Journal of Theoretical and Applied Information Technology

15th December 2023. Vol.101. No 23

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

RANDOM WALK-BASED APPROACH FOR ADDRESSING OF NAVIGATION ALGORITHMS IN SERVICE AND SURVEILLANCE ROBOTS

ALBERTO MALDONADO ROMO[1], JESUS YALJA MONTIEL PEREZ[2], CESAR CASTREJON PERALTA[3], LUIS ENRIQUE ANDRADE GORJOUX[4], JOSE ALBERTO TORRES LEON[5]

Instituto Politécnico Nacional, Centro de Investigación en Computación,
Laboratorio de Robótica y Mecatrónica, México
E-mail: [1]amaldonador2021@cic.ipn.mx, [2]yalja@ipn.mx, [3]castrejonp2021@cic.ipn.mx,
[4]landradeg2022@cic.ipn.mx, [5]jtorresl2019@cic.ipn.mx

ABSTRACT

In the present article, a simulation of a robotics application based on random walk is described. Typically, robots employ navigation algorithms based on planning and adjustments with respect to their working environment. As a specific case for mobile robots with two-dimensional navigation, the navigation algorithm is implemented with the characteristics of a random walk. The route planning module is a stochastic process in which it operates with random numbers following a uniform distribution as input data to the navigation module, and the outputs are movement instructions, both in direction and distance of advance. As a test, a closed scenario is used with 1, 2, and up to 5 robots operating simultaneously. The objective is to cover the area of the scenario. The results include the percentage of visited area, as well as the decision metrics of the stochastic processes involved in navigation. This navigation algorithm is oriented towards applications in service and surveillance robots.

Keywords: *Random Walk, Stochastic Processes, Simulation, Robotics, Random Numbers.*

1. INTRODUCTION

Considering that technological trends are moving towards the Internet of Things and Smart Cities, robotics plays a predominant role. Clear examples include the autonomy of devices and services for both the masses and households. Currently, there is a variety of service robots with multiple applications, such as food preparation, cleaning, surveillance, and more. In this case, mobile robots were studied, which, in their general configuration, have a physical and movement structure, control and power electronics, an electronic data analysis stage, and, if applicable, a communication stage.

In general, the applications of mobile robots are based on algorithms with artificial intelligence, including pattern recognition and decision-making based on sensor measurements. This implies that the robot must have a navigation algorithm that allows it to move, either in response to stimuli or autonomously. The complexity of these algorithms and the quantity or diversity of sensors included in the robots increase their cost and maintenance difficulty [1], [2].

In public spaces where there is an unpredictable flow of people over varying periods, surveillance or cleaning activities may be predominant. In the case of surveillance, if predefined routes are used for either a security guard or a surveillance robot, there is no guarantee of an increase or decrease in incident detection. Considering that incidents may occur opportunistically to some extent randomly or within a context based on opportunity and the surveillance route [3] [4], it is not advisable to employ a navigation algorithm that could be predictable or routine in generating travel routes [5], [6].

The present work describes a navigation algorithm based on a random walk for use in a service robot designed for situations where the robot needs to visit or cover a surface without considering time optimization but rather following a random pattern of visiting each point on the surface. This approach is suitable for potential applications such as surveillance or cleaning within defined areas. For instance, it could be applied in settings like airport waiting rooms, commercial zones, or even urban areas.

2. METHOD

A simulation of robot navigation using a basic random walk algorithm with random numbers following a uniform distribution is presented. The algorithm's test scenario is bounded. Simulation tests are conducted in an obstacle-free scenario. For each scenario, conditions are set for 1 to 5 robots simultaneously. The test variable is the percentage of the scenario visited concerning both time and the number of robots operating concurrently. The simulation is programmed in the Python language.

Below, the random walk algorithm and the definition of the robot's movement direction based on random numbers are described. This algorithm includes details such as the initial position, movement direction, step or path length the robot will move, and handling collisions of the robot with the scenario boundary or other robots.

2.1 Algorithm

Random walk algorithms have a wide range of variations and applications [7], [8]. A random walk is a model of random, independent, and identically distributed events representing the movement of an object in a space.

A random walk is a set of integer numbers that evolve in discrete time $\{x_k: k=0, 1, 2,...\}$. In the simple case, if the position evolves to the left or right, then each subsequent state will have a probability p or q, respectively, such that $p+q=1$. x_k is the state of the stochastic process at time k, valid for any $k>=0$ and for any integers i and j. These probabilities can be expressed as follows:

$$P(x_{k+1} = j | x_k = i) = \begin{cases} p & if\ j = i+1 \\ q & if\ j = i-1 \\ 0 & another\ case \end{cases} \quad (1)$$

As the probabilities do not depend on k, they are said to be homogeneous over time, meaning they remain the same at any time value k. Based on the properties of Markov, the future state of the process depends solely on the current state and not on previously visited states [9]. Therefore, the random walk can be defined as $x1, x2, ...$; a sequence of random values, independent and identically distributed. Thus, $P(x=+1)=p$ and $P(x=-1)=q$, also satisfying $p+q=1$. Then, for $n\geq 1$, it is defined as:

$$x_k := x_0 + \xi_1 + \xi_2 + \cdots + \xi_k \quad (2)$$

Without loss of generality, we can define $x_0=0$. With this, some statistical properties of the process can be derived. For any $k\geq 0$, the expected value is:

$$E(x_k) = \sum_{i=1}^{k} E(\xi_i) = nE(\xi)$$

$$E(x_k) = n(p-q) \quad (3)$$

On the other hand,

$E(x)=p+q=1$ y $E(x)=p-q$ it is necessary $Var(x)=1-(p-q)2=4pq$. Therefore, the variance is:

$$Var(x_k) = \sum_{i=1}^{k} Var(\xi_i) = nVar(\xi)$$

$$Var(x_k) = 4npq \quad (4)$$

If $p>q$, the walk takes steps to the right with a higher probability, so the average state after k steps is a positive number. If $p<q$, the average final state of the walk after k steps is negative, indicating a tendency towards the left. In both cases, the variance increases as the number of steps k grows. The greater the number of steps k, the greater the uncertainty about the final position of the process. When $p\neq q$, the walk is called asymmetric. When $p=q=½$, it is called symmetric, and on average, the process remains in its initial state, as the expectation $E(x_k)=0$. However, the variance of p is $Var(x_k)=n$, which corresponds to the maximum value of $4npq$, for p in the interval (0,1), [9], [10], [11].

For the simulation scenario, the robot is square and initially placed at a random position within the scenario. With these considerations, Algorithm 1 comes into operation.

Algorithm 1: Random walk.

Input:
Output: Visited area of the scenario at time t

```
1:  i ← 1
2:  At ← 0
3:  Positioni ← GenerateRandomPair
4:  repeat
5:    Path ← GenerateDirection(Generate Random)
6:    while DetectObstruction(Path) = false or End
      Path(Path) = false
7:      MoveOneStep(Path)
8:      At ← CalculateVisitedArea(Path)
9:    end while
10:   Position ← StopStep(Path)
11: until SimulationEnds
12: return
```

Variables are initialized, one of which is the percentage of visited area counter, steps 1-3. A loop is started, steps 4-12, in which a random movement direction is obtained. This direction is used to calculate the trajectory from the current point to the

final point of the path, and the step size remains constant throughout the simulation, steps 5-6.

Then, a nested loop begins, steps 7-10. The loop continues as long as there is no collision with a wall of the scenario, a collision with another robot occurs, or the path is completed, step 7. As the robot moves along its path, a count of the percentage of visited area is maintained, and if an area has already been visited, it is not added to the counter, steps 8 and 9.

Once the nested loop is finished, the robot is stopped, and its current position is recorded, step 11.

Steps 5 and 6 are executed again to generate a new direction and path based on a random number. The steps of the algorithm are then executed again until the termination condition is met.

If a collision is detected during the nested loop (step 7), the nested loop terminates, meaning the robot stops, the current position is recorded (step 11), and the algorithm of the outer loop begins again.

The termination conditions of the algorithm are by time, by percentage of visited area, or by the number of steps.

2.2 Random numbers

The random walk algorithm requires random numbers to determine a movement direction at the beginning of each step.

Random numbers were generated using the ROOT data analysis framework with a uniform probability distribution. It's worth noting that there are other distributions available, such as normal, geometric, Landau, among others [12] [13].

The movement direction of the robot is determined by a random number sampled from a uniform distribution to ensure that there are no biases or weights associated with specific directions.

Figure 1 displays the histogram of the distribution of 5 million integer random numbers ranging from 0 to 120, which are used in the simulation.

2.3 Initial position

The random walk algorithm starts with an initial position of the robot, which is obtained using the random number generator from the Python library *numpy* (the *numpy.random* method). This method provides a floating-point number in the half-open interval [0.0, 1.0).

Once the initial position in the scenario is determined, the representation of the robot is a pixel matrix of size *nXm*, where *n* and *m* represent the width and length. In the case of the simulation, the dimensions of the robot are set to *11X11* pixels.

Subsequently, the movement direction of the robot must be generated.

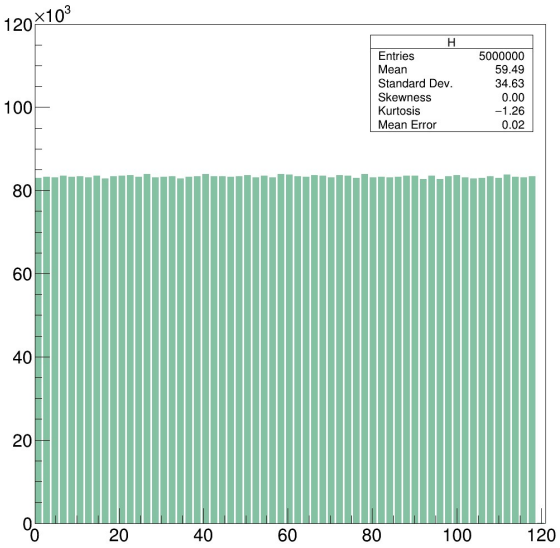

Figure 1: Histogram of the distribution of 5 million random numbers.

2.4 Robot movement

The robot is considered to be in a stationary state both initially in the algorithm and when it needs to change direction. Therefore, before it begins its movement, its direction must be defined, and this direction is random. There are eight possible directions for the robot to move independently of its orientation in its static position, as shown in Figure 2. Each direction has an equal probability of being selected.

In the random walk algorithm, it's important to ensure that the random direction selection process does not transform its uniform distribution into another distribution with a weighted bias (such as the cases of normal, binomial, etc. distributions). This is to maintain the randomness and equal probability of each direction being selected during the random walk.

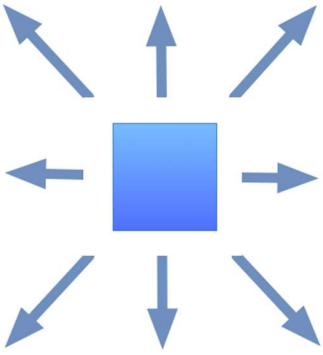

Figure 2: The eight possible movement directions of the robot.

For example, if a random number with a uniform distribution is used to represent an angle in the polar coordinate plane, which has a cyclic property. Then, angles of 0 and 360 degrees represent the same direction, and therefore, this direction will have a higher probability of being selected compared to other directions represented by angles. This results in a distribution that resembles a normal distribution from a uniform distribution.

The movement direction is determined using a roulette technique [14]. The random numbers are in the range of 0.0 to 1.0. This range is divided into 8 sections, with each section corresponding to a direction as indicated in Figure 2.

In the case of the robot's static state, if it has an initial coordinate *(x, y)* and you want to move it in one of the four cardinal directions (up, down, left, right), you can determine the final coordinate as follows:

1. For moving up:
   The final coordinate will be *(x, y + 50)*.
2. For moving down:
   The final coordinate will be *(x, y - 50)*.
3. For moving left:
   The final coordinate will be *(x - 50, y)*.
4. For moving right:
   The final coordinate will be *(x + 50, y)*.

These calculations involve simply adding or subtracting 50 pixels to the respective coordinate depending on the desired direction.

For the diagonal directions (up-right, up-left, down-left, and down-right) as shown in Figure 2, you can use a right triangle approach to determine the final coordinates. In these cases, you should add or subtract 35 pixels to both the *x* and *y* coordinates of the initial position, depending on the direction, to find the final coordinate of the step. Here's how you can calculate it for each diagonal direction:

5. For moving up-right:
   The final coordinate will be *(x + 35, y + 35)*.
6. For moving up-left:
   The final coordinate will be *(x - 35, y + 35)*.
7. For moving down-left:
   The final coordinate will be *(x - 35, y - 35)*.
8. For moving down-right:
   The final coordinate will be *(x + 35, y - 35)*.

These calculations consider a right triangle with legs of 35 pixels each, resulting in the diagonal movement of 50 pixels.

Using the initial and final coordinates of the robot's step, you can construct a linear equation. With the direction conditions, you can determine the coordinates by which the robot will move. This linear equation will help you calculate the exact path the robot will take based on its initial and final positions and the desired direction of movement.

Each step of the robot covers 50 pixels. At each pixel of movement of the robot or robots in the scenario, a check for "valid movement" is performed. This means verifying whether collisions occur or not. Robot collisions can happen when reaching the boundary of the scenario or when colliding with another robot along the trajectory, as shown in Figure 3. In either case, the robot or robots stop, and a new movement direction is obtained.

The simulation scenario is a 1000x1000 pixel square. This scenario is represented by a matrix in CSV format, which is initialized with integer values of *0*. These values change to *1, 2, ..., n*. The number *n* represents the number of robots traveling in the scenario. This matrix likely serves as a representation of the occupancy or presence of robots in different positions within the scenario.

When robot 1 moves one step of 50 pixels in a direction, the pixels that are part of its trajectory are represented in the matrix with integer values of 1. For robot 2, these pixels are represented with the value 2 in the matrix, and so on for n robots. This way, the matrix serves as a representation of the positions occupied by different robots in the simulation.

In Figure 4, a section of the matrix is displayed. Based on the conditions of the algorithm, the robot can revisit the same area multiple times, resulting in updates to the values in the matrix over time. As these updates to the matrix values occur, a count of unvisited pixels (value 0 in the matrix) is maintained relative to the total number of pixels in the scenario. This calculation provides a percentage of visited area over time, as described in Equation 5.

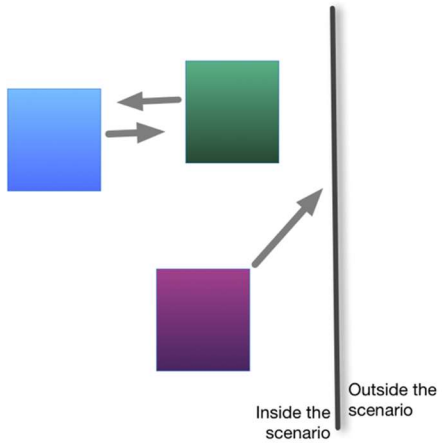

Figure 3: Types of collisions in the simulation: Collisions between robots and collisions with the wall that bounds the scenario.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

col14	col15	col16	col17	col18	col19	col20
1	1	0	0	0	0	1
1	1	1	0	0	1	1
1	1	1	1	1	1	1
1	1	2	1	1	1	1
1	2	2	1	1	1	1
2	2	2	1	1	1	2
2	2	2	2	1	2	2
2	2	2	2	2	2	2
2	2	2	3	2	2	2

Figure 4: Section of the CSV matrix recording the pixels in the scenario visited by the robots.

$$\%visited\ area = \frac{visited\ pixels}{total\ pixeles} x100 \qquad (5)$$

3. SIMULATION AND TESTING

Based on the mentioned conditions, multiple tests are conducted. To visualize the repeatability of the behavior of the graph "*time versus percentage of visited area*" by the robot in the scenario, the same experiment is repeated several times. This experiment involves using 1 robot, and in each experiment, there is a random starting point and different sets of random numbers for directions. This results in the graphs shown in Figure 5, demonstrating the similarity or repeatability of the results.

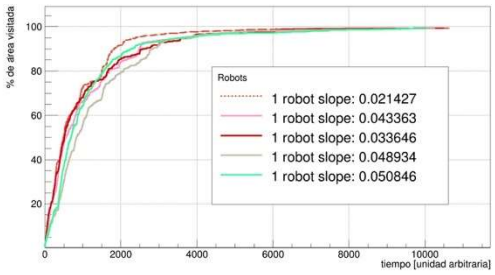

Figure 5: Percentage of area visited over time for 1 robot, in multiple simulation runs.

In the interval of *20-40%* of visited area, the graph is considered linear, and an approximation of the slope is calculated. The slope of the graph indicates how quickly the scenario is visited. After the linear interval, there is a curve that slows down the speed at which the remaining area is covered. Therefore, under the same conditions and in several experiments, there is a standard deviation of *0.004627* with respect to the slopes of each curve. This standard deviation provides a measure of the variability or spread of the slope values in different experiments.

The same process is applied to experiments with 2 robots moving simultaneously in the scenario, as well as for experiments with 3, 4, and 5 robots. The experiments are repeated under the conditions described for 1 robot, and the data is obtained as shown in Table 1. This process helps analyze and compare the behavior and deviations for different numbers of robots operating simultaneously in the scenario.

Table 1: The standard deviation of the experiments.

Number of robots	Standard deviation
1	0.004627
2	0.008017
3	0.007640
4	0.004606
5	0.008901

For each case of the number of robots operating simultaneously in the scenario, the standard deviation defines their behavior in the graph of "*time versus percentage of visited area*". In other words, if the random walk algorithm is executed multiple times with *k* robots moving simultaneously in a scenario, the graphs of "*time versus percentage of visited area*" for each run of the algorithm will be similar to each other and will fall within the standard deviation for that number *k* of robots. This standard deviation helps characterize the variability or spread in the behavior of the algorithm for different robot counts.

The first experiment using the random walk algorithm involves a single robot traveling in a 1000x1000 pixel scenario without obstacles and with a limit of 100,000 steps. For the analysis of the behavior of each robot in the simulation, the directions of movement are grouped into pairs of opposite directions: up-down, left-right, up-left-down-right, up-right-down-left. This is done so that the pairs of directions are in line with the one-dimensional model expressed by Equation (1). Counts are made for right turns, left turns, upward steps, and downward steps with respect to the robot's orientation at the time of counting, considering the 8 possible directions the robot can have. The count of down-up turns is shown in Figure 6. With the measured data and the one-dimensional model given by Equation (1), histograms are generated to characterize the behavior of the robot's random walk, as shown in Figure 7. In the count, there are approximately 55% of downward directions and 45% of upward directions. It's worth noting that the model is one-dimensional.

Journal of Theoretical and Applied Information Technology
15th December 2023. Vol.101. No 23

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

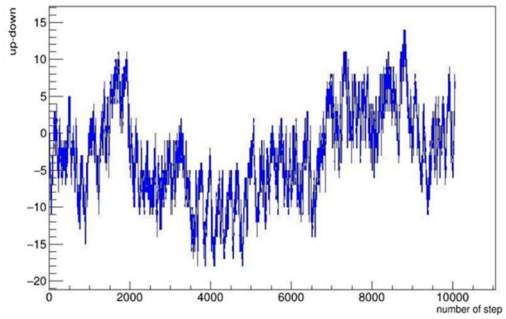

Figure 6: Count of down-up direction decisions in a one-dimensional representation based on the Equation (1).

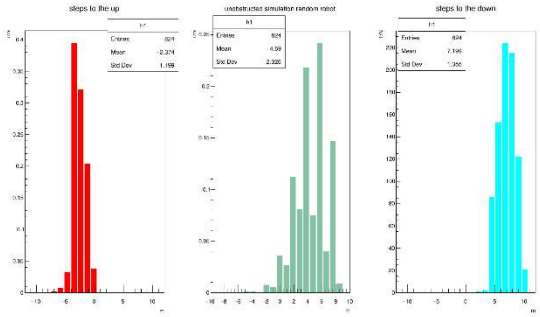

Figure 7: Histogram for down-up direction decisions in a one-dimensional representation.

The same process is carried out for the other directions of movement, yielding similar results since the same algorithm is used.

Based on the definition of the random walk model given by Equation (1), the counts of upward-downward, left-right, upward-rightward-downward-leftward, and upward-leftward-downward-rightward directions follow a binomial distribution. This is because to decide the robot's direction of movement, random numbers with a uniform distribution are used, and therefore, each direction has an equal probability of occurrence.

Experiments are conducted, operating 1 and simultaneously 2, 3, 4, and 5 robots with identical rules and random initial positions. Multiple tests are conducted under the same conditions to observe if there is repeatability, and the results indeed show variations of less than 3%. In the tests with a different number of robots, it is observed that 100% of the visited area is not achieved but rather 99.9%, as seen in Figure 8, which corresponds to random initial positions and 100 steps of progress for 5 robots. This analysis helps assess the behavior of the algorithm with varying numbers of robots and its repeatability.

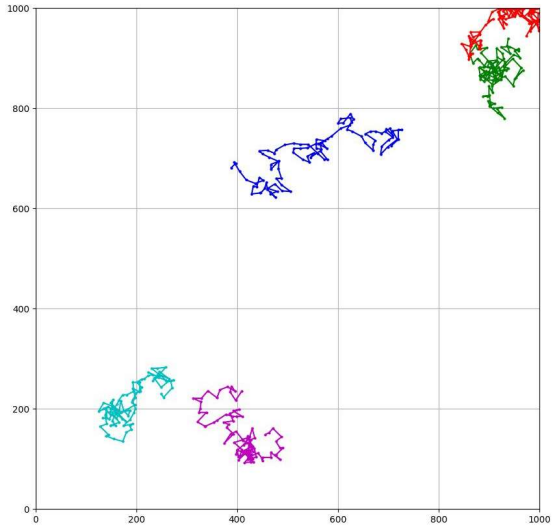

Figure 8: Random initial positions for 5 robots and their trajectories generated after 100 steps.

Starting from step or time *0*, the percentage of area occupied in the scenario increases rapidly, with the rate of occupancy decreasing to approximately *85%* for any experiment involving 1, 2, or up to 5 robots. If we linearize the increase from *0%* to *85%* of the occupied area and calculate the slope, the rate of increase in occupancy for 1 robot is lower than for 5 robots operating simultaneously. It is expected that more area is covered rapidly with 5 robots than with 1 robot. The slope is interpreted as the constant for the rate at which the area of the scenario is visited, regardless of whether previously visited areas are revisited at times. This can be observed in the graphs in Figure 9.

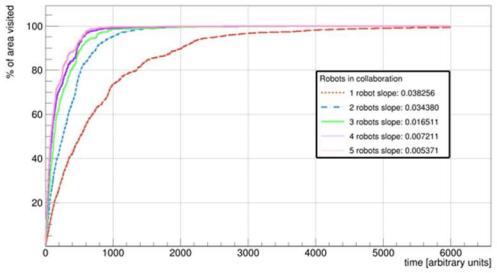

Figure 9: Percentage of visited area for experiments with 1, 2, 3, 4, and 5 robots working simultaneously in a 1000x1000 pixel scenario.

On the x-axis, we have time, which has an arbitrary time unit since each unit corresponds to a step, the discrete state of the system's evolution to decide the next direction of movement, and the data is recorded to generate the graphs. On the y-axis, we have the percentage of visited area, which, as

explained, is not counted multiple times and is calculated using Equation (5). Based on the observations, we calculate the slopes in the indicated region, as shown in Table 2. These results show that from 3 robots operating simultaneously, the speed at which the area of the scenario is covered is no longer faster with 4 or 5 robots for a *1000x1000* pixel scenario. This analysis provides insights into the efficiency of multiple robots in covering an area, Figure 10.

Table 2 Slope corresponding to the highest increase in the area visited in the experiments.

Number of robots	Slope
1	0.038256
2	0.034380
3	0.016511
4	0.007211
5	0.005371

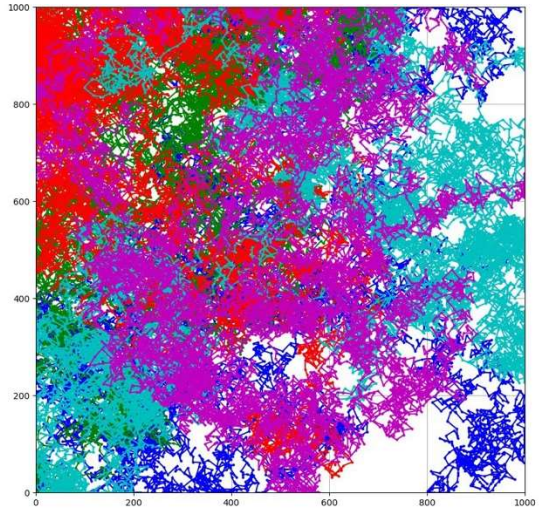

Figure 10: Random walk with a uniform distribution of random numbers for 5 robots, each color represents a robot.

To find a correlation between the speed or slope of the visited area in different combinations of robots working simultaneously, a polynomial regression of degree 1, 2, 3, and 4, as well as exponential, logarithmic, and power types, is performed. As a result, the best fit is a 4th-degree polynomial, which is expected since there are up to 5 robots. However, the best fit is likely to be exponential because it is independent of the amount of slope data or robots used in this calculation. To confirm this, and as future work, a wider range of robot combinations working simultaneously should be tested. For now, there is an approximation of the type of function that governs the speed of robot visits, Figure 11.

Finally, a test was conducted under the same movement rules, using a Gaussian distribution with a standard deviation of 1.5 for the random numbers. In this case, *100,000* steps were taken with 1 robot in a *1000x1000* pixel scenario, which resulted in a concentration of visits to segments in the scenario. At some point, the robot moved to another area and did the same. This behavior is due to the nature of random numbers, where the robot's movement is biased, even though the 8 possible directions have a quasi-binomial distribution, Figure 12.

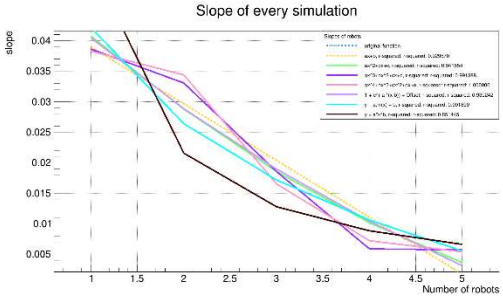

Figure 11: Regression of the slopes or speed factor of scene visits for robot configurations.

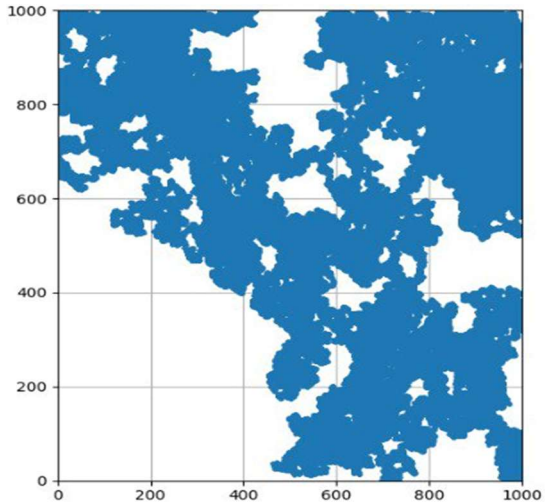

Figure 12: Walk of 1 robot with random numbers following a Gaussian distribution.

4. CONCLUSIONS

The theory and statistical model of random walk are applied to the field of service robots, where time and route efficiency are not critical factors. This is evident in the case discussed in this article, where the robot's objective is to cover a specific area. The

random walk algorithm with random numbers following a uniform distribution ensures that, at some time '*k*', the scenario's area will be covered. For a *500x500* pixel scenario, at least 3 robots are required to cover up to 85% of the area in the shortest possible time.

The analysis of left, right, up, and down turns in the random walk algorithm provides information about the distribution of movement directions and characterizes the behavior of robots. The variability in movement direction generated by using random numbers with a uniform distribution can lead to unpredictable robot behavior, which can be beneficial in surveillance and cleaning applications in dynamic environments.

Multiple tests were conducted with different numbers of robots in a 500x500 pixel scenario, observing that the percentage of visited area increases rapidly at the beginning and then decreases its rate of occupation. The speed of area occupation becomes less dependent on the number of robots when a threshold is exceeded, suggesting that adding more robots may not be significantly beneficial in terms of coverage speed in large scenarios.

Simulating robots using the random walk algorithm is a valuable tool for studying and understanding the behavior of robots in dynamic and random environments. The repeatability of simulation results, with standard deviations of less than 3%, suggests that the random walk algorithm is robust and consistent across different experiments.

It is important to emphasize that the results reported in this work present a random walk algorithm specifically designed for service robots requiring complete coverage of an area without optimization for time efficiency. This characteristic makes the algorithm suitable for applications such as surveillance, cleaning, or disinfection of spaces. The results and arguments put forth reinforce that the stochastic nature of the algorithm does not necessitate knowledge of the dimensions or surface geometry of the environment. Unlike navigation algorithms reported in the literature for mobile robots, there is no need to create predefined travel paths. This work represents a minimalist approach to navigation in unknown environments. Notably, it avoids the use of complex algorithms such as exploration with image or video analysis, reinforcement learning-based scene reconstruction, Simultaneous Localization and Mapping (SLAM) algorithms, bioinspired perception algorithms, among others.

As a continuation of the presented work, the next step would be to implement the algorithm on real robots and evaluate their random characteristics for practical service applications. This would involve taking the theoretical insights and findings from the simulations and applying them in real-world scenarios.

A real-world application scenario could be a waiting room at an airport. Such spaces have defined areas, furniture, and people generally spend a short time in the room. However, their movements can be erratic, forming queues, among other behaviors. This scenario involves dynamic elements. A robot equipped with minimal presence sensors (without cameras) and the random walk algorithm could effectively perform tasks continuously, whether for cleaning or surveillance. Such scenarios are suitable for practical implementation and could be explored and reported in future work.

REFERENCES:

[1] A. A. Nippun Kumaar, Sreeja Kochuvila, ”Reinforcement learning based path planning using a topological map for mobile service robot”, *2023 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT),* Amrita School of Computing India, July 2023, pp. 1-6.

[2] M. Chen, X. Wang, R. Law, M. Zhang, *“Research on the Frontier and Prospect of Service Robots in the Tourism and Hospitality Industry Based on International Core Journals: A Review”*, Behav. Sci., Vol. 13, No. 560, 2023, pp. 1-21.

[3] J. Wang, S. Lin, A. Liu, *“Bioinspired Perception and Navigation of Service Robots in Indoor Environments: A Review”*, Biomimetics, Vol. 8, No. 350, 2023, pp. 1-39.

[4] Paula Dootson, Dominique A. Greer, Kate Letheren, *“Reducing deviant consumer behaviour with service robot guardians”*, Journal of Services Marketing, Vol. 37, No. 3, 2023, pp. 276–286.

[5] Anbalagan Loganathan, Nur Syazreen Ahmad, *“A systematic review on recent advances in autonomous mobile robot navigation”*, International Journal Engineering Science and Technology, Vol 40, 2023,101343.

[6] Liyana Wijayathunga, Alexander Rassau, Douglas Chai, *“Challenges and Solutions for Autonomous Ground Robot Scene Understanding and Navigation in Unstructured Outdoor Environments: A Review”*, Appl. Sci., Vol 13, 2023, 9877.

[7] Chen Chen-Yu, Chang Kuo-Chou, Ho Shing-Hua, *“Improved framework for particle swarm*

optimization: Swarm intelligence with diversity-guided random walking", Expert Systems with Applications, Vol. 38, 2011, pp. 12214–12220.

[8] Cameron Musco, Hsin-Hao Su, Nancy A. Lynch, *"Ant-inspired density estimation via random walks*", Proceedings of the National Academy of Sciences of the United States of America, 2017, Vol. 114, No. 40, pp. 10534-10541.

[9] Q. Liu and G. -C. Zhang, *"Random walk model in the application of game"*, 2010 International Conference on Machine Learning and Cybernetics, Qingdao, China, 2010, pp. 1875-1880.

[10] F. Rong, L. Shasha, L. Fang, L. Aimin, X. Qingzheng, *"Random walk model based statistical analysis and application on transport simulation"*, 2018 Chinese Control And Decision Conference (CCDC), Shenyang, China, 2018, pp. 5429-5434.

[11] Y. Gao, T. Fan and S. Cai, *"Navigation in Complex Networks Using Random Walk Theory and Principal Component Analysis"*, 2019 16th International Computer Conference on Wavelet Active Media Technology and Information Processing, Chengdu, China, 2019, pp. 349-354.

[12] James M. Hill, M. Gulati Chandra, *"The Random Walk Associated by the Game of Roulette",* Journal of Applied Probability, Vol 18, No. 4, 1981, pp. 931–36.