IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 11 March 2025, accepted 27 March 2025, date of publication 31 March 2025, date of current version 10 April 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3556446

TOPICAL REVIEW

A Comprehensive Survey on Coverage Path Planning for Mobile Robots in Dynamic Environments

K. P. JAYALAKSHMI[1,2], VISHNU G. NAIR[1], AND DAYAKSHINI SATHISH[2]

[1]Department of Aeronautical and Automobile Engineering, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, Karnataka 576104, India

[2]Department of Electronics and Communication Engineering, St. Joseph Engineering College, Mangaluru, Karnataka 575028, India

Corresponding author: Vishnu G. Nair (vishnu.nair@manipal.edu)

ABSTRACT Coverage Path Planning (CPP) is a fundamental aspect of mobile robotics, enabling robots to navigate dynamic environments efficiently while ensuring thorough coverage of all areas of interest and avoiding obstacles. CPP is pivotal in diverse applications, from everyday tasks like vacuum cleaning, lawn mowing, and window cleaning to specialized operations such as demining hazardous areas, autonomous underwater imaging, and inspecting complex structures. This survey reviews recent advancements in CPP, focusing on algorithms and methodologies tailored for dynamic environments. It highlights the challenges posed by environmental variability, obstacle dynamics, and real-time computational demands. By synthesizing insights from existing research, this survey aims to guide future developments in CPP, paving the way for smarter and more adaptive robotic systems capable of handling the complexities of real-world scenarios.

INDEX TERMS Coverage path planning, path planning, dynamic environment, obstacle avoidance, autonomous systems.

I. INTRODUCTION

The rapid advancements in mobile robotics have sparked growing interest in autonomous systems, particularly for tasks requiring efficient exploration and coverage. CPP has emerged as a critical area of research, providing valuable solutions across many industries [1]. From agricultural robots [2], [3] tending crops to drones inspecting infrastructure [4] and autonomous vacuums cleaning floors [5], CPP plays a pivotal role in enabling effective and autonomous task execution [6], [7]. However, developing robust CPP algorithms comes with challenges, especially in dynamic environments where obstacles and operating conditions are constantly changing. This paper seeks to explore the various CPP algorithms, their applications, and how they perform in dynamic settings, providing a comprehensive overview of current research in this field. The motivation to study Computational CPP stems from the growing demand for automation in real-world tasks. Mobile robots must cover a given area completely, efficiently, and autonomously. In the past, mobile robotics was primarily focused on navigation and obstacle avoidance. However, as robots have become more integrated into everyday tasks, the importance of ensuring thorough area coverage while avoiding unnecessary overlap has gained importance. For example, an autonomous vacuum cleaner must clean every inch of a floor, and a drone surveying a farm should capture data from the entire field without gaps [8]. The emergence of CPP algorithms has transformed industries and improved the overall efficiency of robotic systems. From search and rescue to automated agriculture, CPP techniques are being applied to optimize performance across a range of scenarios. Achieving complete coverage with minimal overlap and reduced time/energy consumption is an intricate problem, especially in dynamic environments like a changing warehouse or cluttered household. These complexities drive ongoing research into CPP, prompting the development of algorithms that can handle static and dynamic environments efficiently.

The associate editor coordinating the review of this manuscript and approving it for publication was Giulio Reina.

CPP is a fundamental task in mobile robotics, focused on navigating a robot through an area to efficiently cover the entire region without missing any spots or retracing steps unnecessarily. This is especially crucial in applications where the robot's objective is to perform tasks like cleaning, surveying, or inspecting large spaces. CPP has relevance across diverse fields, including household robotics, agriculture, industrial automation, healthcare, and even space exploration. For example, in agriculture, autonomous drones equipped with CPP algorithms are used to inspect vast fields, gathering essential data on crop health, irrigation, and soil conditions [9]. Similarly, CPP is applied in industrial settings, where robots may autonomously inspect or maintain infrastructure, ensuring comprehensive coverage without redundancy. CPP also finds significant use in healthcare environments, where disinfection robots need to thoroughly cover entire rooms to sterilize surfaces, particularly during public health crises.

The scope of CPP extends beyond static, obstacle-free environments. A key challenge is developing algorithms that can handle diverse environmental conditions, from simple, structured spaces to complex, unstructured, and dynamic settings. While static environments may have predictable obstacles, dynamic environments like busy warehouses or public spaces introduce moving elements and evolving conditions, making path planning far more complicated. Adapting to these varied contexts is a crucial part of the CPP domain. As CPP technology evolves, its applications continue to expand, addressing increasingly complex and dynamic scenarios. Beyond ensuring efficient coverage, the field now focuses on optimizing factors like energy consumption, time efficiency, and adaptability to real-time changes. Given its extensive and growing use, CPP has become a vital area of research within the broader field of autonomous systems, contributing to innovations that enhance the performance and autonomy of mobile robots in real-world environments.

Dynamic environments add an extra layer of complexity to CPP. Unlike static scenarios where all obstacles and conditions are known in advance, dynamic environments are characterized by unpredictable changes that can occur at any time. This might include moving obstacles, shifting terrain, or unpredictable weather in outdoor applications [10]. In such environments, a CPP algorithm must not only plan an efficient path, but also be capable of real-time adaptations. For example, a drone inspecting a construction site faces a constantly changing layout due to ongoing work, new materials, or relocated machinery. The drone's path-planning algorithm must be flexible enough to detect these changes and adjust the path accordingly, while ensuring complete coverage of the designated area. Similarly, warehouse robots need to avoid human workers, other robots, and newly placed obstacles that weren't present during initial planning. The ability of CPP algorithms to effectively handle dynamic changes is crucial for successful robotic missions without delays, rework, or wasted energy. This adaptability improves the safety and reliability of robotic systems, especially where human-robot interaction is inevitable. Advancements in real-time sensing and data processing have enabled CPP algorithms to become more reactive, allowing them to handle increasingly complex dynamic scenarios. As the field evolves, there is a growing emphasis on developing efficient and resilient algorithms that can adapt to unpredictable environmental changes.

This paper is categorised into several key sections. The introduction provides a brief overview of CPP, highlighting its importance in mobile robotics and various real-world applications. In the next section, we delve deeper into the Classification of different CPP environments, laying the groundwork for understanding the significance of this field. Next section includes a detailed analysis of CPP techniques in static environments. Next, we present a discussion on the techniques of CPP used in dynamic environments. We also discuss Multi robot coordination in Dynamic CPP. Finally, the paper concludes with discussion on applications of dynamic CPP and discussing the overview of challenges and future research directions, identifying gaps in the current body of knowledge and suggesting areas that warrant further investigation [11]. The CPP process must take into account aspects such as time, energy, or degree of difficulty of the environment. The intention is to make sure that all points in the given area are monitored at least once while eliminating redundant movements and circumferencing obstructions and performing other dynamic tasks [12]. Key terms used in CPP aid in outlining the area and the targets of various approaches. For example, the notion of 'coverage' here would mean that the robot shall go to every part of the space set. Path planning means the whole process of finding a way, or combination of ways, for the robot to operate in order to cover all areas. Besides, every system involves an operating environment, this refers to the volume within which the robot is working and it can be from a limited fixed space (such as a room) to extensive active settings (like a factory or crop field).Other noteworthy terms include 'obstacles', which are objects or limits that the robot has to bypass while sweeping the area, and 'revisit', which is when the robot goes back over where it has been, a practice that CPP algorithms usually try to avoid. It is also employed the term ''coverage rate'', which is designed to find out how productive the robot was in covering the working space. The terms like ''localization'' and ''mapping'' are often linked to CPP, as the robot must know its own position within the workspace and be aware of the layout to plan its path effectively [13], [14].The CPP concepts are shown in Figure 1.

A. CATEGORIES OF CPP ALGORITHMS

CPP can be vaguely arranged into different degrees of applicability depending on the type of coverage needed and coverage planning method used. Familiarity with these classifications is useful in formulating methods that answer given requirements and limitations [15].

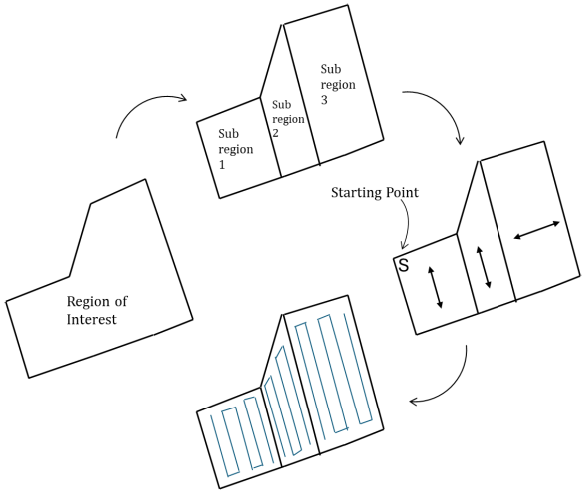

FIGURE 1. The Concept of CPP: The region of interest is subdivided into three subregions, each covered using CPP algorithms.

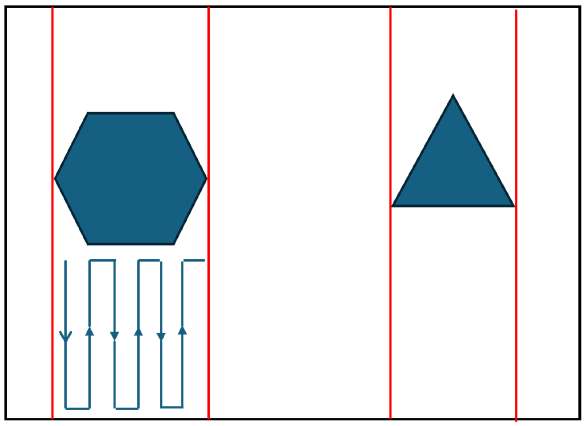

FIGURE 2. Boustrophedon Decomposition: The workspace is divided into subregions (red divisions), with obstacles represented by shaded areas. The blue line indicates the robot's coverage path.

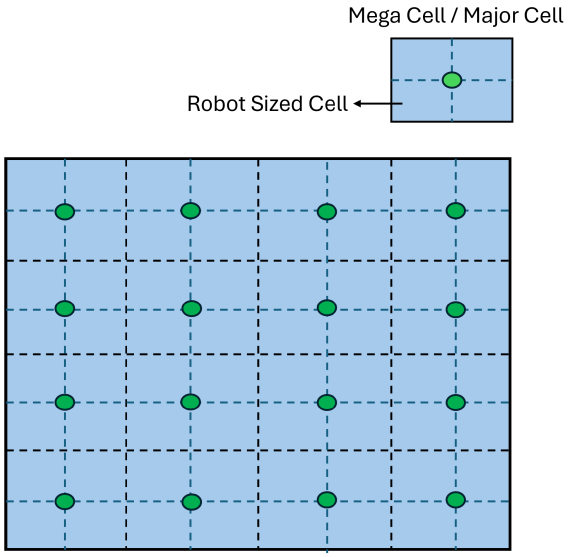

FIGURE 3. Approximate Cellular Decomposition: The workspace is divided into major cells, each consisting of a 2D × 2D grid. The center of each cell, referred to as the major node, is represented by a green dot.

1) COMPLETE VS PARTIAL COVERAGE

Among the most basic categorizations in CPP is the one on the class of coverage provided that is either complete coverage or total coverage. Also called complete coverage planning, this applies to situations when the robot has to traverse every location in the space. All of these points in the containment area have to be reached which is common in cleaning applications (e.g. floor cleaning) [15]. Although coverage is not the problem being solved, in [16] writers integrate exploration and goal-seeking challenges in a way that could result in comprehensive coverage of the region. A CPP method based on boustrophedon was proposed by Choset [17] as shown in Figure 2. The problem that the authors of [18] focus on is that the robot may need to return to its starting position, recharge, and then resume coverage when its battery is restricted (defined as the maximum distance it can go when completely charged). An online coverage technique based on approximate cellular decomposition that incorporates exploration is presented by Song and Gupta [19].

Several algorithms like approximation cellular decomposition-based CPP algorithms, as those found in [20] and [21], concentrate on covering only the cells that are totally free of impediments as shown in Figure 3. These cells are usually the same size as the robot's or the coverage tool's footprint. The robot may occasionally need to retrace its course with graph-based CPP algorithms such as BSA, extended-BSA [22], which use approximate cellular decomposition, and BDC [17], which utilize precise cellular decomposition. It is challenging to completely avoid path retracing because this occurs when the algorithm reaches a dead end and needs to retract and restart. In [23], the authors offer an approach based on approximate cellular decomposition, attempting to reduce coverage overlap while achieving exact coverage.

Creating complete coverage methods requires the robotics system ever visit a point aimed for all surface of the floor with in its defined area and it works where the efficiency of the systems targets the use of complicated route making systems to prevent the same places from being covered net types [24].On the other hand, in some cases, it is acceptable not to provide coverage for all the locations, but only for a few essential ones. For instance, in the case of surveillance, there may be no need to perform a full surveillance of an area, and instead only specific areas need to be monitored for activity. Partial coverage is also important in conducting retrieval operations during which objects are perceived in specific locations, such as determining the air temperature or air contamination in a factory or outdoor areas [25].The choice between complete and partial coverage depends on the task's objectives and the constraints imposed by time, energy, or the robot's capabilities. While complete coverage ensures that every point in the target area is visited, partial coverage

focuses on monitoring or inspecting only specific regions of interest. This is particularly useful in applications such as surveillance, where only critical areas require attention, or in retrieval operations where specific environmental data, like air temperature or contamination levels, need to be collected [26].

2) ONLINE VERSUS OFFLINE PLANNING

In CPP, the distinction between online and offline planning and control strategy is also an important area of differentiation [27], [28]. Offline planning implies a circumstance where the robot works in a known environment and related tasks since all environmental data is provided in advance [29], [30]. Hence a long and efficient path planning predetermination can be done, especially in static or semi-static circumstances. For instance, a robot moving in a well known environment, say, the floor of the warehouse or an office building that has already been mapped out, can have its entire movement programmed before it embarks on the task. This one is rather straightforward but is usually applied when the robot has to cope with a changing or a completely unknown environment. It is frequent in dynamic environments where there are going to be moving obstacles or new obstacles appearing at unexpected times. A rescue robot in a disaster area or a delivery robot in a crowded environment will have to plan where to go while on the go. Online planning is also more demanding on computation resources since the robot has to collect and process a lot of data from the location before any move is made out [31]. In addition to online and offline planning, other classification criteria also exist in CPP, encompassing various factors such as environmental characteristics, robot capabilities, and task-specific requirements.

B. KEY CHALLENGES IN CPP

CPP continues to be a significant challenge in the field of robotics. It revolves around improving the efficiency of planning a path that covers an area completely, while also keeping computational demands low and ensuring that the robot avoids collisions. The ideal coverage path would reduce needless expenses including energy use, travel time, overlapping routes, and sharp curves. However, CPP faces several key challenges, including dealing with uncertainties, navigating unexpected obstacles in complex environments, and optimizing the robot's path—all of which are critical hurdles in advancing robotics technology [32]. Several studies have demonstrated the effectiveness of using a single robot for area coverage, often in tasks like cleaning a room or other small, confined spaces. However, when it comes to covering larger areas, depending on just one autonomous vehicle can be risky. Issues like mechanical failures, sensor malfunctions, or battery depletion could jeopardize the mission. To address these concerns, researchers are increasingly focusing on improving coverage efficiency by using multi-robot systems. This approach offers clear benefits over single-robot methods, including shorter operation times and increased reliability in coverage. Despite these advantages, advancing multi-robot CPP technology presents significant challenges, especially when dealing with the constraints of complex and expansive environments [33].At the same time, it's essential to address the limitations of sensing capabilities and communication challenges, which can lead to positioning failures in multi-robot systems [?]db@bib:34. To overcome scalability issues caused by these constraints, distributed control networks are employed using either centralized or decentralized approaches. Moreover, ensuring that the team of robots is strategically resilient is critical. This means that neighboring robots should be capable of taking over and re-planning tasks if another robot fails [34]. Poorly managed task scheduling can lead to robots idling unnecessarily. Since one robot's position greatly affects the other, efficient coordination and task distribution are consequently critical issues in multi-robot systems for area coverage. Strong coordination and task allocation techniques must be used to cut down on overall coverage time and divide the workload equally among all robots in order to maximize CPP's efficiency. Ultimately, using multiple robots instead of just one offers more fault tolerance and system redundancy.

Avoiding obstacles is similar to steering a car to prevent accidents and damage. In large areas, especially when multiple robots are involved, their paths are often pre-planned due to limitations in sensors and battery life. Typically, these robots can only plan their paths in a flat, two-dimensional space because accounting for movement in three dimensions such as elevation changes is challenging. As a result, they struggle to effectively cover areas that aren't flat, like underwater environments. While 2D planning is simpler, it can lead to issues such as sensor overlap, which might leave some areas uncovered. Since the real world is rarely flat, planning in 3D would provide a more accurate approach, particularly for applications involving drones or underwater robots. Cellular decomposition is one of the most straightforward ways to break down an environment into smaller parts when it is already known [35]. This method divides the region into smaller pieces, such squares or other forms. In three-dimensional spaces, CPP often focuses on covering specific areas of interest, like critical parts of a structure, to evaluate its quality [36].Achieving the most efficient path, known as path optimality, typically involves finding the shortest route, similar to solving the Traveling Salesman Problem (TSP), where the goal is to plan a path with minimal travel cost to visit all points across various regions of interest (ROIs) [37], [38], [39], [40]. This presents a major challenge in CPP, as both TSP and CPP are NP-hard problems. Numerous studies have tried to integrate TSP and CPP by using TSP solvers to determine the visiting order for regions and then planning the optimal path to cover all sub-regions effectively [41], [42], [43]. The link between local and global coverage paths, including the path inside each ROI, the order of sub-region visits, and the entry-exit paths,

must therefore be taken into account while tackling integrated TSP and CPP challenges [4], [44], [45], [46].

C. METRICS FOR EVALUATING CPP ALGORITHMS

In evaluating CPP algorithms, various metrics are utilized to assess their performance, helping both researchers and practitioners gauge coverage efficiency, computational complexity, adaptability to dynamic environments, and persistence in surveillance tasks [47]. Coverage percentage is a key metric that measures how effectively the robot covers the designated area. A high coverage percentage indicates thorough coverage, while a low percentage suggests missed spots or gaps, which is crucial in tasks such as cleaning or disinfection [48]. Energy efficiency is another critical factor, particularly for mobile robots with limited battery capacity [27]. Algorithms that enable the robot to cover the area while minimizing energy consumption are highly valued, especially in outdoor scenarios where access to charging stations may be limited. Path length plays a significant role in evaluating coverage, as it directly impacts energy consumption and operational time. A shorter path generally improves efficiency, but this must be balanced with factors like obstacle avoidance and real-time adaptability [49]. Algorithm computation time is crucial in real-time applications, as it affects the robot's ability to quickly respond to environmental changes [50]. A highly efficient algorithm ensures fast decision-making without compromising coverage performance. Time-to-complete-coverage, particularly relevant in persistent surveillance tasks (e.g., urban security monitoring or UAV-UGV coordination), is a vital performance metric. Faster coverage completion ensures timely updates in dynamic environments, making it an essential consideration for real-world applications [51]. Redundancy and overlap rate also influence performance, as excessive overlap leads to inefficiency, while too little may result in uncovered areas. Optimizing coverage without unnecessary repetition enhances both speed and resource utilization [52]. Robustness is essential for handling uncertainties and unexpected environmental changes without compromising efficiency or path length. This is especially important in unstructured or dynamic environments, such as search-and-rescue missions or agricultural robotics, where adaptability is key. By integrating these diverse evaluation metrics, researchers can holistically assess the effectiveness of CPP algorithms across various applications.

II. CLASSIFICATION OF ENVIRONMENTS

The features of the environment greatly influence the design, choice, and application of algorithms in CPP. The complexities of the environment directly influence the robot's path-planning approach, including adapting to changes, avoiding obstacles, and navigating unstructured terrains. Classifying these environments helps understand the challenges faced by mobile robots and guides algorithm development. This section explores different environment classifications, including static and dynamic settings, as well as structured and unstructured spaces. It also discusses the challenges posed by dynamic obstacles and time-varying constraints, along with real-world applications illustrating the practical implications of these classifications [53].

A. STATIC VERSUS DYNAMIC ENVIRONMENTS

One of the fundamental ways to classify environments in CPP is based on whether they are static or dynamic. A static environment is one in which the layout, obstacles, and other features remain constant throughout the robot's operation. In these environments, the robot has the advantage of being able to plan its path in advance, as all information about the space is known beforehand. For example, cleaning robots operating in an empty office at night or drones inspecting a pre-mapped agricultural field would typically operate in static environments [54]. These scenarios are relatively predictable, allowing for optimal path-planning algorithms that can minimize energy consumption, reduce redundancy, and ensure efficient coverage. However, in real-world scenarios, environments are often less predictable. This brings us to dynamic environments, where the conditions, obstacles, and layout change over time, requiring the robot to continuously adapt to its surroundings. A dynamic environment is characterized by unpredictable factors such as moving objects, changing terrain, or even the presence of humans or animals. For instance, a drone monitoring a busy construction site, where materials and machinery are constantly in flux, or a warehouse robot navigating among workers and moving pallets, would need to operate in a dynamic setting. The main challenge in dynamic environments is the need for real-time adaptability [55]. The robot must be equipped with sensors and algorithms that allow it to detect changes in the environment and update its path accordingly. While static environments allow for offline planning, dynamic environments require online planning, where the robot makes decisions on the fly. This increases the computational complexity, as the robot must balance the need for real-time responses with the goal of achieving efficient coverage. Moreover, the ability to handle dynamic environments opens the door to a broader range of applications, such as search-and-rescue missions, where robots must navigate through rapidly changing disaster zones, or autonomous vehicles that must safely operate on busy urban roads.

B. STRUCTURED VERSUS UNSTRUCTURED ENVIRONMENTS

CPP environments can be classified as structured or unstructured based on spatial organization, predictability, and obstacle distribution. Structured environments are well-organized and follow predefined layouts with clearly defined boundaries, regular shapes, and predictable obstacles, as seen in office buildings, factories, warehouses, and residential homes, where robots can rely on predefined maps and systematic path planning for efficient navigation [56], [57].

In contrast, unstructured environments are irregular and unpredictable, lacking predefined boundaries or patterns. These include forests, mountains, disaster zones, and construction sites, where obstacles vary in size, shape, and location, and the terrain is uneven. Robots operating in such environments must depend on real-time sensor data and adaptive algorithms rather than pre-existing maps to dynamically generate paths [58]. Despite these challenges, advancements in sensor technology, deep learning, and real-time data processing have significantly improved robotic performance in complex environments, enabling machine learning-based perception, predictive modeling, and adaptive path planning to enhance navigation and coverage efficiency [59], [60].

C. DYNAMIC OBSTACLES AND TIME-VARYING CONSTRAINTS

CPP becomes even more complex when dynamic impediments and time-varying constraints are present, especially in dynamic and unstructured environments. Dynamic obstacles refer to any moving or changing objects in the environment, such as people, vehicles, machinery, or animals. These unpredictable elements create significant challenges for path-planning algorithms, as the robot must continuously update its path to avoid collisions while still ensuring full coverage of the area. For example, a warehouse robot operating alongside human workers must navigate around moving individuals and forklifts while still completing its tasks efficiently [61]. Similarly, autonomous vehicles in outdoor environments like parks or busy streets must navigate through traffic, pedestrians, and other unpredictable elements [62]. Detecting and predicting the future positions of these dynamic obstacles requires real-time sensing and decision-making capabilities [63]. In some cases, robots may even need to temporarily pause or reroute their paths to avoid collisions, which can impact overall coverage efficiency. Time-varying constraints add another layer of complexity to CPP. These constraints arise when certain areas of the environment become accessible or inaccessible at different times. For example, in a factory setting, some sections of the floor may be occupied by machinery or workers during certain periods, while being free for the robot to cover at other times. Similarly, in outdoor environments, weather conditions or changes in lighting can affect the robot's ability to navigate through specific areas. These time-varying constraints force robots to operate with a higher level of situational awareness. The robot must plan its path not only based on the current layout, but also anticipate how the environment will change over time. This requires sophisticated algorithms capable of predicting future states and adjusting the robot's behavior accordingly. Additionally, time-varying constraints often necessitate prioritization in path planning, where certain tasks must be completed before others based on the availability of specific regions [64].The combination of dynamic obstacles and time-varying constraints presents a unique challenge for CPP algorithms. They must strike a balance between adaptability and efficiency. While traditional path-planning may focus solely on covering the area, advanced CPP approaches must account for these additional variables to ensure safe and effective navigation in complex, real-world environments.

D. REAL WORLD APPLICATIONS: EXAMPLES AND USE CASES

The classification of environments into static, dynamic, structured, and unstructured categories, along with the challenges posed by dynamic obstacles and time-varying constraints, directly impacts the real-world applications of Continuous Path Planning (CPP). In this section, we explore various examples and use cases that illustrate how CPP algorithms are applied in different types of environments, highlighting the importance of environmental classification in determining the best approach to path planning. One prominent application of CPP in static, structured environments is in household robotics. Robotic vacuum cleaners, used in homes and offices, operate in relatively predictable spaces where the layout and obstacles (such as furniture) are stationary [65]. These robots typically rely on predefined maps, using efficient path-planning algorithms to ensure complete floor coverage while minimizing energy consumption. The static and structured nature of these environments allows for optimal path planning, where the robot can follow a carefully designed route to clean every area without unnecessary overlap.

In contrast, dynamic, structured environments, such as warehouse automation, present more challenges. Robots used to transport goods, perform inventory checks, or assist in packing and sorting must adapt to changes while still operating within a structured space. The presence of human workers, other robots, and moving equipment creates a dynamic environment, requiring the robot to continuously monitor its surroundings and adjust its path to avoid collisions while ensuring all designated areas are covered [66]. In unstructured environments, such as agriculture, CPP algorithms must prioritize adaptability and real-time decision-making. Agricultural robots, like autonomous tractors or drones, operate in large, outdoor environments with uneven terrain and unpredictable obstacles (e.g., plants, animals, terrain variations). These robots cannot rely on predefined maps and must use sensor data to navigate through the field [67]. Similar to this, robots used in search and rescue missions have to find survivors by navigating through debris, fallen buildings, or hazardous terrain. The success of these missions depends on the robot's capacity to plan and modify its path in real-time in these extremely unpredictable conditions, where the layout may alter over time owing to aftershocks or other hazards [68]. Autonomous vehicles, which function in dynamic and frequently unstructured contexts like city streets or highways, represent another new use of CPP. To ensure safe and effective navigation, these vehicles must avoid people, obey traffic signals, and

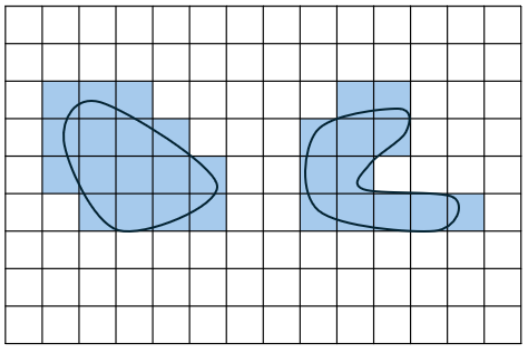

FIGURE 4. Grid cells containing obstacles of different shape. The shaded region shows the area covered by the obstacle.

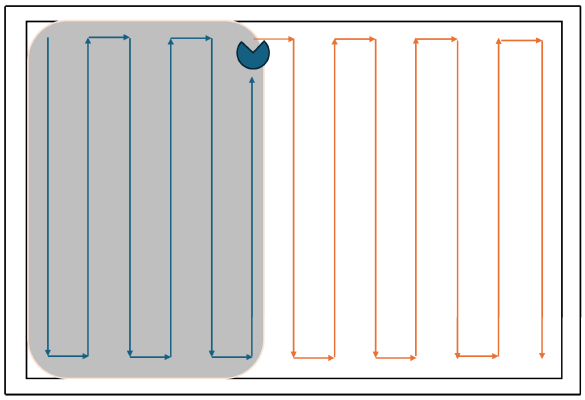

FIGURE 5. Zig-Zag Movement of Robot. The shaded regions represent the areas already covered (darker) and the areas to be covered (lighter) as the robot completes the zigzag path.

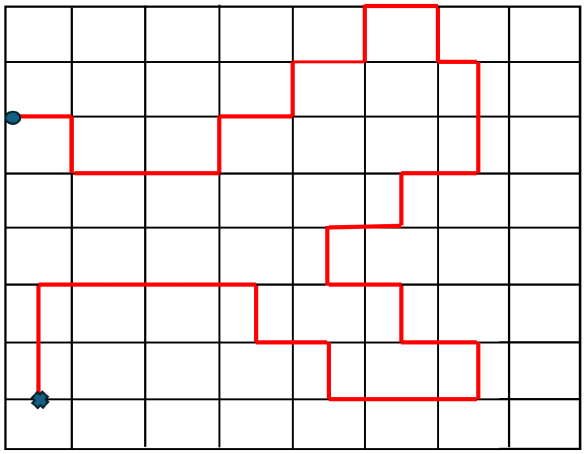

FIGURE 6. Random Walk Method: The robot's movement covering the given workspace is depicted in red.

maneuver through traffic. This calls for sophisticated CPP algorithms [69].

III. CPP TECHNIQUES IN STATIC ENVIRONMENTS

In static environments where the layout remains constant, various CPP techniques have been developed to ensure efficient and comprehensive coverage. These techniques leverage the predictability and fixed nature of static environments to optimize the robot's path, minimizing redundancies and ensuring thorough coverage. This section examines four key approaches used in static environments: traditional grid-based methods, graph-based algorithms, sampling-based techniques, and heuristics and optimization approaches.

A. TRADITIONAL GRID-BASED METHODS

The use of grid-based techniques, as seen in Figure 4, is among the oldest and most natural approaches for CPP in static situations. According to these methods, the surroundings are separated into a consistent grid, with each cell denoting a tiny, distinct region. To make sure the entire region is covered, the robot's job is to visit each of these grid cells. Because of their simplicity and ease of use, grid-based approaches are widely used, especially in structured contexts with well defined layouts and known barriers.

Grid-based methods offer several advantages, particularly in terms of clarity and precision. The robot can easily track which cells it has visited and which remain unvisited, ensuring complete coverage. Additionally, the resolution of the grid can be adjusted to suit the specific task at hand—smaller cells offer higher accuracy, while larger cells allow for faster computation and reduced path length. One of the most well-known grid-based techniques is the lawnmower pattern, in which the robot moves in a back-and-forth, zigzag motion across the grid as shown in figure 5. This approach is highly effective in environments where obstacles are minimal or stationary, such as cleaning robots working in empty office spaces or drones inspecting large fields. The lawnmower pattern ensures that the robot covers the area systematically, without missing any sections or overlapping unnecessarily [27]. Another method called random or stochastic walk is a probabilistic process often used to model animal search patterns or movements when exploring unknown territories as shown in Figure 6. Various variants have been explored in the context of environmental exploration and coverage. The fixed linear method and the variable step method are the two main Random Walk (RW)-based approaches for area coverage. The robot frequently travels straight until it runs into obstacles and rotates at random angles in the fixed linear approach. CPP algorithms that incorporate RW, spiral motion, boustrophedon motion, and wall following in cleaning systems were introduced by Hasan et al. [70]. However to improve coverage rates, Liu et al., suggested an online random coverage technique [71]. To guarantee thorough area coverage, the variable step approach calculates a set of RW directions based on the probability distribution of the robot's step lengths. In collaborative mobile robot swarm systems, such as Brownian motion (BM), the variable step approach is frequently used [72].

Grid-based approaches can have some drawbacks, though. The main difficulty is that as the grid resolution gets finer, the computing complexity rises dramatically. Furthermore, the grid might not correctly depict the actual arrangement

of the space in settings with asymmetrical or unstructured obstructions, which could result in inefficiencies or less-than-ideal routes. Hybrid approaches or more sophisticated algorithms are frequently needed to address these problems, particularly in settings with more intricate layouts or barriers.

B. GRAPH BASED METHODS

In static contexts, graph-based algorithms are yet another effective tool for CPP. Graph-based techniques represent the environment as a network of interconnected nodes, in contrast to grid-based techniques that partition the environment into regular cells. The edges between nodes indicate potential routes the robot could follow, and each node represents a distinct spot in the workspace. The robot's objective is to move around the graph, stopping at every node while making sure the entire area is covered. The Spanning Tree Coverage Algorithm is a popular graph-based technique that creates a spanning tree across the graph that encompasses every node [20], [73]. The robot then covers the entire area by following the tree's boundaries. This method works especially well in structured spaces like factories, warehouses, and office buildings where the layout is simply depicted as a system of walkways or corridors. The robot can effectively cover the workspace while avoiding obstacles and eliminating duplicate travel by using the spanning tree to guide its movement.

As illustrated in Figure 7, the spanning-tree coverage (STC) based CPP algorithm divides the workspace into a finite sequence of distinct cells using either grid-based or cell decomposition techniques. It then creates a spanning tree on the graph of matching mega-cells, splitting them into four sub-cells of the same size as the robot. Using tree traversal methods such as depth-first search, this method enables the robot to effectively navigate each empty cell. The robot does not, however, completely cover a mega-cell if an obstruction takes up a whole sub-cell inside of it. In their paper, the authors suggested a full-STC method that makes sure robots cover free sub-cells in order to optimize area coverage [21]. STC has also been expanded to improve online strategies for multi-robot systems; nevertheless, the path followed is dependent on the initial positions of each robot and can result in high overlap rates and backtracking problems, which can affect energy efficiency [74], [75]. In order to overcome these difficulties, Kapoutsis et al. [76] developed an area division algorithm that optimizes robot placement and task assignment. However, this method is unable to deal with circumstances in which impediments block free sub-cells, especially when robots are positioned along the same axis. Furthermore, a separate method [77] divides the workspace into cells of varied sizes using a hierarchical quadtree structure. This is followed by spanning tree construction that takes into account various edge lengths. Although this approach balances task assignment and reduces repeated coverage, it may cause over-segmentation, which would raise task costs.

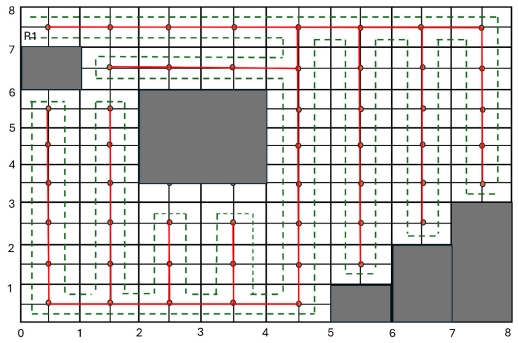

FIGURE 7. Spanning Tree Coverage: The region of interest is divided into grid cells. The shaded grey area represents static obstacles. The red line illustrates the spanning tree generated as the robot moves from one cell to another, while the robot's coverage path is depicted in green.

The STC algorithm was first presented by Gao and Xin [78], who used auction and bidding procedures to solve multi-robot CPP problems. Building a pseudo-STC to create virtual edges was another method [79] that was proposed in a different study, particularly when barriers fill mega-cells. Using a wall-following algorithm, robots moved through sub-nodes by navigating along obstacle boundaries. The method was further improved by Pham et al. [80], who concentrated on reducing backtracking and raising coverage rates. In order to accomplish this, they planned routes along spanning-tree edges in an anticlockwise orientation to reach unvisited mega-cells, taking into account mega-cells that were partially occupied by barriers when creating the C-space boundary contour. Comparing the experimental findings to the full-STC technique, better coverage rates were shown. An adjacency graph structure based on connectedness between minor nodes was also proposed in another study [81], which allowed robots to cover mega-cells that were partially filled by barriers. Frontier-based exploration and STC algorithms were combined in a hybrid solution [82] to overcome energy consumption issues in online CPP. Centralized control methods are frequently used in recent research on multi-robot-based STC algorithms, which can increase memory and computing complexity. In order to ensure work allocation among robots and path regeneration in the event of robot failure, Dong et al. [83]. presented an artificially weighted STC based on a decentralization technique to conduct coverage tasks in a distributed way. This method might, however, ignore the workload required to operate robots, leading to an imbalance in workload. In practical situations, fault tolerance is still a major obstacle.

The application of Eulerian and Hamiltonian circuits, as shown in figure 8, is another popular graph-based method. A Hamiltonian circuit makes precisely one visit to each node, whereas an Eulerian circuit makes exactly one visit to each edge of a graph. These circuits are helpful in settings where the robot has to go to every point (or edge) in the workspace without going back to any places too often. The robot can

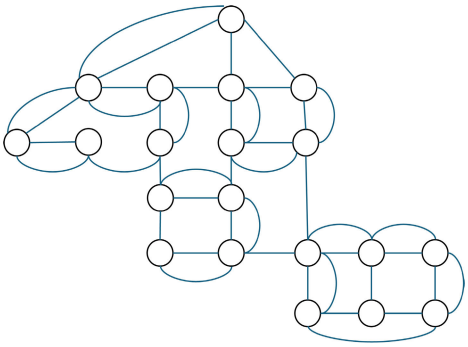

FIGURE 8. Euler and Hamiltonian Graph: The graph nodes are represented as circles, and the paths connecting the nodes are depicted as blue lines.

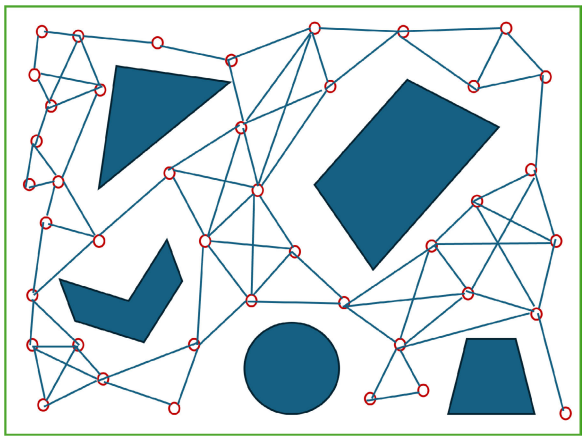

FIGURE 9. Robot Sample-Based Motion Planning: The blue shaded regions represent the obstacles in the workspace, the orange circles depict the sampled nodes, and the blue lines indicate the paths covered by the robot.

ensure full coverage while minimizing its journey length by identifying the best Eulerian or Hamiltonian circuit [84]. One advantage of graph-based algorithms is their ability to handle environments with complex layouts, including those with irregular obstacles or non-uniform structures. By modeling the environment as a graph, these algorithms can find efficient paths that take into account the specific arrangement of obstacles and open spaces. Additionally, graph-based methods can be combined with other techniques, such as grid-based methods or optimization approaches, to further enhance their performance in specific scenarios. However, graph-based algorithms also have some drawbacks. Constructing the graph can be computationally expensive, particularly in large or highly complex environments. Additionally, these algorithms may not always produce optimal paths in environments with non-uniform obstacles or dynamic elements, where real-time adaptability is required.

C. SAMPLING BASED METHODS

By creating random samples of the environment instead of explicitly partitioning it into grids or graphs, sampling-based techniques provide an alternative to CPP. In vast or high-dimensional landscapes, where conventional grid-based or graph-based approaches would be computationally prohibitive, these techniques are very helpful. The main benefit is that sampling-based techniques produce a collection of arbitrary locations or configurations in the surroundings, which the robot can then use to map out its route by joining them in a fashion that guarantees total coverage. In order to tackle coverage and planning challenges, the traditional method uses random sampling. In recent years, algorithms for probability sampling-based planning (SBP) have become powerful instruments for solving complicated planning problems with optimum and heuristic solutions. A node sampling approach is typically used by these algorithms to map the environment from configuration space, producing a set of nodes within the search environment at random. Sensor-based (visual-based) inspection tasks can be optimized with SBP's probabilistic completeness, especially when it comes to exploration. One of the probability sampling-based plans is depicted in Figure 9.

By creating a roadmap inside the configuration space, the Probabilistic Roadmap (PRM) planner is a technique for path planning and querying [85]. The roadmap is created during the planning stage by randomly generating a predefined number of nodes within the robot's configuration space and connecting pairs of nodes with straight lines while avoiding obstacles. The outcomes of the planning phase are then used to plan a path between the initial and goal configurations during the query phase [86]. A grid-based PRM was used by Dias et al. [87] for search and rescue missions during earthquakes. PRM is frequently used in conjunction with search algorithms such as the A* algorithm [88] to optimize path planning and obstacle avoidance. For example, the PRM algorithm and the A* algorithm were used in a specific study [89] to produce collision-free paths and optimal sequence paths between industrial robot measurement points, respectively. Simulation studies showed that by using a Traveling Salesman Problem (TSP) solution, this method might shorten cycle time. However, because nodes are positioned randomly, PRM may limit robot coverage close to barriers and borders. In addition, when obstacles collide, PRM eliminates the matching nodes and edges. A highly effective search planner, the Rapidly Exploring Random Tree (RRT) algorithm builds a graph for exploration in free or obstacle configuration spaces using an incremental technique in tree structure form [90]. It is especially made to manage kinodynamic planning and efficiently traverse high-dimensional spaces. Because it doesn't need a sampling setup to create a roadmap during the learning phase, RRT is quicker than PRM for single-query situations [91]. According to Zaheer et al. [92], RRT generates smoother routes and performs better in terms of calculation time than PRM. Furthermore, a bidirectional search strategy [93] put out by a different study makes it easier for initial and goal trees to grow quickly together, joining them to provide the shortest path for consistent searching. RRT-generated

pathways, however, might not be the best option for resolving planning issues. By providing an asymptotically optimal solution, RRT*, a modified form of RRT, can improve path quality [94]. Apart from sensor-based planning techniques, the sampling-based view planning method provides an additional way to optimize activities that need both view planning and motion planning features. The main uses for view planning are in modeling and exploring tasks. In order for the robot's vision system to handle perspective planning and CPP for target coverage, sensors are essential. Techniques such as the Traveling Salesman Problem (TSP) and the Set Cover Problem (SCP) identify the smallest number of perspectives required to cover the goal structure and the viewpoints, respectively. Coverage planning problems are then solved using a variety of planning algorithms, including decompose planners, optimal methods, and greedy techniques. Numerous research use the Next-Best-View (NBV) technique to choose appropriate perspectives for online CPP tasks based on sensor data and the robot's current location. Before the planner creates the best coverage path, the robot's onboard sensors investigate and sense the target area.

Motion planning techniques based on sampling have a number of important benefits, particularly in static settings. They are ideal for complex setups or vast workstations because to their excellent scalability. Second, the robot can create its coverage path using the samples gathered during exploration, negating the need for a predetermined map of the surroundings. This adaptability is especially helpful in situations when the environment is hard to model using conventional grid-based or graph-based representations or is only partially known. Methods based on sampling have drawbacks. Even though they use random sampling, the distribution of the samples can affect how well the coverage path works. Due to unequal sampling, the robot can overlook crucial locations or be unable to create an effective path. Furthermore, these techniques may necessitate substantial computational resources, particularly in complicated situations with obstructions or high-dimensional landscapes.

D. HEURISTICS AND OPTIMIZATION APPROACHES

To improve the efficacy and efficiency of CPP in static situations, researchers have used optimization approaches and heuristics. By taking into consideration variables like energy consumption, path length, and obstacle avoidance, these strategies use mathematical models, heuristics, and optimization techniques to identify the most effective coverage paths.

1) GREEDY ALGORITHM

One popular technique for resolving optimization issues is the greedy algorithm. It operates by making a sequence of decisions and following through on them without taking subsequent actions into account [95]. In order to strive for an overall optimal solution, it frequently chooses the best choice available at each stage [96]. Because it concentrates on short-term profits, it doesn't always identify the optimum solution, despite being straightforward, simple to use, and generally quick. To plan and optimize coverage paths, graph search algorithms such as A*, D*, and Theta* mix various motion patterns. These algorithms determine the shortest path between two points in a graph, modifying the path in the event that the robot runs into hidden spots or obstructions. They are essential for effectively resolving the CPP problem, but because of their high computing costs, finding paths in big grid maps can be difficult. Algorithms for searching nodes in a network include Depth First Search and Breadth First Search [97]. Despite their efficiency, both have disadvantages. Deep spaces can be challenging for DFS, as it might not always discover the shortest path. Because of its search mechanism, BFS consumes a lot of memory. While BFS can produce paths with fewer turns, DFS is better at reducing overlap and turns in coverage planning [98], [99], [100]. DFS has been used by researchers to design cleaning pathways, although it may necessitate sophisticated robots [101]. Coverage paths with few turns have been produced using BFS, especially in grid-based workspaces [102].

A popular technique for determining the shortest path from a single source node in a network with all edges having non-negative costs is Dijkstra's algorithm [103]. By visiting nearby vertices from the initial node according to their individual cost functions, the algorithm creates a shortest path tree. Almadhoun et al. [104] used Dijkstra's algorithm to explore and visit every node at a low cost, demonstrating effective path coverage in indoor situations. Dijkstra's algorithm was used to determine the least weighted path after Yehoshua et al. [105] presented a spiral STC technique to optimize coverage paths. The coverage percentage is improved by this combination and an approximation approach. In order to decrease revisiting nodes and maximize area coverage inside each stripe layer, Cheng et al. [106] used Dijkstra's algorithm to calculate the shortest pathways between subgraphs of stripe layers. The A* algorithm uses a heuristic function to evaluate surrounding vertices by calculating the cost of the path from the present point to the target [107]. It efficiently finds the shortest path by intelligently choosing nodes rather than thoroughly examining the full map. This approach has demonstrated efficacy in reducing turns and expediting path-finding tasks by utilizing cost functions [108], [109]. In order to achieve optimal coverage, Viet et al. [110] used the A* algorithm with a backtracking strategy for CPP, even though doing so resulted in a substantial memory use for storing backtracking points. The A* method was used by Cai et al., to find the shortest route between avoiding dead zones and arriving to uncovered areas [111]. The robot's diagonal movement around barriers presents difficulties, though, as it may cause excessive overlap and revisits without sufficiently covering adjacent cells during obstacle avoidance [112]. Dynamic environments are easily navigated by the D* algorithm [113]. It is a variant of the optimal A* algorithm that is intended to optimize cost solutions in order to adaptively replan

courses when the robot encounters barriers. Dakulovic et al., used the D* algorithm to determine cost values with the goal of minimizing path overlap and avoiding node revisits throughout the replanning process [114].

2) META HEURISTIC ALGORITHMS

To identify the best answers to a variety of optimization problems, evolutionary algorithms (EAs) imitate natural evolution [115]. They evaluate a fitness function to determine the quality of the solution and incorporate variation operators such as crossover and mutation. This function rates each answer according to how effective it is. EAs play a key role in improving the effectiveness of genetic searches, especially when it comes to solving practical optimization problems in mobile robot CPP [116].

Inspired by biogenetics principles, the Genetic Algorithm (GA) is a population-based stochastic meta-heuristic that emphasizes survival and breeding of the fittest to solve search problems [117], [118]. It provides near-optimal results quickly, particularly when processing in parallel. In order to solve the Traveling Salesman Problem (TSP) under CPP, Wang and Bo applied GA to solve the Traveling Salesman Problem (TSP) under CPP [119]. In order to reduce path overlap and expense, Hameed et al. [120], [121] used GA to optimize driving direction and track sequence. Shen et al. [122] used GA to optimize path connection order across many fields, hence increasing energy efficiency. In contrast to circular and sampling-based CPP techniques, Ellefsen et al. [123] used a multi-objective planner with EA in Autonomous Underwater Vehicles (AUVs) to create collision-free coverage trajectories, guaranteeing a balanced approach to coverage and energy consumption. For area coverage problems, the Genetic Algorithm (GA) has robust global search capabilities; however, stability issues arise from the intricacy of its search space, requiring a significant amount of computation time [124], [125]. Sadek et al. [126] addressed this by introducing a multi-objective GA in conjunction with Dynamic Programming (DP) for online CPP. This improved the speed of convergence towards optimal solutions by substituting a deterministic crossover process for the randomized one [127]. In their implementation of GA for pool cleaning, Batista and Zampirolli [128] used a twofold fitness function to evaluate chromosome efficiency and lower robot energy consumption.

Inspired by the collective behavior of live organisms, swarm intelligence was developed by Beni and Wang [129] with the goal of utilizing the collective intelligence that results from swarm agents cooperating with one another [130]. Because of its adaptability and great efficiency, this method is especially helpful for tackling global and non-linear optimization problems in real-world scenarios. Its goal is to build probability-based search algorithms for optimization issues [131]. Particle Swarm Optimization (PSO) [132], Ant Colony Optimization (ACO) [133], and Bee Colony Optimization (BCO) [134] are some of the

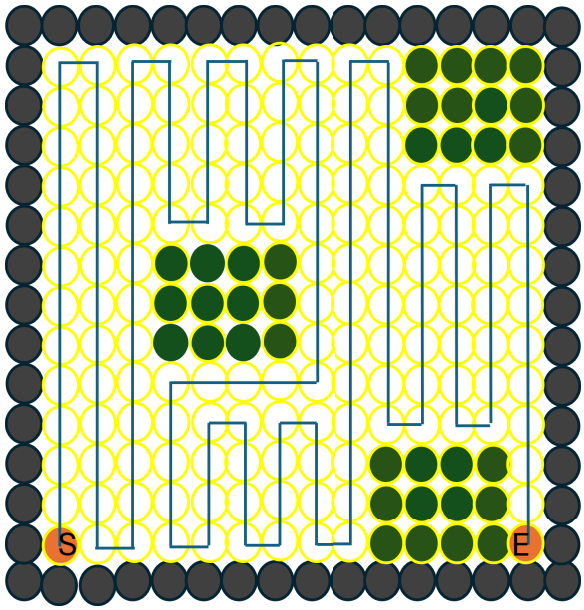

FIGURE 10. CPP using Genetic Algorithm: The green shaded regions represent obstacle areas, while the blue line illustrates the robot's coverage movement within the workspace using the genetic algorithm.

optimization algorithm classes used in CPP. These swarm intelligence algorithms provide optimal coverage solutions by using the movements of particle populations to determine the shortest path or reach a target in the fewest amount of time. Inspired by the swarming behavior of organisms, PSO is a meta-heuristic algorithm used in many applications [135]. Lee et al. [136] used an online CPP approach based on PSO to create smooth coverage paths in high-resolution grid maps, while another study used PSO in conjunction with clustering distribution factors to facilitate area coverage in each division map, and Sahu and Choudhury [137] used PSO to create trajectories for global target coverage. In order to optimize dynamic route planning, Lin [138] investigated both single-objective and multi-objective PSO [139]. In contrast to more conventional techniques like cattle herding, Wang et al. [140] demonstrated that PSO-based CPP produced less duplicate coverage. Even though PSO can search globally at first, it may have trouble finding local minima, which could cause slower convergence rates later on in the search process [141]. ACO, a probabilistic method used to identify the best paths in complex optimization problems, was inspired by the foraging activity of ants [142], [143], [144]. Benefits of its implementation include parallel computation [145] and resilience [146], [147]. Nevertheless, ACO algorithms might face difficulties like slower convergence rates and a propensity to become trapped in local optima [148], [149]. Better ACO algorithms have been developed to overcome these problems using improved pheromone update methods to avoid local minimum traps. Using GA and ACO algorithms to reduce energy consumption, Le et al. [150] presented novel cleaning and tiling robots (hTetro, hTetrakis [151], and hTrihex [152]) made for CPP tasks.

These robots may change their shape to maximize coverage effectiveness in certain areas. Meanwhile, Han et al. [153] employed gliders with back-and-forth motion to span sea levels, applying the ACO algorithm to travel the shortest obstacle-avoiding paths, even in tough conditions affected by dynamic factors such thermocline-induced variations in communication radius. The BCO is a swarm intelligence method based on bio-inspired machine learning, similar to ACO and PSO. Caliskanelli et al. [154] presented a hybrid BCO-ACO approach to handle communication loss in multi-robot scenarios [155], as well as a pheromone-based algorithm developed from BCO [156] for multi-robot coverage. Another noteworthy method, the Firefly Algorithm (FA), is based on natural processes and is frequently used to explore uncharted territory, particularly for jobs like mine disarming [157], [158], [159]. FA is used by multi-robot systems for exploration, mining area coverage, and obstacle avoidance path optimization. The performance of FA, PSO, and BCO in coordinating swarm robotics systems with respect to energy consumption was the subject of a comparative study by Palmeiri et al. [160], [161] and Gul et al. [162].Inspired by the social hierarchies and hunting dynamics of grey wolves, the Grey Wolf Optimizer (GWO) is a modern meta-heuristic technique [163]. Wolves are divided into alpha, beta, delta, and omega groups under this paradigm, each of which represents a different role in the optimization process [164], [165], [166].

However, optimization approaches also have some limitations. They can be computationally expensive and may require significant processing time, especially in large environments or scenarios with complex constraints. Furthermore, finding the global optimum is not always guaranteed, particularly in environments with a high degree of complexity or uncertainty. Despite these challenges, optimization techniques have shown considerable promise in enhancing the efficiency and performance of CPP algorithms, particularly in static environments where the layout remains constant.

IV. CPP TECHNIQUES IN DYNAMIC ENVIRONMENTS

CPP in dynamic environments presents unique challenges. Unlike static environments where a robot can rely on a fixed map, dynamic environments require the robot to adapt its coverage path in real-time. This is due to factors like moving obstacles, shifting layouts, or time-sensitive constraints that can change the landscape. To address these dynamic challenges, researchers have developed various approaches. Reactive strategies deal with immediate changes, while learning-based methods help the robot predict and adapt to future changes. Other key techniques include adaptive and decentralized CPP, real-time path re-planning, and environment modeling for dynamic changes. This section will focus into these key CPP techniques for handling the complexities of dynamic environments. By breaking the original into two focused paragraphs, the rewrite provides a clearer, more structured overview of the topic.

A. REACTIVE AND PREDICTIVE APPROACHES

In dynamic environments, robots often use reactive methods to instantly respond to changes in their surroundings [167]. These approaches prioritize quick adjustments to avoid collisions and alter the robot's path as needed without requiring long-term planning [168]. Reactive techniques, such as potential fields, vector fields, and behavior-based models, enable robots to make rapid local decisions based on real-time sensor data [169]. However, these methods may lead to local minima or inefficient paths due to their myopic nature [170]. On the other hand, predictive methods focus on anticipating environmental changes using sensor data, historical information, or probabilistic models. Techniques such as Kalman filters, Bayesian networks, and deep reinforcement learning allow robots to forecast future states and adjust their paths proactively [171]. By identifying movement patterns of dynamic obstacles, predictive models can optimize coverage paths and reduce unnecessary detours [172]. While reactive methods excel in environments with frequent and unpredictable changes, predictive techniques are particularly effective in structured dynamic settings where motion trends can be learned and anticipated. Many advanced CPP systems combine reactive and predictive approaches to balance immediate responses to obstacles with long-term path optimization, ensuring both safety and efficiency in dynamic environments [173].

B. ADAPTIVE AND LEARNING BASED METHODS

As robots operate in more complex and unpredictable environments, adaptive and learning-based methods are essential for effective CPP. These methods enable robots to learn from experience and dynamically adjust their behavior, thereby improving coverage performance in dynamic settings. By continuously updating strategies based on real-time feedback, rather than relying on fixed rules, adaptive systems allow robots to more effectively cope with changing conditions. In machine learning, reinforcement learning (RL) is unique in that it allows agents to learn the best course of action by navigating through successive decision-making processes [174]. RL uses trial-and-error learning techniques to learn from experience, in contrast to supervised or unsupervised learning. RL challenges are defined within the framework of Markov's decision process (MDP). Figure 11 illustrates RL's flow diagram. Here, the agent interacts with an uncertain environment, selecting actions based on the current state s_t at each time step t. Subsequently, the environment responds, transitioning to a new state s_t+1 while providing feedback in the form of rewards r_t to the agent. By continually receiving new data (s_t, a_t, r_t, s_t+1), the agent iteratively self-optimizes, ultimately developing a policy pi through the training process.

In robotics, reinforcement learning (RL) is widely used, especially in recent studies on CPP [175]. The intricacy of transition probability matrices makes classical Dynamic Programming (DP) less successful when dealing with large-scale

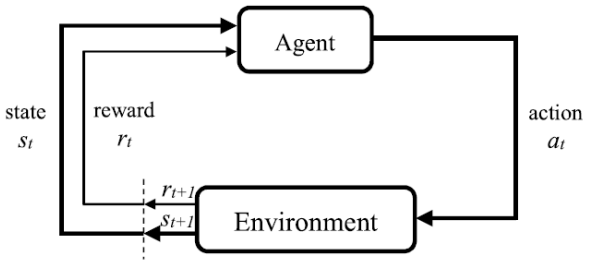

FIGURE 11. The dynamic interplay between the agent and its environment in reinforcement learning [27].

Markov decision issues, even though it excels at solving optimal planning problems. As a result, RL has become a strong substitute, providing close to optimum solutions for complex and large-scale Markov decision problems. Model-free RL techniques have recently shown promise in practical settings [176], [177], [178].

Shakeri et al. [179] highlighted how RL may be used in CPP. A 3D surface inspection technique for manufacturing lines was presented by Jing et al. who used a greedy Forward Tree Search (FTS) and Markov Decision Process (MDP) to create an online inspection planning strategy. This approach reduced the average cycle time across eight target models by 24%, outperforming the Next-Best-View (NBV) approach. A policy gradient algorithm called Proximal Policy Optimization (PPO) was used in industrial coverage spray painting. Le et al. [180] optimized the Traveling Salesman Problem (TSP) and identified low-cost routes using the PPO algorithm, which is based on RL reward functions. Although coverage efficiency may decline with environmental changes, the PPO algorithm with intrinsic rewards in [181] showed strong coverage ratios and collision prevention capabilities. To optimize CPP trajectories, Piardi et al. [182] presented a Q-learning technique that uses a grid map. To enhance coverage efficiency and give stable local optimal coverage solutions within constrained communication distances, [183] used an information map to develop a distributed Q-learning algorithm for cooperative multi-agent systems. Managing a large state-action space can be difficult in real-world situations because memory constraints make it impracticable to store information for every state-action pair. This problem is addressed by Deep Reinforcement Learning (RL), which uses function approximation rather than tabular functions. One example of this is the Deep Q-network (DQN) for mobile robot exploration and path planning [184], [185]. However, overestimation of action values might cause instability in DQN training. Luis et al. [186] developed a Double Deep Q-learning CPP technique for efficient patrolling tasks in order to counteract this. In order to maximize coverage in pertinent zones depending on observations, Piciarelli and Foresti [187] presented a bi-dimensional relevance map fed into a convolutional layer using Double DQN. Although zigzag pathways are often outperformed by RL techniques, it is crucial to note that these studies usually only show one outcome. In general, significant unpredictability and slow convergence are characteristics of DQN training. Actor-critic techniques such as the asynchronous advantage actor-critic (A3C) network and the deep deterministic policy gradient (DDPG) algorithm have been developed to overcome these difficulties. DDPG makes use of an architecture with experience replay, in which a target network is isolated from the main model and every environment sample is regularly used. A similar approach using the dual-stream Q-learning technique for target search in exploring unfamiliar settings was used by Cao et al. [188], however they ran into difficulties with task allocation. Using an actor-critic framework and an experience replay algorithm (off-policy implementation of A3C), the cleaning robot hTetro optimizes coverage time and energy efficiency, reducing coverage time by 25.88% and 29.11%, respectively, when compared to ACO and GA methods [47]. Using a long short-term memory network (building blocks for layers of a recurrent neural network), Kyaw et al. [189] addressed the TSP on deconstructed cells, resulting in a modest reduction in path length and overlapping rate. The efficiency of the RL strategy (or deep RL approach) in solving the TSP was demonstrated in studies [190], [191]. Nevertheless, the concept works well with self-reconfigurable robots in a 2D workspace; otherwise, it might greatly increase the number of turns, making traditional robots' pathways expensive. As a result, one of the biggest challenges in robotics is still how to adapt the RL technique to appropriate robot platforms in dynamically changing situations.

C. MULTIAGENT AND DECENTRALIZED CPP

Since a single robot might not be able to cover the region in a fair amount of time in dynamic environments, multi-agent systems are used, which enable numerous robots to collaborate effectively. The robots must coordinate their motions to prevent collisions and redundant coverage while adjusting to environmental changes, which creates additional issues for multi-agent CPP [192]. One intriguing method for multi-agent CPP is decentralized planning. Every robot in this decentralized system works autonomously while cooperating with other robots. Robots exchange environmental information, such as the positions of obstacles and covered regions [193]. However, each robot makes its own navigation decisions based on local data and predefined rules. This decentralized approach is especially useful in dynamic environments, where centralized control may be too slow or complex to handle rapid changes [194]. Decentralized cooperative path planning methods often leverage swarm intelligence, where numerous robots work together towards a shared objective without centralized command [195]. Emulating the collective behavior of social insects like ants and bees, swarm-based CPP systems enable robots to coordinate their actions through simple communication protocols, granting them remarkable adaptability to dynamic environments. For instance, in disaster response operations,

a swarm of drones could be deployed to search for survivors in a constantly shifting, unstable setting, with each drone dynamically adjusting its path based on the movements of the others.

D. REAL-TIME PATH PLANNING

The capacity to replan the coverage path in real-time is one of the most important prerequisites for effective CPP in dynamic contexts. In order to make sure the robot can manage unforeseen changes, like the sudden appearance of an obstacle or the discovery of a new area that needs to be covered, real-time path replanning entails dynamically modifying the robot's course as new information becomes available [167]. Real-time replanning algorithms are designed to be fast and efficient, allowing the robot to make adjustments without significantly interrupting its coverage process. One common approach is local path replanning, where the robot only adjusts the portion of its path that is immediately affected by the change, rather than recalculating the entire coverage plan. This minimizes the computational burden and allows the robot to maintain high coverage efficiency even in rapidly changing environments [196].

In some cases, real-time replanning is combined with global optimization techniques to ensure that the robot's new path remains as efficient as possible. For example, if a robot is tasked with covering a factory floor and an unexpected obstacle appears, it might replan its path locally to avoid the obstacle while still adhering to the overall coverage strategy. This combination of local adjustments and global optimization allows the robot to handle dynamic environments without sacrificing coverage quality or efficiency.

E. ENVIRONMENT MODELING FOR DYNAMIC CHANGES

Robots need a precise and current model of their surroundings in order to navigate dynamic situations efficiently. To enable the robot to make well-informed judgments about where to go and how to cover the area, environment modeling entails generating a representation of the workspace that takes into account both static and dynamic components. [197]. Environment models need to be updated frequently to account for changes in dynamic environments, where layout elements or impediments may alter over time. Occupancy grids, in which the environment is depicted as a grid of cells, each of which is designated as either free, occupied, or unknown, are one popular method. As the robot moves through the environment, it updates the grid in real-time based on sensor data, ensuring that its model remains accurate. In addition to occupancy grids, probabilistic models are often used to represent dynamic environments. These models take into account the uncertainty associated with sensor data and the movement of obstacles, allowing the robot to make predictions about how the environment might change in the future. For example, in a dynamic warehouse, a probabilistic model might predict that certain areas are more likely to become blocked due to the movement of forklifts or other machinery, allowing the robot to plan its coverage path accordingly [198]. Effective environment modeling is crucial for successful CPP in dynamic environments, as it enables the robot to anticipate changes, avoid obstacles, and ensure complete coverage. By continuously updating its model and using predictive techniques, the robot can adapt to even the most unpredictable environments, ensuring that it remains efficient and effective in its coverage tasks.

V. MULTI ROBOT COORDINATION IN DYNAMIC CPP

Coordination of many robots is necessary for successful CPP in dynamic situations where circumstances and obstacles can change quickly. Larger or more complicated environments can be handled by multi-robot systems, which also provide improved coverage efficiency. To guarantee that numerous robots cooperate well, managing them calls for close coordination. The main facets of multi-robot coordination in dynamic CPP are examined in this section, including with the function of swarm intelligence, job allocation and role assignment, communication and consensus processes, cooperative and competitive strategies, and more [199].

A. COOPERATIVE VERSUS COMPETITIVE STRATEGIES

Cooperative strategies involve robots working together towards a common goal. Each robot shares information and collaborates to cover the environment more efficiently [200]. This approach leverages the collective strengths of the robots, such as their ability to cover more ground simultaneously or coordinate to avoid obstacles. Cooperative strategies are particularly effective in dynamic environments, where robots can adapt their actions based on the real-time information shared among them. For instance, if one robot encounters a newly introduced obstacle, it can notify the other robots, allowing them to adjust their paths and avoid potential collisions. Cooperative approaches also enable load balancing, where tasks and coverage responsibilities are evenly distributed among the robots to prevent any single robot from becoming overwhelmed.

Competitive strategies, on the other hand, involve robots operating with a degree of competition, often seeking to optimize their own performance independently. In this approach, robots may compete for the same resources or areas of coverage, which can lead to conflicts or inefficiencies. However, competitive strategies can be advantageous in scenarios where individual robots are tasked with achieving specific goals or metrics, such as covering the most ground or reaching particular locations first. By fostering a competitive environment, robots might push each other to perform better, though this can sometimes lead to suboptimal overall coverage if not managed properly [201]. Balancing cooperative and competitive elements is crucial in dynamic environments. Many multi-robot systems incorporate both strategies, allowing robots to cooperate when it benefits the collective goal and compete when individual performance metrics are emphasized. This hybrid approach can enhance

the overall efficiency and effectiveness of the coverage process.

Mitra and Saha presented a centralized algorithm that allows concurrent planning and execution for multi-robot coverage tasks. Unlike traditional horizon-based methods, this approach plans paths for subsets of robots while others execute their assigned paths, optimizing both robotic and computational resources. The algorithm guarantees complete coverage of unknown workspaces and has demonstrated scalability in simulations with up to 512 robots, achieving up to 1.6× speedup compared to existing methods [202].

B. TASK ALLOCATION AND ROLE ASSIGNMENT

Effective task allocation and role assignment are central to the success of multi-robot coordination in dynamic environments. Each robot must be assigned specific tasks or roles to ensure that the entire environment is covered efficiently and that robots work together without redundant efforts [203]. Task allocation involves determining which robot is responsible for covering which area or performing which task. This process can be based on factors such as the robot's current location, its capabilities, or the state of the environment. For example, in a warehouse with moving obstacles, task allocation might involve assigning robots to different sections of the warehouse or specific tasks like obstacle detection and area cleaning. The goal is to ensure that all areas are covered without overlap and that robots can handle dynamic changes in the environment effectively.

Role assignment refers to defining the specific functions or roles that each robot will perform within the multi-robot system. This might include roles such as leader, follower, scout, or coordinator. For instance, a leader robot might be responsible for making high-level decisions and guiding the other robots, while follower robots execute specific tasks based on the leader's instructions. Role assignment helps to streamline the coordination process and ensures that each robot contributes effectively to the overall coverage objective [204]. Task allocation and role assignment can be dynamic, adapting to changes in the environment or the status of the robots. In a dynamic environment, the system must be flexible enough to reassign tasks and roles as needed to respond to new obstacles, changes in layout, or shifts in the robots' performance. Effective task allocation and role assignment enhance the system's ability to handle dynamic challenges and ensure comprehensive coverage. Gong et.al., introduced a method for adaptive redistribution of coverage paths in the event of robot failures. Utilizing the Boustrophedon Cellular Decomposition (BCD) algorithm, the approach divides the environment into cells and assigns coverage paths to each robot. In case of a robot failure, its coverage area is redistributed to neighboring robots through an iterative reassignment strategy, ensuring balanced coverage and maintaining mission objectives efficiently despite dynamic changes in robot availability [205].

C. COMMUNICATION AND CONSENSUS MECHANISMS

Communication and consensus mechanisms are vital for coordinating the actions of multiple robots in dynamic environments. Robots must share information and reach agreements on their actions to work together effectively and avoid conflicts [206]. Robots communicate information through protocols and methods called communication mechanisms. This can involve direct communication, in which robots use wireless networks to exchange data, or indirect communication, in which data is communicated via ambient markers or signals. Robots could use communication channels, for instance, to discuss coverage progress, report the discovery of new impediments, or ask for help. To make sure that every robot is aware of the situation of the environment at any given time and can coordinate their actions accordingly, dependable and effective communication is crucial.

Robots that use consensus methods are better able to come to choices and take actions when there is uncertainty or a chance for disagreement. Thanks to these systems, robots may coordinate their movements and decide as a group based on shared data. Robots that use consensus algorithms, like voting or averaging, can settle disputes and decide on the optimal course of action. When two robots approach the same region, for instance, they may utilize a consensus process to choose how to divide up the coverage duties or which robot should cover the area first [207]. Maintaining coordination and preventing conflicts in dynamic situations requires effective systems for consensus-building and communication. They guarantee that robots can cooperate well, adjust to changing circumstances, and effectively accomplish their coverage objectives.

Tang et.al., proposed the MSTC* algorithm, an enhancement of the Spiral Spanning Tree Coverage (Spiral-STC) method. MSTC* incorporates physical constraints such as terrain traversability and material load capacity to achieve a well-balanced workload distribution among robots. Simulation results indicate that MSTC* significantly outperforms existing Spiral-STC-based methods, minimizing the overall time required to complete coverage tasks [208].

Recent advancements in multi-robot exploration strategies have introduced innovative techniques for efficient task distribution and coordination [209]. One approach utilizes Genetic Algorithms (GAs) for spatial partitioning of a known topo-metric map, ensuring effective workload distribution. The Hungarian method is employed for task assignment, while the Bully Algorithm facilitates leader election. In case of robot failures, tasks are dynamically reassigned using graph re-partitioning and single-item auctions, demonstrating improved performance over existing Delaunay triangulation-based methods [210].Another approach extends the frontier tree data structure, previously used in single-robot exploration, to multi-robot systems. This method introduces a 'group' abstraction, wherein multiple robots maintain a common frontier tree for synchronized information sharing and goal assignment. Groups merge seamlessly when their explored regions overlap, ensuring a unified state

of exploration. Simulation results show that this strategy outperforms seven state-of-the-art methods in multi-robot exploration tasks [211].A decentralized relay-based approach (D-MRFTE) has been proposed for multi-robot exploration in unknown environments, particularly under communication constraints. This method employs relay robots to establish a high-latency decentralized network that maintains distributed copies of exploration data. To ensure data consistency and completeness, periodic meetups are scheduled whenever network fragmentation occurs, facilitating reliable information exchange [212].

D. SWARM INTELLIGENCE IN CPP

The idea of swarm intelligence was inspired by the way social insects, like termites, ants, and bees, collaborate to find solutions to complicated issues through straightforward local interactions. Swarm intelligence is a potent method for coordinating several robots in dynamic environments within the framework of CPP [195]. The foundation of swarm intelligence is the notion that a collection of basic agents, all of whom adhere to a set of basic rules, may cooperate to produce sophisticated and useful behavior. This means that in a multi-robot system, robots cooperate based on mutual information and local interactions instead than depending on a main controller or intricate global plans. Robots can adapt to dynamic changes, avoid collisions, and efficiently cover an environment by emulating the behavior of natural swarms. Self-organization, in which robots dynamically modify their behavior based on local interactions and environmental variables, is one of the fundamental ideas of swarm intelligence in CPP. For instance, a robot can modify its course and coordinate its actions with other robots nearby if it detects an obstruction or a change in the surroundings. The system can successfully handle unforeseen changes and adapt to dynamic settings because to its decentralized architecture [213].

One further significant component of swarm intelligence is collective behavior. Swarm systems of robots cooperate to accomplish shared objectives, including exploring the whole area or dodging impediments. Robots can jointly optimize their coverage paths and react to dynamic changes in real-time by exchanging messages and adhering to a few basic criteria. When a robot comes upon a novel impediment, for instance, it can alert other robots nearby to steer clear of the area and take detours. In dynamic contexts, swarm intelligence provides CPP with a number of benefits. It makes coordination scalable and adaptable so the system can handle big or complex areas with several robots and adjust to changes. Swarm-based methods can also increase fault tolerance and resilience since they allow the system to keep working even in the event that a few robots malfunction or have problems [214]. However, implementing swarm intelligence in multi-robot systems also presents challenges, such as managing communication, ensuring consistent behavior, and addressing potential conflicts. Despite these challenges, swarm intelligence remains a promising approach for enhancing the efficiency and effectiveness of multi-robot coordination in dynamic environments.

VI. APPLICATIONS OF DYNAMIC CPP

In many real-world applications, CPP has become essential, especially in dynamic situations where conditions are always changing. Dynamic CPP is being used more and more in many industries to handle challenging issues as technology develops. The adaptability of CPP in dynamic environments is rooted in its capacity to manage erratic situations, adjust to evolving impediments, and effectively accomplish tasks requiring prompt decision-making. This section examines the various uses of dynamic CPP, emphasizing its importance in industrial inspection and maintenance, autonomous vehicle navigation, agricultural robots, disaster response and search and rescue operations, surveillance, and security.

A. DISASTER RESPONSE AND SEARCH AND RESCUE

Search and rescue and disaster response are two of the most important and significant uses of dynamic CPP [7]. Hazardous settings can make it challenging for typical human-led search teams to locate survivors or assess damage following natural catastrophes like earthquakes, floods, or landslides. By deploying autonomous robots or drones quickly across these chaotic and constantly shifting landscapes, dynamic CPP makes it possible to promptly cover important areas. In dynamic disaster environments, debris, collapsed structures, and shifting terrains present a significant challenge. Rescue robots can continuously adjust their coverage path to take new obstructions or structural instabilities into consideration by using adaptive techniques and real-time path replanning. Robots with dynamic CPP, for example, can swiftly modify their paths to avoid dangerous situations and concentrate on the most accessible locations in the event of a building collapse or rising floodwaters [215]. Every second counts in search and rescue, and being able to move swiftly and effectively over wide regions might mean the difference between life and death. Furthermore, in order to optimize coverage, multi-robot cooperation is frequently used in search and rescue operations. Drones or ground robot swarms can operate in tandem and communicate with one another to split the task and cover different areas of a catastrophe zone. While some robots may be assigned to search for survivors, others may be charged with mapping the region thoroughly in order to support human rescue teams. These robots can work together seamlessly by taking advantage of dynamic CPP, making sure that no area of the disaster zone is overlooked.

B. AGRICULTURAL ROBOTICS

A technological revolution has occurred in agriculture, and one important aspect of contemporary farming is agricultural robotics. For these robots to efficiently traverse vast, intricate agricultural areas, dynamic CPP is essential. Farm landscapes are very dynamic, in contrast to static ones,

with impediments including moving machinery, shifting topography, and growing crops that differ in height and density. Whether they are used for crop monitoring, pest management, or harvesting, agricultural robots depend on dynamic CPP to plan the best routes that maximize coverage and adjust to changing environmental conditions [216]. For instance, in a wide field, a robot can employ dynamic CPP to avoid areas that may be momentarily inaccessible owing to equipment in use or different soil conditions, and instead concentrate on areas that are accessible. The robot can reroute its journey to get over previously prohibited parts if the conditions alter. Additionally, real-time data acquiring from sensors installed on the robots helps improve dynamic CPP in agriculture. These sensors collect data on crop growth, soil health, and moisture content, allowing robots to modify their paths according to the unique requirements of the field. To ensure that resources are used effectively and crops receive the attention they require, the robot can, for example, prioritize coverage in a field where additional irrigation or insect control is needed [217]. Farmers can minimize human intervention in large-scale farming operations while optimizing productivity and reducing waste by integrating dynamic CPP with precision agriculture advancements. Robots are essential to the future of sustainable farming because of the adaptability of CPP in agriculture, which guarantees that they can withstand the inherent volatility of farming situations.

C. INDUSTRIAL INSPECTION AND MAINTENANCE

To maintain safety and operational effectiveness in industrial contexts, huge infrastructures like factories, power plants, and oil rigs must be inspected and maintained. The dynamic nature of these locations, where people, machines, and equipment are always moving, presents difficulties for autonomous robots assigned to do inspections. Robots must be able to maneuver around these intricate industrial settings in order to conduct inspections without interfering with current processes, which is made possible by dynamic CPP [218]. Robotic inspectors are used to evaluate huge areas of infrastructure to detect any problems like corrosion, cracks, or malfunctioning equipment [219]. These inspectors are outfitted with sensors and cameras. These robots may modify their inspection routes in real time in response to environmental information, like the existence of moving machinery or movable obstacles, thanks to dynamic CPP. The robot may change its course to avoid obstacles and still cover important regions in the event that one appears, such as a worker or crane entering the area. In industrial settings, downtime is costly, and it is essential that inspections are conducted efficiently without halting production. Dynamic CPP enables non-disruptive maintenance, allowing robots to work around moving objects or personnel while continuously updating their path to cover the entire area. This reduces the need for human inspectors to enter hazardous or hard-to-reach areas, improving safety and operational efficiency. In addition to inspection, dynamic CPP can be applied to automated repair tasks. For instance, if a robot identifies a fault, such as a broken pipeline or a malfunctioning piece of machinery, it can replan its path to perform the necessary repairs, working in coordination with other robots or human operators. By ensuring continuous, adaptive coverage in dynamic industrial environments, dynamic CPP enhances both the safety and productivity of industrial operations [220].

D. AUTONOMOUS VEHICLE NAVIGATION

The rise of autonomous vehicles has sparked a revolution in transportation, and dynamic CPP plays a fundamental role in enabling these vehicles to navigate safely and efficiently through complex, ever-changing environments. Whether it's a self-driving car navigating through city streets or an autonomous drone delivering packages, dynamic CPP is key to ensuring that these vehicles can adapt to unexpected changes in their surroundings, such as pedestrians, traffic, or temporary road closures [221]. Autonomous cars must continually recalculate their routes in order to maintain efficiency and safety in urban settings where traffic patterns and obstacles are very dynamic. These cars can react to alterations in real time like the unexpected entrance of a pedestrian or the presence of a construction zone thanks to dynamic CPP. Autonomous vehicles can ensure seamless navigation by combining reactive and predictive techniques. This allows them to react to sudden changes and foresee possible obstacles based on past data or sensor input [222]. Additionally dynamic CPP-enabled multi-robot coordination is advantageous for autonomous vehicles working in fleets like self-driving taxis or drone delivery services. These vehicles can coordinate and operate more efficiently if they can communicate with one another to optimize paths that avoid traffic and shorten travel times. By avoiding moving obstacles like other drones or outside variables like wind dynamic CPP in logistics makes sure that drones can deliver packages safely. In order to ensure that autonomous cars can operate safely in a variety of unpredictable environments and to push the limits of what is possible for autonomous transportation systems dynamic CPP will be necessary as these vehicles develop.

E. SURVEILLANCE AND SECURITY

In the domain of security and surveillance where autonomous drones and robots are used to continuously scan and patrol wide areas dynamic CPP has also found extensive use. For surveillance robots to effectively cover their assigned areas they must be able to adapt to changing environments including around moving people vehicles and possible intruders. Robots can effectively cover large areas and react to events in real time with dynamic CPP in large-scale surveillance operations like border patrol airport security and critical infrastructure protection [223]. A security drone sweeping a facility for instance might have to alter its

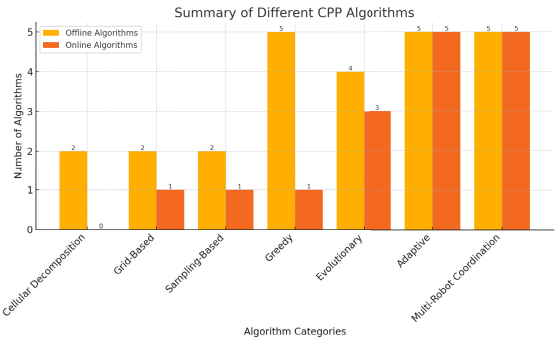

FIGURE 12. Algorithm Catergories Vs Online/Offline.

coverage path to track down a suspicious person or get around a passing car that might act as a temporary obstruction. The robot can manage these changes without sacrificing its overall surveillance objectives thanks to the flexibility offered by dynamic CPP. Multiple drones or ground robots collaborate in multi-robot surveillance systems to continuously cover sensitive areas [67]. These robots can coordinate their paths stay clear of each other and communicate to exchange real-time information about their surroundings thanks to dynamic CPP. A coordinated and adaptable surveillance network can be created if one robot recognizes a possible security threat and notifies other robots to change their routes and concentrate more on the area of concern. Security robots operating in dynamic environments might also have to contend with time-varying constraints like shifting weather crowds or lighting. A drone might have to modify its flight path for example to avoid inclement weather or to maximize its coverage path depending on the amount of foot traffic that passes through it during the day. Because dynamic CPP can manage these time-varying constraints it is the best option for dependable and efficient surveillance under a variety of circumstances. In addition to monitoring dynamic CPP facilitates responsive security measures like following and tracking intruders or reacting to alarms. Robots can for instance dynamically reroute their routes to intercept an intruder or provide real-time updates to human security teams in the event of a security breach. Surveillance systems can handle complex real-world scenarios with greater resilience and responsiveness when they utilize dynamic CPP [224].

F. RECENT STUDIES ON CPP METHODOLOGIES IN DYNAMIC ENVIRONMENTS

One notable study presents a novel CPP algorithm designed for autonomous mobile patrol robots, emphasizing fast revisit times in dynamic settings. The algorithm utilizes a pre-recorded 2D occupancy grid map to update the surveilled region within the robot's field of view at each step, subsequently creating waypoints that maximize coverage area. This approach has been experimentally validated, demonstrating its effectiveness in dynamic environments [225]. Another study introduces a complete-coverage path planning algorithm for multiple mobile robots, based on a grid reliability function. This method addresses the challenges posed by dynamic environments by ensuring that multiple robots can collaboratively achieve complete coverage. Experimental results indicate that this algorithm enhances coverage efficiency and robustness in the face of environmental changes [226]. Additionally, a comprehensive review of recent path-planning techniques in dynamic environments highlights various methodologies, including obstacle detection, path-planning strategies, formation control, and communication styles. The review discusses key trends, challenges, and gaps in current methods, emphasizing the need for more efficient and robust algorithms capable of handling complex and unpredictable dynamic environments [227].

VII. CHALLENGES AND FUTURE DIRECTIONS

As CPP continues to evolve, it is encountering several critical challenges that need to be addressed to fully unlock its potential, particularly in dynamic and complex environments. These challenges include issues of scalability, handling uncertainties, fostering effective human-robot interactions, and grappling with ethical concerns. Additionally, the rise of advanced technologies like artificial intelligence (AI), deep learning, and edge computing presents both opportunities and new challenges for CPP. This section explores these challenges in depth and discusses promising future directions for CPP research.

A. SCALABILITY AND COMPUTATIONAL COMPLEXITY

Scalability is one of CPPs main problems particularly in dynamic environments. The computational demands necessary to create effective coverage paths rise sharply as robots are deployed in bigger more complicated environments. This is especially true for systems with multiple robots as it takes exponentially more processing power to coordinate the actions and tasks of multiple robots. A major issue in this area is making sure CPP algorithms are scalable without compromising effectiveness or performance [228]. Computational complexity is another closely related issue. With large-scale environments or intricate dynamic obstacles many existing CPP algorithms find it difficult to strike a balance between producing optimal paths and computing them quickly. Robots are often required to plan their paths quickly while adapting to changing conditions in real-world applications such as industrial inspection disaster response and agricultural robotics. However creating these pathways frequently requires spending a lot of computational resources to solve extremely complicated optimization problems. In order to overcome these obstacles more effective algorithms that manage large-scale settings without taxing computer resources excessively must be created. Using distributed computing techniques where the computational load is divided among several agents or devices approximation algorithms and hierarchical planning approaches may be used in this [15].

TABLE 1. Summary of different CPP algorithms.

Category	Approach	Offline/Online	Environments Handled	Remarks
Cellular Decomposition [21]	Trapezoidal	Offline	Static Environments	Divides space into trapezoidal cells; suitable for structured environments.
	Boustrophedon	Offline	Static Environments	Efficient for structured and semi-structured environments; uses back-and-forth sweeping motion.
Grid-Based [72]	Wavefront	Offline	Static Environments	Uses breadth-first search to propagate from the start point; efficient for simple grids.
	Spiral STC	Online	Static and Dynamic Environments	Covers the area in a spiral pattern; suitable for various grid sizes and dynamic updates.
Sampling-Based Methods [89]	PRM (Probabilistic Roadmap)	Offline	Static Environments	Constructs a roadmap by sampling random points; efficient for high-dimensional spaces.
	RRT (Rapidly-Exploring Random Tree)	Online	Dynamic and Complex Environments	Real-time pathfinding in complex spaces; extends paths using random sampling.
Greedy Algorithm [99]	A*	Offline	Static and Dynamic Environments	Finds shortest path using heuristic-based cost; combines DFS/BFS features.
	D*	Online	Dynamic Environments	Adapts to environment changes in real-time; efficient for dynamic updates.
	Theta*	Offline	Static Environments	Extends A* with line-of-sight; provides shorter paths.
	DFS/BFS	Offline	Static Environments	DFS explores deeper first; BFS guarantees shortest path in unweighted graphs.
	Dijkstra's	Offline	Static Environments	Guarantees shortest path for weighted graphs; does not use heuristics.
Meta heuristic Algorithms [119]	Genetic Algorithms	Offline	Static and Dynamic Environments	Mimics natural selection; useful for complex optimization problems.
	PSO (Particle Swarm Optimization)	Online	Dynamic Environments	Swarm-based optimization; explores space using personal and global bests.
	BCO (Bee Colony Optimization)	Online	Dynamic and Complex Environments	Inspired by foraging behavior of bees; balances exploration and exploitation.
	GWO (Grey Wolf Optimizer)	Offline	Dynamic Environments	Simulates leadership hierarchy of grey wolves; suitable for global optimization.
Adaptive and Learning-Based Methods [178]	Reinforcement Learning	Online	Dynamic and Complex Environments	Learns optimal policies through trial and error; adaptable to changes.
	PPO (Proximal Policy Optimization)	Online	Dynamic Environments	Policy-gradient method balancing exploration and exploitation; efficient.
	Q-Learning	Online	Dynamic Environments	Model-free RL; finds best action using a Q-value table.
	DQN (Deep Q-Network)	Online	Complex Dynamic Environments	Combines Q-learning with deep neural networks; excels in high-dimensional spaces.
	DDPG (Deep Deterministic Policy Gradient)	Online	Continuous Action Spaces	Extends DQN for continuous spaces; efficient for robotic control tasks.
Multi-Robot Coordination in Dynamic CPP [205]	Cooperative Strategies	Online	Dynamic Environments	Robots collaborate to achieve goals; efficiency through shared information.
	Competitive Strategies	Online	Dynamic Environments	Robots compete for tasks/resources; requires interaction management.
	Task Allocation and Role Assignment	Online	Dynamic Environments	Dynamic task allocation based on capabilities and environmental changes.
	Communication Mechanisms	Online	Dynamic Environments	Communication protocols enhance coordination and decision-making.
	Consensus Algorithms	Online	Dynamic Environments	Achieves agreement on actions/paths/tasks among robots; ensures coherence.

TABLE 2. Characteristics of the motion planning problem.

Attributes	Details	Grading Scale					Best Grade
		1	2	3	4	5	
Search Efficiency	Ability to quickly identify unexplored regions or targets	Very Slow	Slow	Moderate	Fast	Very Fast	5
Path Quality	Efficiency of the path in terms of coverage or sequence	Poor	Fair	Average	Good	Excellent	5
Adaptability	Capability to perform in a changing environment	Poor	Fair	Average	Good	Excellent	5
Computational Demand	Assessment of time and resource usage	Very Low	Low	Medium	High	Very High	1
Convergence Efficiency	Speed at which a solution or sequence is achieved	Very Slow	Slow	Moderate	Fast	Very Fast	5

B. HANDLING UNCERTAINTIES AND NOISY DATA

Robots frequently have to work in noisy uncertain or incompletely supplied environments in real-world applications. In dynamic settings where the environment is ever-changing and sensor data may not always be precise or current this is especially true [27]. A major challenge for dynamic CPP is managing these uncertainties and making trustworthy decisions when dealing with noisy data. Robots may find it challenging to plan exact routes in agricultural fields due to environmental factors like wind rain or uneven terrain that can cause sensor inaccuracies. Similar to this path planning becomes even more difficult in industrial inspection when robots come across erratic obstacles or moving machinery. Under such circumstances robots ability to sift through noise and reach sound judgments even in the face of incomplete data becomes invaluable. A potential solution to these issues is the creation of stochastic planning techniques and probabilistic algorithms. These methods let robots create coverage paths that take environmental uncertainties into account and instantly adjust to shifting conditions. Additionally promising for enhancing the resilience of CPP algorithms in ambiguous circumstances are sensor fusion techniques which merge information from various sources to produce a more accurate image of the surroundings [229].

C. HUMAN ROBOT INTERACTION

Human-robot interaction (HRI) is becoming a critical field of study as robots are incorporated into increasingly human-centered environments. Robots and humans must collaborate in environments like hospitals workplaces and public areas which presents new difficulties for trust safety and communication [230]. The challenge lies in coordinating human operators and autonomous systems in a seamless manner while keeping CPP effective. Designing CPP systems with the ability to predict human behavior and modify their trajectories accordingly presents a significant challenge. For instance a robots ability to anticipate a persons next move can help prevent collisions and guarantee smooth interaction in a shared workspace where humans and robots work together. Likewise in situations such as search and rescue robots need to be able to successfully communicate with human operators in order to notify them of potential hazards and give them real-time updates on their coverage progress. The future of CPP depends on creating natural and seamless human-robot interaction through the development of intuitive interfaces and collaborative systems. To enable human operators to direct or communicate with autonomous systems this may entail employing gesture controls speech recognition or augmented reality. Improving human-robot interaction will increase safety and boost robot performance in practical uses [231], [232].

D. ETHICAL AND SAFETY CONSIDERATIONS

The ethical and safety concerns surrounding the use of robots are gaining prominence as these machines become more autonomous and are used in more delicate or important settings [233]. As the field develops a significant challenge that researchers must overcome is ensuring that CPP systems function in a way that is morally sound safe and consistent with human values. In scenarios where robots must make crucial decisions like emergency response or security applications one of the main ethical questions is autonomy in decision-making. Complex moral dilemmas that need to be carefully considered when designing CPP algorithms for high-stakes applications include for example whether an autonomous robot should prioritize saving more lives at the risk of jeopardizing its own safety or whether it should follow pre-set rules even if doing so limits its effectiveness. The question of accountability is another. Its critical to establish clear policies regarding who bears responsibility when a robot malfunctions or causes harm. This responsibility can fall on the shoulders of the robot its operator or the robot itself. Building trust in autonomous systems requires implementing strong safety protocols and making sure CPP systems follow strict ethical standards. Furthermore safety must always come first especially in settings where robots may interact with people or other machines. It takes careful planning and the application of fail-safe mechanisms to ensure that robots can navigate safely without causing harm especially in dynamic or unpredictable environments. One of the continuous challenges in CPP is creating algorithms that put safety first without compromising performance [234].

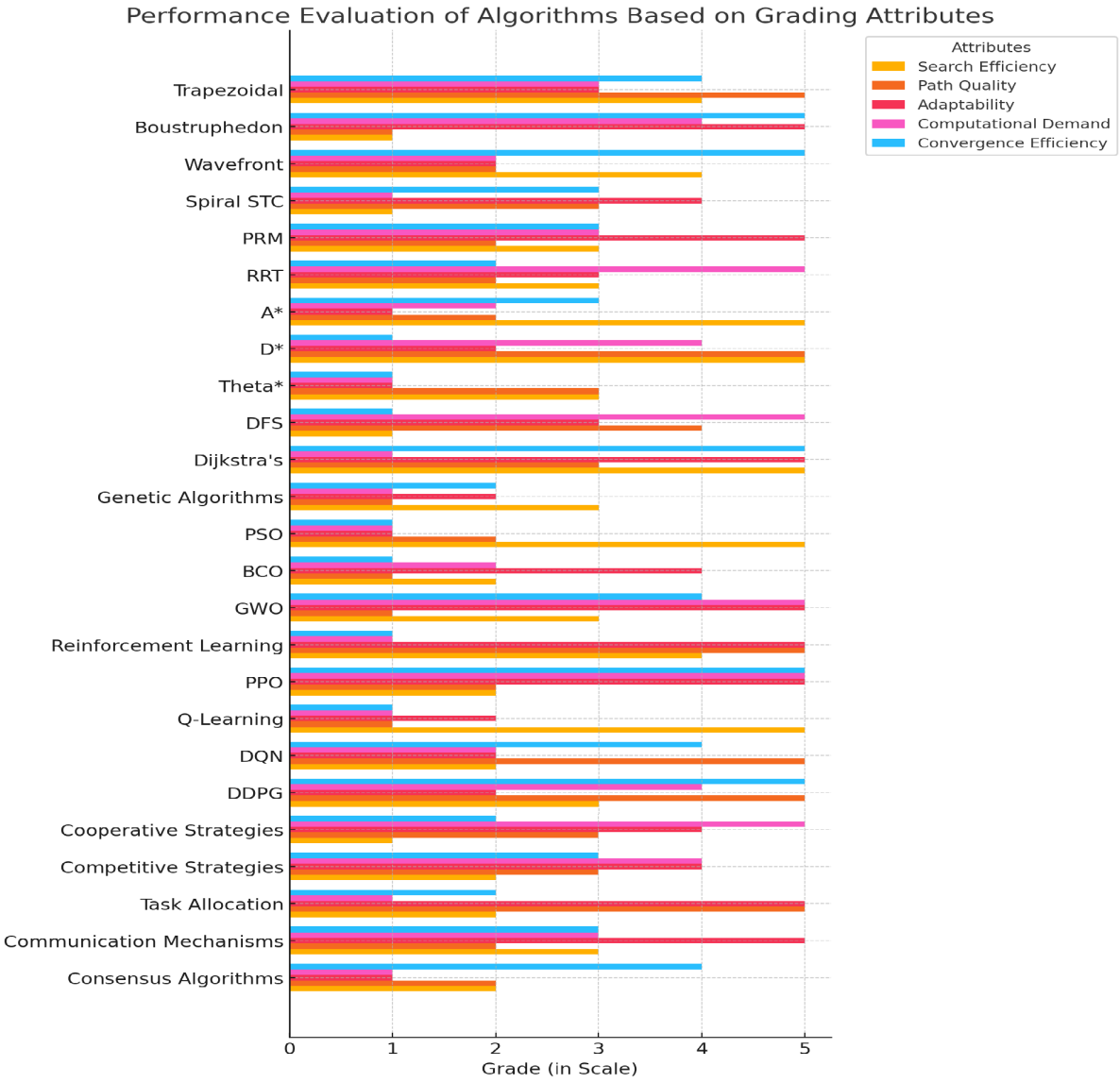

FIGURE 13. Performance evaluation of different algorithms based on five key attributes (as described in Table 2).

E. EMERGING TRENDS: AI, DEEP LEARNING, AND EDGE COMPUTING

Artificial intelligence (AI) deep learning and edge computing are some of the exciting new technologies that will shape the future of CPP. The way robots approach CPP could be completely changed by these developments especially in dynamic environments [197]. Robotics and deep learning are changing how robots perceive and engage with their surroundings. Deep learning algorithms can help robots learn from experience and become more adept at creating effective coverage paths in chaotic and complex environments by utilizing vast amounts of data. Robots can learn to modify their routes in response to particular environmental factors for instance allowing for more adaptable and sophisticated coverage strategies than those that depend solely on pre-programmed rules. Another interesting application of CPP is reinforcement learning which is more promising than learning-based methods especially in dynamic environments where robots have to constantly adjust to newly created conditions [235], [236], [237]. Robots may learn to balance exploration and exploitation to maximize coverage while minimizing computational overhead by experimenting and learning how to optimize their routes in real-time. Another new development that could improve the efficiency and scalability of CPP algorithms is edge computing. Robots can make decisions more quickly and with less latency by processing data locally on the robot or at the network edge instead of depending on centralized cloud servers. This is especially significant for applications like autonomous driving and industrial inspection where quick decisions are essential. Edge computing also makes distributed systems possible enhancing the overall effectiveness and resilience

of coverage planning by allowing several robots to work together and exchange information in real time. These cutting-edge technologies will probably be crucial in helping CPP overcome many of its present problems as they develop pushing the envelope in terms of scalability adaptability and practicality.

VIII. CONCLUSION

CPP or CPP is becoming an essential part of deploying autonomous systems in a variety of dynamic environments as technology advances. In this review we have looked at the fundamental ideas of CPP the main methods in static and dynamic environments and the practical applications of these approaches. Along with talking about the difficulties CPP is facing we also outlined some exciting future developments like the fusion of edge computing deep learning and artificial intelligence.

A. SUMMARY OF KEY FINDINGS

The flexibility and adaptability of CPP algorithms is among the most significant lessons to be learned from this review. Depending on the environment static or dynamic these techniques whether grid based graph-based sampling-based or optimization driven offer different benefits. While sampling-based and heuristic approaches work well in more complex unstructured environments grid-based methods for example offer precision and are particularly effective in structured spaces. These methods adapt to the changing needs of robotics enabling robots to operate in both straightforward and extremely dynamic environments. We observed that in dynamic environments the combination of adaptive and learning-based techniques with reactive and predictive approaches allows robots to instantly adapt to changing conditions. An exciting new frontier in robotics is the integration of multi-agent and decentralized CPP which enables flexible scalable operations in large-scale unpredictable environments by enabling groups of robots to collaborate efficiently on tasks. Furthermore the agility and responsiveness of robots in these environments have been greatly improved by developments in real-time path re-planning and environment modeling.

Important tactics like cooperative versus competitive methods task distribution and role assignment and communication and consensus mechanisms were discovered during our investigation into multi-robot coordination. These tactics are essential for the effective implementation of large-scale robotic teams. Future robotic systems that mimic the collective behavior observed in nature are glimpsed in the study of swarm intelligence and this could result in reliable and highly scalable CPP solutions. Dynamic CPP has equally compelling applications. Dynamic CPP is showing to be a vital tool in a variety of fields including automated vehicle navigation industrial inspection disaster response, search and rescue agricultural robotics, and surveillance and security. These examples show how widely applicable CPP is in transforming industries, improving safety and offering creative answers to challenging real-world issues. The summary of various CPP algorithms are shown in Table 1 and Figure 12. The performance evaluation of the same is shown in figure 13 with respect to different attributes like search efficiency, path quality, adaptability, computational demand and convergence efficiency as tabulated in table 2.

B. IMPLICATIONS OF FUTURE RESEARCH

Even though CPP has come a long way there are still many unanswered questions many of which offer opportunities for more research. Scalability is still a major problem especially in large dynamic environments and multi-robot systems. Subsequent investigations ought to concentrate on crafting algorithms that are more effective and computationally controllable capable of tackling the incessantly complex real-life situations. It may be possible to overcome these scalability issues by combining hierarchical planning techniques with distributed computing. Robots are being deployed in environments where they must rely on incomplete or imperfect information more and more so the ability to handle uncertainties and noisy data remains a critical challenge. To improve the robustness of CPP systems future studies should investigate more complex stochastic planning techniques and probabilistic algorithms. To further enhance accuracy and decision making in unpredictable environments, more work could be done on sensor fusion techniques and AI-driven error correction.

Another important area where research can advance significantly in the future is the interaction between humans and robots. For autonomous systems to be more widely integrated into society it is imperative that they be able to anticipate human behavior work well with human operators and communicate clearly in real time. Improved human-robot interaction and more approachable and user-friendly autonomous systems could result from research into more natural and intuitive interfaces such as voice commands in augmented reality and gesture recognition. Additionally the moral and security ramifications of robot use grow in significance as they acquire greater autonomy. Future studies in CPP must address ethical issues, especially those pertaining to autonomous systems accountability and high-stakes decision making. Ensuring the responsible use of robotic systems in dynamic environments requires the establishment of strong safety protocols and regulatory frameworks. Ultimately CPP has a lot of exciting opportunities as a result of the development of AI deep learning and edge computing. The adaptability effectiveness and scalability of CPP algorithms could be greatly improved by these technologies. In order to increase the capabilities of robots in real-world settings, future research should concentrate on how these cutting-edge technologies can be smoothly incorporated into already in-use systems. Examples of areas that could be investigated to improve CPP paths include reinforcement learning and edge computing which could allow for real-time coordination and decision-making amongst several robots resulting in more intelligent and adaptable systems. In summary, even though

CPP has advanced significantly, there is still much to learn. Undoubtedly, the persistent development of self-governing systems propelled by breakthroughs in artificial intelligence and robotics will yield revolutionary resolutions to the present-day predicaments. There are many exciting opportunities for engineers and researchers to push the limits of what is feasible in the field of CPP.

ACKNOWLEDGMENT

The authors would like to thank Manipal Institute of Higher Education, MAHE, Manipal, and St. Joseph Engineering College, Mangaluru, for their support and assistance during this research.

AUTHOR CONTRIBUTIONS

K. P. Jayalakshmi: Conceptualization, Methodology, and Writing the Original Draft; Vishnu G. Nair: Supervision, Review, Editing, and Submitting; and Dayakshini Sathish: Supervision, Review, and Editing.

CONFLICTS OF INTEREST

The authors affirm that there are no competing financial interests or personal relationships that could have influenced the work presented in this article. Additionally, they have no relevant financial or non-financial interests to disclose.

REFERENCES

[1] J. Chen, F. Ling, Y. Zhang, T. You, Y. Liu, and X. Du, "Coverage path planning of heterogeneous unmanned aerial vehicles based on ant colony system," *Swarm Evol. Comput.*, vol. 69, Mar. 2022, Art. no. 101005.

[2] M. Ollis and A. Stentz, "First results in vision-based crop line tracking," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 1, Aug. 1996, pp. 951–956.

[3] M. Ollis and A. Stentz, "Vision-based perception for an automated harvester," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot Syst. Innov. Robot. Real-World Appl.*, vol. 3, Grenoble, France, Sep. 1997, pp. 1838–1844.

[4] B. Englot and F. S. Hover, "Sampling-based coverage path planning for inspection of complex structures," in *Proc. 22nd Int. Conf. Automated Planning Scheduling*, vol. 22, Sao Paulo, Brazil, May 2012, pp. 29–37.

[5] F. Yasutomi, M. Yamada, and K. Tsukamoto, "Cleaning robot control," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 1988, pp. 1839–1841.

[6] S. Hert, S. Tiwari, and V. Lumelsky, "A terrain-covering algorithm for an AUV," *Auto. Robots*, vol. 3, nos. 2–3, pp. 91–119, 1996.

[7] D. W. Gage, "Randomized search strategies with imperfect sensors," *Proc. SPIE*, vol. 2058, pp. 270–279, Oct. 1994.

[8] Z. L. Cao, Y. Huang, and E. L. Hall, "Region filling operations with random obstacle avoidance for mobile robots," *J. Robotic Syst.*, vol. 5, no. 2, pp. 87–102, Apr. 1988.

[9] M. Farsi, K. Ratcliff, J. P. Johnson, C. R. Allen, K. Z. Karam, and R. Pawson, "Robot control system for window cleaning," in *Proc. Amer. Control Conf.*, vol. 1, 1994, pp. 994–995.

[10] K. R. Jensen-Nau, T. Hermans, and K. K. Leang, "Near-optimal area-coverage path planning of energy-constrained aerial robots with application in autonomous environmental monitoring," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 3, pp. 1453–1468, Jul. 2021.

[11] E. M. Arkin and R. Hassin, "Approximation algorithms for the geometric covering salesman problem," *Discrete Appl. Math.*, vol. 55, no. 3, pp. 197–218, 1994.

[12] H. Choset, "Coverage for robotics—A survey of recent results," *Ann. Math. Artif. Intell.*, vol. 31, pp. 113–126, Oct. 2001.

[13] J. Palacin, T. Palleja, I. Valganon, R. Pernia, and J. Roca, "Measuring coverage performances of a floor cleaning mobile robot using a vision system," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 4236–4241.

[14] V. J. Lumelsky, S. Mukhopadhyay, and K. Sun, "Dynamic path planning in sensor-based terrain acquisition," *IEEE Trans. Robot. Autom.*, vol. 6, no. 4, pp. 462–472, Apr. 1990.

[15] R. N. De Carvalho, H. A. Vidal, P. Vieira, and M. I. Ribeiro, "Complete coverage path planning and guidance for cleaning robots," in *ISIE 97 Proc. IEEE Int. Symp. Ind. Electron.*, vol. 2, Jul. 1997, pp. 677–682.

[16] V. R. Jisha and D. Ghose, "Frontier based goal seeking for robots in unknown environments," *J. Intell. Robotic Syst.*, vol. 67, nos. 3–4, pp. 229–254, Sep. 2012.

[17] H. Choset, "Coverage of known spaces: The boustrophedon cellular decomposition," *Auton. Robots*, vol. 9, no. 3, pp. 247–253, 2000.

[18] I. Shnaps and E. Rimon, "Online coverage of planar environments by a battery powered autonomous mobile robot," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 425–436, Apr. 2016.

[19] J. Song and S. Gupta, "A star: An online coverage path planning algorithm," *IEEE Trans. Robot.*, vol. 34, no. 2, pp. 526–533, Feb. 2018.

[20] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," *Ann. Math. Artif. Intell.*, vol. 31, pp. 77–98, Nov. 2002.

[21] Y. Gabriely and E. Rimon, "Competitive on-line coverage of grid environments by a mobile robot," *Comput. Geometry*, vol. 24, no. 3, pp. 197–224, Apr. 2003.

[22] E. Gonzalez, O. Alvarez, Y. Diaz, C. Parra, and C. Bustacara, "BSA: A complete coverage algorithm," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 2040–2044.

[23] K. R. Guruprasad and T. D. Ranjitha, "ST-CTC: A spanning tree-based competitive and truly complete coverage algorithm for mobile robots," in *Proc. Conf. Adv. Robot.*, Jul. 2015, pp. 1–6.

[24] J. H. Lee, J. S. Choi, B. H. Lee, and K. W. Lee, "Complete coverage path planning for cleaning task using multiple robots," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, Oct. 2009, pp. 3618–3622.

[25] M. M. G. Plessen, "Partial field coverage based on two path planning patterns," *Biosystems Eng.*, vol. 171, pp. 16–29, Jul. 2018.

[26] T. Cabreira, L. Brisolara, and P. R. Ferreira Jr., "Survey on coverage path planning with unmanned aerial vehicles," *Drones*, vol. 3, no. 1, p. 4, Jan. 2019.

[27] C. S. Tan, R. Mohd-Mokhtar, and M. R. Arshad, "A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms," *IEEE Access*, vol. 9, pp. 119310–119342, 2021.

[28] M. Govindaraju, D. Fontanelli, S. S. Kumar, and A. S. Pillai, "Optimized offline-coverage path planning algorithm for multi-robot for weeding in paddy fields," *IEEE Access*, vol. 11, pp. 109868–109884, 2023.

[29] J. Xie, L. R. G. Carrillo, and L. Jin, "An integrated traveling salesman and coverage path planning problem for unmanned aircraft systems," *IEEE Control Syst. Lett.*, vol. 3, no. 1, pp. 67–72, Jan. 2019.

[30] J. Wan, S. Tang, H. Yan, D. Li, S. Wang, and A. V. Vasilakos, "Cloud robotics: Current status and open issues," *IEEE Access*, vol. 4, pp. 2797–2807, 2016.

[31] G. Sharma, A. Dutta, and J. Kim, "Optimal online coverage path planning with energy constraints," in *Proc. 18th Int. Conf. Auto. Agents Multiagent Syst.*, May 2019, pp. 1189–1197.

[32] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robot. Auto. Syst.*, vol. 61, no. 12, pp. 1258–1276, Dec. 2013.

[33] R. Sun, C. Tang, J. Zheng, Y. Zhou, and S. Yu, "Multi-robot path planning for complete coverage with genetic algorithms," in *Intelligent Robotics and Applications* (Lecture Notes in Computer Science), vol. 11744, H. Yu, J. Liu, L. Liu, Z. Ju, Y. Liu, and D. Zhou, Eds., Cham, Switzerland: Springer, 2019, pp. 349–361.

[34] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," in *Cooperative Robots and Sensor Networks* (Studies in Computational Intelligence), vol. 604. Cham, Switzerland: Springer, 2015, pp. 31–51.

[35] J. Song and S. Gupta, "CARE: Cooperative autonomy for resilience and efficiency of robot teams for complete coverage of unknown environments under robot failures," *Auto. Robots*, vol. 44, nos. 3–4, pp. 647–671, Mar. 2020.

[36] W. He, Z. Li, and C. L. P. Chen, "A survey of human-centered intelligent robots: Issues and challenges," *IEEE/CAA J. Autom. Sinica*, vol. 4, no. 4, pp. 602–609, Apr. 2017.

[37] Z. Zeng, L. Lian, K. Sammut, F. He, Y. Tang, and A. Lammas, "A survey on path planning for persistent autonomy of autonomous underwater vehicles," *Ocean Eng.*, vol. 110, pp. 303–313, Dec. 2015.

[38] J. Kim, D.-E. Lee, and J. Seo, "Task planning strategy and path similarity analysis for an autonomous excavator," *Autom. Construct.*, vol. 112, Apr. 2020, Art. no. 103108.

[39] A. Ghaddar, A. Merei, and E. Natalizio, "PPS: Energy-aware grid-based coverage path planning for UAVs using area partitioning in the presence of NFZs," *Sensors*, vol. 20, no. 13, p. 3742, Jul. 2020.

[40] S. Xing, R. Wang, and G. Huang, "Area decomposition algorithm for large region maritime search," *IEEE Access*, vol. 8, pp. 205788–205797, 2020.

[41] Y. Choi, Y. Choi, S. Briceno, and D. N. Mavris, "Energy-constrained multi-UAV coverage path planning for an aerial imagery mission using column generation," *J. Intell. Robotic Syst.*, vol. 97, no. 1, pp. 125–139, Jan. 2020.

[42] E. M. Arkin, S. P. Fekete, and J. S. B. Mitchell, "Approximation algorithms for lawn mowing and milling," *Comput. Geometry*, vol. 17, nos. 1–2, pp. 25–50, Oct. 2000.

[43] H. Azpúrua, G. M. Freitas, D. G. Macharet, and M. F. M. Campos, "Multi-robot coverage path planning using hexagonal segmentation for geophysical surveys," *Robotica*, vol. 36, no. 8, pp. 1144–1166, Aug. 2018.

[44] J. Xie, L. R. Garcia Carrillo, and L. Jin, "Path planning for UAV to cover multiple separated convex polygonal regions," *IEEE Access*, vol. 8, pp. 51770–51785, 2020.

[45] C. Dornhege, A. Kleiner, A. Hertle, and A. Kolling, "Multirobot coverage search in three dimensions," *J. Field Robot.*, vol. 33, no. 4, pp. 537–558, Jun. 2016.

[46] A. Jamshidpey, M. Wahby, M. K. Heinrich, M. Allwright, W. Zhu, and M. Dorigo, "Centralization vs. decentralization in multi-robot coverage: Ground robots under UAV supervision," 2024, *arXiv:2408.06553*.

[47] A. K. Lakshmanan, R. E. Mohan, B. Ramalingam, A. Vu Le, P. Veerajagadeshwar, K. Tiwari, and M. Ilyas, "Complete coverage path planning using reinforcement learning for tetromino based cleaning and maintenance robot," *Autom. Construct.*, vol. 112, Apr. 2020, Art. no. 103078.

[48] M. Mohammadpour, L. Zeghmi, S. Kelouwani, M.-A. Gaudreau, A. Amamou, and M. Graba, "An investigation into the energy-efficient motion of autonomous wheeled mobile robots," *Energies*, vol. 14, no. 12, p. 3517, Jun. 2021.

[49] T. Kong, H. Gao, and X. Chen, "Research on full-coverage path planning method of steel rolling shop cleaning robot," in *Proc. Int. Conf. Neural Comput. Adv. Appl.* Singapore: Springer, Jan. 2023, pp. 455–466.

[50] L. Yang, P. Li, S. Qian, H. Quan, J. Miao, M. Liu, Y. Hu, and E. Memetimin, "Path planning technique for mobile robots: A review," *Machines*, vol. 11, no. 10, p. 980, Oct. 2023.

[51] S. Kakolu and M. A. Faheem, "Autonomous robotics in field operations: A data-driven approach to optimize performance and safety," *Iconic Res. Eng. J.*, vol. 7, no. 4, pp. 565–578, 2023.

[52] L. Wijayathunga, A. Rassau, and D. Chai, "Challenges and solutions for autonomous ground robot scene understanding and navigation in unstructured outdoor environments: A review," *Appl. Sci.*, vol. 13, no. 17, p. 9877, Aug. 2023.

[53] M. Kapanoglu, M. Alikalfa, M. Ozkan, A. Yazıcı, and O. Parlaktuna, "A pattern-based genetic algorithm for multi-robot coverage path planning minimizing completion time," *J. Intell. Manuf.*, vol. 23, no. 4, pp. 1035–1045, Aug. 2012.

[54] S. Khanna and S. Srivastava, "Path planning and obstacle avoidance in dynamic environments for cleaning robots," *Quart. J. Emerg. Technol. Innov.*, vol. 8, no. 2, pp. 48–61, 2023.

[55] P. Arena, C. F. Blanco, A. Li Noce, S. Taffara, and L. Patanè, "Learning traversability map of different robotic platforms for unstructured terrains path planning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–8.

[56] P. Yee Leong and N. S. Ahmad, "Exploring autonomous load-carrying mobile robots in indoor settings: A comprehensive review," *IEEE Access*, vol. 12, pp. 131395–131417, 2024.

[57] F. Torchio, "Survey on automated systems for smart warehouses," Ph.D. dissertation, Dept. Mech. Eng., Politecnico di Torino, Turin, Italy, 2023.

[58] F. Lorson, A. Fügener, and A. Hübner, "New team mates in the warehouse: Human interactions with automated and robotized systems," *IISE Trans.*, vol. 55, no. 5, pp. 536–553, May 2023.

[59] F. Sibona, "Robots learn to behave: Improving human–robot collaboration in flexible manufacturing applications," Ph.D. dissertation, Electron. Commun. Eng., Graduate School Politecnico di Torino, 2023.

[60] Y. Zhang, W. Zhao, J. Wang, and Y. Yuan, "Recent progress, challenges and future prospects of applied deep reinforcement learning : A practical perspective in path planning," *Neurocomputing*, vol. 608, Dec. 2024, Art. no. 128423.

[61] I. Kubasakova, J. Kubanova, D. Benco, and D. Kadlecová, "Implementation of automated guided vehicles for the automation of selected processes and elimination of collisions between handling equipment and humans in the warehouse," *Sensors*, vol. 24, no. 3, p. 1029, Feb. 2024.

[62] M. Prédhumeau, A. Spalanzani, and J. Dugdale, "Pedestrian behavior in shared spaces with autonomous vehicles: An integrated framework and review," *IEEE Trans. Intell. Vehicles*, vol. 8, no. 1, pp. 438–457, Jan. 2023.

[63] H. H. James, R. Pawel, and G. Saduf, "Autonomous vehicles and robust decision-making in dynamic environments," *Fusion Multidisciplinary Res., An Int. J.*, vol. 1, no. 2, pp. 110–121, 2020.

[64] B. Bergqvist and H. Rm, "Designing for trust in domestic robots: Understanding how design affects users' trust in robotic vacuum cleaners," Tech. Rep., 2020.

[65] J. Fottner, D. Clauer, F. Hormes, M. Freitag, T. Beinke, L. Overmeyer, S. N. Gottwald, R. Elbert, T. Sarnow, T. Schmidt, and K. B. Reith, "Autonomous systems in intralogistics: State of the art and future research challenges," *Logistics Res.*, vol. 14, p. 2, 2021, doi: 10.23773/2021_2.

[66] S. Chakraborty, D. Elangovan, P. L. Govindarajan, M. F. ELnaggar, M. M. Alrashed, and S. Kamel, "A comprehensive review of path planning for agricultural ground robots," *Sustainability*, vol. 14, no. 15, p. 9156, Jul. 2022.

[67] J. P. Queralta, J. Taipalmaa, B. C. Pullinen, V. K. Sarker, T. N. Gia, H. Tenhunen, M. Gabbouj, J. Raitoharju, and T. Westerlund, "Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision," *IEEE Access*, vol. 8, pp. 191617–191643, 2020.

[68] Z. Shen, J. Song, K. Mittal, and S. Gupta, "CT-CPP: Coverage path planning for 3D terrain reconstruction using dynamic coverage trees," *IEEE Robot. Autom. Lett.*, vol. 7, no. 1, pp. 135–142, Jan. 2022.

[69] L. Annamalai, S. Vigneshwar, and G. Subashini, "Grid traversal path planning for robot foraging," in *Proc. 2nd Int. Conf. Adv. Comput., Commun. Control Netw. (ICACCCN)*, Dec. 2020, pp. 940–945.

[70] K. M. Hasan, Abdullah-Al-Nahid, and K. J. Reza, "Path planning algorithm development for autonomous vacuum cleaner robots," in *Proc. Int. Conf. Informat., Electron. Vis. (ICIEV)*, May 2014, pp. 1–6.

[71] Y. Liu, X. Lin, and S. Zhu, "Combined coverage path planning for autonomous cleaning robots in unstructured environments," in *Proc. 7th World Congr. Intell. Control Autom.*, 2008, pp. 8271–8276.

[72] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein, "Robotic exploration, Brownian motion and electrical resistance," in *Proc. 2nd Int. Workshop Randomization Approximation Techn. Comput. Sci.*, Barcelona, Spain. Berlin, Germany: Springer, Oct. 1998, pp. 116–130.

[73] N. Hazon and G. A. Kaminka, "Redundancy, efficiency and robustness in multi-robot coverage," in *Proc. IEEE Int. Conf. Robot. Autom.*, Oct. 2005, pp. 735–741.

[74] N. Agmon, N. Hazon, and G. A. Kaminka, "Constructing spanning trees for efficient multi-robot coverage," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Jun. 2006, pp. 1698–1703.

[75] K. S. Senthilkumar and K. K. Bharadwaj, "Spanning tree based terrain coverage by multi robots in unknown environments," in *Proc. Annu. IEEE India Conf.*, Dec. 2008, pp. 120–125.

[76] A. C. Kapoutsis, S. A. Chatzichristofis, and E. B. Kosmatopoulos, "DARP: Divide areas algorithm for optimal multi-robot coverage path planning," *J. Intell. Robotic Syst.*, vol. 86, nos. 3–4, pp. 663–680, Jun. 2017.

[77] X. Huang, M. Sun, H. Zhou, and S. Liu, "A multi-robot coverage path planning algorithm for the environment with multiple land cover types," *IEEE Access*, vol. 8, pp. 198101–198117, 2020.

[78] G.-Q. Gao and B. Xin, "A-STC: Auction-based spanning tree coverage algorithm formotion planning of cooperative robots," *Frontiers Inf. Technol. Electron. Eng.*, vol. 20, no. 1, pp. 18–31, Jan. 2019.

[79] T. D. Ranjitha and K. R. Guruprasad, "Pseudo spanning tree-based complete and competitive robot coverage using virtual nodes," *IFAC-PapersOnLine*, vol. 49, no. 1, pp. 195–200, 2016.

[80] H. V. Pham, P. Moore, and D. X. Truong, "Proposed smooth-STC algorithm for enhanced coverage path planning performance in mobile robot applications," *Robotics*, vol. 8, no. 2, p. 44, Jun. 2019.

[81] K. R. Guruprasad and T. D. Ranjitha, "CPC algorithm: Exact area coverage by a mobile robot using approximate cellular decomposition," *Robotica*, vol. 39, no. 7, pp. 1141–1162, Jul. 2021.
[82] P. M. M. Falaki, A. Padman, V. Nair, and K. Guruprasad, "Simultaneous exploration and coverage by a mobile robot," in *Control Instrumentation Systems*. Singapore: Springer, 2020, pp. 33–41.
[83] W. Dong, S. Liu, Y. Ding, X. Sheng, and X. Zhu, "An artificially weighted spanning tree coverage algorithm for decentralized flying robots," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 4, pp. 1689–1698, Oct. 2020.
[84] L. Li, D. Shi, S. Jin, S. Yang, C. Zhou, Y. Lian, and H. Liu, "Exact and heuristic multi-robot Dubins coverage path planning for known environments," *Sensors*, vol. 23, no. 5, p. 2560, Feb. 2023.
[85] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Jun. 1996.
[86] O. Salzman, "Sampling-based robot motion planning," *Commun. ACM*, vol. 62, no. 10, pp. 54–63, Sep. 2019.
[87] A. Dias, T. Fernandes, J. Almeida, A. Martins, and E. Silva, "3d path planning methods for unmanned aerial vehicles in search and rescue scenarios," in *Human-Centric Robotics*, M. F. Silva, G. S. Virk, M. O. Tokhi, B. Malheiro, and P. Guedes, Eds., Singapore: World Scientific, 2017, pp. 213–220.
[88] Á. Madridano, A. Al-Kaff, D. Martín, and A. A. D. L. de la Escalera, "3D trajectory planning method for UAVs swarm in building emergencies," *Sensors*, vol. 20, no. 3, p. 642, Jan. 2020.
[89] M. Ulrich, G. Lux, L. Jürgensen, and G. Reinhart, "Automated and cycle time optimized path planning for robot-based inspection systems," *Proc. CIRP*, vol. 44, pp. 377–382, Jan. 2016.
[90] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, B. Donald, K. Lynch, and D. Rus, Eds., Boca Raton, FL, USA: CRC Press, 2001, pp. 293–308.
[91] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.
[92] S. Zaheer and T. Gulrez, "Performance analysis of path planning techniques for autonomous mobile robots," in *Proc. IEEE Int. Conf. Electr., Comput. Commun. Technol. (ICECCT)*, Coimbatore, India, Mar. 2015, pp. 1–5.
[93] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. ICRA. Millennium Conf., IEEE Int. Conf. Robot. Autom. Symposia*, vol. 2, San Francisco, CA, USA, Apr. 2000, pp. 995–1001.
[94] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011.
[95] Y. Wu and J. Wang, *Algorithm Design Practice for Collegiate Programming Contests and Education*. Boca Raton, FL, USA: CRC Press, 2018.
[96] P. Joshi, *Artificial Intelligence With Python*. Birmingham, U.K.: Packt, 2017.
[97] A. Zdesar, S. Blazic, G. Klancar, and I. Skrjanc, *Wheeled Mobile Robotics: From Fundamentals Towards Autonomous Systems*. Oxford, U.K.: Butterworth-Heinemann, 2017.
[98] G. Zuo, P. Zhang, and J. Qiao, "Path planning algorithm based on sub-region for agricultural robot," in *Proc. 2nd Int. Asia Conf. Informat. Control, Autom. Robot. (CAR)*, Wuhan, China, Mar. 2010, pp. 197–200.
[99] X. Wang and D. Li, "Coverage path planning for UAVs in unknown directional regions," *Int. J. Wireless Mobile Comput.*, vol. 8, no. 3, p. 285, 2015.
[100] P. S. Pratama, J.-W. Kim, H.-K. Kim, S.-M. Yoon, T.-K. Yeu, S. Hong, S.-J. Oh, and S.-B. Kim, "Path planning algorithm to minimize an overlapped path and turning number for an underwater mining robot," in *Proc. 15th Int. Conf. Control, Autom. Syst. (ICCAS)*, Busan, South Korea, Oct. 2015, pp. 499–504.
[101] A. M. Kabir, K. N. Kaipa, J. Marvel, and S. K. Gupta, "Automated planning for robotic cleaning using multiple setups and oscillatory tool motions," *IEEE Trans. Autom. Sci. Eng. (from July 2004)*, vol. 14, no. 3, pp. 1364–1377, Jul. 2017.
[102] A. Barrientos, J. Colorado, J. D. Cerro, A. Martinez, C. Rossi, D. Sanz, and J. Valente, "Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots," *J. Field Robot.*, vol. 28, no. 5, pp. 667–689, Sep. 2011.
[103] M. Zhou and N. Gao, "Research on optimal path based on Dijkstra algorithms," in *Proc. 3rd Int. Conf. Mechatronics Eng. Inf. Technol. (ICMEIT)*, 2019, pp. 884–892.
[104] R. Almadhoun, T. Taha, L. Seneviratne, and Y. Zweiri, "Multi-robot hybrid coverage path planning for 3D reconstruction of large structures," *IEEE Access*, vol. 10, pp. 2037–2050, 2022.
[105] R. Yehoshua, N. Agmon, and G. A. Kaminka, "Robotic adversarial coverage of known environments," *Int. J. Robot. Res.*, vol. 35, no. 12, pp. 1419–1444, Oct. 2016.
[106] K. P. Cheng, R. E. Mohan, N. H. K. Nhan, and A. V. Le, "Graph theory-based approach to accomplish complete coverage path planning tasks for reconfigurable robots," *IEEE Access*, vol. 7, pp. 94642–94657, 2019.
[107] X. Liu and D. Gong, "A comparative study of A-star algorithms for search and rescue in perfect maze," in *Proc. Int. Conf. Electric Inf. Control Eng.*, Apr. 2011, pp. 24–27.
[108] A. M. Chaudhari, M. R. Apsangi, and A. B. Kudale, "Improved a-star algorithm with least turn for robotic rescue operations," in *Computational Intelligence, Communications, and Business Analytics*, vol. 776, J. Mandal, P. Dutta, and S. Mukhopadhyay, Eds., Cham, Switzerland: Springer, 2017, pp. 614–627.
[109] S. Dogru and L. Marques, "A-based solution to the coverage path planning problem," in *Proc. 3rd Iberian Robot. Conf. (Advances Intell. Syst. Computing)*, Nov. 2017, pp. 240–248.
[110] H. H. Viet, V.-H. Dang, S. Choi, and T. C. Chung, "BoB: An online coverage approach for multi-robot systems," *Int. J. Speech Technol.*, vol. 42, no. 2, pp. 157–173, Mar. 2015.
[111] Z. Cai, S. Li, Y. Gan, R. Zhang, and Q. Zhang, "Research on complete coverage path planning algorithms based on A algorithms," *Open Cybern. Syst. J.*, vol. 8, no. 1, pp. 418–426, 2014.
[112] A. Le, V. Prabakaran, V. Sivanantham, and R. Mohan, "Modified A-Star algorithm for efficient coverage path planning in tetris inspired self-reconfigurable robot with integrated laser sensor," *Sensors*, vol. 18, no. 8, p. 2585, Aug. 2018.
[113] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, San Diego, CA, USA, May 1994, pp. 3310–3317.
[114] M. Dakulović, S. Horvatić, and I. Petrović, "Complete coverage D algorithm for path planning of a floor-cleaning mobile robot," *IFAC Proc. Volumes*, vol. 44, no. 1, pp. 5950–5955, Jan. 2011.
[115] M. Faria, R. Marín, M. Popović, I. Maza, and A. Viguria, "Efficient lazy Theta path planning over a sparse grid to explore large 3D volumes with a multirotor UAV," *Sensors*, vol. 19, no. 1, p. 174, Jan. 2019.
[116] M. Faria, A. S. Ferreira, H. P. Leon, I. Maza, and A. Viguria, "Autonomous 3D exploration of large structures using a uav equipped with a 2D LiDAR," *Sensors*, vol. 19, no. 22, pp. 1–24, 2019.
[117] T. Raja, M. Shanmugapriya, and K. Manivannan, "Metaheuristic algorithms for robot path planning: A comprehensive exploration," in *Metaheuristic and Machine Learning Optimization Strategies for Complex Systems*. Pennsylvania, PA, USA: IGI Global, 2024, pp. 215–238.
[118] M. N. Ab Wahab, A. Nazir, A. Khalil, W. J. Ho, M. F. Akbar, M. H. M. Noor, and A. S. A. Mohamed, "Improved genetic algorithm for mobile robot path planning in static environments," *Expert Syst. Appl.*, vol. 249, Sep. 2024, Art. no. 123762.
[119] Z. Wang and Z. Bo, "Coverage path planning for mobile robot based on genetic algorithm," in *Proc. IEEE Workshop Electron., Comput. Appl.*, Ottawa, ON, Canada, May 2014, pp. 732–735.
[120] I. A. Hameed, D. D. Bochtis, and C. G. Sorensen, "Driving angle and track sequence optimization for operational path planning using genetic algorithms," *Appl. Eng. Agricult.*, vol. 27, no. 6, pp. 1077–1086, 2011.
[121] I. A. Hameed, D. Bochtis, and C. A. Sørensen, "An optimized field coverage planning approach for navigation of agricultural robots in fields involving obstacle areas," *Int. J. Adv. Robotic Syst.*, vol. 10, no. 5, pp. 1–9, May 2013.
[122] M. Shen, S. Wang, S. Wang, and Y. Su, "Simulation study on coverage path planning of autonomous tasks in hilly farmland based on energy consumption model," *Math. Problems Eng.*, vol. 2020, pp. 1–15, Aug. 2020.
[123] K. O. Ellefsen, H. A. Lepikson, and J. C. Albiez, "Multiobjective coverage path planning: Enabling automated inspection of complex, real-world structures," *Appl. Soft Comput.*, vol. 61, pp. 264–282, Dec. 2017.

[124] W.-C. Tung and J.-S. Liu, "Genetic algorithm with modified operators for an integrated traveling salesman and coverage path planning problem," in *Proc. 16th Int. Conf. Appl. Comput.*, Cagliari, Italy, Nov. 2019, pp. 205–216.

[125] L. Qiu, "Research on a hierarchical cooperative algorithm based on genetic algorithm and particle swarm optimization," in *Computational Intelligence and Intelligent Systems* (Communications in Computer and Information Science), vol. 874, K. Li, W. Li, Z. Chen, and Y. Liu, Eds., Singapore: Springer, 2018, pp. 16–25.

[126] M. G. Sadek, A. E. Mohamed, and A. M. El-Garhy, "Augmenting multi-objective genetic algorithm and dynamic programming for online coverage path planning," in *Proc. 13th Int. Conf. Comput. Eng. Syst. (ICCES)*, Cairo, Egypt, Dec. 2018, pp. 475–480.

[127] C.-T. Cheng, K. Fallahi, H. Leung, and C. K. Tse, "A genetic algorithm-inspired UUV path planner based on dynamic programming," *IEEE Trans. Syst., Man, Cybern., C Appl. Rev.*, vol. 42, no. 6, pp. 1128–1134, Nov. 2012.

[128] V. R. Batista and F. A. Zampirolli, "Optimising robotic pool-cleaning with a genetic algorithm," *J. Intell. Robotic Syst.*, vol. 95, no. 2, pp. 443–458, Aug. 2019.

[129] G. Beni and J. Wang, "Swarm intelligence in cellular robotics system," in *Robots and Biological Systems: Towards a New Bionics* (NATO ASI Series), vol. 102, P. Dario, G. Sandini, and P. Aebischer, Eds., Berlin, Germany: Springer, 1993, pp. 703–712.

[130] E. Cuevas, F. Fausto, and A. Gonzalez, *New Advancements in Swarm Algorithms: Operators and Applications* (Intelligent Systems Reference Library). Cham, Switzerland: Springer, 2020.

[131] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural To Artificial Systems*. London, U.K.: Oxford Univ. Press, 1999.

[132] C. Blum and X. Li, "Swarm intelligence in optimization," in *Swarm Intelligence* (Natural Computing Series). Berlin, Germany: Springer, 2008, pp. 43–85.

[133] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4, Perth, WA, Australia, Nov. 2002, pp. 1942–1948.

[134] M. Dorigo and T. Stzle, *Ant Colony Optimization*. Cambridge, MA, USA: MIT Press, 2004.

[135] D. Teodorovi, "Bee colony optimization (BCO)," in *Innovations in Swarm Intelligence* (Studies in Computational Intelligence), vol. 248. Berlin, Germany: Springer, 2009, pp. 39–60.

[136] T.-K. Lee, S.-H. Baek, Y.-H. Choi, and S.-Y. Oh, "Smooth coverage path planning and control of mobile robots based on high-resolution grid map representation," *Robot. Auto. Syst.*, vol. 59, no. 10, pp. 801–812, Oct. 2011.

[137] S. Sahu and B. B. Choudhury, "PSO based path planning of a six-axis industrial robot," in *Computational Intelligence in Data Mining* (Advances in Intelligent Systems and Computing), vol. 990, H. Behera, J. Nayak, B. Naik, and D. Pelusi, Eds., Singapore: Springer, 2020, pp. 213–220.

[138] Y.-H. Lin, S.-M. Wang, L.-C. Huang, and M.-C. Fang, "Applying the stereo-vision detection technique to the development of underwater inspection task with PSO-based dynamic routing algorithm for autonomous underwater vehicles," *Ocean Eng.*, vol. 139, pp. 127–139, Jul. 2017.

[139] Y.-H. Lin, L.-C. Huang, S.-Y. Chen, and C.-M. Yu, "The optimal route planning for inspection task of autonomous underwater vehicle composed of MOPSO-based dynamic routing algorithm in currents," *Appl. Ocean Res.*, vol. 75, pp. 178–192, Jun. 2018.

[140] S. Wang, Y. Bai, and C. Zhou, "Coverage path planning design of mapping UAVs based on particle swarm optimization algorithm," in *Proc. Chin. Control Conf. (CCC)*, Guangzhou, China, Jul. 2019, pp. 8236–8241.

[141] M. S. Couceiro, R. P. Rocha, and N. M. F. Ferreira, "A novel multi-robot exploration approach based on particle swarm optimization algorithms," in *Proc. IEEE Int. Symp. Saf., Secur., Rescue Robot.*, Kyoto, Japan, Nov. 2011, pp. 327–332.

[142] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.

[143] H. Duan, "Ant colony optimization: Principle, convergence and application," in *Handbook of Swarm Intelligence* (Adaptation, Learning, and Optimization), vol. 8, B. Panigrahi, Y. Shi, and M. Lim, Eds., Berlin, Germany: Springer, 2011, pp. 373–388.

[144] J. Ning, Q. Zhang, C. Zhang, and B. Zhang, "A best-path-updating information-guided ant colony optimization algorithm," *Inf. Sci.*, vols. 433–434, pp. 142–162, Apr. 2018.

[145] A. Borisenko and S. Gorlatch, "Optimizing a gpu-parallelized ant colony Metaheuristic by parameter tuning," in *Parallel Computing Technologies* (Lecture Notes in Computer Science), vol. 11657, V. Malyshkin, Ed., Cham, Switzerland: Springer, 2019, pp. 151–165.

[146] M. Starzec, G. Starzec, A. Byrski, and W. Turek, "Distributed ant colony optimization based on actor model," *Parallel Comput.*, vol. 90, Dec. 2019, Art. no. 102573.

[147] C. Pan, H. Wang, J. Li, and M. Korovkin, "Path planning of mobile robot based on an improved ant colony algorithm," in *Convergent Cognitive Information Technologies* (Communications in Computer and Information Science), vol. 1140, V. Sukhomlin and E. Zubarev, Eds., Cham, Switzerland: Springer, 2020, pp. 132–141.

[148] B. Li and T. Li, "Vehicle path optimization with time window based on improved ant colony algorithm," in *Data Processing Techniques and Applications for Cyber-Physical Systems* (Advances in Intelligent Systems and Computing), vol. 1088, C. Huang, Y. W. Chan, and N. Yen, Eds., Singapore: Springer, 2020, pp. 167–175.

[149] W. Zhang, X. Gong, G. Han, and Y. Zhao, "An improved ant colony algorithm for path planning in one scenic area with many spots," *IEEE Access*, vol. 5, pp. 13260–13269, 2017.

[150] A. V. Le, P. C. Ku, T. T. Tun, N. H. K. Nhan, Y. Shi, and R. E. Mohan, "Realization energy optimization of complete path planning in differential drive-based self-reconfigurable floor cleaning robot," *Energies*, vol. 12, no. 6, pp. 1–23, Mar. 2019.

[151] A. V. Le, N. H. K. Nhan, and R. E. Mohan, "Evolutionary algorithm based complete coverage path planning for tetriamond tiling robot," *Sensors*, vol. 20, no. 2, pp. 1–14, 2020.

[152] A. V. Le, R. Parween, R. E. Mohan, N. H. K. Nhan, and R. E. Abdulkader, "Optimization complete area coverage by reconfiguring htrihex tiling robot," *Sensors*, vol. 20, no. 11, pp. 1–20, 2020.

[153] G. Han, Z. Zhou, T. Zhang, H. Wang, L. Liu, Y. Peng, and M. Guizani, "Ant-Colony-Based complete-coverage path-planning algorithm for underwater gliders in ocean areas with thermoclines," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8959–8971, Aug. 2020.

[154] I. Caliskanelli, B. Broecker, and K. Tuyls, "Multi-robot coverage: A bee pheromone signaling approach," in *Artificial Life and Intelligent Agents* (Communications in Computer and Information Science), vol. 519, C. J. Headleand, W. J. Teahan, and L. A. Cenydd, Eds., Cham, Switzerland: Springer, 2015, pp. 124–140.

[155] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Global Optim.*, vol. 39, no. 3, pp. 459–471, Oct. 2007.

[156] B. Broecker, I. Caliskanelli, K. Tuyls, E. I. Sklar, and D. Hennes, "Hybrid insect-inspired multi-robot coverage in complex environments," in *Towards Autonomous Robotic Systems* (Lecture Notes in Computer Science), vol. 9287, C. Dixon and K. Tuyls, Eds., Cham, Switzerland: Springer, 2015, pp. 56–68.

[157] X. S. Yang, *Nature-Inspired Metaheuristic Algorithms*. Frome, U.K.: Luniver Press, 2008.

[158] F. De Rango, N. Palmieri, X. S. Yang, and S. Marano, "Bio-inspired exploring and recruiting tasks in a team of distributed robots over mined regions," in *Proc. Int. Symp. Perform. Eval. Comput. Telecommun. Syst. (SPECTS)*, Chicago, IL, USA, Jul. 2015, pp. 1–8.

[159] M. N. Ab Wahab, A. Nazir, A. Khalil, B. Bhatt, M. H. Mohd Noor, M. F. Akbar, and A. S. A. Mohamed, "Optimised path planning using enhanced firefly algorithm for a mobile robot," *PLoS ONE*, vol. 19, no. 8, Aug. 2024, Art. no. e0308264.

[160] N. Palmieri, F. de Rango, X. She Yang, and S. Marano, "Multi-robot cooperative tasks using combined nature-inspired techniques," in *Proc. 7th Int. Joint Conf. Comput. Intell.*, Lisbon, Portugal, 2015, pp. 74–82.

[161] N. Palmieri, X.-S. Yang, F. De Rango, and S. Marano, "Comparison of bio-inspired algorithms applied to the coordination of mobile robots considering the energy consumption," *Neural Comput. Appl.*, vol. 31, no. 1, pp. 263–286, Jan. 2019.

[162] F. Gul, I. Mir, D. Alarabiat, H. M. Alabool, L. Abualigah, and S. Mir, "Implementation of bio-inspired hybrid algorithm with mutation operator for robotic path planning," *J. Parallel Distrib. Comput.*, vol. 169, pp. 171–184, Nov. 2022.

[163] Y. Ou, P. Yin, and L. Mo, "An improved grey wolf optimizer and its application in robot path planning," *Biomimetics*, vol. 8, no. 1, p. 84, Feb. 2023.
[164] S. Liu, S. Liu, and H. Xiao, "Improved gray wolf optimization algorithm integrating A* algorithm for path planning of mobile charging robots," *Robotica*, vol. 42, no. 2, pp. 536–559, Feb. 2024.
[165] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Jan. 2014.
[166] K. Albina and S. G. Lee, "Hybrid stochastic exploration using grey wolf optimizer and coordinated multi-robot exploration algorithms," *IEEE Access*, vol. 7, pp. 14246–14255, 2019.
[167] N. AbuJabal, T. Rabie, M. Baziyad, I. Kamel, and K. Almazrouei, "Path planning techniques for real-time multi-robot systems: A systematic review," *Electronics*, vol. 13, no. 12, p. 2239, Jun. 2024.
[168] C. Zhou, B. Huang, and P. Fränti, "A review of motion planning algorithms for intelligent robots," *J. Intell. Manuf.*, vol. 33, no. 2, pp. 387–424, Feb. 2022.
[169] L. Huber, J.-J. Slotine, and A. Billard, "Fast obstacle avoidance based on real-time sensing," *IEEE Robot. Autom. Lett.*, vol. 8, no. 3, pp. 1375–1382, Mar. 2023.
[170] T. Fuke, M. Endo, K. Honda, and G. Ishigami, "Towards local minima-free robotic navigation: Model predictive path integral control via repulsive potential augmentation," in *Proc. IEEE/SICE Int. Symp. Syst. Integr. (SII)*, Jan. 2025, pp. 1112–1117.
[171] M. Saad and Y. Alazzawi, "Improved mobile robot manoeuvring using Bayes filter algorithm within the planned path," *Int. J. Multiphys.*, vol. 18, no. 3, pp. 1–12, 2024.
[172] H. Qin, S. Shao, T. Wang, X. Yu, Y. Jiang, and Z. Cao, "Review of autonomous path planning algorithms for mobile robots," *Drones*, vol. 7, no. 3, p. 211, Mar. 2023.
[173] M. Hamidaoui, M. Z. Talhaoui, M. Li, M. A. Midoun, S. Haouassi, D. E. Mekkaoui, A. Smaili, A. Cherraf, and F. Z. Benyoub, "Survey of autonomous vehicles' collision avoidance algorithms," *Sensors*, vol. 25, no. 2, p. 395, Jan. 2025.
[174] S. V. Albrecht, F. Christianos, and L. Schäfer, *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press, 2024.
[175] A. Billard, S. Mirrazavi, and N. Figueroa, *Learning for Adaptive and Reactive Robot Control: A Dynamical Systems Approach*. Cambridge, MA, USA: MIT Press, 2022.
[176] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., Cambridge, MA, USA: MIT Press, 2018.
[177] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, Sep. 2013.
[178] E. Barrett and S. Linder, "Autonomous hvac control, a reinforcement learning approach," in *Machine Learning and Knowledge Discovery in Databases* (Lecture Notes in Computer Science), vol. 9286, A. Bifet, M. May, B. Zadrozny, R. Gavalda, D. Pedreschi, F. Bonchi, J. Cardoso, and M. Spiliopoulou, Eds., Cham, Switzerland: Springer, 2015, pp. 3–19.
[179] R. Shakeri, M. A. Al-Garadi, A. Badawy, A. Mohamed, T. Khattab, A. K. Al-Ali, K. A. Harras, and M. Guizani, "Design challenges of multi-UAV systems in cyber-physical applications: A comprehensive survey and future directions," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3340–3385, 4th Quart., 2019.
[180] A. V. Le, R. Parween, P. T. Kyaw, R. E. Mohan, T. H. Q. Minh, and C. S. C. S. Borusu, "Reinforcement learning-based energy-aware area coverage for reconfigurable hRombo tiling robot," *IEEE Access*, vol. 8, pp. 209750–209761, 2020.
[181] X. Xia, T. Roppel, J. Y. Hung, J. Zhang, S. C. Periaswamy, and J. Patton, "Balanced map coverage using reinforcement learning in repeated obstacle environments," in *Proc. IEEE 29th Int. Symp. Ind. Electron. (ISIE)*, Delft, The Netherlands, Jun. 2020, pp. 41–48.
[182] L. Piardi, J. Lima, A. I. Pereira, and P. Costa, "Coverage path planning optimization based on Q-learning algorithm," in *Proc. Central Eur. Symp. Thermophys. (CEST)*, 2019, pp. 1–4.
[183] J. Xiao, G. Wang, Y. Zhang, and L. Cheng, "A distributed multi-agent dynamic area coverage algorithm based on reinforcement learning," *IEEE Access*, vol. 8, pp. 33511–33521, 2020.
[184] L. Tai and M. Liu, "Mobile robots exploration through CNN-based reinforcement learning," *Robot. Biomimetics*, vol. 3, no. 1, pp. 1–8, Dec. 2016.
[185] J. Xin, H. Zhao, D. Liu, and M. Li, "Application of deep reinforcement learning in mobile robot path planning," in *Proc. Chin. Autom. Congr. (CAC)*, Jinan, China, Oct. 2017, pp. 7112–7116.
[186] S. Y. Luis, D. G. Reina, and S. L. T. Marín, "A deep reinforcement learning approach for the patrolling problem of water resources through autonomous surface vehicles: The Ypacarai lake case," *IEEE Access*, vol. 8, pp. 204076–204093, 2020.
[187] C. Piciarelli and G. L. Foresti, "Drone patrolling with reinforcement learning," in *Proc. 13th Int. Conf. Distrib. Smart Cameras*, Trento, Italy, Sep. 2019, pp. 1–6.
[188] X. Cao, C. Sun, and M. Yan, "Target search control of AUV in underwater environment with deep reinforcement learning," *IEEE Access*, vol. 7, pp. 96549–96559, 2019.
[189] P. T. Kyaw, A. Paing, T. T. Thu, R. E. Mohan, A. Vu Le, and P. Veerajagadheswar, "Coverage path planning for decomposition reconfigurable grid-maps using deep reinforcement learning based travelling salesman problem," *IEEE Access*, vol. 8, pp. 225945–225956, 2020.
[190] A. Koval, S. S. Mansouri, and G. Nikolakopoulos, "Online multi-agent based cooperative exploration and coverage in complex environment," in *Proc. 18th Eur. Control Conf. (ECC)*, Naples, Italy, Jun. 2019, pp. 3964–3969.
[191] F. Balampanis, I. Maza, and A. Ollero, "Spiral-like coverage path planning for multiple heterogeneous UAS operating in coastal regions," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Miami, FL, USA, Jun. 2017, pp. 617–624.
[192] L. Jiao, Z. Peng, L. Xi, S. Ding, and J. Cui, "Multi-agent coverage path planning via proximity interaction and cooperation," *IEEE Sensors J.*, vol. 22, no. 6, pp. 6196–6207, Mar. 2022.
[193] D. E. Soltero, M. Schwager, and D. Rus, "Decentralized path planning for coverage tasks using gradient descent adaptive control," *Int. J. Robot. Res.*, vol. 33, no. 3, pp. 401–425, Mar. 2014.
[194] K. Champagnie, F. Arvin, and J. Hu, "Decentralized multi-agent coverage path planning with greedy entropy maximization," in *Proc. IEEE Int. Conf. Ind. Technol. (ICIT)*, Mar. 2024, pp. 1–6.
[195] P. Cao, L. Lei, S. Cai, G. Shen, X. Liu, X. Wang, L. Zhang, L. Zhou, and M. Guizani, "Computational intelligence algorithms for UAV swarm networking and collaboration: A comprehensive survey and future directions," *IEEE Commun. Surveys Tuts.*, vol. 26, no. 4, pp. 2684–2728, 4th Quart., 2024.
[196] M. Ramesh, F. Imeson, B. Fidan, and S. L. Smith, "Anytime replanning of robot coverage paths for partially unknown environments," *IEEE Trans. Robot.*, vol. 40, pp. 4190–4206, 2024.
[197] H. S. Hewawasam, M. Y. Ibrahim, and G. K. Appuhamillage, "Past, present and future of path-planning algorithms for mobile robot navigation in dynamic environments," *IEEE Open J. Ind. Electron. Soc.*, vol. 3, pp. 353–365, 2022.
[198] Y. Lian, W. Xie, and L. Zhang, "A probabilistic time-constrained based heuristic path planning algorithm in warehouse multi-AGV systems," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 2538–2543, 2020.
[199] J. Tang, Y. Gao, and T. L. Lam, "Learning to coordinate for a worker-station multi-robot system in planar coverage tasks," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 12315–12322, Oct. 2022.
[200] V. G. Nair, R. S. Adarsh, K. P. Jayalakshmi, M. V. Dileep, and K. R. Guruprasad, "Cooperative online workspace allocation in the presence of obstacles for multi-robot simultaneous exploration and coverage path planning problem," *Int. J. Control, Autom. Syst.*, vol. 21, no. 7, pp. 2338–2349, Jul. 2023.
[201] L. Antonyshyn, J. Silveira, S. Givigi, and J. Marshall, "Multiple mobile robot task and motion planning: A survey," *ACM Comput. Surveys*, vol. 55, no. 10, pp. 1–35, Oct. 2023.
[202] R. Mitra and I. Saha, "Online concurrent multi-robot coverage path planning," 2024, *arXiv:2403.10460*.
[203] P. Fong, C. Fang, and J. He, "Optimal attack against coverage path planning in multi-robot system," in *Proc. 5th Chin. Conf. Swarm Intell. Cooperat. Control*. Singapore: Springer, Jul. 2022, pp. 1760–1772.
[204] R. J. Alitappeh and K. Jeddisaravi, "Multi-robot exploration in task allocation problem," *Int. J. Speech Technol.*, vol. 52, no. 2, pp. 2189–2211, Jan. 2022.
[205] J. Gong, H. Kim, and S. Lee, "Resilient multi-robot coverage path redistribution using boustrophedon decomposition for environmental monitoring," *Sensors*, vol. 24, no. 23, p. 7482, Nov. 2024.
[206] Y. Wu, X. Ren, H. Zhou, Y. Wang, and X. Yi, "A survey on multi-robot coordination in electromagnetic adversarial environment: Challenges and techniques," *IEEE Access*, vol. 8, pp. 53484–53497, 2020.

[207] P. Mahato, S. Saha, C. Sarkar, and M. Shaghil, "Consensus-based fast and energy-efficient multi-robot task allocation," *Robot. Auto. Syst.*, vol. 159, Jan. 2023, Art. no. 104270.
[208] J. Tang, C. Sun, and X. Zhang, "MSTC: Multi-robot coverage path planning under physical constrain," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 2518–2524.
[209] K. Shibata, T. Jimbo, and T. Matsubara, "Deep reinforcement learning of event-triggered communication and consensus-based control for distributed cooperative transport," *Robot. Auto. Syst.*, vol. 159, Jan. 2023, Art. no. 104307.
[210] A. Gautam, V. S. Shekhawat, and S. Mohan, "A graph partitioning approach for fast exploration with multi-robot coordination," in *Proc. IEEE Int. Conf. Syst., Man Cybern. (SMC)*, Oct. 2019, pp. 459–465.
[211] A. Soni, C. Dasannacharya, A. Gautam, V. S. Shekhawat, and S. Mohan, "Multi-robot unknown area exploration using frontier trees," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2022, pp. 9934–9941.
[212] A. Soni, C. Dasannacharya, A. Gautam, V. S. Shekhawat, and S. Mohan, "D-MRFTE: A decentralized relay-based approach for multi-robot unknown area exploration," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2023, pp. 1–7.
[213] J. Zhang, J. Shen, and C. Liu, "Grouping and angle partitioning-based coverage path planning for swarm robots using only local communication," in *Proc. Int. Conf. Swarm Intell.* Singapore: Springer, Aug. 2024, pp. 36–45.
[214] A. Phadke and F. A. Medrano, "Towards resilient UAV swarms—A breakdown of resiliency requirements in UAV swarms," *Drones*, vol. 6, no. 11, p. 340, Nov. 2022.
[215] B. Ai, M. Jia, H. Xu, J. Xu, Z. Wen, B. Li, and D. Zhang, "Coverage path planning for maritime search and rescue using reinforcement learning," *Ocean Eng.*, vol. 241, Dec. 2021, Art. no. 110098.
[216] K. Lochan, A. Khan, I. Elsayed, B. Suthar, L. Seneviratne, and I. Hussain, "Advancements in precision spraying of agricultural robots: A comprehensive review," *IEEE Access*, vol. 12, pp. 129447–129483, 2024.
[217] A. Saddik, R. Latif, M. Elhoseny, and A. El Ouardi, "Real-time evaluation of different indexes in precision agriculture using a heterogeneous embedded system," *Sustain. Comput., Informat. Syst.*, vol. 30, Jun. 2021, Art. no. 100506.
[218] J.-P. Kaiser, J. Gäbele, D. Koch, J. Schmid, F. Stamer, and G. Lanza, "Adaptive acquisition planning for visual inspection in remanufacturing using reinforcement learning," *J. Intell. Manuf.*, vol. 2024, pp. 1–27, Aug. 2024.
[219] S. Halder and K. Afsari, "Robots in inspection and monitoring of buildings and infrastructure: A systematic review," *Appl. Sci.*, vol. 13, no. 4, p. 2304, Feb. 2023.
[220] A. Prorok, M. Malencia, L. Carlone, G. S. Sukhatme, B. M. Sadler, and V. Kumar, "Beyond robustness: A taxonomy of approaches towards resilient multi-robot systems," 2021, *arXiv:2109.12343*.
[221] P. C. Cheng, "Safe navigation and human–robot interaction in assistant robotic applications," Tech. Rep., 2023.
[222] N. Van Thanh and T. Linh, "Real-time trajectory planning for autonomous vehicles in dynamic traffic environments: A survey of modern algorithms and predictive techniques," *J. Intell. Connectivity Emerg. Technol.*, vol. 7, no. 12, pp. 1–25, 2022.
[223] H. Abas and N. Nasir, "Drone patrolling applications, challenges, and its future: A review," Tech. Rep., 2023.
[224] S. Rivera and R. State, "Securing robots: An integrated approach for security challenges and monitoring for the robotic operating system (ROS)," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, Bordeaux, France, 2021, pp. 754–759.
[225] K. Keung, "Designing computational intelligence and data-driven cyber-physical approach for robotic mobile fulfillment system," Tech. Rep., 2023.
[226] S. Kachavarapu, T. Doernbach, and R. Gerndt, "Fast-Revisit coverage path planning for autonomous mobile patrol robots using long-range sensor information," 2025, *arXiv:2501.07343*.
[227] S. Zhang and J. Zhang, "Complete-coverage path planning algorithm of multiple mobile robots based on reliability functions," in *Proc. 3rd Int. Conf. Service Robot. Technol.*, Mar. 2022, pp. 15–21.
[228] N. AbuJabal, M. Baziyad, R. Fareh, B. Brahmi, T. Rabie, and M. Bettayeb, "A comprehensive study of recent path-planning techniques in dynamic environments for autonomous robots," *Sensors*, vol. 24, no. 24, p. 8089, Dec. 2024.
[229] È. Pairet, J. D. Hernández, M. Carreras, Y. Petillot, and M. Lahijanian, "Online mapping and motion planning under uncertainty for safe navigation in unknown environments," *IEEE Trans. Autom. Sci. Eng.*, pp. 1–23, 2021, doi: 10.1109/TASE.2021.3118737.
[230] D. Riek, "Hospitals of the future: Designing interactive robotic systems for resilient emergency departments," Tech. Rep., 2022.
[231] I. N. Weerarathna, D. Raymond, and A. Luharia, "Human–robot collaboration for healthcare: A narrative review," *Cureus*, vol. 15, no. 11, pp. 1–11, Nov. 2023.
[232] S. Li, P. Zheng, S. Liu, Z. Wang, X. V. Wang, L. Zheng, and L. Wang, "Proactive human–robot collaboration: Mutual-cognitive, predictable, and self-organising perspectives," *Robot. Comput.-Integr. Manuf.*, vol. 81, Jun. 2023, Art. no. 102510.
[233] L. Battistuzzi, C. T. Recchiuto, and A. Sgorbissa, "Ethical concerns in rescue robotics: A scoping review," *Ethics Inf. Technol.*, vol. 23, no. 4, pp. 863–875, Dec. 2021.
[234] A. Zacharaki, I. Kostavelis, A. Gasteratos, and I. Dokas, "Safety bounds in human robot interaction: A survey," *Saf. Sci.*, vol. 127, Jul. 2020, Art. no. 104667.
[235] M. Chong, "Coverage path planning in dynamic environment through guided reinforcement learning," Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, Tech. Rep. UCB/EECS-2022-123, 2022. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-123.html
[236] X. Ren, C. Qiu, X. Wang, Z. Han, K. Xu, H. Yao, and S. Guo, "AI-bazaar: A cloud-edge computing power trading framework for ubiquitous AI services," *IEEE Trans. Cloud Comput.*, vol. 11, no. 3, pp. 2337–2348, Mar. 2022.
[237] J. P. Carvalho and A. P. Aguiar, "A reinforcement learning based online coverage path planning algorithm," in *Proc. IEEE Int. Conf. Auto. Robot Syst. Competitions (ICARSC)*, Apr. 2023, pp. 81–86.

K. P. JAYALAKSHMI received the M.Tech. degree in digital electronics and communication from Visvesvaraya Technological University. She is currently pursuing the Ph.D. degree with the Department of Aeronautical and Automobile Engineering, Manipal Institute of Technology, Manipal Academy of Higher Education. She is also an Assistant Professor with the Department of Electronics and Communication Engineering, St. Joseph Engineering College, Mangaluru. Her research interests include robotics and artificial intelligence, signal processing, and control engineering.

VISHNU G. NAIR received the Ph.D. degree from the National Institute of Technology, Karnataka, India. He is with the Department of Aeronautical and Automobile Engineering, Manipal Academy of Higher Education. His research interests include control, motion planning, and multi robotic systems.

DAYAKSHINI SATHISH received the Ph.D. degree from Manipal Institute of Technology, Manipal. She is currently a Professor and the Head of the Department of Electronics and Communication Engineering, St. Joseph Engineering College, Mangaluru. Her research interests include biomedical signal processing, digital signal processing, and artificial intelligence.