EMITTER International Journal of Engineering Technology
Vol. 13, No. 2, December 2025, pp. 268~291
DOI : 10.24003/emitter.v13i2.994

The Impact of Social Force Model Parameters On Frontier-Based Exploration Performance

Asyam Irsyad[1], Bima Sena Bayu Dewantara[2], Setiawardhana[2]

[1]Department of Electrical Engineering, Politeknik Elektronika Negeri Surabaya, Surabaya, Indonesia.
[2]Department of Computer Engineering, Politeknik Elektronika Negeri Surabaya, Surabaya, Indonesia.
Corresponding Author: bima@pens.ac.id

Received September 11, 2025; Revised October 20, 2025; Accepted November 26, 2025

Abstract

Autonomous exploration is one of the most challenging tasks in mobile robotics, particularly in environments that contain dynamic obstacles and require fully autonomous mapping without human intervention. This study addresses the dual problem of enabling navigation in the presence of potential static obstacles and achieving autonomous map building. To solve this, we utilize the Social Force Model (SFM), which offers a behavior-based approach suitable for dynamic and uncertain environments. The objective of this research is to investigate how different SFM parameters—Gain (k^s), Radius (r_R), and Effective range (ψ^s)—influence the effectiveness of autonomous exploration. Experiments were conducted using a TurtleBot3 robot in a simulated 155 m^2 environment, where various parameter combinations were tested. Evaluation metrics included mapping completion, failure types, travel distance, and exploration duration. Results indicate that tuning the SFM parameters significantly affects the robot's ability to explore autonomously and avoid obstacles. Extremely low parameter values led to collisions, while excessively high values caused unstable or inefficient behavior. The Radius parameter had a major impact on spatial awareness, and moderate effective range values contributed to stable tracking. Furthermore, higher frontier sensing latency resulted in longer exploration times. This study provides practical insights into the sensitivity of SFM parameters and offers guidance for optimizing navigation systems for fully autonomous exploration in both simulated and real-world settings.

Keywords: Autonomous exploration, Social Force Model (SFM), Frontier-based exploration, TurtleBot3, Obstacle avoidance, SLAM.

1. INTRODUCTION

Autonomous exploration represents a critical challenge in mobile robotics, particularly in scenarios requiring robots to operate effectively within unknown and dynamic environments. Out of all the methods, frontier–based exploration has become the most popular one, in which robots locate

and target frontiers, or the borders between mapped and unmapped areas, as exploration targets [1], [2], [3], [4], [5], [6], [7], [8], [9]. Since Yamauchi's foundational work [8], this methodology has gained widespread adoption due to its balanced approach to maximizing coverage while maintaining exploration efficiency.

Many improvements have been made to the methods of finding and choosing frontiers thanks to research in this area. Some important new ideas are fast, computer–friendly frontier detection methods that make sure navigation is safe and targets can be reached [8], [10]. Adding robot direction to frameworks for choosing targets has also been shown to make a big difference in how smoothly paths are formed and how naturally robots move [10]. In dynamic environments, robots must continuously adapt their exploration strategies to avoid redundant coverage and potential hazards. Frontier-based exploration, while effective, can struggle in rapidly changing settings without real-time path adjustment. As shown by [9], adaptive motion planning is crucial in dynamic urban environments, where robots adjust their trajectories in response to human movement—highlighting the broader need for continuous path re-evaluation in evolving maps.

Recent studies have further refined this paradigm. For instance, [11] proposed an enhanced frontier-based exploration model that optimizes the cost function of frontier selection through real-time map updates, leading to higher exploration efficiency in complex spaces. Similarly, [12] compared various exploration strategies—particularly frontier-based and RRT-based methods—demonstrating that frontier-based exploration maintains superior coverage consistency in cluttered or partially structured environments. These findings emphasize that frontier-based methods remain highly competitive, especially when properly tuned for computational and spatial efficiency.

In order to get around these problems, people have looked into combining frontier-based discovery with complex multi–step path planning and navigation methods [5]. Even though these merged methods often give better results, they often require a lot of extra work to be done on the computer. Researchers [3] discuss that while heuristic planning frameworks for exploration improve efficiency, the challenge of processing large amounts of data remains a bottleneck, affecting real-time performance, which is a key issue in dynamic situations.

The Social Force Model (SFM) is a hopeful framework for navigation in these kinds of situations because it combines goal–directed movement with forces that push people away from obstacles in a very nice way. SFM was first originally developed to model pedestrian dynamics [10], but it has since been successfully applied to robotics problems like human–aware navigation [13], [14], [15] and dynamic obstacle avoidance. But using SFM in self–driving exploration systems comes with its own problems. For example, exploration success depends on finding the right balance between pursuing the frontier aggressively and staying safe in unstructured environments.

Recent research also demonstrates the versatility of SFM. Meanwhile, [16] applied the Social Force Model to robot navigation on inclined terrains, showing that parameter tuning significantly affects both stability and obstacle avoidance performance. This reinforces the importance of systematic parameter evaluation—particularly for parameters such as Gain (k), Radius (rR), and Effective range (ψ)—in different operating contexts.

Therefore, the problem addressed in this research is the lack of systematic understanding of how Social Force Model (SFM) parameters affect the performance of autonomous exploration using frontier-based approaches, especially in dynamic environments. This paper looks into the study questions:

1. What effects do certain SFM parameters (k, r_R, ψ) have on important exploration metrics like coverage time, path length, and collision frequency?
2. What are the main trade–offs between being aggressive in discovery and being safe during operations in different parameter settings?

We answer these questions by doing a lot of experiments with a ROS2–based application in both simulated and real–world settings. The SLAM–toolbox was chosen for our work because it has a lot of useful features [17], [18]. This toolbox simplifies SLAM implementation by providing a ready-to-use framework, eliminating the need to develop algorithms from scratch. What's more, it also includes useful occupancy grid mapping features.

To summarize, the problem addressed in this research is the lack of systematic understanding of how Social Force Model (SFM) parameters affect the performance of autonomous exploration using frontier-based approaches, especially in dynamic environments. While SFM has been applied in navigation contexts, its role and parameter sensitivity in exploration tasks remains underexplored. In this research our contributions include:

1. Empirical evidence quantifying SFM parameter sensitivity in exploration tasks, establishing clear relationships between k, rR, ψ and key performance metrics.
2. Practical insights into optimal parameter tuning strategies for SFM–based navigation systems deployed in autonomous robotic exploration scenarios.

The novelty of this work lies in integrating SFM with frontier-based exploration and systematically analyzing parameter effects in both simulated and real-world scenarios using ROS2, which has not been extensively studied in prior literature.

2. RELATED WORKS

Related research in the field of autonomous exploration using Simultaneous Localization and Mapping (SLAM) showcases various approaches to addressing the challenges of robot navigation in dynamic environments. Several key works in this domain involve SLAM techniques to facilitate accurate mapping in unknown environments, as well as real-time sensor data processing for dynamic obstacle avoidance.

1. Map Making in Social Indoor Environments Through Robot Navigation Using Active SLAM. The authors [19] introduced a solution for mapping dynamic indoor environments using Active SLAM adapted for social settings. Their approach integrates orientation tracking with adaptive squashing functions in artificial neural networks to improve mapping efficiency, highlighting the importance of adapting to environmental changes. Experimental results show that this method outperforms exhaustive search techniques for mapping in human-populated spaces with dynamic obstacles.
2. Autonomous Robotic Exploration and Navigation System Using Tri-Layered Mapping and Geometrical Path Planning Techniques. The authors [20] developed an autonomous exploration system leveraging SLAM-based mapping and geometric path-planning techniques for robot navigation. Their approach advocates the use of a tri-layered mapping technique to generate more accurate maps, as well as a geometrical boundary-based path planning algorithm to enhance robot mobility in dynamic environments. Experimental results demonstrate that this system outperforms conventional path-planning techniques in handling complex exploration tasks.
3. Bayesian Mapping-Based Autonomous Exploration and Patrol of 3D Structured Indoor Environments with Multiple Flying Robots. The authors [21] proposed a framework for autonomous exploration and patrol using cooperative flying robots with Bayesian-based mapping. They developed a 3D occupancy grid map and exploration algorithm designed to maximize the information gain of the map. This study integrates path planning techniques to avoid dynamic obstacles, offering valuable insights for multi-robot exploration in larger, more complex environments.
4. In addition to frontier-based exploration, several studies have investigated graph search algorithms for autonomous exploration. For example, [22] proposed optimized graph search algorithms that combine depth-first search (DFS) and breadth-first search (BFS) with Dijkstra's shortest path method to enable efficient navigation in unknown environments. Their hybrid approach leverages DFS or BFS for initial boundary detection and then applies Dijkstra's algorithm to minimize the return path, thus achieving faster and more efficient coverage. The experimental results with real robots demonstrated that the DFS+Dijkstra hybrid algorithm outperformed other methods in terms of exploration distance and time. This work highlights the relevance of graph-based approaches as an alternative to frontier-based exploration strategies, providing useful insights for comparison with the method proposed in this study.

These works demonstrate significant advancements in the field of autonomous exploration with SLAM and active mapping. Each study integrates SLAM with advanced navigation techniques to improve mapping accuracy and

exploration efficiency in dynamic environments, making them highly relevant to our research, which aims to optimize parameters in the Social Force Model (SFM) for autonomous navigation.

3. ORIGINALITY

This paper presents a novel exploration of the Social Force Model (SFM) in autonomous robotic navigation, specifically investigating how varying SFM parameters—Gain (k^s), Radius (r_R), and Effective range (ψ^s)—affect frontier-based exploration performance. While the Social Force Model has been widely applied in dynamic obstacle avoidance, this study uniquely focuses on how parameter tuning influences exploration efficiency, obstacle avoidance, and mapping completion in a dynamic environment. By conducting experiments with a TurtleBot3 robot in a controlled 155 m^2 simulated environment, the study provides new insights into the optimization of SFM parameters, offering practical guidance for improving autonomous exploration in real-world scenarios. The originality lies in the specific combination of these parameters and their direct impact on the robot's exploration capability, providing a novel contribution to the field of autonomous navigation.

4. SYSTEM DESIGN

This study utilizes a Differential Drive Mobile Robot (DDMR) to develop an autonomous exploration system based on Frontier–Based Exploration (FBE) and SFM for navigation. Simultaneous Localization and Mapping (SLAM) is employed to continuously build and update the map of the environment. A 2D LiDAR sensor, integrated into the robot, serves as the primary tool for mapping the surroundings and detecting obstacles. The SLAM Toolbox is used to generate an occupancy grid map, which is then processed to detect frontiers—transitional areas between known and unknown space—serving as candidate exploration goals.

The system operates in a continuous loop: the robot detects frontiers from the current map, selects a goal point, and navigates toward it using the SFM algorithm. Upon reaching the frontier, the robot updates the map using SLAM, and the process repeats until the entire environment has been fully explored. A detailed block diagram of the system is presented in Figure 1.

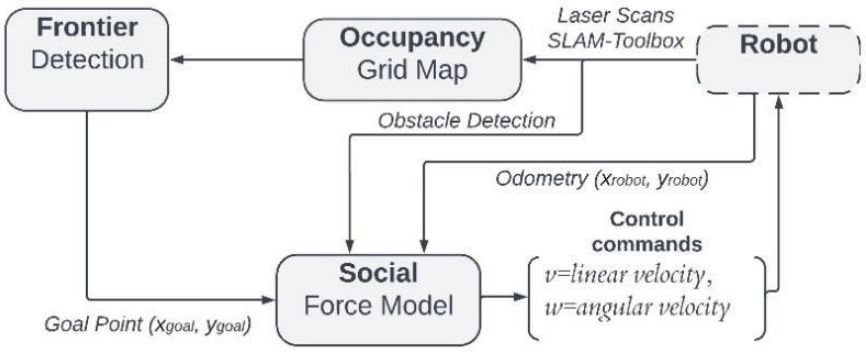

Figure 1. Block Diagram: Frontier-Based Exploration with SFM Navigation.

4.1 Simultaneous Localization and Mapping (SLAM)

SLAM or simultaneous localization and mapping, is a basic method that lets a mobile robot make a map of an unknown area while also figuring out where it is on that map. It uses sensor data, mathematical models, and algorithms to figure out what's going on around it and guess the robot's path in real time. SLAM is a key part of independent navigation because it lets robots work in places where previous maps aren't available or aren't reliable [23].

The SLAM process in this work is carried out using SLAM Toolbox, a well–known and widely–used SLAM system made available by the ROS 2 community. SLAM Toolbox is a very good way to do 2D mapping and localization tasks. It works especially well in structured indoor settings with fixed features. With this set of tools, the robot can make accurate population grid maps and stay reliably located in its surroundings [17], [18], [23].

This choice to use SLAM Toolbox instead of creating a SLAM algorithm from scratch was made because it saves time and is more reliable. Creating and testing a unique SLAM system would take a lot of time, especially to make sure it works correctly and stays stable in a variety of situations. SLAM Toolbox, on the other hand, has been tried and proven to work in the robotics community, so it was a good choice for this study.

A 2D occupancy grid map, which shows the surroundings as a matrix, is what SLAM Toolbox gives you as an output. Based on what the sensors see, each cell in the grid is marked as either filled, free, or unknown. This occupancy grid is the basis for the next step in the discovery process, which is called "frontier detection." This is where unknown areas that border known space are found as possible places to explore.

4.2 Robot Model

The robot used in this study is a Differential Drive Mobile Robot (DDMR), which consists of two independently driven wheels placed on either side of the chassis and one passive caster wheel in front for balance. Changing the speeds of the two drive wheels in this setup lets the robot move around. The DDMR's kinematic model is important for figuring out how the robot moves in reaction to control signals and for figuring out the right wheel speeds needed to get to a certain point in space.

The following equations are used to figure out the robot's linear and rotational speeds. These equations show how the angular speeds of the right (ω_R) and left (ω_L) wheels are related to their linear speeds (V_R and V_L shown in Equation (1) – (2), taking into account the radii of each wheel (r_R and r_L).

$$V_R = \omega_R \cdot r_R \tag{1}$$

$$V_L = \omega_L \cdot r_L \tag{2}$$

Its linear speed, v shown in Equation (3), is equal to the sum of its two wheels' linear speeds. Its angular speed, ω shown in Equation (4), is equal to half of the difference between their angular speeds:

$$v = \frac{V_R + V_L}{2} \tag{3}$$

$$\omega = \frac{\omega_R - \omega_L}{2} \tag{4}$$

These expressions tell the robot how to move around in its surroundings by using the rotation of its wheels. If you know the robot's linear and angular speeds, you can use the following set of differential equations to change its pose (position and direction) in a two-dimensional plane:

$$\dot{x} = v \cdot \cos\theta \tag{5}$$

$$\dot{y} = v \cdot \sin\theta \tag{6}$$

$$\dot{\theta} = \omega \tag{7}$$

In this case, (x, y) shows where the robot is in the world coordinate frame, and θ is the robot's angle of orientation with respect to a reference direction, which is usually the x−axis. These formulae are the basis for the robot's kinematic model and are very important for planning its path and controlling its movement in the navigation system. Figure 2 shows both these kinematic connections and the DDMR configuration in a visual way.

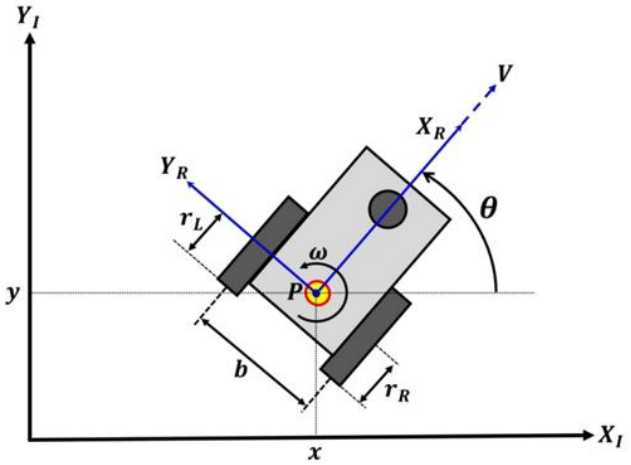

Figure 2. Illustration of Equations on the DDMR Robot [24].

4.3 Frontier Detection

Frontier detection is one of the most important techniques used in autonomous exploration to find possible target spots that tell a robot how to move in new places. This method finds cells on the edge of studied (known) and unexplored (unknown) areas, which is especially helpful when used with an occupancy grid. Frontier cells, which are these boundary cells, are very

important for setting research goals because they show the area where mapped and unmapped space meet [2], [10], [17], [18], [25], [26].

Frontier detection is used in this study using a matrix–based method. Each cell in the occupancy grid has a unique value: 0 means the cell is free (known to be empty), 1 means the cell is filled (known to be blocked), and –1 means the cell is unknown (not mapped).

A cell is a frontier if it has a value of –1 (unknown), is next to at least one open cell with a value of 0, and is next to at least one cell with a known value (not –1) [2], [5]. Algorithm 1 formalizes this process of classifying by looking at neighboring cells to figure out the frontier state of each individual cell.

Algorithm 1: isFrontierCell

```
Input: Matrix B, coordinates (x, y)
Output: True if (x, y) is frontier cell, otherwise False

if (x, y) is out of bounds or B[y, x] ≠ – 1 then
    return False;

Define neighbors as the 8 surroundings cells
   (including diagonals);
hasZeroNeighbor ← False;
allNeighborsUnknown ← True;
for each (nx, ny) in neighbors do
    if (nx, ny) is within bounds then
        neighborValue ← B[ny, nx];
        if neighborValue == 0 then
            hasZeroNeighbor ← True;
         if neighborValue ≠ –1 then
            allNeighborsUnknown ← False;

return (hasZeroNeighbor AND NOT
    allNeighborsUnknown);
```

Utilizing Algorithm 1, the system finds all frontier cells. Next, Algorithm 2 groups these cells into highly connected components (SCC). Kosaraju's Depth–First Search (Kosaraju_DFS) is used for this grouping. This method is good for finding connections in directed graphs [27]. Every SCC is made up of a group of border cells that are all connected. To get rid of noise and small pieces, Algorithm 2 keeps only parts that have more than eight cells [2].

The centroid is found for each good component found by Algorithm 2. The centers of these frontier groups are possible exploration goals that are saved in a list, the calculation of the centroid is shown in Equation (8). Then, the closest centroid to where the robot is now is chosen as the next exploration target. This makes the best use of both coverage and trip distance [10]. Figure 3 illustrates how frontiers are detected and subsequently converted into

frontier points, which serve as potential goal positions for the robot. From these points, the one closest to the robot is chosen as the next goal.

Algorithm 2: Detect Frontiers

Input: A Matrix B of size width × height
Output: List of centroids of strongly connected component

```
Create empty set frontier;
for x ← 0 to B.width – 1 do
    for y ← 0 to B.height – 1 do
        if is frontier cell then
            Add (x, y) to frontier;

SCC_list ← Kosaraju_DFS(frontier);
centroids ← [];
for each component in SCC_list do
    if component.size > 8 then
        centroid ← calculateCentroid(component);
        Add centroid to centroids;

return centroids;
```

$$centroid = \left(\frac{\sum_{i=1}^{n} x_i}{n}, \frac{\sum_{i=1}^{n} y_i}{n}\right) \quad (8)$$

Where centroid is coordinate (x, y) obtained by averaging the positions of all points along the frontier line.

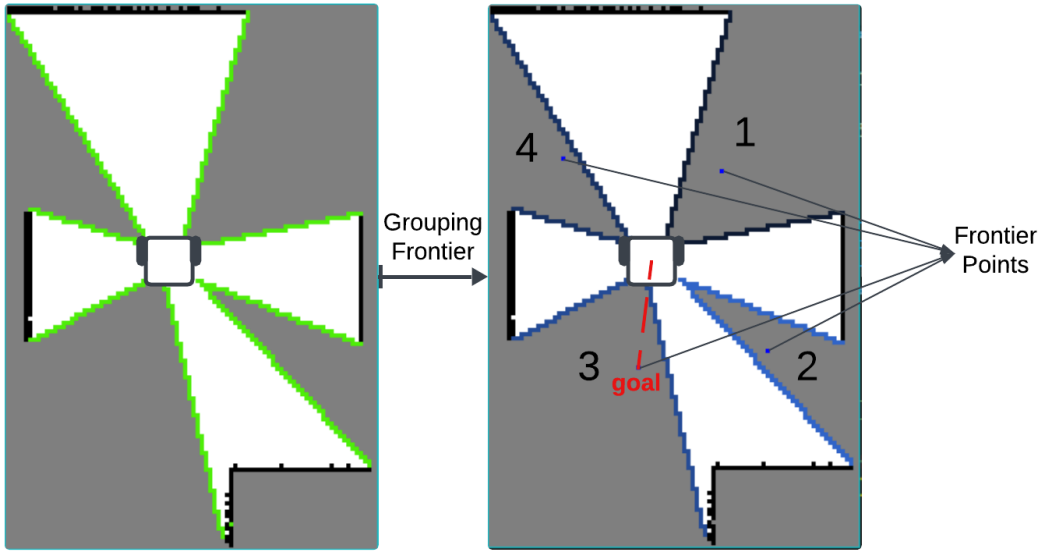

Figure 3. Illustration of the frontier detection.

Frontier recognition algorithms do a good job of finding and analyzing structural boundaries within the occupancy grid. This helps autonomous systems make full maps of their surroundings. In addition to robots, these cutting–edge detection methods have been used in areas like image segmentation and network connectivity analysis, where finding structural changes is just as important [10].

4.4 Social Force Model (SFM)

The Social Force Model (SFM) is a way to use math to figure out how people will move in changing settings by simulating "social forces." These forces show not only how things interact physically, but also how people act on purpose, like wanting to reach a goal while avoiding crashes and keeping personal space. SFM is used a lot in robotics for socially–aware navigation, which lets robots automatically avoid both fixed and moving obstacles.

The SFM is used in this study to figure out how much total navigation force is needed for the robot to move toward its goal while staying away from fixed objects. The total navigation force, written as $\vec{F}_{nav}$, is made up of two main parts: $\vec{F}_g$, which pulls the object toward the goal, and $\vec{F}_s$, which pushes it away from static obstacles. The equation (9) shows this relationship:

$$\vec{F}_{nav} = \vec{F}_g + \vec{F}_s \tag{9}$$

The attractive force $\vec{F}_g$ moves the robot toward its destination. This force is computed based on Newton's second law (see Equation 10), where m is the robot's mass, v is the desired velocity vector, v_0 is the current velocity vector, and t is the time interval over which the acceleration is applied:

$$\vec{F}_g = m \cdot \vec{a} = m \cdot \frac{\vec{v} - \vec{v}_0}{t} \tag{10}$$

The repulsive force from static obstacles, denoted as $\vec{F}_s$, is the combination of social repulsion force ($\vec{f}^{\,s}_{soc}$) and physical repulsion force ($\vec{f}^{\,s}_{phy}$), as described in Equation (11):

$$\vec{F}_s = \vec{f}^{\,s}_{soc} + \vec{f}^{\,s}_{phy} \tag{11}$$

The social repulsion force $\vec{f}^{\,s}_{soc}$ is modeled as an exponential function that decreases with distance, indicating a gradual increase in social discomfort as the robot approaches a static obstacle. This force is given in Equation (12):

$$\vec{f}^{\,s}_{soc} = k^s \cdot exp\left(\frac{r_R - d^s_R}{\psi^s}\right) . \overrightarrow{e_s} \tag{12}$$

The physical repulsion force $\vec{f}^{s}_{phy}$ becomes significant when the obstacle is within the robot's proxemic boundary. It is calculated based on the linear overlap between the robot and the obstacle, as shown in Equation (13):

$$\vec{f}^{s}_{phy} = k^{s} \cdot (r_R - d^{s}_{R}).\overrightarrow{e_s} \tag{13}$$

In these equations:

- r_R is the radius of the robot's proxemic (personal) area.
- d^{s}_{R} is the distance from the robot to the nearest static obstacle.
- k^{s} is a gain parameter that adjusts the strength of the repulsive forces.
- ψ^{s} is the effective range that modulates the range of social interaction.
- $\overrightarrow{e_s}$ is a unit vector pointing from the static obstacle to the robot.

The robot's motion planner uses these forces all the time to make sure it avoids obstacles in a way that is safe and considerate of other people, all while still moving toward its research goals.

By combining this model with the SLAM–based mapping system and frontier–based exploration, the robot can change its path on the fly, making sure it stays safe and finds its way quickly while exploring on its own. Figure 4 show the SFM forces.

In this study, the Gain (k^{s}), Radius (r_R), and Effective range (ψ^{s}) were chosen because they are the core constants in the SFM equations, directly influencing repulsive force magnitude, obstacle influence range, and force decay rate. Variations in these parameters are therefore expected to significantly impact navigation behavior and exploration performance.

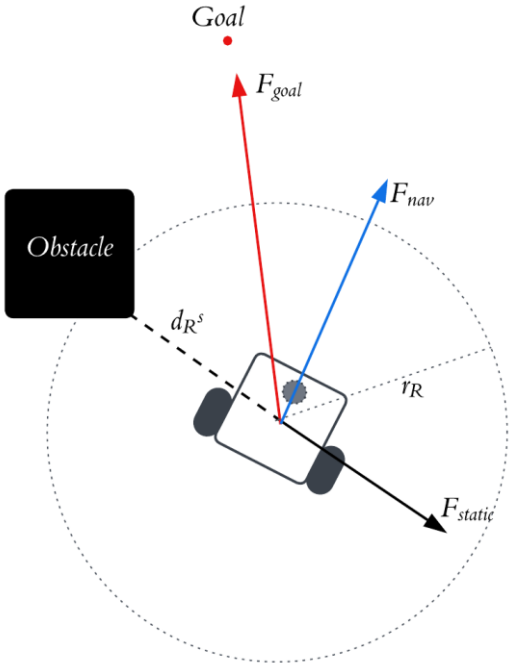

Figure 4. Forces acting within the SFM framework.

4.5 Simulation Setup & Scenario

An experimental situation was designed so that the system's performance could be tested in a simulated setting. The scenario is mostly about trying how well a robot can map a room on its own while navigating it with obstacles using different Social Force Model (SFM) parameters. The goal of the experiment was to evaluate how effectively the robot can make a full occupancy grid plan of its surroundings while staying away from obstacles. The simulation takes place in a room that was specially built and covers about 155 m^2, as shown in Figure 5.

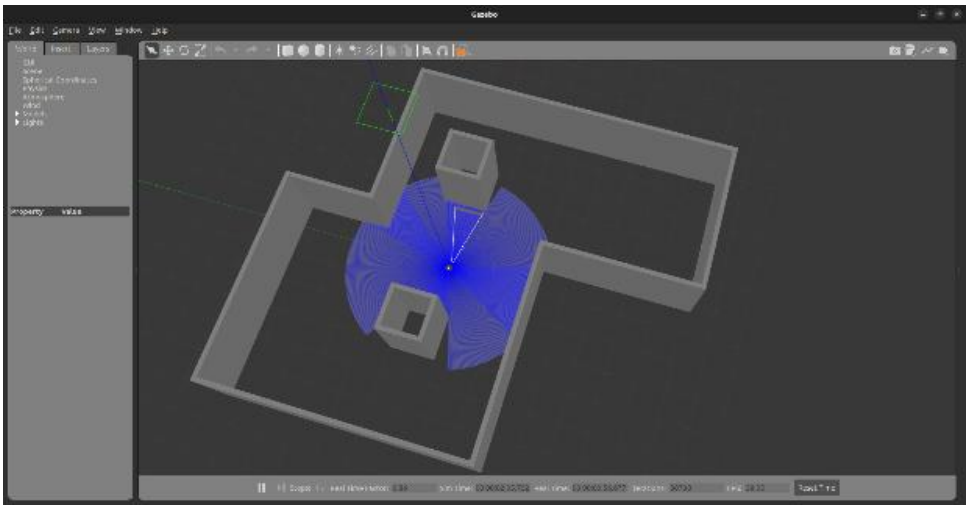

Figure 5. Experimental layout room scenario.

In the experiment, the TurtleBot3 [28] (see Figure 6) was used as a robot platform. The ROS group made this widely used mobile robot. The fact that it has a differential drive system and sensors that can be used for SLAM-based navigation, including a 2D LiDAR with a maximum range of 3.5 m, makes it a great choice for this simulation-based study.

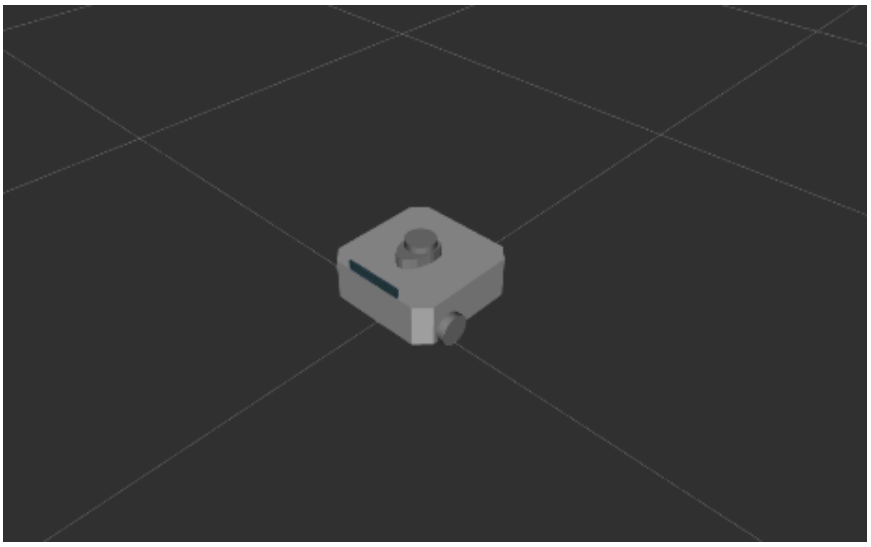

Figure 6. Turtlebot3 model in simulation.

The Gazebo model was used to simulate how the robot interacted with its surroundings, and RViz was used to see the robot's state, SLAM mapping progress, and path planning in real time. Gazebo and RViz are both part of the ROS2 environment, which means they can talk to the SLAM Toolbox and the navigation stack without any problems. ROS2 nodes running in parallel regulated and watched how the robot behaved.

The simulation environment and associated control nodes were developed using Visual Studio Code (VSCode) as the primary IDE. Additional support libraries such as OpenCV 4.9.0 were utilized to assist with perception and potential image processing tasks. Here are the basic requirements for running the simulation:

- Computer: Intel NUC
- Processor: Intel Core i5-1135G7 (up to 4.2 GHz)
- Memory: 8 GB DDR4 RAM
- Graphics: Intel Iris Xe Integrated Graphics
- Operating System: Ubuntu 22.04 LTS
- Storage: 128 GB NVMe M.2 SSD

This setup ensured sufficient computational resources to run real-time simulations and data processing jobs, like SLAM, avoiding obstacles, and exploring. The full integration of Gazebo, RViz, ROS2, and supporting libraries made it possible to try experiments in a simulation environment that could be relied on and repeated.

5. EXPERIMENT AND ANALYSIS

This chapter presents the outcomes of the simulation experiments conducted using the TurtleBot3 robot in a virtual environment of approximately 155 m^2. The primary objective of these experiments was to evaluate the robot's ability to autonomously map an environment while avoiding static obstacles, by testing various parameter combinations of the Social Force Model (SFM). The tested parameters included Gain (k^s), Radius (r_R), and Effective range (ψ^s). The performance metrics observed were Mapping Status, Failure Type (if any), Total Travel Distance, and Total Duration.

5.1 Simulation Result

This section analyzes the results of 12 simulation scenarios conducted using various SFM parameter configurations. The goal was to evaluate the robot's ability to complete the mapping task successfully while navigating autonomously. These scenarios tested how different values of Gain (k^s), Radius (r_R), and Effective Range (ψ^s) influenced robot performance in terms of collision avoidance and path clarity.

Table 1 provides a summary of the outcomes. Out of the 12 scenarios, only four were successful in completing the map, while the remaining eight failed due to issues such as collisions or confusion during navigation.

Table 1. Summarizes the results of 12 simulation scenarios using different combinations of sfm parameters.

No	Gain (k^s)	Radius (r_R)	Effective range (ψ^s)	Mapping Status	Failure Status	Total Distance (m)	Total Duration (s)
1	3	1.3	1	success	-	75.01	**662.85**
2	1.5	1	0.6	success	-	**62.75**	786.63
3	3	0.5	0.3	success	-	66.07	723.14
4	1.5	1.5	0.6	success	-	65.78	795.63
5	0.5	2	1	failed	collision	-	-
6	4.5	1.5	0.6	failed	confusion	-	-
7	0.2	1.5	0.6	failed	collision	-	-
8	0.5	1.5	0.6	failed	collision	-	-
9	1.5	3	0.6	failed	confusion	-	-
10	1.5	0.5	0.6	failed	collision	-	-
11	1.5	1.5	3	failed	collision	-	-
12	1.5	1.5	0.1	failed	confusion	-	-

Only four of the 12 simulation scenarios (see Table 1) were able to complete the mapping job successfully. These were Scenarios 1, 2, 3, and 4. The other eight scenarios failed because of collisions or confusion, which made it impossible to make good navigation choices. A summary of what happened:

1. Successful mapping: 4 scenarios (33.3%)
2. Failed mapping: 8 scenarios (66.7%)
    a. 5 due to collisions (41.7%)
    b. 3 due to confusion (25%)

We found three main factors that affect how well the robot navigates using the Social Force Model (SFM). These are based on the results shown in the table above. These features were studied by looking at different parameter settings and the results they produced in a number of different situations. Below, we talk about how each parameter—Gain (k^s), Radius (r_R), and Effective range (ψ^s)—affects the mapping process and look at how far and how long things take to move:

1) Effect of Gain (k^s)

In the Social Force Model (SFM), the Gain parameter (k^s) modulates the magnitude of repulsive forces generated in response to nearby obstacles. In our experiments, Scenarios 7 and 8—configured with low gain values (0.2 to 0.5)—resulted in insufficient repelling force, causing the robot to collide with obstacles. Conversely, Scenario 6, which used a very high gain value (4.5), exhibited erratic and unstable robot behavior due to overreaction to minor obstacles.

These findings align with the study by [14], who introduced an adaptive gain control using a Fuzzy Inference System (FIS). Their work demonstrated

that dynamic adjustment of the gain parameter significantly reduced unstable responses and allowed robots to respond more naturally to environmental stimuli. This supports our observation that improper gain settings—whether too low or too high—can severely hinder the robot's navigation performance.

2) Effect of Radius (r_R)

The Radius parameter (r_R) defines the effective spatial range within which social forces influence the robot's motion. Our results show that extremely small (<1.0) or large (>2.0) radius values tend to degrade performance. For example, although Scenario 3 succeeded with a small radius (0.5), this was only effective due to the specific interplay with other parameters. Meanwhile, Scenario 10, which also used a small radius, failed due to poor parameter interaction.

This is consistent with the findings by [29], who explored adaptive tuning of both gain and radius parameters using FIS in mobile robot navigation. Their study emphasized that optimal performance requires careful balancing of these parameters, as mismatches often lead to inefficient or unsafe navigation. Our results support this view, highlighting that success is not only tied to a specific value, but rather how the radius parameter interacts with gain and other motion constraints.

3) Effect of Effective range (ψ^s)

The Effective range (ψ^s) parameter governs the angular preference or directionality of the social force. Our experiments revealed that values at the extremes, such as 0.1 and 3.0 (Scenarios 12 and 11), led to navigation failures due to directional confusion and poor path planning. Scenarios with moderate ψ^s values between 0.3 and 1.0 consistently produced better navigation outcomes.

This observation parallels the work of [30], who utilized adaptive SFM for human-robot interaction and path stabilization. Their results showed that tuning directional parameters like ψ^s was crucial for avoiding oscillations and maintaining smooth trajectories. Our findings reinforce this, suggesting that extreme directional biases disrupt effective frontier-based exploration, while moderate tuning provides better path coherence and stability.

4) Statistical Validation of Parameter Influence

Due to the small sample size (n = 12 runs) and binary outcome (success vs failure), a Fisher's exact test was employed to evaluate the association between parameter category (optimal vs. non-optimal) and exploration success for each SFM parameter: Gain (k^s), Radius (r_R), and Effective range (ψ^s). Optimal ranges were determined from preliminary observations of parameter combinations that yielded successful mapping (k^s=1.5–3.0, r_R=1.0–1.5, and ψ^s=0.3–1.0).

Table 2. Statistical analysis of SFM parameters and exploration success.

Parameter	Table 2×2 (Non-optimal / Optimal × Fail / Success)	Odds Ratio	p-value
Gain (k^s)	Non-optimal: 4F / 0S Optimal: 4F / 4S	∞	0.208
Radius (r_R)	Non-optimal: 3F / 1S Optimal: 5F / 3S	1.80	1.000
Effective range (ψ^s)	Non-optimal: 2F / 0S Optimal: 6F / 4S	∞	0.515

To assess the statistical relationship between SFM parameters and exploration success, Fisher's exact tests were performed by classifying each parameter into optimal and non-optimal ranges. The statistical results are summarized in Table 2. The results showed that all successful trials occurred within the optimal ranges of Gain and Effective range, yielding odds ratios approaching infinity, which indicates a very strong practical association. However, because of the limited sample size (12 trials, with only 4 successful cases), the resulting p-values (all $p > 0.05$) did not reach conventional levels of statistical significance. This suggests that while the statistical tests were underpowered, the observed odds ratios provide strong evidence that parameter settings outside the optimal ranges substantially reduce the likelihood of successful exploration. Therefore, although formal significance could not be established, the experimental trends strongly support the hypothesis that inappropriate parameter values, whether too small, too large, or unbalanced, lead to mapping failures.

5.2 Exploration Performance Analysis

Frontier-based navigation and the Social Force Model (SFM) were used to finish four scenarios of autonomous exploration successfully in this work. Different combinations of the SFM factors gain k^s, repulsion radius r_R, and angular preference ψ^s were used in each case. Table 3 shows a summary of the data collected for each scenario, which included the total journey distance, the time spent exploring, and the average time it took to find frontier points.

Table 3. Summarizes the results of 12 simulation scenarios using different combinations of sfm parameters.

No	Gain (k^s)	Radius (r_R)	Effective range (ψ^s)	Total Distance (m)	Total Duration (s)	Frontier Points Avg. Detection Time (s)
1	3	1.3	1	75.01	**662.85**	**2.04**
2	1.5	1	0.6	**62.75**	786.63	2.86
3	3	0.5	0.3	66.07	723.14	2.29
4	1.5	1.5	0.6	65.78	795.63	3.45

Figure 7 shows that when we look more closely, the SFM parameters don't seem to have a strong or direct link with the total exploration time. For instance, both Scenario 2 and Scenario 4 used a gain value of 1.5 and a similar ψs. However, Scenario 4 took 795.63 seconds longer to finish the exploration than Scenario 2, which took 786.63 seconds longer. Scenario 1 had the shortest length (662.85 seconds), even though it had the highest gain. This suggests that frontier processing latency may be a more important factor in choosing exploration time.

The SFM settings don't seem to have a direct effect on the time, but they might have an effect on the total distance travelled. In this case, Scenario 1 has the longest path length (75.01 m), thanks to its high gain (3.0) and fairly large radius (1.3). The quickest path (62.75 m) is found in Scenario 2, where both gain and radius are lowered. This trend could mean that when the robot's repulsion strength and range are higher, it takes longer, less direct routes, which means it travels further.

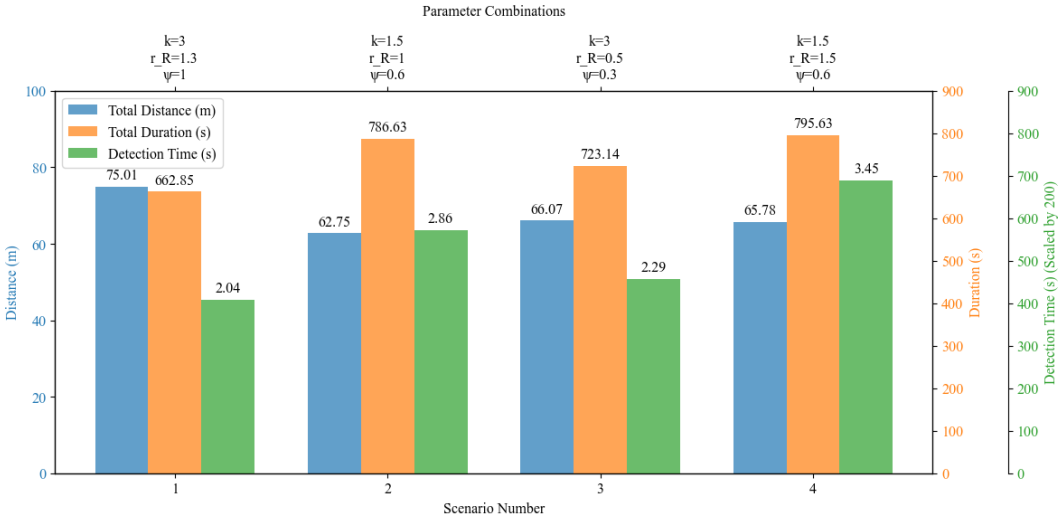

Figure 7. Exploration efficiency by parameter combination.

Figure 8 shows a clear link between the average time it takes to find the border and the total time it takes to explore. Longer exploration times are linked to scenarios with higher detection latency (e.g., 3.45 s in Scenario 4), while shorter exploration times are linked to scenarios with lower latency (e.g., 2.04 s in Scenario 1). This is because as the average frontier detection laten-cy rises, so does the total duration. This suggests that the robot's ability to quickly and efficiently continue its path planning and exploration cycles is greatly affected by how often and how quickly the border is updated.

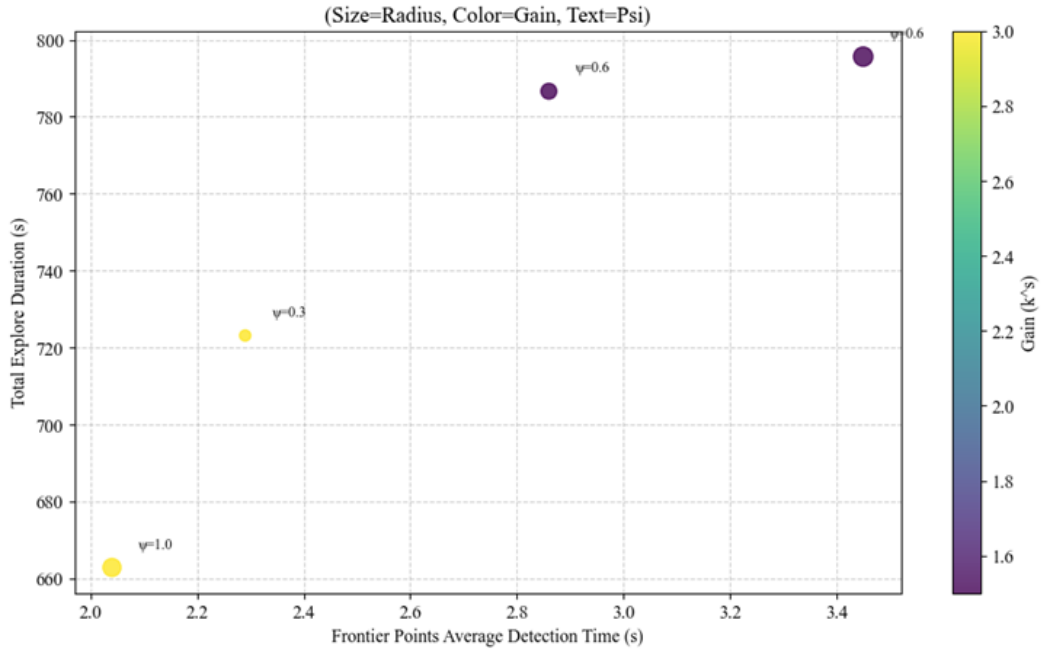

Figure 8. Exploration efficiency: frontier detection time vs duration.

It is important to keep in mind that the SFM settings do not change the average detection time of the frontier points. Instead, it is a measure of how well a computer program works, and it is based on things like processing power, map complexity, and the frontier recognition algorithm's internal logic. This means that different frontier detection delays can happen even when the SFM settings are the same. This is because of factors in the backend computing.

5.3 Visualization of Exploration Results

Figure 9 illustrates Scenario 1, which had the longest travel distance but had the shortest exploration time. The trajectory shows a winding and complex path, with the robot exploring the full area. The robot takes less direct routes to the frontiers, which may be attributed to the high Gain and Radius parameters in the SFM, leading to stronger social forces and wider avoidance behavior. Additionally, the blue dots on the map represent goal points, obtained from frontier detection, that have already been traversed along the robot's trajectory, and the red dots on the map represent the frontier points that detected by the robot.

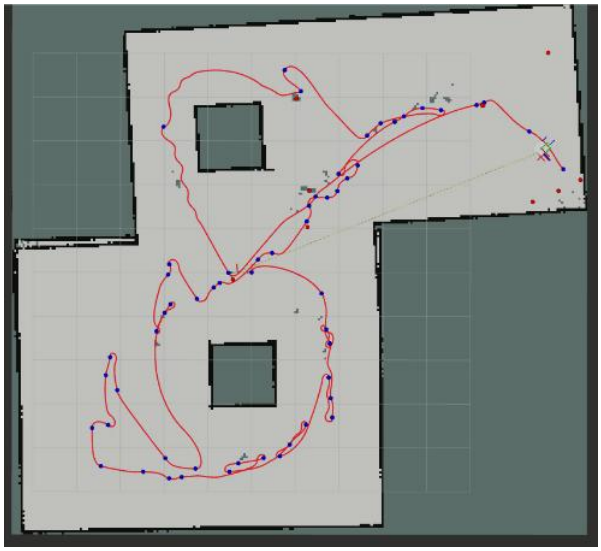

Figure 9. Exploration trajectory in Scenario 1 with the longest travel distance.

In contrast, Figure 10 presents Scenario 2, which demonstrates a more efficient exploration pattern. The robot explores broader areas with fewer redundant motions, resulting in the shortest total travel distance. The lower Gain and moderate Radius parameters likely contributed to smoother and more balanced movement toward the frontiers.

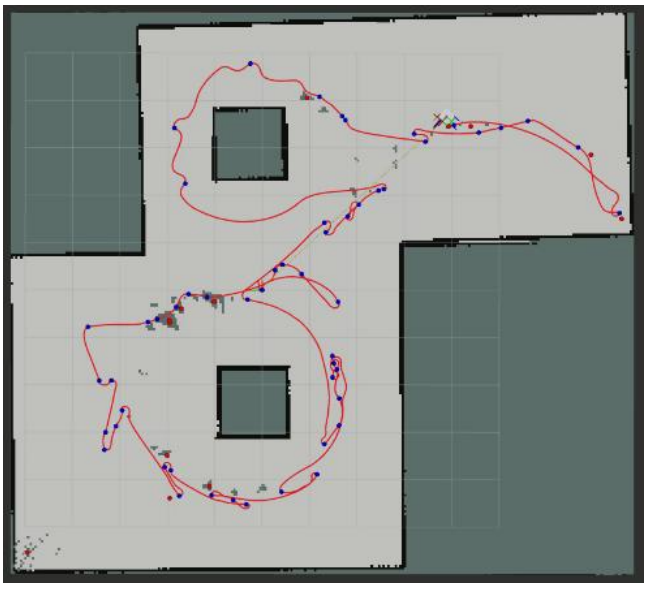

Figure 10. Exploration trajectory in Scenario 2 with the shortest travel distance.

Figure 11 shows a failed scenario 4 due to a collision. The impact caused a shift in the robot's orientation (roll and pitch), leading to mapping distortions where obstacles were falsely detected. The trajectory became distorted, and the robot prematurely terminated the exploration. This failure

is attributed to weak static repulsive forces, which prevented the robot from reacting appropriately to nearby obstacles.

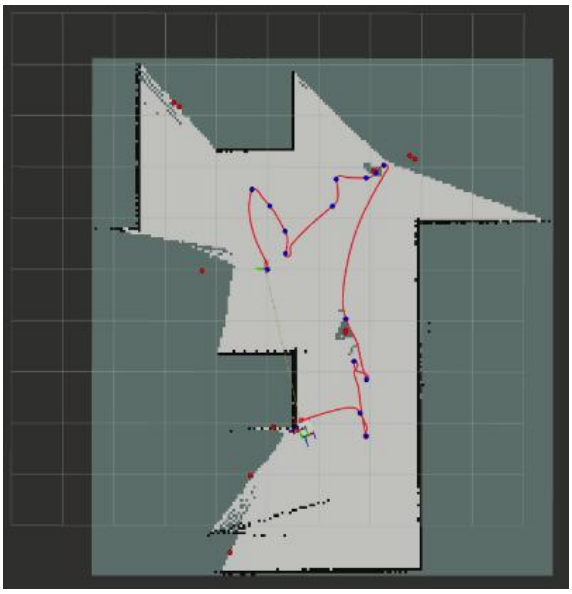

Figure 11. Exploration trajectory in failed scenario due to collision.

Finally, Figure 12 depicts a scenario 6 in which the robot exhibited circular and repetitive movements in a confined space. This behavior is a result of excessive sensitivity to nearby obstacles, likely due to overly strong static repulsive forces in the SFM. The robot was unable to escape the area and ended up oscillating between obstacles, causing exploration failure.

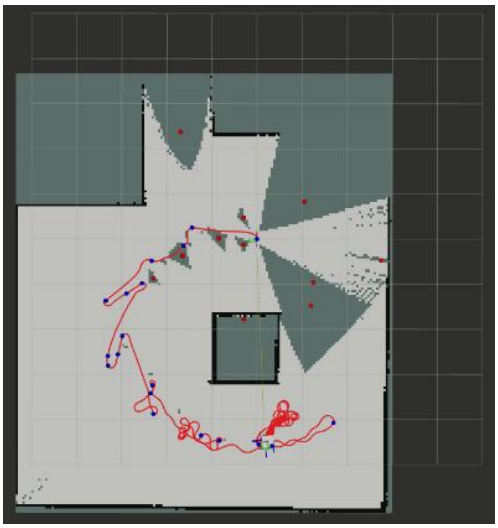

Figure 12. Exploration trajectory in failed scenario due to local oscillation.

6. CONCLUSION

The point of this study was to find out how well the Social Force Model (SFM) works for autonomous exploration, especially for frontier-based

navigation for mobile robots working in changing, unknown settings. The study's main goal was to find out how the robot's ability to accurately map its surroundings and avoid objects changed when certain SFM parameters were changed. These parameters were Gain (k^s), Radius (r_R), and Effective range (ψ^s). We did a lot of tests with the TurtleBot3 robot in simulated environments and discovered that the SFM parameters had a big impact on how well it explored. In particular, low Gain values caused crashes because the robots couldn't avoid obstacles well enough, and high Gain values made the robots move in strange ways. The Radius parameter was also very important. Extremely small or large values degraded performance, while values in the middle helped with navigation. In the same way, behaviour that was disoriented was caused by very high Effective range values, while stable navigation was caused by intermediate Effective range values. The study also found that the SFM parameters didn't have a big effect on the exploration time. However, the total travel distance got longer as the Gain and Radius settings got higher. Frontier detection delay was also found to be a major factor affecting the exploration time. Overall, this study gives real-world examples of how sensitive SFM factors are and useful tips for making autonomous navigation systems work better. The results help make exploration algorithms that work better and more reliably by showing how important it is to find a balance between exploration aggression and safety. More work could be done in the future to improve strategies for tuning parameters and try the system in more complex real-world settings to confirm its performance even more. Another goal could be to shorten the time it takes to find the frontier, which would lower the latency.

REFERENCES

[1] A. Gautam and others, **FAST: Synchronous Frontier Allocation for Scalable Online Multi-Robot Terrain Coverage**, *Journal of Intelligent & Robotic Systems*, vol. 87, pp. 545–564, 2017.

[2] B. Yamauchi, **A frontier-based approach for autonomous exploration**, in *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 1997, pp. 146–151.

[3] J. Wu, S. Jiang, and C. Wei, **A Heuristic Planning Framework Based on Fast Frontier Extraction**, in *2023 5th International Symposium on Robotics & Intelligent Manufacturing Technology (ISRIMT)*, Changzhou, China: IEEE, Sep. 2023, pp. 147–152. doi: 10.1109/ISRIMT59937.2023.10428304.

[4] F. Liu *et al.*, **Anticipatory Robot Navigation: Incorporating Estimated Obstacle Behaviors with the Social Force Model**, in *Proceedings of the IEEE International Conference on Consumer Electronics (ICCE)*, 2024, pp. 1–6.

[5] B. Fang, J. Ding, and Z. Wang, **Autonomous robotic exploration based on frontier point optimization and multistep path planning**, *IEEE*

Access, vol. 7, pp. 46104–46113, 2019, doi: 10.1109/ACCESS.2019.2909307.

[6] A. Topiwala, P. Inani, and A. Kathpal, **Frontier Based Exploration for Autonomous Robot**, Jun. 10, 2018, *arXiv*: arXiv:1806.03581. doi: 10.48550/arXiv.1806.03581.

[7] M. F. Ibrahim, A. B. Huddin, M. H. M. Zaman, A. Hussain, and S. N. Anual, **An enhanced frontier strategy with global search target-assignment approach for autonomous robotic area exploration**, *International Journal of Advanced Technology and Engineering Exploration*, 2021, [Online]. Available: https://api.semanticscholar.org/CorpusID:233834148

[8] E. A. N. Sumer and H. Temeltas, **RRT Based Frontier Point Detection for 2D Autonomous Exploration**, *2022 7th International Conference on Robotics and Automation Engineering (ICRAE)*, pp. 305–311, 2022.

[9] E. Repiso, G. Ferrer, and A. Sanfeliu, **On-line adaptive side-by-side human robot companion in dynamic urban environments**, in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, BC: IEEE, Sep. 2017, pp. 872–877. doi: 10.1109/IROS.2017.8202248.

[10] Y. Sun and C. Zhang, **Efficient and Safe Robotic Autonomous Environment Exploration Using Integrated Frontier Detection and Multiple Path Evaluation**, *Remote Sensing*, vol. 13, no. 23, p. 4881, Dec. 2021, doi: 10.3390/rs13234881.

[11] C. Liu *et al.*, **Enhancing autonomous exploration for robotics via real time map optimization and improved frontier costs**, *Sci Rep*, vol. 15, no. 1, p. 12261, Apr. 2025, doi: 10.1038/s41598-025-97231-9.

[12] D. Djabir, B. Khadir, N. Abdelkrim, B. Zineddine, and D. Rihem, **Study of Unknown Environments Exploration Techniques Based on ROS-Enabled Omnidirectional Mobile Robot**, presented at the The 12th International Conference of Control, Dynamic Systems, and Robotics, Jul. 2025. doi: 10.11159/cdsr25.145.

[13] H. Kivrak, F. Çakmak, H. Kose, and S. Yavuz, **Social navigation framework for assistive robots in human inhabited unknown environments**, *Engineering Science and Technology, an International Journal*, 2021, [Online]. Available: https://api.semanticscholar.org/CorpusID:225278356

[14] A. T. Rifqi, B. S. B. Dewantara, D. Pramadihanto, and B. S. Marta, **Fuzzy Social Force Model for Healthcare Robot Navigation and Obstacle Avoidance**, in *2021 International Electronics Symposium (IES)*, 2021, pp. 445–450. doi: 10.1109/IES53407.2021.9594052.

[15] H. M. Mu'allimi, B. Sena Bayu Dewantara, D. Pramadihanto, and B. S. Marta, **Human Partner and Robot Guide Coordination System Under Social Force Model Framework Using Kinect Sensor**, in *2020 International Electronics Symposium (IES)*, Surabaya, Indonesia: IEEE, Sep. 2020, pp. 260–264. doi: 10.1109/IES50839.2020.9231872.

[16] M. F. Daffa, B. S. B. Dewantara, and S. Setiawardhana, **Robot navigation on inclined terrain using social force model**, *IJRA*, vol. 13, no. 2, p. 131, Jun. 2024, doi: 10.11591/ijra.v13i2.pp131-139.
[17] S. Macenski and I. Jambrecic, **SLAM Toolbox: SLAM for the dynamic world**, *JOSS*, vol. 6, no. 61, p. 2783, May 2021, doi: 10.21105/joss.02783.
[18] S. Macenski, **On Use of SLAM Toolbox, A fresh(er) look at mapping and localization for the dynamic world**, in *ROSCon 2019*, Macau, SAR China, 2019.
[19] K. J. Singh *et al.*, **Map Making in Social Indoor Environment Through Robot Navigation Using Active SLAM**, *IEEE Access*, vol. 10, pp. 134455–134465, 2022, doi: 10.1109/ACCESS.2022.3230989.
[20] S. Gowtham, R. Praveen, M. Parthiban, P. Sai Charan, N. Seenu, and R. Kuppan Chetty, **Autonomous Robotic Exploration and Navigation System using Tri-layered Mapping and Geometrical Path Planning Techniques**, *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 1055, no. 1, p. 012003, Feb. 2021, doi: 10.1088/1757-899X/1055/1/012003.
[21] E. Kaufman, K. Takami, Z. Ai, and T. Lee, **Bayesian Mapping-Based Autonomous Exploration and Patrol of 3D Structured Indoor Environments with Multiple Flying Robots**, *J Intell Robot Syst*, vol. 98, no. 2, pp. 403–419, May 2020, doi: 10.1007/s10846-019-01066-2.
[22] A. Gullu and H. Kuşçu, **Optimized Graph Search Algorithms for Exploration with Mobile Robot**, *EMITTER Int'l J. of Engin. Technol.*, vol. 9, no. 2, pp. 222–238, Dec. 2021, doi: 10.24003/emitter.v9i2.614.
[23] H. Taheri and Z. C. Xia, **SLAM; definition and evolution**, *Engineering Applications of Artificial Intelligence*, vol. 97, p. 104032, 2021.
[24] N. Chindakham, Y.-Y. Kim, A. Pirayawaraporn, and M.-H. Jeong, **Simultaneous Calibration of Odometry and Head-Eye Parameters for Mobile Robots with a Pan-Tilt Camera**, *Sensors*, vol. 19, no. 16, p. 3623, Aug. 2019, doi: 10.3390/s19163623.
[25] J. Oršulić, D. Miklić, and Z. Kovačić, **Efficient Dense Frontier Detection for 2-D Graph SLAM Based on Occupancy Grid Submaps**, *IEEE Robotics and Automation Letters*, vol. 4, pp. 3569–3576, 2019.
[26] W. Qiao, Z. Fang, and B. Si, **Sample-Based Frontier Detection for Autonomous Robot Exploration**, *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1165–1170, 2018.
[27] P. Quin, D. D. K. Nguyen, T. L. Vu, A. Alempijevic, and G. Paul, **Approaches for Efficiently Detecting Frontier Cells in Robotics Exploration**, *Frontiers in Robotics and AI*, vol. 8, 2021, [Online]. Available: https://api.semanticscholar.org/CorpusID:232041530
[28] ROBOTIS-GIT, **turtlebot3_simulations**. 2025. [Online]. Available: https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
[29] Alif Wicaksana Ramadhan, Bima Sena Bayu Dewantara, and Setiawardhana, **Optimization of Fuzzy Social Force Model Adaptive Parameter using Genetic Algorithm for Mobile Robot Navigation**

Control, Jurnal Rekayasa Elektrika, vol. 19, no. 1, 2023, doi: 10.17529/jre.v19i1.28330.

[30] P. Patompak, S. Jeong, I. Nilkhamhang, and N. Y. Chong, **Learning Proxemics for Personalized Human–Robot Social Interaction**, *Int J of Soc Robotics*, vol. 12, no. 1, pp. 267–280, Jan. 2020, doi: 10.1007/s12369-019-00560-9.