise reduction, and effective visualization results in a comprehensive and accurate 2D LiDAR map (see Figure 5).

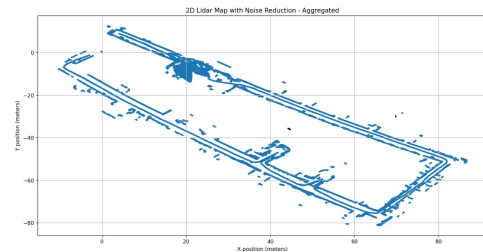


Figure 5: The LiDAR map with noise reduction.

When creating a 2D LiDAR map, the quality of clustering results is crucial for accurately identifying

Table 1: Average metrics for ground floor.

Average Silhouette Score	0.6360463511896636
Average Calinski–Harabasz Index	1.753.594.2288643694
Average Davies–Bouldin Index	0.4843305298989922

and separating different objects or features in the environment. The provided clustering metrics contribute to the quality of a 2D LiDAR map as shown in Table 1.

**Silhouette Score.** (Rousseeuw, 1987) helps determine the cohesion and separation of the clusters in the 2D LiDAR map. The score ranges from -1 to 1:

- 1:** The sample is far from the neighbouring clusters.
- 0:** The sample is on or very close to the decision boundary between two neighbouring clusters.
- 1:** The sample is assigned to the wrong cluster.

The Silhouette score of 0.636 in the context of a 2D LiDAR map indicates that the points belonging to each cluster are well-grouped together and distinctly separated from other clusters. This is important for accurately identifying different objects or structures, such as walls and static objects.

**Calinski-Harabasz Index.** (Wang et al., 2021) measures the dispersion within and between clusters. The Calinski-Harabasz Index score of 1.753.594.229 indicates that the clusters are compact and well-separated. In a 2D LiDAR map, this translates to clear and distinct boundaries between different objects or features. Such clear boundaries are essential for accurate mapping and object detection.

**Davies-Bouldin Index.** (Xiao et al., 2017; Petrovic, 2006) evaluates the average similarity ratio between clusters. The 0.484 value of the Davies-Bouldin Index indicates minimal similarity between clusters. For a 2D LiDAR map, the features identified by the clustering algorithm are distinctly different. This distinct separation ensures that different objects or obstacles are not misidentified or merged, enhancing the reliability of the map.

## 6 COMPARISON OF RESULTS

To achieve a more accurate comparison of the two 2D maps generated using the LiDAR and those with the Sonar sensor, a second scanning operation was

conducted where only 15 meters of the building’s ground floor was scanned using the LiDAR, as shown in Figures 6 and 7. The LiDAR scanning process is more complex, taking into account multiple parameters such as  $x$ ,  $y$ ,  $\theta$ ,  $\alpha$ , and  $r$ . In contrast, the Sonar sensor only collects  $x$  and  $y$  coordinates, resolution, and offset. The data collected using the Pepper robot’s Sonar sensor are inconsistent, showing clusters with points that do not exist on the map. Consequently, these clusters are neither well-defined nor precise, leading to a loss of coherence in the 2D map creation.

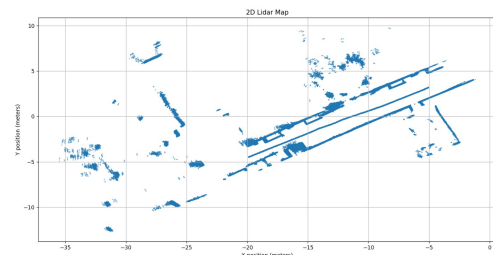


Figure 6: Partial LiDAR map without noise reduction.

Both LiDAR and Sonar-based mapping share key steps. For data collection, LiDAR captures up to 1512 float values per frame, generating a high volume of data, while Sonar collects data within a 15m radius, but only 0.5m at a time. In data conversion, LiDAR uses range and angle for Cartesian coordinates, whereas Sonar converts pixel values into metric coordinates based on resolution and origin offset. Both approaches utilize the DBSCAN algorithm for noise reduction, filtering out outliers. Processed data is visualized via scatter plots to highlight navigable areas (Oliveira and Marçal, 2023).

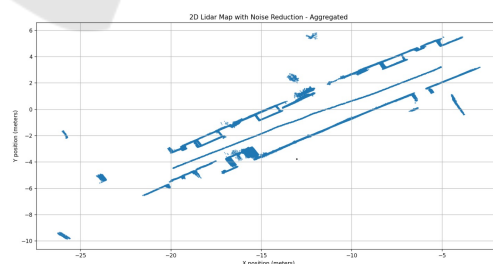


Figure 7: Partial LiDAR map with noise reduction.

The mapping results differ notably between the two approaches. LiDAR with the Scout Mini Robot yields high resolution with 3,763,368 points, capturing detailed features like walls and doors. Initial noise from reflective surfaces was mitigated by DBSCAN, resulting in well-defined clusters (Silhouette Score: 0.636, Calinski-Harabasz Index: 79,720.67, Davies-

Table 2: Average metrics for 15m of ground floor.

Average Silhouette Score	0.6304812788202012
Average Calinski–Harabasz Index	79,720.67387352433
Average Davies–Bouldin Index	0.4860425979366158

Bouldin Index: 0.484).

Conversely, Sonar mapping with the Pepper Robot has lower resolution due to a half-meter scanning limit, suitable for simpler environments. Despite using DBSCAN for noise filtering, Sonar maps are less precise. The clustering involved converting pixel values into metric coordinates, resulting in lower precision for navigable areas.

LiDAR excels in high-resolution mapping and complex environments, though it faces noise challenges and requires complex processing. Sonar is effective for basic navigation in simpler settings, with lower computational demands but also reduced detail and effectiveness.

In summary, while both methods follow similar processes, LiDAR offers superior detail and accuracy for complex environments, while Sonar is suited for simpler navigation tasks.

The average metrics were successfully obtained using the LiDAR for 15m scanning, as presented in Table 2.

This was possible because the data was initially stored in the robot and then processed on a separate computational machine. However, all computations were performed directly on the Pepper robot's hardware. Consequently, these metrics could not be obtained due to the inability to remove noise from the map.

## 7 CONCLUSIONS AND FUTURE WORK

This paper presents several contributions, aiming to utilize Pepper as an autonomous robot for guiding individuals within a university environment, where the challenge of generating 2D maps was tackled. The necessity for creating these maps arises due to the insufficient data available about the environment in which the robot is required to navigate. Accurate and detailed maps are essential for Pepper to orient itself and provide reliable guidance effectively. In this process, Pepper will be used to explore and document the university's environment, make the robot autonomously move, and assist people with accurate directions.

Our approach demonstrated that while Pepper's built-in Sonar sensor can create basic maps, an Agilex Scout Mini robot with a LiDAR sensor significantly enhances map resolution and reliability. By implementing a ROS-based system to synchronize and log LiDAR and odometry data, robust data collection for accurate mapping was ensured.

In future work, the authors would also like to use an algorithm like A\* or Dijkstra to calculate the shortest path the robot can take to get from one point to another. Crowd control is also an important step in recognizing people that are moving into the environment such that the robot will avoid them.

In order to have a correct interpretation of the data, formally verifying the DBSCAN algorithm for LiDAR maps it is important to ensure its reliability and robustness. LiDAR maps provide detailed spatial data, and any errors in clustering could lead to significant misinterpretations of the environment, potentially causing safety risks or incorrect decisions. By using formal methods to verify the algorithm, it can be mathematically proven that DBSCAN will correctly and consistently identify clusters, handle noise, and perform as expected under various conditions. This verification process helps build trust in the algorithm's outputs, ultimately contributing to safer and more efficient use of LiDAR technology.

## ACKNOWLEDGEMENTS

This work is co-funded by the European Union through the Erasmus+ project *AiRobo: Artificial Intelligence-based Robotics*, 2023-1-RO01-KA220-HED-000152418.

## REFERENCES

- Arvai, L. (2021). Convolutional neural network-based activity monitoring for indoor localization. *Pollack Period.*, 16(3):7–12.
- Arvai, L. and Homolya, S. (2019). Filtering and fingerprint matching methods for Wi-Fi radio map based indoor localization. In *2019 20th International Carpathian Control Conference (ICCC)*. IEEE.
- Arvai, L. and Homolya, S. (2020). Efficient radio map update algorithm for indoor localization. In *2020 21th International Carpathian Control Conference (ICCC)*. IEEE.
- Ayyalasomayajula, R., Arun, A., Wu, C., Sharma, S., Sethi, A. R., Vasisht, D., and Bharadia, D. (2020). Deep learning based wireless localization for indoor navigation. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*,

- MobiCom '20, New York, NY, USA. Association for Computing Machinery.
- Benet, G., Blanes, F., Simó, J. E., and Pérez, P. (2002). Using infrared sensors for distance measurement in mobile robots. *Robotics and Autonomous Systems*, 40(4):255–266.
- Blackmore, S., Stout, B. A., Wang, M., Runov, B. I., and Stafford, J. V. (2005). Robotic agriculture - the future of agricultural mechanisation? In *Proceedings of European Conference on Precision Agriculture 2005*, pages 621–628.
- Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231.
- Ivanov, P., Raitoharju, M., and Piché, R. (2018). Kalman-Type Filters and Smoothers for Pedestrian Dead Reckoning. In *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 206–212.
- Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45.
- Kia, G., Ruotsalainen, L., and Talvitie, J. (2022). Toward Accurate Indoor Positioning: An RSS-Based Fusion of UWB and Machine-Learning-Enhanced WiFi. *Sensors*, 22(9).
- Leonard, J. J. and Durrant-Whyte, H. (1991). Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–382.
- Macenski, S., Foote, T., Gerkey, B., Lalancette, C., and Woodall, W. (2022). Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074.
- Mostofi, N., Elhabiby, M., and El-Sheimy, N. (2014). Indoor localization and mapping using camera and inertial measurement unit (IMU). In *2014 IEEE/ION Position, Location and Navigation Symposium-PLANS 2014*, pages 1329–1335. IEEE.
- Oliveira, M. I. and Marçal, A. R. (2023). Clustering LiDAR Data with K-means and DBSCAN. In *ICPRAM*, pages 822–831.
- Perafan-Lopez, J. C., Ferrer-Gregory, V. L., Nieto-Londoño, C., and Sierra-Pérez, J. (2022). Performance Analysis and Architecture of a Clustering Hybrid Algorithm Called FA+GA-DBSCAN Using Artificial Datasets. *Entropy*, 24(7):875.
- Petrovic, S. (2006). A comparison between the silhouette index and the davies-bouldin index in labelling ids clusters. In *Proceedings of the 11th Nordic workshop of secure IT systems*, volume 2006, pages 53–64. Cite-seer.
- Rousseeuw, P. (1987). Rousseeuw, P.J.: Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.
- Salamah, A. H., Tamazin, M., Sharkas, M. A., and Khedr, M. (2016). An enhanced WiFi indoor localization system based on machine learning. In *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–8.
- Shim, Y., Chung, J., and Choi, I.-C. (2005). A Comparison Study of Cluster Validity Indices Using a Nonhierarchical Clustering Algorithm. In *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, volume 1, pages 199–204.
- Singh, N., Choe, S., and Punmiya, R. (2021). Machine Learning Based Indoor Localization Using Wi-Fi RSSI Fingerprints: An Overview. *IEEE Access*, 9:127150–127174.
- Suddrey, G., Jacobson, A., and Ward, B. (2018). Enabling a Pepper Robot to provide Automated and Interactive Tours of a Robotics Laboratory. In *Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2018)*. Australian Robotics and Automation Association (ARAA), Australia.
- Thrun, S. (2003). *Robotic mapping: a survey*. Morgan Kaufmann Publishers Inc.
- Umesh, P. (2012). Image Processing in Python. *CSI Communications*, 23.
- Wang, J., Zhang, W., Hua, T., and Wei, T.-C. (2021). Unsupervised learning of topological phase transitions using the Calinski-Harabaz index. *Phys. Rev. Res.*, 3:013074.
- Xiao, J., Lu, J., and Li, X. (2017). Davies Bouldin Index based hierarchical initialization K-means. *Intelligent Data Analysis*, 21(6):1327–1338.
- Zhang, J. and Singh, S. (2014). LOAM: Lidar Odometry and Mapping in Real-time. In *Robotics: Science and Systems*.
- Zhu, J. and Xu, H. (2019). Review of RFID-Based Indoor Positioning Technology. In Barolli, L., Xhafa, F., Javaid, N., and Enokido, T., editors, *Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 632–641, Cham. Springer International Publishing.