

FMEA-Based Coverage-Path-Planning Strategy for Floor-Cleaning Robots

Isira D. Wijegunawardana, S. M. Bhagya P. Samarakoon, M. A. Viraj J. Muthugala,* and Mohan Rajesh Elara

Floor-cleaning robots play a crucial role in both industrial and domestic spaces. However, these robots often face challenges due to hazardous components in their environment, which can cause them to fail and prevent them from performing at their best. This situation necessitates continuous research in the field of floor-cleaning robots. However, most of these efforts focus on improving the robots' perception capabilities by incorporating additional sensors. Nevertheless, incorporating more sensors is an expensive solution for most cleaning robots. Alternatively, in this research, the feasibility of introducing a safe path on a predefined hazard map is explored. The proposed method aims to trade-off between area coverage and the safety of the cleaning robot. Herein, the failure mode and effect analysis (FMEA) method is introduced as a tool to classify the hazards and implement a safety-ensured coverage path-planning process. In this approach, the risk factor defined for a point in the environment serves as the key parameter to assess the safety of the algorithm's suggested path. To validate and evaluate the proposed method, this article utilizes the hTetro mobile robot. In the experimental results, it is demonstrated that the proposed method can reduce high-risk movements of the robot compared to existing methods.

improving their performance. Specifically, these research efforts have focused on areas such as cleaning systems,^[3] dirt detection,^[4] and area coverage.^[5]

In the past decade, there has been significant research focused on enhancing the cleaning performance of robots by increasing their coverage.^[6] Consequently, coverage path planning (CPP) has gained considerable attention in this field.^[7] CPP involves determining a path that covers all points of interest while avoiding obstacles. According to the literature, CPP can be performed through online and offline methods. Offline approaches assume a static and fully observable environment, allowing for the generation of optimized navigation paths in advance. In contrast, the online (or sensor-based) approach can adapt to dynamic changes in the environment and update the path plan in real time.^[7] However, when dealing with static cleaning environments, path-

planning approaches tend to favor offline methods due to their lower power consumption and reduced sensor requirements.


CPP algorithms can also be classified based on the strategies employed to decompose the environment.^[7] A decomposition strategy involves dividing the environment into smaller units, known as subregions or cells. One classic method of unit decomposition involves using simple shapes like trapezoids and triangles to separate the robot's space.^[7] In each unit, a straightforward pattern such as spiral motion or boustrophedon motion is then deployed to achieve coverage. Another frequently observed approach in CPP-related research is grid-based decomposition, initially proposed by Moravec and Elfes.^[8] Grid-based methods typically divide the environment into square-shaped cells, although occasionally other shapes, such as triangular cells, can be used depending on the robot's shape.^[9] The use of grid-based decomposition significantly reduces the computational complexity required to determine a coverage path. However, in larger environments, computational complexity can become an issue unless hierarchical grids are employed.

To cover the decomposed cells, various techniques can be employed. In most commercial cleaning robots, simple algorithms such as random coverage, spiral motion, and boustrophedon motion have been utilized for robot path planning.^[10] However, in grid-based CPP research, more emphasis has been placed on efficient navigation in terms of performance factors

1. Introduction

There is a growing demand for robot-assisted floor cleaning in both industrial and domestic settings. This demand stems from various factors, including socioeconomic backgrounds and the busy day-to-day lifestyles of individuals, which make manual cleaning difficult.^[1] Additionally, concerns regarding labor health have arisen due to the repetitive nature of cleaning tasks.^[2] The high demand for floor-cleaning robots, driven by the aforementioned reasons, has led to continuous research aimed at

I. D. Wijegunawardana, S. M. B. P. Samarakoon, M. A. V. J. Muthugala, M. R. Elara
Engineering Product Development Pillar
Singapore University of Technology and Design
Singapore 487372, Singapore
E-mail: viraj_jagathpriya@sutd.edu.sg

 The ORCID identification number(s) for the author(s) of this article can be found under <https://doi.org/10.1002/aisy.202300260>.

© 2023 The Authors. Advanced Intelligent Systems published by Wiley-VCH GmbH. This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

DOI: 10.1002/aisy.202300260

such as energy, time, and distance. This is achieved through the use of techniques such as greedy algorithms,^[10] spanning tree algorithms,^[11] neural networks,^[12] genetic algorithms,^[13] and graph theory.^[14] CPP algorithms can prioritize different optimization factors, including finding the shortest path, avoiding collisions, balancing maximal coverage with time constraints, or generating attractive and intuitive movement patterns.^[15] Some grid-based CPP methods employ greedy search algorithms for optimization. In this approach, each cell is assigned a cost value, and the path is optimized to minimize the overall cost. As a result, evaluations of CPP algorithms typically consider elapsed time for coverage, energy consumption, and area coverage.^[16–18]

Despite extensive research in robot perception, floor-cleaning robots still experience frequent failures due to their limited ability to detect and respond to hazards in their environment. These failures can result in various levels of effects, ranging from minor performance reductions to irreversible damages.^[19] In past years, there have been frequent involvements in analyzing the safety of humans in collaborative human–robot environments, where robots work alongside humans.^[20] These studies primarily address the reliability and safety of manipulative robots operating in controlled environments with fewer uncertainties. However, mobile service robots, such as cleaning robots, operating in unpredictable environments are more susceptible to damage from hazardous components.

In the existing literature, a limited number of studies have specifically examined the safety of service robots from the perspective of the robot itself.^[19,21–24] These studies provide valuable insights into establishing and maintaining the safety of robots throughout their operation. One notable approach is the development of a safety-driven control system by Woodman et al.^[21] This system evaluates sensory data and incorporates predefined safety policies to directly control the robot's locomotion. When encountering high-risk situations, the robot either halts or navigates away from the hazard, thus ensuring its safety. Dogramadzi et al.^[22] conducted a comprehensive hazard analysis focusing on identifying non-mission interactions in human–robot interaction using guide words. Guiochet et al.^[24] performed a safety analysis of an assistive robot designed for tasks such as standing up, sitting down, walking, and health-state monitoring for elderly individuals. They provided recommendations for safer handling of the robot based on identified hazards and their sources. Ng et al.^[19] conducted a safety analysis of a telepresence robot operating in academic institutions to identify potential failures and hazards associated with its use. These studies contribute to the understanding and management of safety risks in different applications of service robots.

Various methods can be employed to analyze the environmental hazards that contribute to failures in a robot's operation, including fault tree analysis, probability tree analysis, Markov analysis, and failure mode and effect analysis (FMEA).^[25] Among these methods, probability tree analysis provides a probabilistic representation of the cause-and-effect occurrence of failures. The fault tree analysis utilizes a logical flow diagram illustrating different events that ultimately lead to a failure. Markov analysis involves representing different states in the robot's operation and presenting the probabilities associated with transitioning between these states. Consequently, the total

probability of different failure states can be determined from any predefined state of the robot. However, the most promising method for these research genres is the FMEA method.^[19] The main reason is that FMEA provides the failure cause, failure mode, and failure effects, where the classification and interconnection of the failure events are more convenient and systematic. The FMEA method is also flexible and scalable for diverse application domains.^[26] Here, the experiments conducted based on the FMEA method have identified static hazards such as furniture, inconsistencies in building layout and decorations, and dynamic hazards such as humans and animals that compromise the performance of the robots.

Furthermore, there have been occasional references in the literature to the concept of robot inclusivity, also known as the friendliness of an environment toward robots.^[19,23,27,28] These research papers suggest that environments with lower robot inclusivity require a higher level of autonomy from the robot to minimize failure rates.^[23] Moreover, they provide guidelines for developing robot spaces in a manner that promotes inclusivity.^[23,27,28] These guidelines aim to enhance the robot inclusivity by altering the environment such that it improves a robot's accessibility, reducing hazards that compromise the robot's safety, and allowing uninterrupted performance of its designated tasks.

These guidelines encompass various considerations to enhance the robot's environment for improved inclusivity and safety. They involve strategic placements of doors to facilitate the robot's movement and enable it to manipulate doorknobs effectively. Designing recessed areas allows the robot to pause its work when necessary. Installing user-friendly mechanisms for opening and closing doors, such as push buttons, easy-to-grasp handles, or touchless interfaces, simplifies operation. Selecting nonslip, nonreflective, level, and even floor surfaces ensures stability. Providing adequate protection for edges along pathways helps prevent falls while using signs on stairs, steps, or areas with surface or level changes contributes to avoiding robot accidents. As a result, the increased robot friendliness of the environment allows for a relaxation of the robot's advanced autonomy requirements. However, no attention has yet been paid to developing CPP methods that minimize the possible failures of robots.

This article introduces a grid-based offline CPP strategy that prioritizes the safety of the robot as the key criterion for success. The proposed method utilizes the FMEA technique to analyze failure cause and effect data, thereby determining the risk priority number (RPN) for each point in the robot's environment. As a result, this path-planning algorithm is guided by the distribution of risk factors (RFs) throughout the robot's environment. In contrast to standard CPP methods that solely focus on avoiding individual obstacles, the method proposed in this article goes a step further by classifying hazards based on varying levels of risk. This categorization allows the algorithm to effectively manage the trade-off between coverage and the safety of the cleaning robot while accounting for the specific hazards present in the environment. The remainder of the article is organized as follows to present the aforementioned aspects. Section 2 provides a brief overview of the proposed path-planning method. In Section 3, the results obtained from simulations and real-world evaluations of the proposed path-planning algorithm are presented. Finally, Section 4 concludes the article by discussing the key findings

regarding the feasibility and effectiveness of the proposed system, along with suggesting potential future directions.

2. FMEA-Based Path-Planning Method

To the best of our knowledge, the path-planning strategy proposed in this article represents the first attempt to utilize the FMEA as a tool for CPP applications. In this approach, the FMEA is employed to identify potential failure causes in the robot's environment and understand their corresponding effects. Subsequently, a parameter called the RF is introduced, as per the mathematical relationships suggested in the following subsections. The RF parameter defined in this article is a discrete function within the robot's 2D grid space. This choice is based on the fact that the robot's interaction, as considered in this article, is limited to a 2D space. The RF function serves as the key parameter for establishing a safety-assured path for the robot, ensuring that it covers the maximum feasible area within the environment while avoiding any hazards that may potentially lead to operational failures.

In the proposed method, the first step is to construct a map that identifies the hazards localized within each respective cell. This map can be created manually or generated using a robot with advanced perception capabilities. In this study, the site inspection and hazard map creation were conducted based on the robot-inclusive FMEA method introduced by ref. [19]. Subsequently, the generated map, which indicates the types and locations of hazards, can be utilized to pre-generate a path plan for a cleaning robot that does not possess high-level perception capabilities.

2.1. FMEA

FMEA is a systematic method that examines product components or processes to identify potential failures, their causes, and the subsequent effects on the system. It can be employed as an inductive reasoning process to identify and understand potential failures in each component of a product, with the aim of reducing and minimizing safety incidents. This method was initially introduced in the early 1950s to evaluate flight control system designs^[29] and later gained popularity in identifying failures in various commercial production applications.^[30] More recently, there have been instances where FMEA techniques have been utilized to analyze robot failures.^[19]

The FMEA approach analyzes failure causes, failure modes, and their associated effects in a process, resulting in the determination of an RPN. The RPN represents the likelihood of a specific failure mode occurring due to a particular cause and resulting in a specific effect. In this computation, FMEA considers three parameters with ratings assigned to each failure mode: the severity of the effect (S), the likelihood of the failure mode occurring (O), and the probability of detecting the cause and stopping the failure (D). The RPN value is calculated as $RPN = S \times O \times D$. Subsequently, the RPN for each failure is compared to determine the possible failure mode and its cause and effects.

The framework proposed in this article is based on the standard FMEA approach. However, to align with the application at

hand, this research has modified the FMEA approach to obtain a discrete mathematical function that represents the failure possibility or risk level of a particular location in the robot's space. This function is based on the RPN calculation in the standard FMEA approach. Unlike the standard method, which compares and prioritizes risks for different types of failures, this method accumulates all potential failure risks at a specific point. The failure causes, failure modes, and effects associated with the floor-cleaning robot are predefined inputs to the model. By feeding the locations of the failure causes into the model, it can provide the risk levels of the environment as a heatmap.

Failures in cleaning robots can arise from various factors. **Figure 1** illustrates a few instances where the hTetro cleaning robot encountered interruptions during the experiments. Therefore, it is crucial to analyze and classify these failure causes to develop a systematic model. In the adapted FMEA approach, the failure causes have been categorized into four different classes (see **Table 1**). This categorization is based on the detectability level of the failure cause. Failure causes that can be detected by the light detection and ranging (LIDAR) sensor or any other optical proximity sensor operating in the robot's navigation plane are classified under class C_1 . This class includes blind obstacles such as walls and furniture. Obstacles that are not detected by these sensors due to their transparent nature are placed in class C_2 , which includes glass obstacles. Class C_3 encompasses floor irregularities that are also not detectable by the aforementioned sensors. Lastly, class C_4 encompasses sudden floor-level changes, such as staircases, edges, cliffs, and curbs. The defined classes and their detectability ratings are illustrated in **Table 1**. However, this analysis excludes dynamic obstacles like humans, animals, and other robots, as well as changes in conditions such as heat, humidity, fog, and light intensity, due to the high uncertainty surrounding their occurrence.

Failures in cleaning robots can lead to various types of effects, each carrying different severity levels. These severity levels are defined based on the damage incurred by the robot and the level of intervention required for its autonomy. For the purpose of this article, five levels of effects have been defined. These effects and their corresponding severity ratings are illustrated in **Table 2**.

Each potential hazard that may cause a failure is associated with one or more potential modes that can arise from such failures, as depicted in **Figure 2**. **Table 1** and **Table 2** summarize these failure modes. The parameter F_j ($j = 1, 2, 3, 4$) represents the identified failure modes for cleaning robots, which include veering off course due to minor collisions or wheel slipping, getting trapped in a particular location, tipping over due to instability, and bouncing back after colliding with an object due to inadequate detection capabilities. Each failure mode, F_j ($j = 1, 2, 3, 4$), is assigned a probability of occurrence, which can happen due to a particular hazard class, C_i ($i = 1, 2, 3, 4$), and is denoted by $P_{(C_i, F_j)}$ (refer to **Table 1**). This research assumes that all the failure modes can only happen due to the defined causes C_i for $i = 1, 2, 3, 4$. Hence, the sum of probabilities of occurring a particular failure mode F_j due to each all failure causes, $\sum_{j=1}^4 P_{(C_i, F_j)}$, should be equal to 1.

Furthermore, each failure mode may have one or more impacts on the performance and safety of the robot, with varying

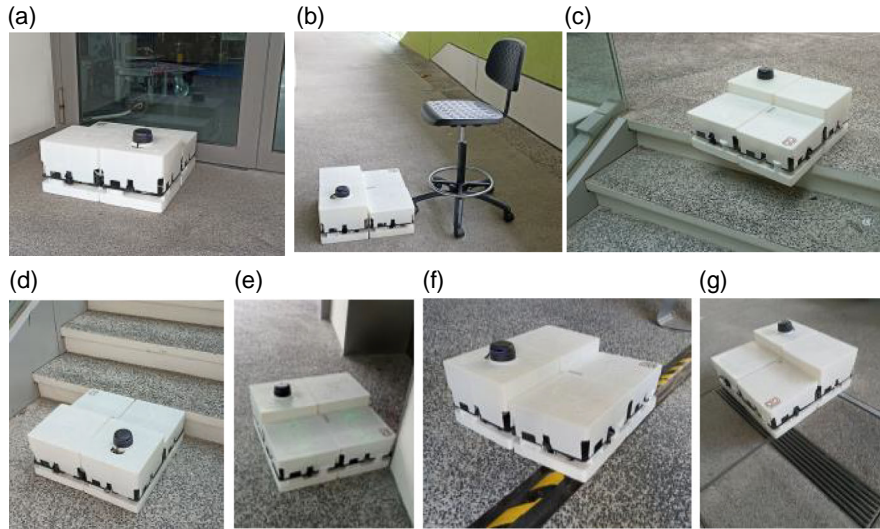


Figure 1. Some instances where cleaning robots failed to perform their duties due to environmental hazards: a) robot collides with a glass door, b) robot gets stuck in a chair, c) robot failing to identify the staircase, d) robot colliding with steps in staircase entrance, e) robot veer-off by colliding with a wall, f) robot get stuck in wire protection, and g) robot get stuck in rough terrain.

Table 1. The probability of each failure mode resulted due to each failure cause.

| Failure causes | | | Failure modes | | | | |
|----------------|--------------------|---|---------------|-----------------------|-------------------|-------------------------|--------------------|
| | | | Categories | Detectability ranking | Veer off F_1 | Stuck in place F_2 | Roll over F_3 |
| C_1 | Blind objects | (Ex: Blind walls, furniture, warning signs) | 1 | $P_{(C_1, F_1)}$ | $P_{(C_1, F_2)}$ | $P_{(C_1, F_3)}$ | $P_{(C_1, F_4)}$ |
| C_2 | Transparent object | (Ex: Glass walls, glass doors) | 2 | $P_{(C_2, F_1)}$ | $P_{(C_2, F_2)}$ | $P_{(C_2, F_3)}$ | $P_{(C_2, F_4)}$ |
| C_3 | Level change | (Ex: Staircase, cliffs, curbs, thresholds) | 3 | $P_{(C_3, F_1)}$ | $P_{(C_3, F_2)}$ | $P_{(C_3, F_3)}$ | $P_{(C_3, F_4)}$ |
| C_4 | Rough terrain | (Ex: Grooves, carpet, humps, cable covers) | 4 | $P_{(C_4, F_1)}$ | $P_{(C_4, F_2)}$ | $P_{(C_4, F_3)}$ | $P_{(C_4, F_4)}$ |

Table 2. The probability of each failure effect resulted due to each failure mode.

| Failure modes | | | | Failure effects | | |
|-------------------|-------------------------|--------------------|----------------------|-----------------|---|--|
| Veer off F_1 | Stuck in place F_2 | Roll over F_3 | Bounce back F_4 | Categories | Severity ranking | |
| $P_{(F_1, E_1)}$ | $P_{(F_2, E_1)}$ | $P_{(F_3, E_1)}$ | $P_{(F_4, E_1)}$ | E_1 | Robot does not get any damage but takes more time to complete the task alone. 1 | |
| $P_{(F_1, E_2)}$ | $P_{(F_2, E_2)}$ | $P_{(F_3, E_2)}$ | $P_{(F_4, E_2)}$ | E_2 | Robot gets recoverable damage but completes the task alone. 2 | |
| $P_{(F_1, E_3)}$ | $P_{(F_2, E_3)}$ | $P_{(F_3, E_3)}$ | $P_{(F_4, E_3)}$ | E_3 | Robot gets recoverable damage and needs human support to complete the task. 3 | |
| $P_{(F_1, E_4)}$ | $P_{(F_2, E_4)}$ | $P_{(F_3, E_4)}$ | $P_{(F_4, E_4)}$ | E_4 | Robot gets irrevocable damage but completes the task. 4 | |
| $P_{(F_1, E_5)}$ | $P_{(F_2, E_5)}$ | $P_{(F_3, E_5)}$ | $P_{(F_4, E_5)}$ | E_5 | Robot gets irrevocable damage and cannot complete the task. 5 | |

levels of severity. These effects are defined based on the potential damage and autonomy loss of the robot. Therefore, this study considers five distinct levels of effects, which are described and rated according to their severity in Table 2. The likelihood of each effect (E_k) occurring due to each failure mode (F_j) is expressed as $P_{(F_j, E_k)}$, where $j = 1, 2, 3, 4$ and $k = 1, 2, 3, 4, 5$ (refer to Table 1). This work assumes that a particular failure mode, F_j ($j = 1, 2, 3, 4$), will only result in the defined effects E_k where for $k = 1, 2, 3, 4, 5$. Hence, the sum of probabilities of occurring all

the effects due to a particular failure mode F_j , $\sum_{k=1}^5 P_{(F_j, E_k)}$, should be equal to 1.

2.2. FMEA-Based Mathematical Model for RF

The FMEA method serves as a tool for assessing the risk level of the robot's environment. In this section, the mathematical modeling modifies this method to derive an analytical function that represents the level of risk at specific locations within the

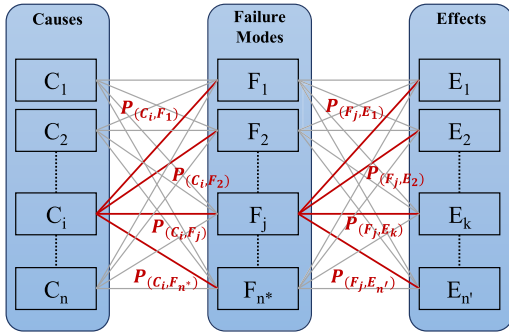


Figure 2. Model of dependencies between failure causes, failure modes, and failure effects.

environment. The robot's environment is divided into grid cells, with each cell assigned a risk value based on its respective location. Figure 2 illustrates the dependencies among the failure causes, failure modes, and effects. The probability of the F_j failure mode occurring due to the C_i cause is denoted as P_{C_i, F_j} , while the probability of the E_k effect occurring due to the F_j failure mode is denoted as P_{F_j, E_k} . Moreover, this model considers a total of n causes, n^* failure modes, and n' failure effects.

Based on the aforementioned model, a set of mathematical relationships has been developed to quantify the level of risk at specific locations on the grid map. The parameter denoted by $I_{H_i}(x_r, y_r)$, where $(i = 1, 2, \dots, N)$, represents the impact of the hazard cell H_i located at coordinates (x_i, y_i) on a particular location in the environment denoted by (x_r, y_r) . Let's assume that this hazard cell H_i belongs to the failure cause class C_i . Consequently, it inherits the detectability characteristic D_i associated with the C_i failure cause. Based on the detectability of H_i and the distance between the considered location and the hazard cell, its impact $I_{H_i}(x_r, y_r)$ can be defined as shown in Equation (1), which is derived from the relationship between these parameters.

$$I_{H_i}(x_r, y_r) = \frac{2 \times D_i}{20 + \sqrt{((x_i - x_r)^2 + (y_i - y_r)^2)}} \quad (1)$$

Equation (2) establishes the relationship between failure modes and hazards (failure causes) by incorporating the occurrence possibility (P_{H_i, F_j}). In other words, $I_{F_j}(x_r, y_r)$ represents the impact of the F_j failure mode ($j = 1, 2, \dots, n^*$) at a specific location (x_r, y_r) in the environment, considering all available failure causes present in the environment. These failure causes or hazards, denoted as H_i , can belong to any class of failure causes C_i ($i = 1, 2, 3, 4$).

$$I_{F_j}(x_r, y_r) = \sum_{i=1}^N P_{(H_i, F_j)} \times I_{H_i}(x_r, y_r) \quad (2)$$

Equation (3) establishes the relationship between failure modes and failure effects by incorporating the occurrence possibility (P_{F_j, E_k}). In other words, $I_{E_k}(x_r, y_r)$ represents the impact of effect E_k ($k = 1, 2, \dots, n'$) at a specific location (x_r, y_r) in the environment, considering all the available failure modes. Furthermore, Equation (4) defines the aggregated risk denoted

by $RF(x_r, y_r)$. This parameter is associated with a specific location defined by x_r and y_r , taking into account all the potential effects resulting from the failure causes present in that environment. Here, S_k represents the severity of effect E_k .

$$I_{E_k}(x_r, y_r) = \sum_{j=1}^{n^*} P_{(F_j, E_k)} \times I_{F_j}(x_r, y_r) \quad (3)$$

Table 3. The probability of each failure mode parameter for hTetro robot.

| | | Failure modes | | | |
|-----------------|-------|---------------|-------|-------|-------|
| | | F_1 | F_2 | F_3 | F_4 |
| Failure causes | C_1 | 0.1 | 0.2 | 0 | 0.7 |
| | C_2 | 0.1 | 0.2 | 0 | 0.7 |
| | C_3 | 0 | 0.1 | 0.8 | 0.1 |
| | C_4 | 0.3 | 0.7 | 0 | 0 |
| Failure effects | E_1 | 0.75 | 0.1 | 0 | 0.55 |
| | E_2 | 0.1 | 0 | 0 | 0.3 |
| | E_3 | 0.1 | 0.9 | 0.3 | 0.1 |
| | E_4 | 0.05 | 0 | 0 | 0.05 |
| | E_5 | 0 | 0 | 0.7 | 0 |

Algorithm 1. Assigning Cell Costs.

```

1: Input: Start, End, Obstacles
2: Output: Cell Costs
3: procedure Wave Front Algorithm
4:   Set Visited  $\leftarrow$  []
5:   Set Evaluated  $\leftarrow$  []
6:   Find Free Cells in the environment
7:   Set Current  $\leftarrow$  Start
8:   while Visited = Free do
9:     Include Current in Visited
10:    Set Priority of the Current  $\leftarrow$  1st
11:    Find non-visited free Neighbours for Current
12:    for Cell in Neighbours do
13:      if Cell in TNeighbours then
14:        CellCost = 15 - 2  $\times$  RF(Cell)
15:        Add Cell to Evaluated with Priority + 1
16:      elseif Cell in XNeighbours then
17:        CellCost = 20 - 2  $\times$  RF(Cell)
18:        Add Cell to Evaluated with Priority + 1
19:      end if
20:    end for
21:    Set Current  $\leftarrow$  Cell in Evaluated with Highest Priority and Not in Visited
22:    if Neighbours = [] and Not Visited = Free then
23:      Set Free  $\leftarrow$  Terrain Obstacle with Min-RF
24:    end if
25:  end while
26: end procedure

```

$$RF(x_r, y_r) = \sum_{k=1}^{n'} S_k \times I_{E_k}(x_r, y_r) \quad (4)$$

The riskiness parameter defined in this approach has been combined with the distance score to calculate the fitness function. Equation (5) presents the total score of the CPP fitness function, denoted as TotalScore_r. Here, x_r and y_r represent the cell location. The cost function, as defined in Equation (5), is then utilized within the path-planning algorithm to determine the optimal path for area coverage. In this equation, the RF component has been normalized by dividing the difference between the RF value for a given cell and the minimum RF value for the environment by the RF range of the specific environment. The parameter K is a positive constant that balances the contribution of the riskiness component and the distance score to the total score. It is configured in a way that ensures TotalScore_r > 0.

$$\text{Total Score}_r = \text{Dist Score}_r - K \times \frac{RF(x_r, y_r) - RF_{\min}}{RF_{\max} - RF_{\min}} \quad (5)$$

2.3. FMEA-Based CPP Algorithm

This research incorporates the path transform (PT) method proposed in ref. [31], where the cell cost is determined by the distance from the goal and the distance from obstacles. However, instead of relying on the distance from obstacles, this

method utilizes the FMEA-based RF calculation method described in the previous subsection. Furthermore, the dependencies defined in the FMEA method and the probabilities of each dependency occurring ($P_{(C_i, F_j)}$ and $P_{(F_i, E_j)}$) have been predetermined or estimated based on early experiments conducted with the hTetro floor-cleaning robot. These experiments include both on-campus experiments (as shown in Figure 1) and experiments conducted in public food courts.^[32] The estimation of these dependencies follows the process described in ref. [19], with adaptations made to suit the context of cleaning robots. The estimated values are presented in Table 3. Here, Table 3 provides the robot-specific probabilities for the mathematically defined probabilities for the general case given in Table 1 and 2

The CPP algorithm proposed in this article consists of three sections. The first part of the algorithm, presented in Algorithm 1, assigns cost values to the respective cells using a wave front algorithm starting from the End cell, which is the destination of the robot's path. In this step, the total cost value for each cell is determined by combining the distance from the End cell and the RF value calculated using the FMEA-based equations proposed in the previous section of the article (refer to Figure 3c). Equation (5) is used for this purpose. The distance cost is evaluated by assigning a score of 15 units to neighboring cells in the up, down, right, and left directions (referred to as T directions), and a score of 20 units to cells in the up-right, right-down, down-left, and left-up directions (referred to as X directions). The value of the constant K in Equation (5) is set to 50. In

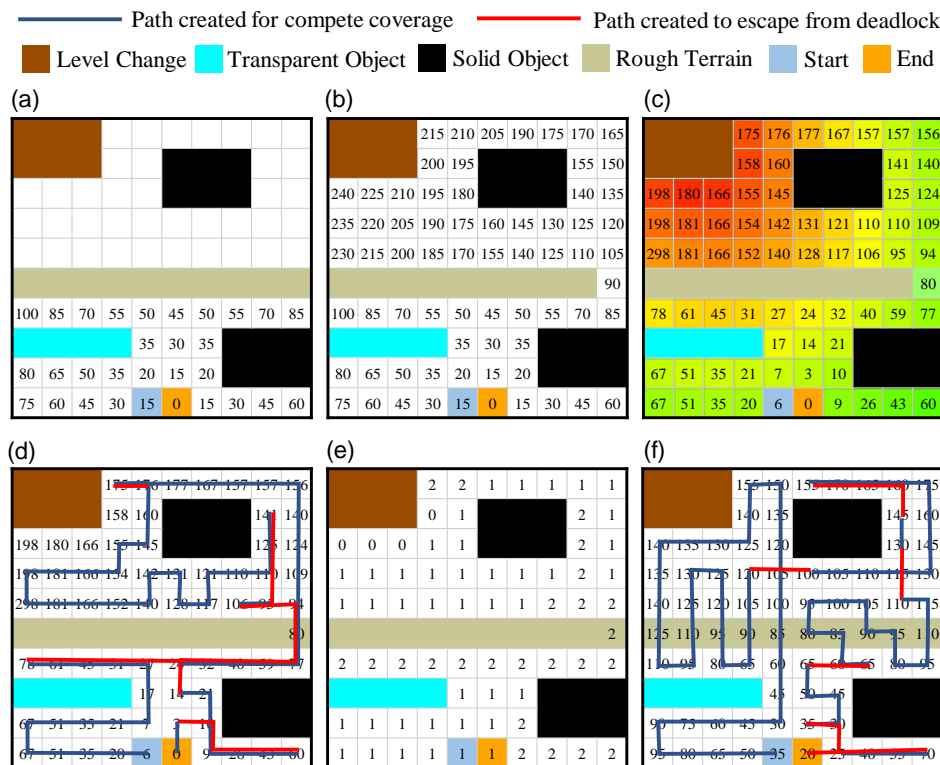


Figure 3. Failure mode and effect analysis (FMEA)-based coverage path planning (CPP) presented in a simulation environment: a) the wave front for cell cost evaluation getting terminated when there is a terrain obstacle across the environment, b) removing the terrain obstacle cell with the least risk factor (RF) and evaluating the distance cost, c) total cell costs when both distance cost and RF combined, d) CPP developed by the proposed algorithm, e) number of visits resulted in the proposed algorithm, f) CPP developed by conventional path transform (PT) algorithm.

the proposed method, the cost evaluation may encounter a situation where the grid is divided into two sections by a terrain obstacle (as shown in Figure 3a). In such cases, the distance wave front terminates before reaching the second section. Here, the terrain obstacle is considered a hazard that can be crossed, but only a minimum number of times to achieve full area coverage. Therefore, the algorithm checks if the wave front terminates before assigning cost values to all the cells. If this occurs, the terrain hazard cell with the lowest risk value, determined through the FMEA calculation, will be allowed for crossover (as illustrated in Figure 3b). Subsequently, the wave front algorithm is performed again to evaluate all the cells.

The second section of the CPP algorithm defines the criteria for creating the cell sequence that represents the path output of the algorithm (refer to Algorithm 2). The cell sequence begins from the Starting cell and selects the next cell as the highest-cost cell among its four neighbors in the T direction (up, down, right, and left) whenever applicable. As the algorithm progresses, the cells that have already been covered are saved in the visited cell list. Additionally, if any cells in the neighboring cell list belong to obstacles (failure causes) or have already been visited, those cells

Algorithm 2. FMEA-Based Path Transform Algorithm.

```

1: Input: Start, End, Obstacles, Cell Costs, Size
2: Output: Path Cell List
3: procedure Path Transform
4:   Set Visited  $\leftarrow$  [ ]
5:   Set Current  $\leftarrow$  Start
6:   Set  $N_{FreeCells} \leftarrow Size^2 - len(Obstacles)$ 
7:   while  $N_{FreeCells} \geq len(Visited)$  do
8:     Include Current in Visited
9:     Find Neighbours for Current
10:    for Cell in Neighbours do
11:      if Cell in Visited OR Cell in Obstacles then
12:        remove Cell from Neighbours
13:      end if
14:    end for
15:    if  $len(Neighbours) = 0$  then
16:      if  $N_{FreeCells} = len(Visited)$  then
17:        Break the Loop
18:      else
19:        Set Next  $\leftarrow$  ClosestNonVisitedCell
20:      end if
21:    else
22:      Next  $\leftarrow$  Cell in Neighbours corresponding to relative max in CellCosts
23:    end if
24:    Set Current  $\leftarrow$  Next
25:    Include Current in PathCellList
26:  end while
27:  return PathCellList
28: end procedure

```

Algorithm 3. Shortest Path for the Closest Non Visited Cell.

```

1: Input: Visited, Cell Costs
2: Output: Path for ClosestNonVisitedCell
3: procedure A* Path Planning Algorithm
4:   while Nonvisited in NeighbourRing is False do
5:     if Nonvisited in NeighbourRing then
6:       Set Nonvisited in NeighbourRing  $\leftarrow$  True
7:       Set NeighbourRing  $\leftarrow$  SelectedRing
8:     else
9:       NextNeighbourRing
10:    end if
11:  end while
12:  Get the ClosestNonVisitedCell in SelectedRing
13:  Create the Shortest Path for the Closest Non Visited Cell
14: end procedure

```

are removed from the list of neighboring cells. The cell sequence follows this pattern until it covers the entire area.

However, if the cell sequence reaches a point where the neighboring cells list is empty, and there are non-visited cells remaining other than the end point, this situation is considered a deadlock in the CPP algorithm. In such cases, a third section of the algorithm is initiated, which involves another wave front algorithm to locate the nearest non-visited cell (refer to Algorithm 3). This wave front algorithm considers the neighbors within an expanding square ring. Subsequently, the shortest path to this non-visited cell is determined using the A* path-planning algorithm. In Figure 3e, the path generated by this method is represented by the red lines. As a result, the cell sequence will ultimately reach the end cell after visiting all the cells.

In contrast to the conventional PT method shown in Figure 3d, which only considers solid obstacles, the proposed FMEA-based CPP method offers several advantages. Notably, when comparing the resulting paths from the two methods, it is evident that the proposed method effectively restricts the crossover of terrain obstacles. Furthermore, the proposed method, with risk treated as a continuously distributed function, effectively keeps the robot away from areas with level changes by avoiding cells with high-risk values (treating them as obstacle cells). In this method, the cells with high risk are defined as the top 15% of cells with the highest risk in the presence of level changes, and the top 3% of cells with the highest risk in the absence of level changes. These risk thresholds were selected through optimization to strike a balance between coverage and risk management.

3. Experimental Validation of Proposed Method

To validate the effectiveness of the proposed FMEA-based algorithm for CPP, a series of experiments were conducted in both simulation and physical environments. The results obtained from the proposed method were compared with the conventional

PT CPP algorithm introduced by Dakulovi et al.^[31] The purpose of this comparison was to evaluate the performance and efficacy of the proposed method. The experimental tests were conducted using the hTetro cleaning robot as the subject. In this section, we provide a detailed explanation of the experimental setup and present the results obtained from these experiments.

3.1. Experimental Setup

The proposed CPP method has been validated using the hTetro robot, which is a state-of-the-art cleaning robot developed by the Robotics and Automation Research Laboratory at the Singapore University of Technology and Design (SUTD). In the experiments conducted for this article, the hTetro robot was utilized in a square configuration. The robot's dimensions combine to form a total length and width of 50 cm, with a height of 10 cm. It consists of four sets of independently driven Mecanum wheels in each block, allowing for holonomic motion. For navigation, the robot is equipped with geared DC motors with encoders. A Raspberry Pi3 controller is utilized to handle both the low-level kinematic control functions of the robot and high-level control and processing tasks such as mapping

and localization. The mapping and localization are facilitated by the RP-LIDAR A3 and Vectornav V-100 IMU sensor used in the robot's setup.

Three distinct test sites within the premises of the SUTD were chosen for conducting the experiments (refer to **Figure 4a,d,g**). The selection of these test sites was based on the availability of failure causes, and additional failure causes were introduced at specific locations to enhance the experimental setup. Subsequently, the test sites were mapped using the LIDAR sensor to obtain occupancy data (refer to Figure 4b,e,h). The obtained LIDAR maps were then recreated as 2D grid maps in the simulation environment to facilitate the simulations, as depicted in Figure 4c,f,i. This allowed for a realistic representation of the test sites within the simulated environment.

3.2. Simulation-Based Performance Evaluation

The path-planning simulations were conducted in a Python-based simulation environment. This environment was configured as a square grid, with each grid cell representing the size of the hTetro robot. The simulation space consisted of both free cells and hazard cells. The hazard cells corresponded to

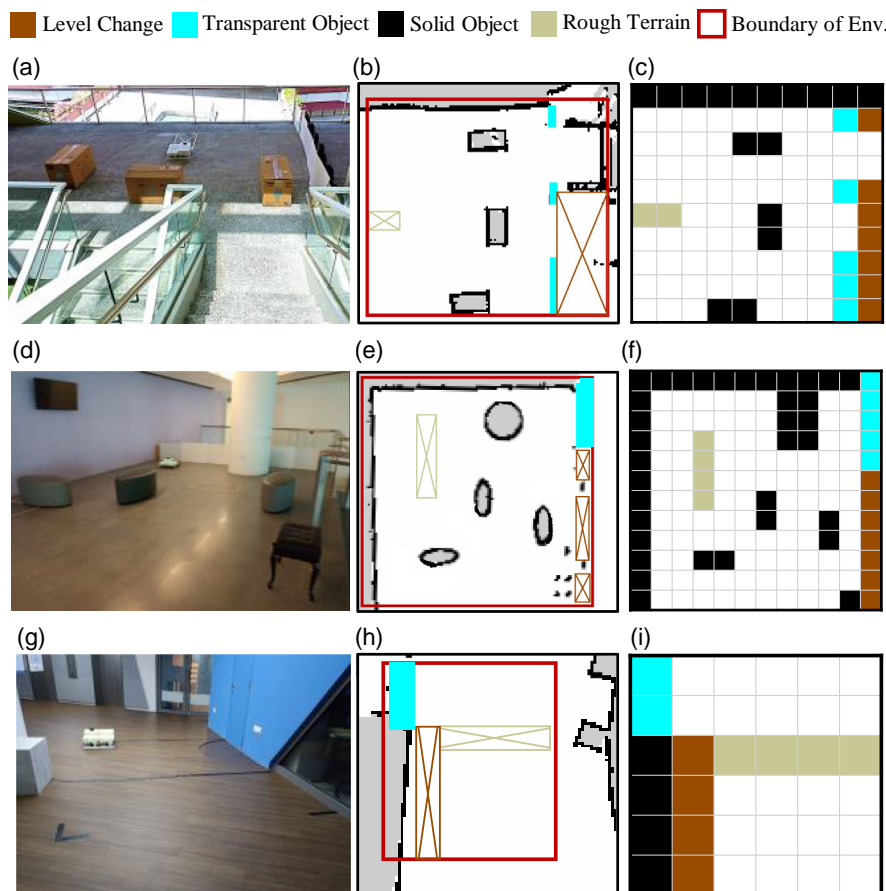


Figure 4. Test sites considered for the experiments: a) first test site (free space near the staircase in the Singapore University of Technology and Design [SUTD]— $550 \times 550 \text{ cm}^2$), b) LIDAR map of the first test site, c) simulation representation of the first test site ($11 \times 11 \text{ cells}^2$), d) second test site (in front of the SUTD auditorium— $600 \times 600 \text{ cm}^2$), e) LIDAR map of the second test site, f) simulation representation of the second test site ($12 \times 12 \text{ cells}^2$), g) third test site (in-door common area in SUTD— $300 \times 300 \text{ cm}^2$), h) LIDAR map of the third test site, and i) simulation representation of the third test site ($6 \times 6 \text{ cells}^2$).

components or factors that had the potential to cause failures in the robot. These hazard cells were categorized into the four classes of failure causes as defined in Table 1. To aid readability, each cell type was assigned a distinct color code (refer to Figure 3). For the experiments, maps of the three physical environments were utilized to perform the CPP simulations (refer to Figure 4). The results obtained from the proposed CPP method were then compared with those of the conventional PT method (refer to Figure 5b,e,h for the proposed CPP method and Figure 5a,d,g for the conventional PT method).

The comparison between the proposed FMEA-based PT method and the conventional PT method was based on three key factors for each test site. These factors included the total RF for the path, the number of passes over cells belonging to different RF ranges, and the total number of cell visits. Based on the analysis, the FMEA-based PT method demonstrated a reduction in the cumulative RF (CRF) compared to the

conventional PT method. Specifically, the CRF was reduced by 3.35%, 23.77%, and 16.94% for sites 1, 2, and 3, respectively. These results indicate that the proposed method effectively reduces risk, particularly when dealing with larger environments with more failure causes. Furthermore, the analysis of the RFs of the visited cells (as shown in Figure 6) revealed that the FMEA-based PT method had a lower number of visits in regions with higher RFs compared to the conventional method. This suggests that the FMEA-based path significantly reduces the frequency of visiting cells with higher RF values, particularly in Site 2, which had the highest number of hazard cells.

3.3. Validating the System Through On-Site Experiments

To validate the usability of the proposed CPP algorithm, physical experiments were conducted at the selected three test sites.

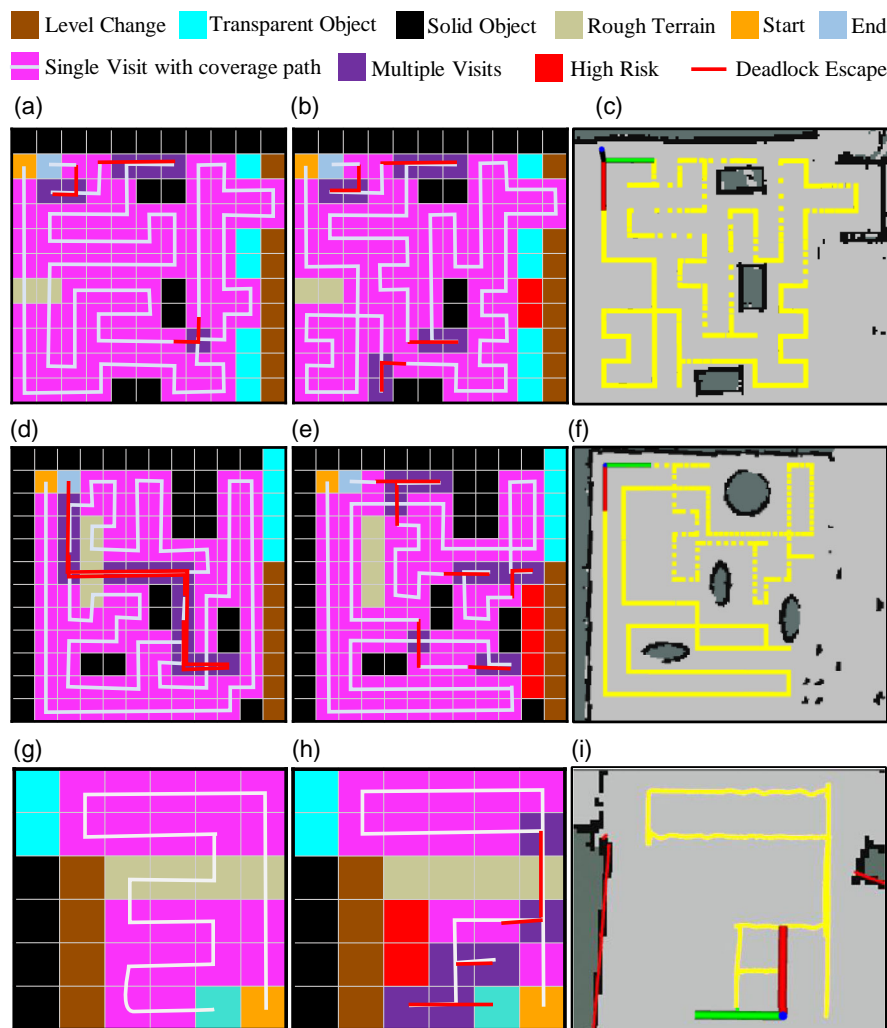


Figure 5. Results obtained by experimenting with the FMEA-based CPP in physical environments: a) path plan generated by the conventional PT method for the first site using the proposed method, b) path plan generated by the proposed method for the first site, c) the actual path followed by the hTetro robot during the experiment at the first site, d) path plan generated by the conventional PT method for the second site, e) path plan generated by the proposed method for the second site, f) the actual path followed by the hTetro robot during the experiment at the second site using the proposed method, g) path plan generated by the conventional PT method for the third site, h) path plan generated by the proposed method for the third site, and i) the actual path followed by the hTetro robot during the experiment at the third site using the proposed method.

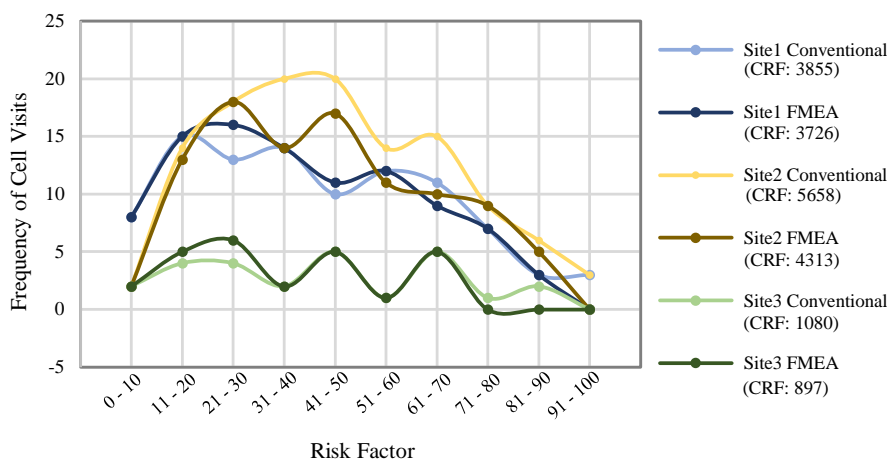


Figure 6. Comparative RF analysis results between path generated using conventional PT and FMEA-based PT (CRF: cumulative RF for the path).

The hTetro robot was programmed to navigate according to the paths generated by the FMEA-based PT algorithm. The paths taken by the robot in the on-site experiments are shown in Figure 5c,f,i. The results indicate that the robot closely followed the proposed path, demonstrating the feasibility of the path plan generated by the FMEA-based algorithm. Furthermore, the effectiveness of the proposed path plan was compared to a conventional path plan. While the robot successfully executed the proposed path plan (Figure 5i), it encountered difficulties with the conventional path plan. In the conventional plan, the robot became stuck in the terrain obstacle (wire protection) when attempting to move sideways on the obstacle (Figure 7). Human intervention was required multiple times, resulting in inaccurate localization and interruptions in autonomous tasks. In contrast, the FMEA-based path plan successfully crossed the terrain hazard only once and avoided cells near the level change hazard. As a result, the robot completed the coverage task without requiring any human intervention.

4. Conclusions and Future Directions

The proposed FMEA-based path-planning method introduced in this article aims to create a path that ensures safety in a predefined environment for robot navigation. This method utilizes the

FMEA-based robot safety assessment criteria to determine the risk levels of the environment. One notable advantage of this method is its ability to define risk levels by employing adapted FMEA-based robot safety assessment criteria, which enables the classification and analysis of various hazard types present in the environment. Compared to other grid-based path-planning methods like distance transform and PT, this method offers the flexibility to choose between limiting or completely avoiding visits to hazardous cells during robot navigation. Additionally, the method incorporates a fitness function that considers both the distance from the goal and the safety levels of the cells. As a result, the proposed algorithm can effectively cover a maximum area of the environment while balancing the trade-off between coverage and safety.

The FMEA-based path-planning algorithm proposed in this article has the potential for further improvement through additional experimentation. Increasing the number of experiments would allow for the identification of more failure causes, failure modes, and failure effects, while also enabling the observation of correlations between them. In future developments of this framework, the classification of failure causes can be expanded, leading to a more accurate safety cost map. This can be achieved by automating the identification of failure causes and the creation of hazard maps using an inspection robot equipped with deep-learning-based place recognition and reconstruction

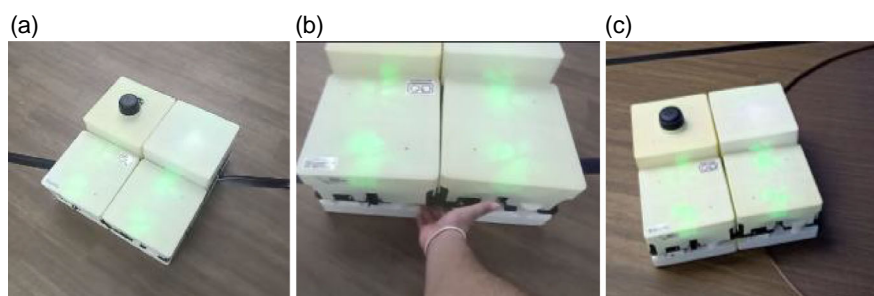


Figure 7. Failures occurred during the trialing conventional method in site 3: a) the robot got stuck on the wire protection hazard, b) needed human intervene to recover, and c) the robot lost the localization and went over the level change hazard.

capabilities. Moreover, integrating artificial intelligence (AI)-based learning into the robot can be explored in future research. The robot could initially start with predefined FMEA parameters and then fine-tune them based on its own experiences and interactions with the environment. This adaptive learning approach has the potential to enhance the robot's ability to effectively navigate and respond to different failure scenarios. By conducting more experiments, refining the failure cause classification, automating hazard map creation, and incorporating AI-based learning, the FMEA-based path-planning algorithm can be further enhanced in terms of accuracy, adaptability, and overall performance.

Acknowledgements

This research is supported by the National Robotics Programme under its National Robotics Programme (NRP) BAU, Ermine III: Deployable Reconfigurable Robots, Award No. M22NBK0054 and also supported by A*STAR under its "RIE2025 IAF-PP Advanced ROS2-native Platform Technologies for Cross sectoral Robotics Adoption (M21K1a0104)" program.

Conflict of Interest

The authors declare no conflict of interest.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Keywords

coverage path planning, floor-cleaning robots, Failure Mode and Effect Analysis (FMEA), robot inclusivity, robot safety

Received: May 18, 2023

Revised: July 16, 2023

Published online: August 27, 2023

- [1] C. Thuesen, I. Tommelein, M. Lepech, in *18th Annual Engineering Project Organization Conf. 2020*, Virtual, October **2020**.
- [2] L. E. Charles, D. Loomis, Z. Demissie, *Work* **2009**, *34*, 105.
- [3] H.-Y. Park, J.-M. Lee, in *2019 19th Int. Conf. on Control, Automation and Systems (ICCAS)*, IEEE, Piscataway, NJ **2019**, pp. 678–680.
- [4] R. Bormann, X. Wang, J. Xu, J. Schmidt, in *2020 IEEE Int. Conf. on Robotics and Automation (ICRA)*, IEEE, Piscataway, NJ **2020**, pp. 1977–1983.
- [5] S. B. P. Samarakoon, M. V. J. Muthugala, M. R. Elara, *Expert Syst. Appl.* **2022**, *201*, 117060.
- [6] S. B. P. Samarakoon, M. V. J. Muthugala, M. R. Elara, in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Piscataway, NJ **2022**, pp. 5744–5751.
- [7] E. Galceran, M. Carreras, *Rob. Auton. Syst.* **2013**, *61*, 1258.
- [8] H. Moravec, A. Elfes, in *Proc. 1985 IEEE Int. Conf. on Robotics and Automation*, vol. 2, IEEE, Piscataway, NJ **1985**, pp. 116–121.
- [9] J. S. Oh, Y. H. Choi, J. B. Park, Y. Zheng, *IEEE Trans. Ind. Electron.* **2004**, *51*, 718.
- [10] A. Khan, I. Noreen, Z. Habib, *J. Inf. Sci. Eng.* **2017**, *33*, 101.
- [11] Y. Gabriely, E. Rimon, in *Proc. 2002 IEEE Int. Conf. on Robotics and Automation (Cat. No.02CH37292)*, vol. 1, IEEE, Piscataway, NJ **2002**, pp. 954–960.
- [12] M. V. J. Muthugala, S. B. P. Samarakoon, M. R. Elara, *Expert Syst. Appl.* **2022**, *187*, 115940.
- [13] S. B. P. Samarakoon, M. V. J. Muthugala, M. R. Elara, In *2022 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, Piscataway, NJ **2022**, pp. 1–8.
- [14] B. Nasirian, M. Mehrandehz, F. Janabi-Sharifi, *Front. Rob. AI* **2021**, *8*, 624333.
- [15] R. Bormann, F. Jordan, J. Hampp, M. Hägele, in *2018 IEEE Int. Conf. on Robotics and Automation (ICRA)*, IEEE, Piscataway, NJ **2018**, pp. 1718–1725.
- [16] M. A. V. J. Muthugala, S. M. B. P. Samarakoon, M. R. Elara, In *2023 Int. Conf. on Robotics and Automation (ICRA)*, IEEE, Piscataway, NJ **2023**.
- [17] R. Li, C. Zhou, Q. Dou, B. Hu, *J. Field Rob.* **2022**, *39*, 1012.
- [18] M. V. J. Muthugala, S. B. P. Samarakoon, M. R. Elara, *IEEE Access* **2020**, *8*, 76267.
- [19] Y. Ng, M. S. Yeo, Q. Ng, M. Budig, M. Muthugala, S. Samarakoon, R. Mohan, *Sci. Rep.* **2022**, *12*, 3408.
- [20] B. Dhillon, A. Fashandi, K. Liu, *J. Qual. Maint. Eng.* **2002**, *8*, 170.
- [21] R. Woodman, A. F. Winfield, C. Harper, M. Fraser, *Int. J. Rob. Res.* **2012**, *31*, 1603.
- [22] S. Dogramadzi, M. E. Giannaccini, C. Harper, M. Sobhani, R. Woodman, J. Choung, *J. Intell. Rob. Syst.* **2014**, *76*, 73.
- [23] N. Tan, R. E. Mohan, A. Watanabe, *Autom. Constr.* **2016**, *69*, 68.
- [24] J. Guiochet, D. Martin-Guillerez, D. Powell, in *2010 IEEE 12th Int. Symp. on High Assurance Systems Engineering*, IEEE, Piscataway, NJ **2010**, pp. 104–113.
- [25] B. Dhillon, in *Proc. of the 11th National Conf. on Machines and Mechanics*, IEEE, Piscataway, NJ **2003**, pp. 86–93.
- [26] A. B. Mohan, C. Ambilikumar, in *2022 Second Int. Conf. on Next Generation Intelligent Systems (ICNGIS)*, IEEE, Piscataway, NJ **2022**, pp. 1–6.
- [27] M. R. Elara, N. Rojas, A. Chua, in *2014 IEEE Int. Conf. on Robotics and Automation (ICRA)*, IEEE, Piscataway, NJ **2014**, pp. 5593–5599.
- [28] R. E. Mohan, N. Tan, K. Tjoelsen, R. Sosa, *Dig. Commun. Netw.* **2015**, *1*, 267.
- [29] J. S. Countinho, *Trans. N.Y. Acad. Sci.* **1964**, *26*, 564.
- [30] R. J. Mikulak, R. McDermott, M. Beauregard, *The Basics of FMEA*, 2nd ed., CRC Press, New York, **2017**.
- [31] M. Dakulović, S. Horvatić, I. Petrović, *IFAC Proc. Vol.* **2011**, *44*, 5950.
- [32] A. V. Le, P. Veerajagadeshwar, Y. Shi, R. E. Mohan, M. Y. Naing, N. N. Tan, P. V. Duc, M. B. Vu, *Expert Syst. Appl.* **2022**, *206*, 117810.