# CSIP5202 – Lab 4 Portfolio

A graph of a line graph

Description automatically generated

Victoria Hektor

P2629898

# Abstract

This study explores advanced robotic exploration by combining random wandering with structured wall following. Random wandering allows robots to move without set paths, using a probabilistic algorithm for unstructured exploration. Wall following, on the other hand, uses sonar sensors and a control system to navigate along walls at a constant distance. Together, these strategies aim to better cover different environments. The research also looks into improving how smoothly the robot moves and how effectively it wanders and researching techniques like Levy Flight Random Walks and Context-Aware Random Walks. These approaches bring their own set of challenges, such as uneven coverage and the need to fine-tune the algorithms. A key part of the study is the use of a Finite State Machine (FSM) architecture [26], refined through trial and error, to enhance the algorithms' flexibility for navigating complex spaces. This work helps in choosing the right exploration strategies and adds to the understanding of robotic navigation. [1]

Contents

[CSIP5202 – Lab 4 Portfolio 1](#_Toc153141110)

[Abstract 1](#_Toc153141111)

[Introduction 3](#_Toc153141112)

[Task: Implementation of Random Wandering and Wall Following 3](#_Toc153141113)

[Problem 3](#_Toc153141114)

[Experimentation 3](#_Toc153141115)

[Analysis 4](#_Toc153141116)

[Optional Enhancements: Refining Path Smoothness and Wandering Efficiency 5](#_Toc153141117)

[Experimentation 5](#_Toc153141118)

[Analysis 6](#_Toc153141119)

[Learning Curve 6](#_Toc153141120)

[Researching Better Methods for Wall Following 8](#_Toc153141121)

[Levy Flight Random Walk: 8](#_Toc153141122)

[Context-Aware Random Walks: 8](#_Toc153141123)

[Reinforcement Learning-Based Exploration: 8](#_Toc153141124)

[Graph-Based Exploration: 8](#_Toc153141125)

[Potential Field Methods: 8](#_Toc153141126)

[Bio-Inspired Algorithms: 9](#_Toc153141127)

[Conclusion 9](#_Toc153141128)

[References 10](#_Toc153141129)

[Figure 1 – Initial Path Followed by the Robot 4](#_Toc153141243)

[Figure 2 – Data Analysis (including Angles(Z)) 5](#_Toc153141244)

[Figure 3 – Post Optional Changes, Path Followed by the Robot 6](#_Toc153141245)

[Figure 4 – Data Analysis 6](#_Toc153141246)

[Figure 5 – First Path before tweaking FSM model 7](#_Toc153141247)

[Figure 6 – Path followed post-tweaking 7](#_Toc153141248)

# Introduction

In autonomous robotics, creating effective exploration strategies is essential for improving how robots operate in various settings. This study concentrates on two main tasks: random wandering and structured wall following. Random wandering allows robots to move spontaneously without a set route, reflecting a form of unstructured exploration. In contrast, wall following is a systematic method initiated by sonar detection, guiding robots to follow walls at a steady distance. These strategies are combined to enhance how robots cover and navigate through different areas.

A significant part of this project was transitioning from a basic while loop to a more sophisticated Finite State Machine (FSM) structure using switch/case statements. This shift has allowed for clearer code logic and state management, essential for the robot to switch seamlessly between wandering and following modes. This advancement underscores the project's focus on refining navigation efficiency and contributes to a more profound understanding of autonomous robotic exploration. [2]

# Task: Implementation of Random Wandering and Wall Following

The random wandering task employs a probabilistic algorithm to determine the robot's movement vectors. Each cycle, the robot randomly generates both the distance to travel and the angle to rotate within specified limits. This strategy promotes thorough exploration by preventing predictable paths and ensuring the robot does not become stuck in confined areas. [3]

The wall following task uses a closed-loop control system, [23], guided by side-mounted sonar sensors to maintain a specific distance from walls. This responsive system dynamically adjusts the robot's steering to correct deviations from the wall-following path, adeptly navigating along wall contours even without pre-mapped environmental data.

The FSM architecture underpinning these tasks enables the robot to switch intelligently between random wandering and wall following. State transitions are managed using switch/case logic, reflecting the robot's response to environmental cues such as the proximity to walls detected by its sensors. This FSM approach ensures seamless adaptation to varying environmental conditions, enhancing the robot's autonomous navigation capabilities.

## Problem

The primary challenge in this task was to develop a robotic control algorithm that could effectively balance two distinct modes of operation, random wandering and wall following. The random wandering mode was aimed at simulating aimless exploration, allowing the robot to navigate without a predetermined path. This mode faced the challenge of ensuring that the robot could effectively cover a significant area without unnecessary retracing of steps or getting stuck in confined spaces. [4] The wall following mode, on the other hand, was designed to enable the robot to navigate along walls using sonar sensors. The key challenge here was to maintain a consistent and safe distance from the wall, avoiding collisions while ensuring smooth navigation along the wall's contour.

## Experimentation

#### Random Wandering Implementation

The robot was programmed to travel random distances in straight lines, followed by turns at random angles. The randomness in distance and angle was intended to ensure thorough environmental coverage. The robot's ability to change directions spontaneously was key in preventing it from getting confined to limited areas.

#### Wall Following Implementation

When the robot's sonar sensors detected a wall within a predetermined distance, the algorithm switched to wall following mode. In this mode, the robot adjusted its wheel speeds based on the sonar readings from the side sensors to maintain a set distance from the wall. This required a delicate balance between speed and sensor input to ensure that the robot could smoothly navigate along the wall without veering too far away or colliding with it. [5]

## Analysis

#### Random Wandering Analysis

The random wandering mode proved effective for initial exploration, allowing the robot to cover a wide area. However, the randomness also meant that the robot's path was less predictable and could sometimes lead to inefficient retracing of previously covered areas.

#### Wall Following Analysis

The wall following mode allowed for more structured exploration once a wall was detected. The algorithm effectively adjusted the robot's path to maintain an optimal distance from the wall. However, this mode was somewhat limited in terms of overall area coverage as it depended on the presence of walls and could potentially lead the robot into closed loops along the wall contours. [6]

A graph of a line graph

Description automatically generated

Figure 1 – Initial Path Followed by the Robot

Figure 2 – Data Analysis (including Angles(Z))

# Optional Enhancements: Refining Path Smoothness and Wandering Efficiency

The primary objective of this part of development was to augment the existing autonomous navigation algorithm of a robotic platform, specifically tailored for the iRobot Create. The enhancements focused on two key areas: introducing randomness to the movement pattern for a more complex and less predictable trajectory and implementing a post-processing smoothing algorithm to refine the path data. These modifications aim to simulate a more natural and less deterministic navigational behaviour, akin to exploratory patterns observed in biological entities, while ensuring the resultant path remains coherent and analytically tractable. [7]

## Experimentation

#### Introduction of Randomised Movement

The existing deterministic approach, primarily based on wall-following and obstacle avoidance algorithms, was augmented with a stochastic component. This randomisation was introduced in the form of sporadic alterations in the robot's movement, including random turns, speed variations, and brief halts. Specifically, a probabilistic model was employed where, at each iteration of the main control loop, the robot has a 10% likelihood of executing a random action as opposed to following its standard wall-following or obstacle avoidance behaviour. This random action includes a full range turn (360-degree spectrum), a randomised adjustment of the wheel speeds, or a temporary halt followed by a reversal manoeuvre. [8]

#### Smoothing of the Path Data

Post-movement, the recorded path, comprising a series of coordinates (X, Y) and orientations (angle), was subjected to a smoothing algorithm. Utilising a moving average filter, this process was designed to reduce the abruptness and erratic nature of the path, resulting from both the inherent sensor noise and the introduced random movements. The smoothing function operates on the path data to compute a moving average over a specified window size, effectively ironing out sharp turns and fluctuations to yield a more fluid and visually coherent trajectory. [8]

#### Implementation and Testing

The enhanced algorithm was implemented within the MATLAB environment, leveraging the iRobot Create Toolbox. Rigorous testing was conducted in a simulated setting to evaluate the efficacy of the randomisation in creating diverse navigational patterns and the smoothing algorithm's ability to maintain the path's overall integrity. The resultant paths were visualised graphically, and the smoothed coordinates were exported to an Excel file for further analysis.

## Analysis

The primary challenge addressed in the Optional Enhancements was the need to refine the robot's path smoothness and wandering efficiency. Two key issues were identified: first, the robot's movements during wall following were prone to overshooting and abrupt corrections, especially near sharp turns, leading to jerky and inefficient navigation. Second, in the random wandering mode, the robot often retraced its steps or got confined to limited areas, leading to inefficient coverage of the environment. These issues hindered the robot's ability to navigate smoothly and explore its surroundings effectively. Implementing the smoothing algorithm seemed to offer a smoother and wider path taken. [7], [8].

A graph of a line drawn on a white background

Description automatically generated

Figure 3 – Post Optional Changes, Path Followed by the Robot

A graph with green lines

Description automatically generated

Figure 4 – Data Analysis

# Learning Curve

The initial technical approach to the autonomous exploration project involved a basic nested ‘while’ loop structure. This approach facilitated a form of reactive navigation, where the robot's movements were dictated by real-time sensor data, alternating between random wandering and wall-following behaviours as environmental conditions dictated. However, this rudimentary strategy lacked the finesse and systematic clarity afforded by a Finite State Machine (FSM) architecture, which is something realised late on in the project.

The inherent limitations of the nested loop became apparent through its operational inefficiencies and the complexity it introduced into the debugging process. The robot's traversal paths, while functional, were unpredictable and lacked optimisation, indicative of an absence of strategic state management.

The adoption of an FSM architecture marked a significant pivot in the project's development. With the implementation of FSM, discrete states for random wandering and wall following were clearly delineated, with transitions managed via a switch/case construct. This shift not only facilitated greater code legibility and maintainability but also introduced a structured framework conducive to further enhancements and rigorous testing.

The transformation in the robot's navigation, post-implementation of FSM, was stark. The paths followed by the robot displayed a marked improvement in purposeful exploration and smoothness in transitioning between states. FSM's encapsulation of state logic engendered a more sophisticated decision-making process, enabling strategic navigation that surpassed the limitations of the initial sensor-reactive programming.

This analytical progression from a basic looping construct to the adoption of an FSM framework accentuates the criticality of the underlying architecture in the management of robotic behaviour. FSM's incorporation not only solidified the control framework but also significantly amplified the navigation efficiency, yielding predictability and coherence in the robot's movement across varied terrains.

A graph of a robot

Description automatically generated

Figure 5 – First Path before tweaking FSM model

A graph of a line graph

Description automatically generated

Figure 6 – Path followed post-tweaking

# Researching Better Methods for Wall Following

### Levy Flight Random Walk:

Unlike standard random walks, Levy flights follow a heavy-tailed probability distribution for step lengths. This means the robot will occasionally make very long moves, interspersed with clusters of shorter movements. This can be more efficient for exploring large, open areas because it reduces the likelihood of immediate retracing of paths. [21]

#### Drawbacks:

* Non-Uniform Coverage - This method might lead to non-uniform exploration of the environment, with some areas being over-sampled and others under-sampled.
* Inefficiency in Dense Environments - In environments with many obstacles or complex terrains, the long steps characteristic of Levy flights can result in frequent collisions or the need for constant path recalculations.

### Context-Aware Random Walks:

These algorithms consider environmental factors, such as previously explored areas or detected obstacles. By incorporating sensors' data into the wandering algorithm, the robot can avoid areas it has recently visited or manoeuvre around obstacles more effectively. [11]

#### Drawbacks:

* Dependency on Environmental Data - The effectiveness of these algorithms heavily depends on accurate and comprehensive environmental data, which can be challenging to obtain in unknown or dynamic environments.
* Computational Complexity - Processing and integrating contextual information can be computationally intensive, particularly for robots with limited processing capabilities. [25]

### Reinforcement Learning-Based Exploration:

In this method, the robot learns an optimal exploration strategy through trial and error, guided by a reward mechanism. This approach is particularly useful in complex environments where pre-programmed behaviours may not suffice. Over time, the robot develops a policy that maximises area coverage and minimises redundant paths. [13]

#### Drawbacks:

* Long Training Times - These methods often require substantial training time to develop efficient exploration strategies, which might not be feasible in rapidly changing environments.
* Dependency on Reward Design - The success of reinforcement learning is highly dependent on the design of the reward function, which can be complex and non-intuitive in exploration tasks.

### Graph-Based Exploration:

The environment is represented as a graph, with nodes and edges corresponding to locations and paths, respectively. The robot then uses algorithms like Depth-First Search (DFS) or Breadth-First Search (BFS) to explore the environment systematically. This method ensures thorough coverage and can be combined with random elements to avoid predictability. [10]

#### Drawbacks:

* Overhead in Map Construction - Building and maintaining a graph-based representation of the environment can be resource-intensive, especially in large or highly dynamic areas. Some study’s explore how to overcome this. [24]
* Scalability Issues - As the exploration area grows, the graph becomes increasingly complex, potentially leading to scalability issues in terms of memory and computation.

### Potential Field Methods:

In this approach, areas of the environment are assigned potential values, with unexplored areas having higher potential. The robot is then driven to move towards higher potential areas, naturally leading to exploration of unvisited regions. This method can be dynamic, adjusting potentials based on real-time environmental data. [14], [15]

#### Drawbacks:

* Local Minima Problems - Robots can get stuck in local minima, particularly in environments with complex obstacle configurations. [16]
* Difficulty in Dynamic Environments - These methods can struggle in environments that change over time, as the potential fields need constant updating to remain effective. [16]

### Bio-Inspired Algorithms:

Drawing inspiration from nature, algorithms like Ant Colony Optimization (ACO), [18] or Particle Swarm Optimization (PSO), [19] can be adapted for robotic exploration. These methods mimic the way social insects explore and exploit their environment, leading to efficient and robust exploration strategies. [17]

#### Drawbacks:

* Complex Implementation - These algorithms can be complex to implement and fine-tune, requiring a deep understanding of the biological systems they are based on.
* Unpredictability - The emergent behaviour in bio-inspired systems can sometimes be unpredictable, making it difficult to guarantee consistent performance across different scenarios.

# Conclusion

In conclusion, this exploration into advanced robotic navigation strategies has been a journey of both discovery and technical challenge. The integration of random wandering and wall following methodologies revealed the dynamic and intricate nature of autonomous navigation. Through random wandering, I gained insights into the necessity of algorithmic adaptability, as the robot often encountered situations of getting stuck or looping, highlighting the need for intelligent environmental interaction. Wall following, while offering structured boundary navigation, required meticulous sensor feedback and control logic, especially in varied spaces with obstacles aplenty, underscoring the importance of precision in robotics.

This study reinforced the critical nature of selecting suitable exploration strategies, a process complicated by various technical hurdles. Such a choice must account for the robot's design, the intended task, and the operational environment, balancing algorithm efficiency with robustness and adaptability. My experience with these challenges has not only deepened my understanding of robotic exploration but has also spurred a drive for innovation. This inquiry sets the stage for future advancements in this dynamic and evolving field, paving the way for more sophisticated and resilient autonomous navigation systems. [20]

The process of navigating through complex problem-solving in this project has been both intellectually stimulating and gratifying. Successfully implementing diverse methodologies, such as the transition from a basic loop structure to a more sophisticated Finite State Machine (FSM) architecture, has not only been a journey of technical advancement but also a deeply satisfying learning experience. Delving into various aspects of coding, each challenge surmounted has brought with it a sense of achievement and a deeper understanding of the nuances of robotic programming. Furthermore, the integration of visualisations to monitor the robot's performance has added a tangible dimension to this exploration. These visual tools have not only facilitated a more intuitive understanding of the robot's behaviour and efficiency but have also served as a powerful means of validating the effectiveness of the implemented strategies. The convergence of problem-solving, coding, and visual analytics in this endeavour has provided a comprehensive and enriching experience, underscoring the joys of discovery and innovation in the field of robotics.

# References

[1]Columbia EDU, “MatlabTutorialDemo,” *Columbia.edu*, 2023. <https://www.cs.columbia.edu/~allen/F15/HMWK/MatlabTutorialDemo4.m.txt> (accessed Dec. 04, 2023).

[2]Mathworks, “Path following for a Differential Drive Robot - MATLAB & Simulink - MathWorks United Kingdom,” *uk.mathworks.com*, 2023. <https://uk.mathworks.com/help/robotics/ug/path-following-for-differential-drive-robot.html> (accessed Dec. 04, 2023).

[3]J. Esposito, “Matlab Toolbox for the iRobot Create V2.0,” *uk.mathworks.com*, Dec. 04, 2023. <https://uk.mathworks.com/matlabcentral/fileexchange/32698-matlab-toolbox-for-the-irobot-create-v2-0> (accessed Dec. 04, 2023).

[4]Autonomous Mobile Robots, “Autonomous Mobile Robots,” *GitHub*, Jan. 14, 2020. <https://github.com/autonomousmobilerobots> (accessed Dec. 04, 2023).

[5]Cornell EDU, “CS1112 Fall 2010 Project 3 Part 2 Due Thursday 10/14 at 11pm,” 2011. Accessed: Dec. 04, 2023. [Online]. Available: <https://www.cs.cornell.edu/courses/cs1112/2010fa/Simulator/p3/p3fa10b.pdf>

[6]T. W. Teng, P. Veerajagadheswar, B. Ramalingam, J. Yin, R. Elara Mohan, and B. F. Gómez, “Vision Based Wall following Framework: a Case Study with HSR Robot for Cleaning Application,” *Sensors*, vol. 20, no. 11, p. 3298, Jun. 2020, doi: <https://doi.org/10.3390/s20113298>.

[7]T. Fagerness, “How to Build a Wall-Following Robot - Projects,” *www.allaboutcircuits.com*, Dec. 01, 2015. <https://www.allaboutcircuits.com/projects/how-to-build-a-robot-follow-walls/>

[8]K. Bayer, A. De, and D. Koditschek, “Wall following for Autonomous Navigation,” Jun. 2012. Available: <https://sunfest.seas.upenn.edu/wp-content/uploads/2018/07/12-bayer.pdf>

[9]B. Pang, Y. Song, C. Zhang, H. Wang, and R. Yang, “A Swarm Robotic Exploration Strategy Based on an Improved Random Walk Method,” *Hindawi*, vol. 2019, pp. 1–9, Mar. 2019, doi: <https://doi.org/10.1155/2019/6914212>.

[10]F. Yang, D.-H. Lee, J. G. Keller, and S. Scherer, “Graph-based Topological Exploration Planning in Large-scale 3D Environments,” *arXiv (Cornell University)*, May 2021, doi: <https://doi.org/10.1109/icra48506.2021.9561830>.

[11]J. Quintas, G. S. Martins, L. Santos, Paulo Rossi Menezes, and J. Dias, “Toward a Context-Aware Human–Robot Interaction Framework Based on Cognitive Development,” *IEEE transactions on systems, man, and cybernetics*, vol. 49, no. 1, pp. 227–237, Jan. 2019, doi: <https://doi.org/10.1109/tsmc.2018.2833384>.

[12]S. Trimble, W. Naeem, S. McLoone, and Pantelis Sopasakis, “Context-aware Robotic Arm Using Fast Embedded Model Predictive Control,” *Research Portal (Queen’s University Belfast)*, Jun. 2020, doi: <https://doi.org/10.1109/issc49989.2020.9180217>.

[13]H. Li, Q. Zhang, and D. Zhao, “Deep Reinforcement Learning-Based Automatic Exploration for Navigation in Unknown Environment,” *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2019, doi: <https://doi.org/10.1109/tnnls.2019.2927869>.

[14]J. Barraquand, B. Langlois, and J.-C. . Latombe, “Numerical Potential Field Techniques for Robot Path Planning,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 2, pp. 224–241, 1992, doi: <https://doi.org/10.1109/21.148426>.

[15]M. A. K. Jaradat, M. H. Garibeh, and E. A. Feilat, “Autonomous Mobile Robot Dynamic Motion Planning Using Hybrid Fuzzy Potential Field,” *Soft Computing*, vol. 16, no. 1, pp. 153–164, Jun. 2011, doi: <https://doi.org/10.1007/s00500-011-0742-z>.

[16]Y. ZHU, T. ZHANG, and J.-Y. SONG, “Study on the Local Minima Problem of Path Planning Using Potential Field Method in Unknown Environments,” *Acta Automatica Sinica*, vol. 36, no. 8, pp. 1122–1130, Sep. 2010, doi: <https://doi.org/10.3724/sp.j.1004.2010.01122>.

[17]J. Li, S. X. Yang, and Z. Xu, “A Survey on Robot Path Planning Using Bio-inspired Algorithms,” *IEEE*, Dec. 2019, doi: <https://doi.org/10.1109/robio49542.2019.8961498>.

[18]M. Dorigo and T. Stützle, “Ant Colony Optimization: Overview and Recent Advances,” *International Series in Operations Research & Management Science*, pp. 311–351, Sep. 2018, doi: <https://doi.org/10.1007/978-3-319-91086-4_10>.

[19]D. Wang, D. Tan, and L. Liu, “Particle Swarm Optimization algorithm: an Overview,” *Soft Computing*, vol. 22, no. 2, pp. 387–408, Jan. 2017, doi: <https://doi.org/10.1007/s00500-016-2474-6>.

[20]O. Miglino, Kourosh Nafasi, and C. Taylor, “Selection for Wandering Behaviour in a Small Robot,” *Artificial Life*, vol. 2, no. 1, pp. 101–116, Jan. 1994, doi: <https://doi.org/10.1162/artl.1994.2.101>.

[21]C. Dimidov, G. Oriolo, and V. Trianni, “Random Walks in Swarm Robotics: an Experiment with Kilobots,” *Lecture Notes in Computer Science*, pp. 185–196, 2016, doi: <https://doi.org/10.1007/978-3-319-44427-7_16>.

[22] L. Nascimento, D. Barrios-Aranibar, Santos, D. S. Pereira, Willian Campos Ribeiro, and Pablo Javier Alsina, “Safe Path Planning Algorithms for Mobile Robots Based on Probabilistic Foam,” *Sensors*, vol. 21, no. 12, pp. 4156–4156, Jun. 2021, doi: <https://doi.org/10.3390/s21124156>.

[23] R. LeGrand, “Closed-Loop Motion Control for Mobile Robotics,” *Circuit cellar: The magazine for computer applications*, no. 169, pp. 34–46, Jan. 2004.

[24] C. W. Lee, J. D. Lee, J. Ahn, H. J. Oh, J. K. Park, and H. S. Jeon, “A Low Overhead Mapping Scheme for Exploration and Representation in the Unknown Area,” *Applied Sciences*, vol. 9, no. 15, p. 3089, Jul. 2019, doi: <https://doi.org/10.3390/app9153089>.

[25] C. Zeng, Q. Wang, S. Mokhtari, and T. Li, “Online Context-Aware Recommendation with Time Varying Multi-Armed Bandit,” Aug. 2016, doi: <https://doi.org/10.1145/2939672.2939878>.

[26] Prof. L. Titarenko and Prof. A. Barkalov, “Applied Sciences,” *www.mdpi.com*, Sep. 30, 2022. https://www.mdpi.com/journal/applsci/special\_issues/finite\_state\_machines (accessed May 22, 2023).