CS 898 AW: Assignment 1_3 Report

Schraeder, Logan (x356t577)
WSU School of Computing
MS in Data Science Program
Wichita State University, Wichita, KS
x356t577@wichita.edu

Abstract - This report details the implementation and debugging of a Simultaneous Localization and Mapping (SLAM) simulation for a robot using Python. The simulation involved generating random static and dynamic landmarks and tracking the robot's predicted and real locations. Several code modifications were necessary to address errors. Despite these efforts, a persistent ValueError remains unresolved. The simulation successfully ran through at least one time-step, detecting both static and dynamic landmarks, demonstrating the potential for creating an environmental map. Future work includes resolving the remaining error, streamlining simulation parameters, and introducing control variables for further testing of the SLAM algorithm.

Keywords - SLAM, robot, localization, mapping, computer vision, robotics, Python

I. BACKGROUND

Simultaneous Localization and Mapping (SLAM) is a fundamental problem in robotics, concerning how a robot can construct a map of its environment while simultaneously determining its location within that map. This is crucial for autonomous navigation, enabling robots to operate in unknown environments without human intervention. Effective SLAM allows robots to navigate complex and dynamic spaces, which is essential for various applications, including autonomous vehicles, search and rescue operations, and industrial automation.

SLAM employs a variety of sensors, such as lidar and cameras, to gather data about the surrounding environment. SLAM algorithms process this sensory information to identify key features, like landmarks, and estimate the robot's motion. By integrating these processes, SLAM enables robots to map out unknown environments and use this information for tasks such as path planning, obstacle avoidance, and executing complex behaviors. The technology is challenging, requiring high-quality sensor data and significant computational power, and ongoing research continues to improve the robustness and efficiency of SLAM systems.

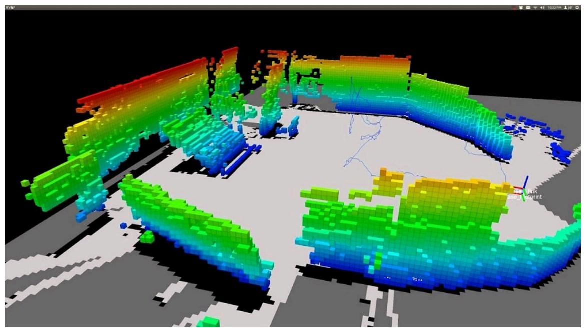


Fig. 1. Robotic 3D SLAM. *(3D representation of a robot's Simultaneous Localization and Mapping activities. Courtesy of Surveying Group Media LTD).*

II. METHOD

In general, implementation of the SLAM method was done by slightly modifying the source files to work with the custom main.py code, and run the simulation over a given number of simulation steps. The details of the code modifications and debugging needed are discussed in the later paragraphs of this section.

main.py established the simulation parameters, generated random landmarks (static and dynamic), initialized the robot, and ran the simulation. It invoked all the necessary and provided source files as modules in order to run the simulation from a single script. As an output, predicted and real/updated locations of the robot were printed to the user's Python console, and environmental maps were provided as plots (see Figure 2).

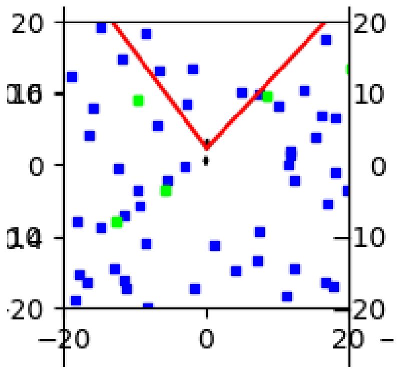


Fig. 2. Robot landmark detection plot. (*Robot SLAM landmark and position plot. Red indicates the robot's FOV, green shows dynamic landmarks, and green shows static landmarks.)*

A. Modifications and Debugging

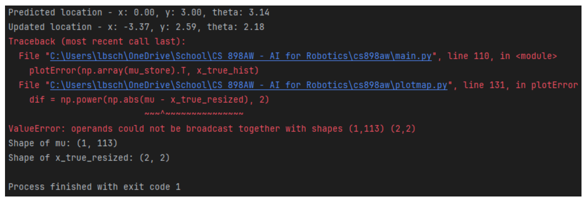
This section details the modifications implemented to rectify errors and enhance the functionality of the SLAM implementation. The use of Google's Gemini 2.0 was heavily used in the modification and debugging of the code as it performs better deep debugging of source code than the author alone. Using this approach, several critical issues were identified and addressed, leading to a more robust and accurate system that worked with the author's custom simulation code implementation.

Firstly, a ValueError related to shape incompatibility within the plotError function of plotmap.py was resolved. This error stemmed from a mismatch in array dimensions when comparing estimated and true robot positions. To correct this, the x_true array was reshaped to align with the dimensions of mu, ensuring consistent time step representation. Specifically, the line x_true_resized = x_true[:2, :mu.shape[1]] was adjusted to accurately slice and align the x_true array. Additionally, a TypeError in the same function, resulting from incorrect indexing due to x_true being a list, was fixed by converting it to a NumPy array and adjusting the indexing logic with x_true = np.asarray(x_true).T[:,:mu.shape[1]].

Furthermore, the main.py script encountered an IndexError during landmark measurement storage. This was due to attempting to access out-of-bounds indices in the z_store list. To address this, z_store was transformed from a list to a dictionary. This change allows for more flexible data storage using landmark IDs as keys, effectively handling arbitrary data access patterns. Lastly, an AttributeError in main.py, caused by the absence of a measurement_model method in the Robot class, was resolved. The measurement_model method was added to robot.py, enabling the calculation of expected range and bearing measurements of landmarks based on the robot's current pose. This addition ensures accurate prediction of landmark observations, crucial for the SLAM process.

B. Persisting Code Problems

Despite the author's debugging efforts, a code problem persisted. When running main.py and referencing plotmap.py, a ValueError "operands could not be broadcast together with shapes (1, 113) (2,2)" was unresolved. This was determined to stem from Numpy being unable to directly subtract the two different-shaped arrays. Future work would involve resolving this and finishing multiple simulation setups.


```
Predicted location - x: 0.00, y: 3.00, theta: 3.14
Updated location - x: -3.37, y: 2.59, theta: 2.18
Traceback (most recent call last):
  File "C:\Users\lbsch\OneDrive\School\CS 898AW - AI for Robotics\cs898aw\main.py", line 110, in <module>
    plotError(np.array(mu_store).T, x_true_hist)
  File "C:\Users\lbsch\OneDrive\School\CS 898AW - AI for Robotics\cs898aw\plotmap.py", line 131, in plotError
    dif = np.power(np.abs(mu - x_true_resized), 2)
                          ~~~^~~~~~~~~~~~~~~~
ValueError: operands could not be broadcast together with shapes (1,113) (2,2)
Shape of mu: (1, 113)
Shape of x_true_resized: (2, 2)

Process finished with exit code 1
```

Fig. 3. Error Code Snippet. *(Exit Code 1 given when running the SLAM simulation, resulting from a Numpy array mismatch).*

III. RESULTS AND ANALYSIS

As seen in Figure 2, the simulation ran through at least one time-step. In this iteration, the robot successfully detected static (blue) and dynamic (green) landmarks that were generated randomly by the code. Given multiple timesteps, the program and robot would have been able form a true map of its environment and location within it, while accounting for motor motion inputs, error, and field of view (FOV) adjustments.

Further work would include fixing the aforementioned Numpy ValueError, streamlining simulation parameter setup, and introducing true control variables to truly test the SLAM algorithm.

ACKNOWLEDGMENT

Acknowledgment is given foremost to Dr. Yan and his well-structured prompt codes. It is also recognized that the performance of this work was heavily thanks to the debugging and troubleshooting help of generative and conversational AI tools, most notably Google Gemini 2.0 and 2.5.

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[1] “What is SLAM and How Does It Works?,” Surveying Group News, https://www.surveyinggroup.com/what-is-slam-and-how-does-it-works/, April 6, 2023

[2] “What Is SLAM?,” MathWorks, https://www.mathworks.com/discovery/slam.html

[3] CS 898 AW: AI for Robotics, Lectures 1-5, Dr. Fujian Yan, Spring 2025

Key AI Prompts

“Can you help me debug [error code] in [file]?”
“I can’t get my main.py and slam_ls.py to work together because of [problem].”
“Can you help me refine or streamline my code?”
“Please reference these two sites and give me a background summary on SLAM.”
“Reference this paper and help me create an abstract for my paper.”