engineering proceedings

MDPI

Proceeding Paper

Robosim: Design, Implementation, and Applications of a Line Follower Robot Simulator †

Samarth Godara

ICAR-Indian Agricultural Statistics Research Institute, New Delhi,110012, India; sgodara@cs.iitr.ac.in or samarth.godara@icar.gov.in

† Presented at the 11th International Electronic Conference on Sensors and Applications (ECSA-11), 26–28 November 2024; Available online: https://sciforum.net/event/ecsa-11.

Abstract: Current line-follower-robot simulation tools often lack the ability to accurately replicate real-world conditions, making them ineffective for testing and optimizing algorithms and inaccessible for educational use. In this scenario, the present study introduces Robosim, a comprehensive line follower robot simulator that mimics physical dynamics, allowing users to develop, test, and refine control algorithms in a virtual environment. The developed Robosim simulator accepts input in the form of a floor map, which represents the environment in which the robot operates. The map is provided in a structured text format, such as a matrix or grid. Each element in the matrix corresponds to a specific type of terrain or path, with different characters or numbers representing different features. The Robosim simulator comprises key modules, including Map Reader for loading the floor map into a 2D array, Robot Initialization for setting start coordinates, Sensor Simulation for environmental sensing, and Robot Movement for executing control algorithms by adjusting coordinates. Users can implement custom algorithms for testing, with the Position Display module showing the robot's location on the map. The proposed software is a very lightweight simulator designed in C++, which makes it easy to use in computationally restricted environments. Moreover, the computer program, along with sample map files, is freely available for download at https://github.com/Samarth-Godara/Robosim_v1. The simulator is a versatile tool that can be utilized in a variety of settings. It can be used in educational institutions for teaching robotics and control systems, in research labs for developing and testing advanced robotic algorithms, and in the robotics industry for prototyping and refining line follower robots before physical deployment. Additionally, its user-friendly interface and comprehensive features make it an intriguing tool for hobbyists and enthusiasts to explore and learn about robotic systems.

Keywords: line follower robot; robotic simulation; virtual environment; path optimization; open-source simulator

Citation: Godara, S. Robosim: Design, Implementation, and Applications of a Line Follower Robot Simulator. *Eng. Proc.* **2024**, *6*, x. https://doi.org/10.3390/xxxxx

Academic Editor(s): Name

Published: 26 November 2024

1. Introduction

Line follower robots are widely used in various domains for efficient problem-solving. In manufacturing, they automate material transport and streamline production lines [1]. In warehouses, these robots enhance inventory management by navigating predefined paths. Additionally, in education and research, they serve as practical tools for teaching robotics and developing advanced control algorithms [2].

In this scenario, there is a growing need for more simulation tools for line-follower robots to enhance the development and testing of control algorithms. Lightweight tools are particularly essential to ensure accessibility and efficiency in various applications. Open-source simulators can greatly benefit the community by fostering collaboration and innovation. Such tools would bridge the gap between theoretical concepts and practical implementation. Robosim addresses these needs by providing a comprehensive, lightweight, and open-source solution.

In the past, numerous simulators have been developed to model the behavior of line follower robots, but many of these systems are limited in their ability to simulate physical dynamics accurately. According to [3], most conventional line follower simulators lack the capability to replicate real-world conditions such as uneven terrain, varying surface materials, or sensor noise, which are critical for testing control algorithms under practical scenarios. In educational settings, such simulators often fail to provide students with an authentic understanding of robot-environment interactions, limiting the scope of learning [4].

Moreover, existing simulation tools tend to be computationally demanding, restricting their use in resource-limited environments. For example, the popular V-REP simulator, as discussed by [5], offers comprehensive features for robotic simulation but is unsuitable for devices with limited processing power due to its high computational cost. This restricts the accessibility of such tools to high-end systems, thus limiting their use in educational institutions, particularly in regions where resources are scarce.

To address these limitations, the Robosim simulator introduces a novel approach by offering a lightweight, easy-to-use platform designed in C++. Unlike its predecessors, Robosim effectively simulates the physical dynamics of a line follower robot, making it suitable for algorithm development, testing, and optimization in virtual environments. The use of a matrix-based map input allows users to model various terrains and paths, offering flexibility in designing simulation environments. Furthermore, it is noted by various studies that the ability to simulate different terrains is crucial for developing more adaptive and robust algorithms [6].

In addition to its realistic simulation capabilities, Robosim's design allows for efficient performance in computationally restricted environments, making it accessible for use on low-end hardware such as older computers or embedded systems. This feature is especially valuable for educational institutions and research labs with limited computational resources. Moreover, in the study, the Robosim simulator is validated by testing it with six different map configurations, each containing a varying number of stations. An AI-based algorithm was used to navigate the maps, and the scanning time for each was recorded. Furthermore, the computer program of the Robosim simulator, along with sample map files, is freely available for download at https://github.com/Samarth-Godara/Robosim_v1. To assist researchers in using the simulator, we have recorded a demonstration video, which is available at the repository link.

Overall, robotic simulators have become a valuable tool in education, allowing students to engage in hands-on learning without the need for physical hardware. In this scenario, Robosim, with its easy-to-understand interface and robust simulation capabilities, is particularly well-suited for teaching robotics and control systems. By allowing students to load custom maps and develop their own control algorithms, Robosim offers a flexible platform for experimentation.

2. Methodology

The methodology for the development and implementation of the Robosim simulator is divided into six key components (Figure 1). These components are designed to accurately simulate the physical dynamics of a line follower robot, while maintaining a lightweight structure to ensure usability in computationally restricted environments. The Robosim simulator is developed in C++ and consists of the following modules. To ensure the reproducibility of this research, the complete code for each of the mentioned modules is available at: https://github.com/Samarth-Godara/Robosim_v1.

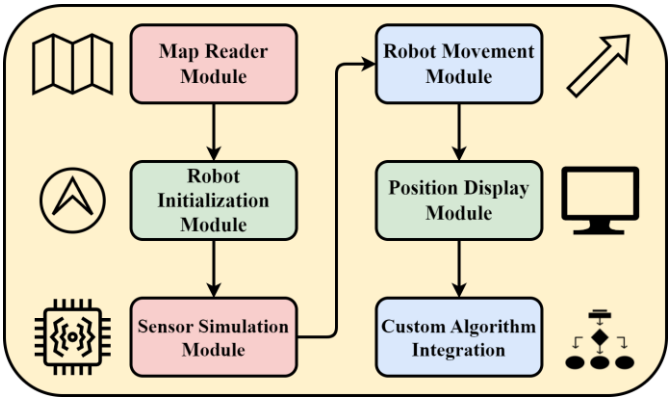


Figure 1. Methodology for the development and implementation of the Robosim simulator.

1. Map Reader Module: is responsible for loading the line follower robot's environment. The environment is represented as a grid or matrix where each element corresponds to a specific terrain or path. The map is provided in a structured text format, with different characters or numbers representing various terrain features such as paths, obstacles, and starting/ending points. In this step, first, the map file is loaded from the disk and read into a 2D array. Later, each element of the 2D array is interpreted according to its value, allowing for different types of terrain (e.g., path, obstacle) to be simulated. This module ensures that the entire map is parsed and ready for the robot's navigation.
2. Robot Initialization Module: initializes the robot's starting position on the map, ensuring that the robot's coordinates align with a valid starting point as defined in the map. The robot's movement parameters, such as speed and direction, are also initialized within this module.
3. Sensor Simulation Module: mimics the behavior of sensors used in a physical line follower robot, such as infrared sensors. These sensors detect the path ahead and provide feedback to the control algorithms. The sensors check the neighboring grid cells to determine whether the robot should continue straight, turn, or stop. The sensor checks the adjacent cells in the direction of movement and provides feedback on whether the robot is on a valid path (e.g., on the line or near obstacles).
4. Robot Movement Module: is responsible for executing the control algorithms that dictate the robot's movement on the map. Based on the sensor feedback, the robot adjusts its position on the grid by moving forward, turning left, or turning right. This module also ensures that the robot avoids obstacles and stays on the designated path. In this step, the key operations include processing the feedback from the sensors, adjusting the robot's coordinates according to the control logic and ensuring that the robot does not move out of bounds or into obstacles.
5. Position Display Module: visually represents the robot's current position on the map. This is done by outputting the robot's coordinates and displaying the corresponding grid. The map is updated in real-time as the robot moves, providing a visual representation of its trajectory.
6. Custom Algorithm Integration: Robosim allows users to integrate their own custom algorithms for controlling the robot's movement. Users can implement advanced control strategies and test them in the simulated environment. The modular design of the simulator enables easy integration of new algorithms.

3. Experiments and Results

Figure 2 provides a step-by-step overview of key features of the Robosim simulator. Figure 2a depicts the initial screen of the Robosim interface upon startup. Figure 2b shows the prompt where the user is required to define the robot's initial position on the map. Figure 2c illustrates the graph generated by Robosim's AI-based navigational lagorithm, which corresponds to the floor map that the robot will traverse. Figure 2d captures the

simulator requesting the user to select the destination station for the robot; upon selecting station 5, Robosim displays a detailed list of junctions and corresponding turns required to navigate the path. Finally, Figure 2e demonstrates the simulator prompting the user to input the next traversal command for the robot's movement. This sequence outlines Robosim's user interaction process, from initializing the robot's position to navigating stations and inputting movement commands, providing a clear representation of its operational flow.

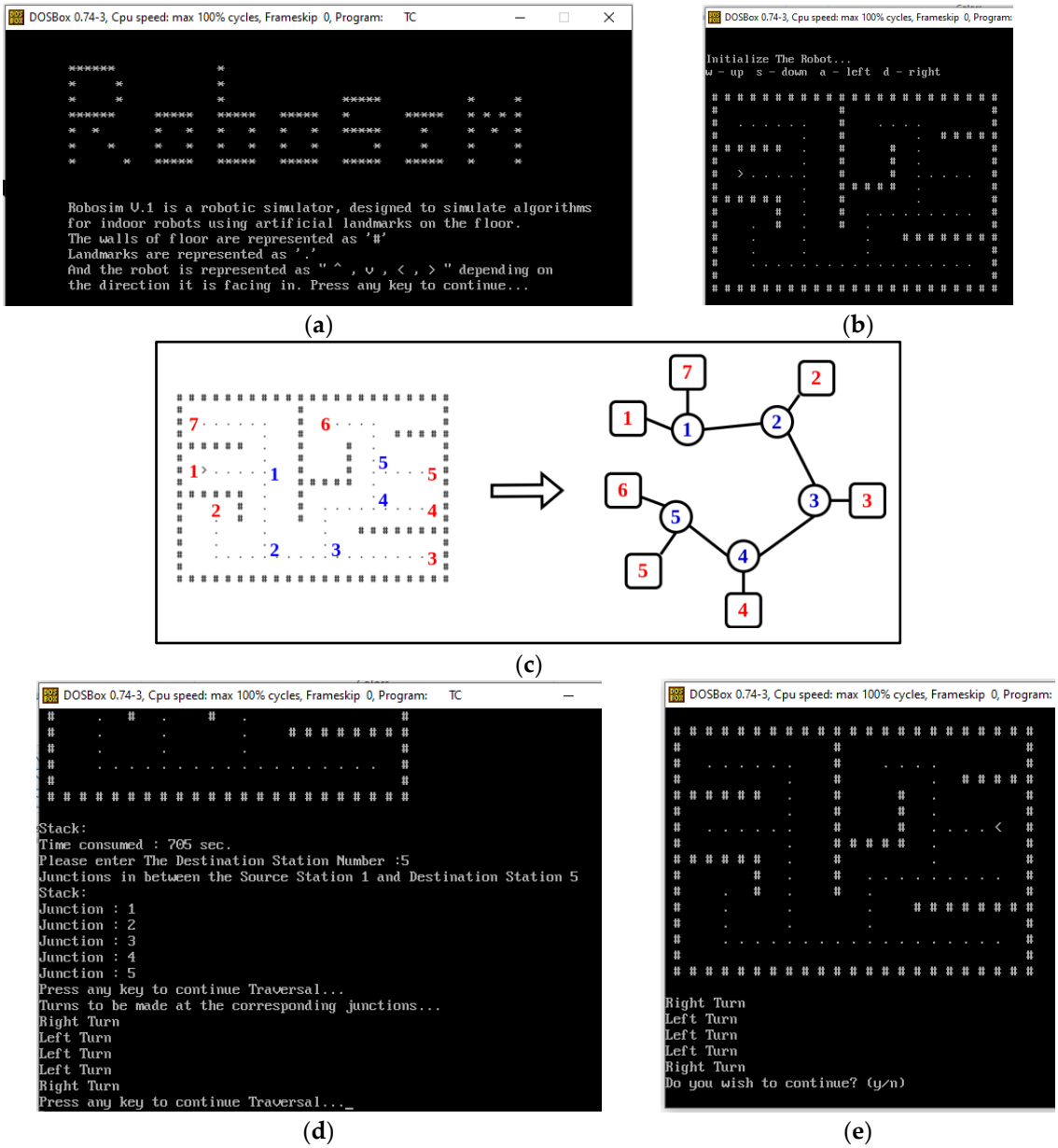


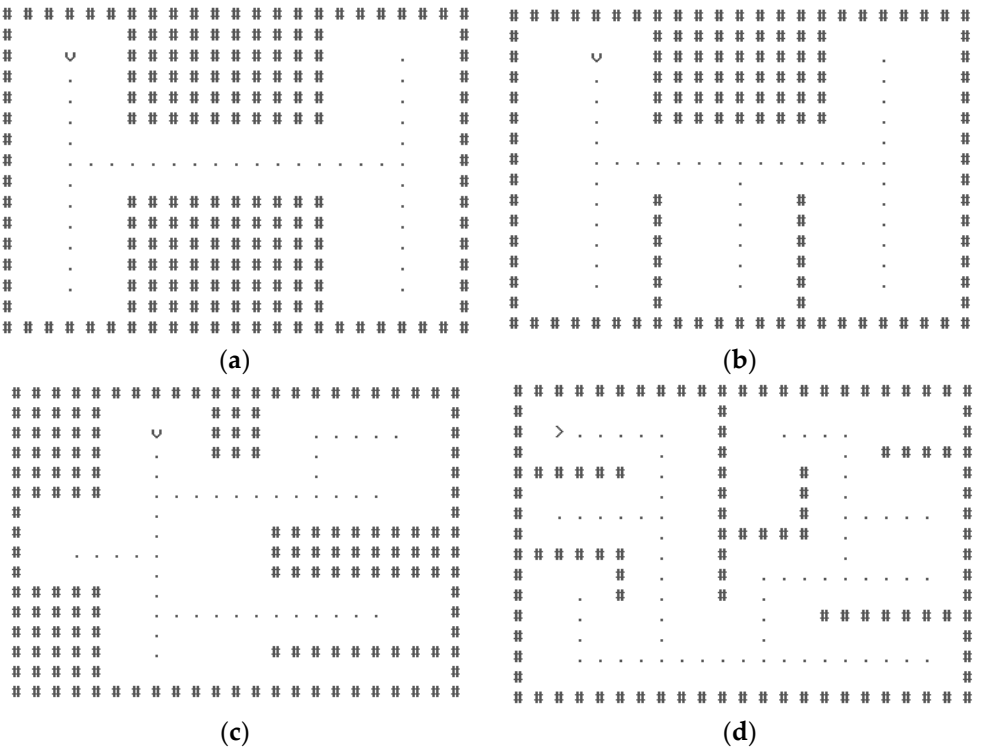
Figure 2. Key Interface Features of the Robosim Simulator, (**a**) The opening screen of the Robosim simulator, (**b**) User prompt for defining the robot's initial position, (**c**) Graph generated by Robosim, corresponding to the floor map, (**d**) Simulator requesting the destination station and displaying junctions and turns for station 5, (**e**) User prompt for inputting the next traversal command.

To evaluate the performance and effectiveness of the Robosim simulator, a series of experiments were conducted using the AI-based navigational algorithm and multiple custom map files (Figure 3). The input map file, structured as a grid, contains various path types and obstacles designed to challenge the robot's navigation. The key goal of these experiments was to test the robot's ability to follow the designated path, avoid obstacles,

and reach the destination using different control algorithms. The Robosim simulator, by default, includes three distinct line-following algorithms, each designed to navigate the robot through a predefined matrix-style map, simulating different terrain types and obstacles. The three algorithms include:

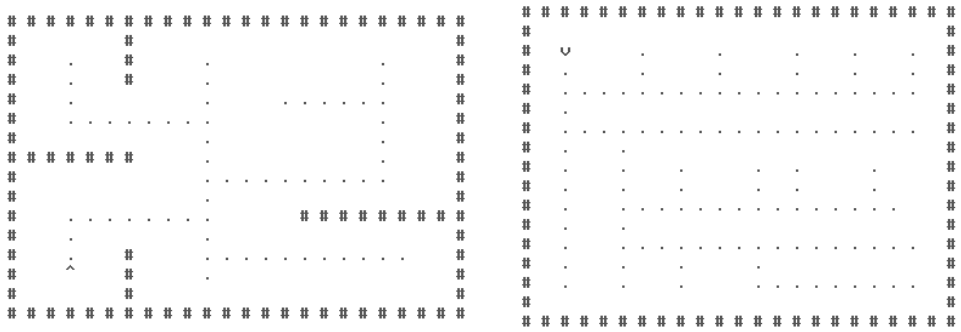
1. Left Path Algorithm: This algorithm prioritizes following the left edge of the path at intersections or decision points.
2. Right Path Algorithm: In contrast, this algorithm prioritizes following the right edge of the path, behaving similarly to the Left Path Algorithm but with a mirrored decision-making process.
3. A.I. based Localization and Navigational Algorithm: This advanced algorithm utilizes sensor feedback and heuristic decision-making to localize the robot within the environment and select the optimal path, accounting for both left and right turns based on the context. The AI-based algorithm begins by scanning and mapping a floor-drawn line network using an m-graph generator, which captures the layout as a graph of stations and junctions. Upon receiving a destination input, the Pruned-BFS algorithm efficiently calculates the optimal path from the source to the target station, minimizing redundant paths. The robot then uses this path, turning at junctions as required, to navigate toward the destination. This process ensures efficient indoor navigation with minimal memory usage, leveraging the m-graph's concise representation. Detailed information regarding the algorithm can be obtained from [7].

Figure 3a–f presents six maps used for robust validation of the simulator, each featuring progressively increasing complexity. Moreover, Figure 4 presents the data for the Robosim simulator's performance using the AI-based algorithm across various maps with different numbers of stations. The details regarding the AI-based algorithm used in the Robosim can be found in the referenced documentation at: https://github.com/Samarth-Godara/Robosim_v1.



(**a**) (**b**)

(**c**) (**d**)



(**e**) (**f**)

Figure 3. Six maps with varying levels of complexity used to validate the performance and robustness of the Robosim simulator. In the maps, the symbol '#' denotes floor walls and obstacles, while '.' represents the lines drawn on the floor.

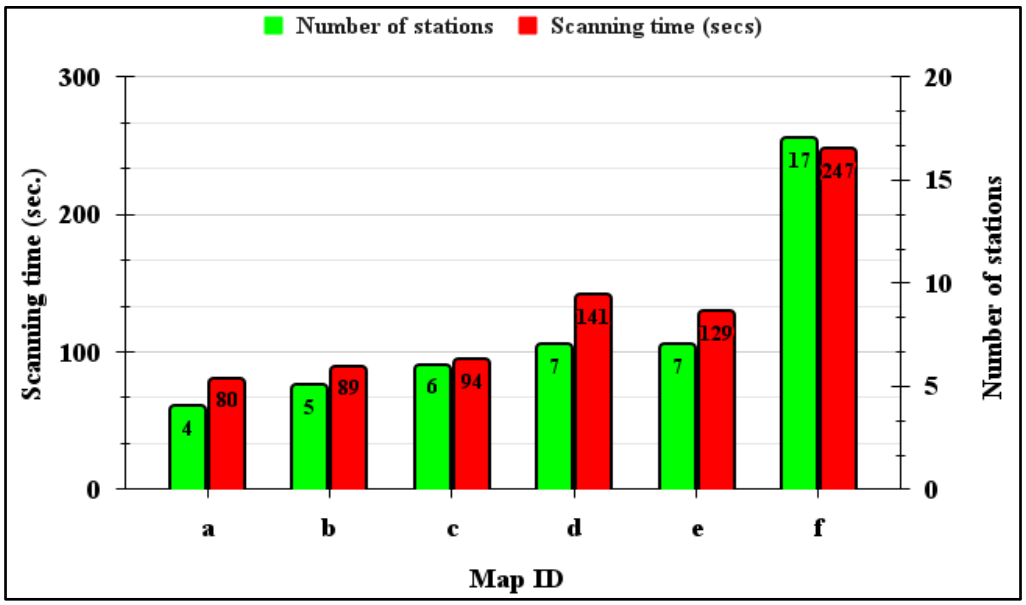


Figure 4. Results of Robosim's Performance on Different Maps.

From the obtained results, it is noted that maps a to c show a steady increase in the number of stations, which correlates with a gradual increase in scanning time taken by the AI-based navigational algorithm. This suggests that as the number of stations increases, the AI-based algorithm takes longer to process the map due to the increased complexity of navigating through more decision points (stations). Furthermore, between maps c (6 stations, 94 s) and d (7 stations, 141 s), there's a noticeable jump in scanning time. This could be due to either the increased complexity in the layout of stations or additional challenges (e.g., obstacles or tighter paths) that make scanning slower despite a small change in station count. Interestingly, map e also has 7 stations but the scanning time is slightly shorter (129 s) than map d. This could indicate that the layout of the stations in map e was simpler for the AI algorithm to navigate, perhaps with fewer obstacles or more direct paths between stations. Map f has 17 stations, and the scanning time spikes to 247 s, which is expected given the much larger number of stations. The exponential increase in scanning time suggests that the complexity of the map significantly affects the performance of the algorithm, especially as the number of stations grows larger.

The AI-based algorithm shows a proportional relationship between the number of stations and the scanning time. However, the map layout and other factors (like obstacles or path structure) also impact the scanning time, as seen in the differences between maps d and e, both with 7 stations but different scanning times. The larger jump in time for map

f highlights the potential challenge in scaling the algorithm to handle significantly more complex networks efficiently. The correlation between the number of stations and the scanning time is 0.98, indicating a strong positive relationship. This suggests that as the number of stations increases, the scanning time also increases almost proportionally.

4. Discussion

In the performed experiments, it is noted that the robot demonstrated excellent adaptability, successfully choosing between left and right turns at intersections based on the optimal route to the destination. It efficiently avoided obstacles, re-routing itself when necessary to bypass blocked paths without getting stuck. The robot reached the destination consistently in both simple and complex environments, even when the path contained multiple branches and intersections.

The Robosim simulator, while effective, has several limitations. It lacks support for simulating sensor noise and environmental variations, such as uneven (elevated) terrain, which can impact algorithm testing. Additionally, it currently supports only line-follower robots, limiting its applicability to other types of robots or more complex control systems. Finally, due to its simplified navigation model, Robosim's performance may degrade with highly complex maps (including cycles in the map) or an increased number of stations.

The future scope of the Robosim simulator includes several key enhancements. Expanding its capabilities to simulate real-world sensor noise, terrain (elevation) variations, and environmental factors would improve its realism. Support for additional robot types and more advanced control algorithms, including multi-robot simulations, could broaden its applicability. Incorporating real-time data feedback and visualization features would enhance user interaction. Finally, optimizing the system for larger and more complex environments could improve scalability, making Robosim suitable for testing more sophisticated robotics systems.

5. Conclusions

This study introduced and evaluated 'Robosim', a lightweight and flexible simulator for testing and optimizing line follower robot algorithms. Robosim offers a realistic virtual environment that mimics the physical dynamics of a line-following robot and accepts a matrix-style map as input, allowing for the simulation of various path and obstacle configurations. The A.I.-based Localization and Navigational Algorithm was tested within the simulator through a series of experiments. The simulator was tested on six different maps, each exhibiting varying complexities and station configurations. The results indicate a strong positive correlation (~0.98) between the number of stations and scanning time, with complexity and layout also affecting performance. Notably, maps with the same number of stations show varied scanning times, highlighting the influence of obstacles and path structures on algorithm efficiency. Robosim's versatility, ease of use, and adaptability make it an ideal tool for educational purposes, research labs, and robotics development environments. By providing a platform for testing and refining algorithms, Robosim bridges the gap between theoretical development and practical deployment, offering valuable insights for the design and optimization of line-following robots. Future work will focus on enhancing the simulator with more complex terrain types, real-time feedback features, and support for additional robot models to further broaden its application scope.

Supplementary Materials: The following supporting information can be downloaded at: www.mdpi.com/xxx/s1, Figure S1: title; Table S1: title; Video S1: title. The supporting information can be downloaded at: https://github.com/Samarth-Godara/Robosim_v1, https://github.com/Samarth-Godara/pony.

Author Contributions: The research was conducted by S.G., Scientist at ICAR-Indian Agricultural Statistics Research Institute, New Delhi, India. S.G. was responsible for the conceptualization, methodology, software development, validation, formal analysis, investigation, resources, data curation,

original draft preparation, review and editing, and visualization. All tasks related to this research were completed by S.G., and he has read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement:

Informed Consent Statement:

Data Availability Statement: "Robosim_v1" at https://github.com/Samarth-Godara/Robosim_v1.

Acknowledgments: We would like to express our sincere gratitude to Geeta Sikka (Head, Department of Computer Science, National Institute of Technology Delhi, New Delhi, India), Rajender Parsad (Director, ICAR-Indian Agricultural Statistics Research Institute, New Delhi, India), and Sudeep Marwaha (Head, Division of Computer Applications, ICAR-Indian Agricultural Statistics Research Institute, New Delhi, India) for their invaluable guidance and support throughout the course of this research.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Hynynen, J. Using artificial intelligence technologies in production management. *Comput. Ind.* **1992**, *19*, 21–35.
2. Miglino, O.; Lund, H.H.; Cardaci, M. Robotics as an educational tool. *J. Interact. Learn. Res.* **1999**, *10*, 25–47.
3. Donà, R.; Ciuffo, B. Virtual testing of automated driving systems. A survey on validation methods. *IEEE Access* **2022**, *10*, 24349–24367.
4. Dietz, G.; Chen, J.K.; Beason, J.; Tarrow, M.; Hilliard, A.; Shapiro, R.B. Artonomous: Introducing middle school students to reinforcement learning through virtual robotics. In Proceedings of the 21st Annual ACM Interaction Design and Children Conference, Braga, Portugal, 27–30 June 2022; pp. 430–441.
5. Rohmer, E.; Singh, S.P.; Freese, M. V-REP: A versatile and scalable robot simulation framework. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 1321–1326.
6. Peng, X.B.; Berseth, G.; Van de Panne, M. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Trans. Graph.* **2016**, *35*, 1–12.
7. Godara, S.; Sikka, G.; Parsad, R.; Marwaha, S.; Faiz, M.A.; Bana, R.S. Pony: Leveraging m-graphs and Pruned-BFS Algorithm to Elevate AI-Powered Low-Cost Self-Driving Robotics. *IEEE Access* **2024**, *12*, 134185–134197. https://doi.org/10.1109/ACCESS.2024.3462102.