



Article

Safe Coverage Control of Multi-Agent Systems and Its Verification in ROS/Gazebo Environment

Fidelia Chaitra Siri ^{1,†}, Jie Song ^{2,†}  and Mikhail Svinin ^{1,*,†} 

¹ Department of Information Science and Engineering, Ritsumeikan University, Kyoto 567-8570, Japan; gr0558hh@ed.ritsumei.ac.jp

² Department of Information Science and Technology, Osaka University, Osaka 565-0871, Japan; j-song@ist.osaka-u.ac.jp

* Correspondence: svinin@fc.ritsumei.ac.jp

† These authors contributed equally to this work.

Abstract: This paper presents safe coverage control algorithms for multi-agent systems, integrating Centroidal Voronoi Tessellation (CVT) and control barrier functions (CBFs). This study aims to ensure safety and spatial optimization by combining CVT and CBFs for obstacle avoidance, testing the controller through simulations, and verifying the results with RT mobile robots. This development of safe coverage control algorithms for multi-agent systems achieves a synergy that addresses both safety and spatial optimization, which are crucial for multi-agent systems. The proposed CVT-CBF-based controller has been validated through extensive simulations in the ROS/Gazebo environment and physical experiments with RT robots, demonstrating its effectiveness in achieving collision-free coverage. This study provides a comprehensive understanding of the integration of CVT and CBFs for safe coverage control with obstacle avoidance in multi-agent systems, highlighting both its potential and the necessary considerations for practical deployment.

Keywords: multi-agent systems; coverage control; collision avoidance; Centroidal Voronoi Tessellation; control barrier function



Citation: Siri, F.C.; Song, J.; Svinin, M. Safe Coverage Control of Multi-Agent Systems and Its Verification in ROS/Gazebo Environment. *Information* **2024**, *15*, 462. <https://doi.org/10.3390/info15080462>

Academic Editors: Nikolaos D. Kouvakas and Zhigang Chu

Received: 30 June 2024

Revised: 26 July 2024

Accepted: 31 July 2024

Published: 2 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Modern research on multi-agent systems covers various aspects, including task allocation, communication protocols, coordination strategies, obstacle avoidance, etc. Each of these aspects plays a crucial role in the overall functionality and performance of the multi-agent systems. In this paper, we cover obstacle avoidance and coverage control for multi-agent systems. These aspects are important for maintaining system efficiency in complex environments.

In recent years, there has been a growing focus on research and studies related to multi-agent coverage control. This evolution has come from the idea of a single robot to the collaboration of many agents in the environment [1]. The main purpose of multi-agent coverage control is to direct the agents to cover a certain area in space, whether it is 2D or 3D. This problem is especially important in applications such as security, surveillance, and search and rescue in disaster areas [2]. An important application of multi-agent management is the monitoring and protection of certain areas through the use of mobile agents equipped with sensors. Additionally, multi-agent systems can also benefit from the integration of tactile sensors such as those in [3,4], which can enhance their sensory capabilities and improve their interaction with complex environments. The benefits of employing multi-agent coverage control methods include increased robustness, scalability, efficiency, and redundancy.

Recent advancements in computational neural network algorithms have further enriched the field of control and navigation in multi-agent systems. Notable methods include observer-based adaptive fuzzy finite-time attitude control for quadrotor UAVs [5,6] and

anti-saturation fixed-time attitude tracking control for uncertain quadrotor UAVs with external disturbances [7]. These approaches offer promising solutions for managing computational burdens and enhancing the system's performance in real-time applications. For instance, observer-based adaptive fuzzy finite-time control leverages neural network techniques to achieve precise and efficient control, while anti-saturation fixed-time control addresses the challenges of computational load and external disturbances, ensuring a robust and reliable performance in practical systems.

Managing the coverage of multiple agents is a complex task, especially in dynamic environments with varying density functions [8–13]. Agents in such environments need to adjust their positioning to ensure effective coverage, taking into account changing factors such as moving obstacles, evolving terrain conditions, and fluctuations in environmental variables like temperature, humidity, and pollution levels. Additionally, the density function reflects the spatial distribution or concentration of a specific attribute or entity within the environment. For instance, in surveillance scenarios, the density function might indicate the probability of activity or events happening in different areas.

Controllers available in the literature [8–10] can adapt to changing environments with varying density functions and in [11–13] address systems with uncertainties. These algorithms are based on the concept of CVT, where agents are represented as points and confined within cells to prevent collisions. However, collisions may still occur when these algorithms are applied to real agents, even if the agents are designed to have zero radius, which is not realistic. To solve this problem, [14–16] suggests several approaches based on the buffered Voronoi cell. The buffered Voronoi cell incorporates a safety radius to adjust the edge of each Voronoi cell, ensuring that collisions are avoided as long as the agents remain within their cells. However, this modification also involves path planning and tracking at each step. The effectiveness of this approach in avoiding collisions in the presence of system uncertainties has not been tested. These uncertainties can potentially cause agents to deviate from the BVC and result in collisions. Successful obstacle avoidance can be achieved when an agent gathers information about its surroundings, combines this information to create a symbolic model of the environment, and then uses this model to make informed decisions. Based on this model, the agent can autonomously plan a path to avoid obstacles [17].

The field of obstacle avoidance algorithms has been extensively researched and developed due to its importance in various applications. However, many algorithms have significant limitations when obstacles are unknown. Traditional obstacle avoidance algorithm methods include the bug algorithm, potential field algorithm, and vector field algorithm [18–20], while intelligent obstacle avoidance algorithm methods include the genetic algorithm, fuzzy logic algorithm, and neural network algorithm [21–24].

The bug algorithm is the simplest obstacle avoidance algorithm, involving finding an obstacle and walking around its outline to bypass it [19]. The most notable bug algorithms include Bug1, Bug2, and Tangent Bug. Bug1 operates by having the robot move directly towards its goal until it encounters an obstacle, at which point it follows the obstacle's boundary until it finds the closest point to the goal before continuing. Bug2 simplifies this by having the robot leave the obstacle boundary at the first intersection point with the line connecting the start and the goal, resuming its direct path towards the goal. Tangent Bug enhances the process by utilizing range sensors to detect obstacles and build a local map, allowing the robot to follow tangent lines along obstacles and dynamically re-plan its path based on real-time sensor data. These algorithms are characterized by their simplicity, sensor-based operation, and completeness, guaranteeing path finding if a path exists under ideal conditions. These algorithms are easy to implement but do not consider robot dynamics and are not reliable in complex environments.

The potential field algorithm is a widely used approach in robotic motion planning for obstacle avoidance, representing the environment as a potential field where the goal exerts an attractive force and obstacles exert repulsive forces. The robot navigates by following the resultant force vector, which is the sum of these attractive and repulsive forces, guiding it

towards the goal while avoiding obstacles. The attractive potential pulls the robot towards the goal, increasing with distance, while the repulsive potential pushes the robot away from obstacles, becoming stronger as the robot approaches them. Mathematically, the attractive force is proportional to the distance to the goal and the repulsive force is inversely proportional to the distance to the obstacle within a certain range. The potential field algorithm is simple to implement and efficient for real-time applications, providing smooth and continuous paths for the robot. However, it can suffer from local minima, where the robot gets stuck in non-goal positions with zero resultant force and may exhibit oscillations around obstacles [20].

The vector field histogram is an advanced real-time obstacle avoidance algorithm used in mobile robotics, designed to navigate around obstacles while moving toward a target. It works by creating a polar histogram that represents the robot's immediate surroundings, where each bin in the histogram corresponds to a specific direction and contains the density of obstacles in that direction. The algorithm involves three main steps: first, it constructs a histogram grid from sensor readings to represent the environment's occupancy; second, it reduces this grid to a one-dimensional polar histogram centered on the robot, reflecting the density of obstacles at various angles; third, it identifies candidate directions with low obstacle densities and selects the most suitable direction based on the goal direction and the robot's dynamic constraints. The vector field histogram algorithm is robust and allows the robot to make quick, reactive decisions to avoid obstacles, balancing the need to reach the goal efficiently with the need to navigate safely. Its ability to dynamically adjust the robot's path in real-time makes it particularly suitable for complex and cluttered environments. However, it requires careful tuning of parameters and may struggle with very tight spaces or highly dynamic environments [21].

The genetic algorithm is an optimization technique inspired by natural selection, used for solving complex problems like obstacle avoidance in robotic motion planning. Over successive generations, the algorithm converges towards an optimal or near-optimal path that effectively avoids obstacles while efficiently guiding the robot to its destination. Genetic algorithms are advantageous for their ability to search large and complex solution spaces without requiring gradient information. However, they can be computationally intensive and may require the careful tuning of parameters such as population size, mutation rate, and crossover rate to balance exploration and exploitation. The genetic algorithm is slow in terms of speed [22].

A fuzzy algorithm for obstacle avoidance in robotics leverages fuzzy logic to handle uncertainties and imprecise information about the environment, providing a flexible and robust approach to navigation. In this method, the robot's sensors collect data about its surroundings, such as the distance to obstacles and the direction of the goal. This sensory information is then fuzzified into linguistic variables like "close", "medium", and "far" for distances, or "left", "straight", and "right" for directions. These variables are processed using a set of fuzzy rules designed to mimic human reasoning, such as "if an obstacle is close and to the right, then turn left". The advantage of a fuzzy algorithm is its ability to smoothly handle the inherent uncertainties and ambiguities in sensor data, resulting in more adaptive and resilient obstacle avoidance behavior compared to rigid, rule-based systems. However, the fuzzy logic algorithm involves a fuzzy controller and requires a large amount of calculation as the number of obstacles increases [23].

A neural network algorithm for obstacle avoidance leverages the learning capabilities of neural networks to enable robots to navigate complex environments. This approach involves training a neural network to recognize and react to obstacles using data collected from sensors such as cameras, lidars, or ultrasonic sensors. During the training phase, the neural network learns to map sensor inputs to appropriate movement commands by being exposed to numerous scenarios involving various obstacle configurations. The network's architecture typically includes multiple layers of neurons that process the sensor data, extract relevant features, and make decisions about the robot's actions. Once trained, the neural network can generalize from the examples it has seen to handle new, unseen obsta-

cles effectively. This method is particularly advantageous in dynamic and unstructured environments where traditional algorithms might struggle. The neural network can adapt to different types of obstacles and complex terrains, providing robust and flexible obstacle avoidance. However, this approach requires a substantial amount of training data and computational resources. Additionally, the performance of the neural network heavily depends on the quality and diversity of the training data, and it may not guarantee safety in all scenarios, particularly if it encounters situations significantly different from its training set [24].

Considering these limitations, carefully selecting a better obstacle-avoidance algorithm is essential. The increasing use of multi-agent systems for surveillance and security purposes has raised concerns about safety and efficiency. Traditional approaches often focus on individual agents, neglecting interactions that can lead to collisions or ineffective coverage. The controversy lies in balancing agents to ensure safe and optimal coverage. This study addresses the concern of proposing a collision avoidance algorithm that guarantees safety while in a complex static environment. The CBF-based controller offers guaranteed safety, scalability, and adaptability to complex real-world environments. Implementing a CBF-based controller for obstacle avoidance has been identified as a feasible solution to address the limitations of traditional algorithms in challenging environments. By leveraging the principles of CBF, the controller has demonstrated the potential to improve the reliability and performance of autonomous systems [25]. Its ability to generate robust and safe trajectories while guaranteeing collision avoidance has made it a promising option for developing advanced navigation systems.

The primary objective of this paper is to explore the combination of CVT and CBFs with obstacle navigation, evaluate the controller through simulations, and validate the outcomes with mobile agents. Integrating CBF-based obstacle avoidance would enable agents to navigate while avoiding collisions with obstacles. Meanwhile, implementing CVT for coverage control would allow agents to efficiently cover the area by dividing it into regions for optimal coverage. This integration of CVT and CBF techniques would improve the agents' ability to navigate challenging terrain while effectively covering the area. The combination of CBFs with CVT presents a robust approach to ensuring safety and coordination in multi-agent systems. However, integrating these complex methodologies poses challenges, such as algorithm and computational complexity, non-convexity, real-time implementation, and parameter tuning. Yet, this integration offers numerous benefits, including safety assurance, collision avoidance, formation control, uncertainty robustness, global optimality, real-time adaptation, flexible applications, scalability, and compliance with safety standards.

This paper's first contribution involves developing a safe coverage controller using CBFs with CVT to simultaneously address safety and spatial optimization in multi-agent systems. The combined approach aims to address the challenges of coordinating multiple agents and facilitating formation control that upholds safety constraints. Another contribution is developing and testing the controller in ROS/Gazebo simulators for mobile robots with obstacles. The ROS/Gazebo simulator provides a realistic and complex simulation environment, supporting sensor integration, ROS integration, dynamic environments, and multi-robot simulation, making it an industry-standard tool. Additionally, this paper involves setting up and testing the CVT-CBF-based controller through experiments using RT mobile robots to examine its effectiveness in various scenarios involving obstacles. In summary, this paper presents a promising solution for ensuring safe and efficient mobile robot navigation.

The rest of the paper is organized as follows. In Section 2, preliminaries about CBF are introduced. In Section 3, the coverage control problem is stated for multi-agent systems and a corresponding control algorithm is given based on the CBF technique. The simulation tools and experimental setup are explained in Section 4. The structure of the proposed framework with the flowchart is explained in Section 5. A comparison case study is given

in Section 6. The proposed algorithm is verified under simulation in Section 7 and under experiments with RT robots in Section 8. Finally, conclusions are drawn in Section 9.

2. Preliminaries

In this section, the concept of the CBF, which is the main tool for our controller design, is reviewed.

The CBF is a level-wise function used to generate formal security guarantees for a nonlinear control affine system. In the artificial optimization of linear control affine systems [24], the barrier function is usually a continuous function. This study focuses on barrier certificates. The barrier certificates play an important role in security and use both Lyapunov functions and barrier functions for safety and stability [26] constraints. Combining CBFs and CVT allows agents to overcome obstacles. Consider the following control affine system:

$$\dot{x} = f(x) + g(x)u, \quad (1)$$

where $x \in \mathbb{R}^n$ is the state, $u \in U \subset \mathbb{R}^m$ and $x \in \mathcal{D} \subset \mathbb{R}^n$ is the set of admissible inputs, and $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$ and $g: \mathbb{R}^n \rightarrow \mathbb{R}^{n \times m}$ are local Lipschitz continuous functions. A set \mathcal{S} is called a forward controlled invariant with respect to system (1) if, for every $x_0 \in \mathcal{S}$, there exists a control signal $u(t)$ such that $x(t; t_0, x_0) \in \mathcal{S}$ for all $t \geq t_0$, where $x(t; t_0, x_0)$ denotes the solution of (1) at time t with initial condition $x_0 \in \mathbb{R}^n$ at time t_0 .

Consider the control system (1) and a safe set

$$\mathcal{C} = \{x \in \mathbb{R}^n : h(x) \geq 0\} \quad (2)$$

with a local Lipschitz continuous function $h: \mathbb{R}^n \rightarrow \mathbb{R}$ referred to as a nonsmooth control barrier function. The function $h(x)$ is called a (zeroing) CBF if there exists a constant $\gamma > 0$ such that

$$\sup_{u \in U} [L_f h(x) + L_g h(x)u + \gamma h(x)] \geq 0, \quad (3)$$

where $L_f h(x) = \frac{\partial h}{\partial x} f(x)$ and $L_g h(x) = \frac{\partial h}{\partial x} g(x)$ are the Lie derivatives of $h(x)$ along the vector fields $f(x)$ and $g(x)$, respectively. Given a CBF $h(x)$, the set of all control values that satisfy (3) for all $x \in \mathbb{R}^n$ is defined as

$$K_{bf}(x) = \{u \in U : L_f h(x) + L_g h(x)u + \gamma h(x) \geq 0\}. \quad (4)$$

It was proven in [16] that any Lipschitz continuous controller $u(x) \in K_{bf}(x)$ for every $x \in \mathbb{R}^n$ will guarantee the forward invariance of \mathcal{C} . The provably safe control law is obtained by solving an online quadratic program (QP) problem that includes the control barrier condition as its constraint.

3. Controller Design

This section describes the multi-agent security problem and the coverage control based on CBF technology. The aforementioned controller utilizes the CVT algorithm to optimally distribute the agents, effectively solving the coverage control problem. Additionally, it integrates the CVT with the CBF to ensure that the system is able to avoid obstacles with ease.

By the definition of Voronoi tessellations, assuming it is in a two-dimensional case, a Voronoi diagram is a partition of the plane. Every partition carries one generator or more than one generator. Thus, we have the relation $\|q - p_i\| \leq \|q - p_j\|$, where $i \neq j, j \in N$. A Voronoi tessellation is formed as

$$V_i(p) = \|q - p_i\| \leq \|q - p_j\|, \quad i \neq j, \quad (5)$$

where V_i indicates the Voronoi region corresponding to the i -th agent, while \mathcal{D} is the partition of the Voronoi tessellation. In addition to Voronoi tessellation, the CVT will generate the center of mass for each Voronoi region.

The Voronoi tessellation also includes density function $\phi: \mathcal{D} \rightarrow (0, \infty)$, which expresses the relative significance of points in the region and, to calculate how well a given point $q \in \mathcal{D}$ is covered by the agent at position $p_i \in \mathcal{D}$, one defines the locational cost

$$\mathcal{H}(p, t) = \sum_{i=1}^n \int_{V_i} \|q - p_i\|^2 \phi(q) dq, \tag{6}$$

where $\phi(q)$ denotes the associated density function, which is assumed to be positive and bounded. This captures the relative importance of a point $q \in \mathcal{D}$ [27].

From (6), the time derivative of \mathcal{H} can be expressed by

$$\dot{\mathcal{H}} = \sum_{i=1}^n \frac{\partial \mathcal{H}}{\partial p_i} \dot{p}_i + \frac{\partial \mathcal{H}}{\partial t}. \tag{7}$$

Assume that, as the density ϕ changes slowly with respect to time, one has $\partial \mathcal{H} / \partial t = 0$ [8]. In [27], it was shown that

$$\frac{\partial \mathcal{H}}{\partial p_i} = 2m_i(p_i - c_i)^\top, \tag{8}$$

where the mass m_i and the center of mass c_i of the i -th Voronoi cell V_i are defined as

$$m_i = \int_{V_i} \phi(q) dq, \quad c_i = \frac{\int_{V_i} \phi(q) q dq}{m_i}. \tag{9}$$

Note that $m_i > 0$ because the density function $\phi(q)$ is strictly positive [28].

At a given time t , an optimal coverage for the domain \mathcal{D} requires a configuration of agents p_i to minimize \mathcal{H} . From (8), one can see that a critical point is

$$p_i(t) = c_i(p, t), \quad i = \{1, 2, \dots, n\}. \tag{10}$$

When (10) is satisfied, an agent's network is said to be in a locally optimal coverage configuration [28]. The corresponding p defines the CVT.

The concept of the controller for CBF is illustrated in Figure 1. The safety range depicts the safety distance between the agents. The CBF controller is designed to function within the sensing range, which is its designated operating range. The sensing range of agent i and the safety range of agent i refers to the distances at which an agent can detect other agents or obstacles in its surroundings.

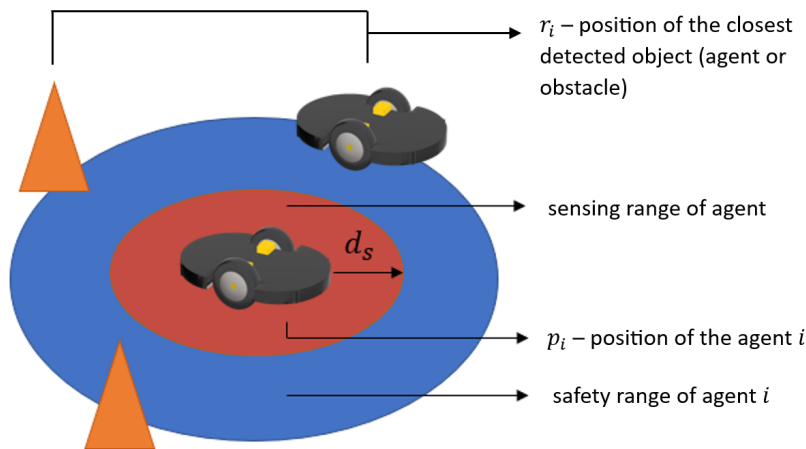


Figure 1. The schematic diagram of CBF controller application to the multi-agent system.

For the safety controller in this study, we model the agent by the first-order integrator,

$$\dot{p}_i = u_i, \quad i = \{1, 2, \dots, n\}, \tag{11}$$

where n is the number of agents, $\mathbf{p}_i \in \mathbb{R}^2$ is the planar position of robot i , and $\mathbf{u}_i \in \mathbb{R}^2$ is its input velocity. A construction of safe set \mathcal{C}_i

$$\mathcal{C}_i = \left\{ \mathbf{p}_i \in \mathbb{R}^2 \mid h_i(\mathbf{p}_i, \mathbf{r}_i) \geq 0 \right\} \tag{12}$$

where

$$h_i(\mathbf{p}_i, \mathbf{r}_i) = \|\mathbf{p}_i - \mathbf{r}_i\|^2 - d_s^2, \tag{13}$$

is defined as a set of a continuous function $h_i \in \mathbb{R}$ where \mathbf{p}_i is the position of the agent i and \mathbf{r}_i is the position of the obstacle or agent that is closest to agent i ; d_s is the minimum safety distance of the agents between each other.

We want to enforce conditions on \mathbf{u}_i to guarantee the system’s safety. However, simply enforcing the condition that $h_i(\mathbf{p}_i, \mathbf{r}_i) > 0$ at a particular time is not useful since $h_i(\mathbf{p}_i, \mathbf{r}_i)$ does not explicitly depend on the control input. The control input influences the state evolution, so we can enforce a safety condition on the time derivative \dot{h}_i . To achieve this, we first adapt the definition of barrier functions from **Definition 1** of [29].

Definition 1 (Nonsmooth Control Barrier Function). *A local Lipschitz continuous function $h : \chi \rightarrow \mathbb{R}$ is a nonsmooth control barrier function (NCBF) if there exists a measurable control law $u : \chi \rightarrow \mathcal{U}$ such that the set $\mathcal{C} = \{x \in \chi : h(x) \geq 0\}$ is forward invariant for the closed-loop system.*

If h is a local Lipschitz continuous function, it is also absolutely continuous and differentiable almost everywhere; then, the following lemma [29] can be used to guarantee safety:

Lemma 1. *Let $\alpha > 0$ be a constant and $h_i : [0, T] \rightarrow \mathbb{R}$ be an absolutely continuous function. If*

$$\dot{h}_i(t) \geq -\alpha \cdot h_i(t) \tag{14}$$

for almost all $t \in [0, T]$ and $h_i(0) > 0$, $h_i(t) \geq h_i(0)e^{-\alpha t} > 0 \forall t \in [0, T]$.

To represent the admissible control space for the given barrier function $h_i(\mathbf{p}_i, \mathbf{r}_i)$, we need to consider the time derivative of barrier function $\dot{h}_i(t)$. The time derivative can be calculated as

$$\dot{h}_i(t) = 2(\mathbf{p}_i - \mathbf{r}_i)^T \dot{\mathbf{p}}_i \tag{15}$$

where $\dot{\mathbf{p}}_i = \mathbf{u}_i$ is the input velocity.

The admissible control space ensures that the safety function $h_i(\mathbf{p}_i, \mathbf{u}_i)$ remains non-negative for all time. Therefore, the admissible control $\mathcal{K}_i(\mathbf{u}_i)$ is defined as

$$\mathcal{K}_i(\mathbf{u}_i) = \left\{ \mathbf{u}_i \in \mathbb{R}^2 \mid \frac{\partial h_i(\mathbf{p}_i, \mathbf{r}_i)}{\partial \mathbf{p}_i} \geq -\alpha h_i(\mathbf{p}_i, \mathbf{r}_i) \right\}. \tag{16}$$

Combining (5) and (16), the velocity input \mathbf{u} needs to satisfy

$$-2(\mathbf{p}_i - \mathbf{r}_i)^T \leq \alpha h_i(\mathbf{p}_i - \mathbf{r}_i), \quad \forall i = \{1, 2, \dots, n\}.$$

This inequality can be treated as a linear constraint on \mathbf{u} when the state \mathbf{p} is given, i.e., by selecting

$$\mathbf{A}_i = -2(\mathbf{p}_i - \mathbf{r}_i)^T, \quad b_i = \alpha h_i(\mathbf{p}_i - \mathbf{r}_i), \tag{17}$$

which becomes

$$\mathbf{A}_i \mathbf{u}_i \leq b_i. \tag{18}$$

The CBF-based obstacle avoidance minimizes the difference between the user’s desired control signal and the actual control signal using a quadratic program (QP) controller

$$\mathbf{u}^* = \arg \min_{\mathbf{u} \in \mathbb{R}^{2n}} J(\mathbf{u}), \tag{19}$$

where

$$J(\mathbf{u}) = \frac{1}{2} \sum_{i=1}^n \|\mathbf{u}_i - \hat{\mathbf{u}}_i\|^2, \quad (20)$$

such that

$$A_i \mathbf{u}_i \leq b_i, \quad \forall i = 1, 2, \dots, n, \quad (21)$$

where $\hat{\mathbf{u}}_i$ is a nominal input and \mathbf{u}_i^* is the actual control command or actual input. The nominal input $\hat{\mathbf{u}}_i$ can be interpreted as the desired or reference value for the control input in each dimension. It serves as a target or baseline for the actual control input \mathbf{u}_i^* and is compared against the cost function. The goal of minimizing this cost function is often to bring the actual control input as close as possible to the desired or nominal input. The linear constraints $A_i \mathbf{u}_i \leq b_i$ ensure safety and are derived from the safety conditions and admissible control space considerations. This QP formulation aims to find the control inputs \mathbf{u}_i^* for all agents that minimize the deviation from the user-specified signals while satisfying safety constraints. The solution to this QP provides the optimal control inputs that balance user preferences with safety requirements.

4. Simulation Tools and Experimental Setup

This section explains the structure of the experimental system, which comprises the hardware and software components that are utilized as well as their interaction with one another, the agents, and the user. The verification is carried out by using simulators and experiments with RT robots.

4.1. Hardware

The hardware used in the experiment, including the robot and the camera for the position-tracking system, is described in detail in this document.

The robot used in this experiment is a two-wheeled Raspberry Pi Mouse V3. A robot equipped with a Raspberry Pi microcomputer manufactured by RT Corporation in Japan. Each device measures 13 cm in width, 10 cm in length, 8.3 cm in height, and weighs 740 g. It runs on a Raspberry Pi running the Ubuntu 20.04 operating system with ROS Noetic. Figure 2 shows a labeled image of the Raspberry Pi Mouse V3.

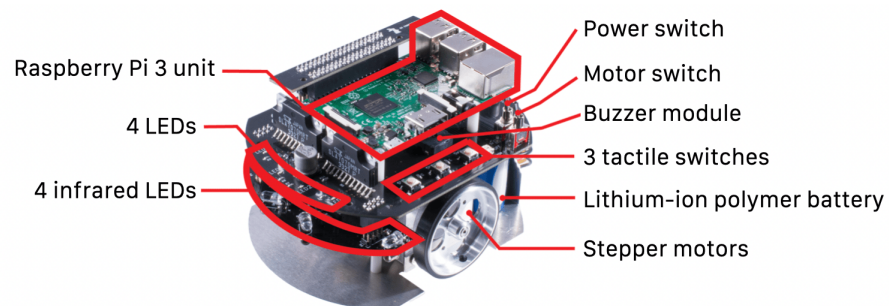


Figure 2. A Raspberry Pi Mouse V3 (RT robot).

Mobile robots must be capable of autonomous localization and navigation. GPS is the tool that is most commonly used for outside localization; however, it cannot be utilized indoors. Among the various tools available for indoor localization, such as Lidar, Ultra Wide Band (UWB), and WiFi, the use of cameras for localization is the method that provides maximal flexibility at the lowest possible cost [30]. ARTag is an augmented reality fiducial marker system.

This technology has the potential to make games, animations, and other forms of virtual content appear more natural in the real world. ARTag can use physical markers to calculate the position and orientation of the camera in real-time, a feature called video tracking. Once the camera position is determined, a virtual camera can be placed in the same location to view virtual objects in the location where the ARTag is placed.

Markers are attached to the top of each robot's body, allowing the camera to track the movements of each robot in turn. Figure 3 shows examples of markers. AR Track Alvar is the name of the package used for ARTag. AR Track Alvar is a free and open-source ARTag tracking library available via the ROS wrapper package for Alvar [31]. The Alvar tool has been used as the basis for many other ROS ARTag packages. Alvar features adaptive thresholding that works under a variety of lighting conditions, optical flow-based tracking for more reliable body pose estimation, and an improved tag recognition engine that does not slow down as the number of tags increases.

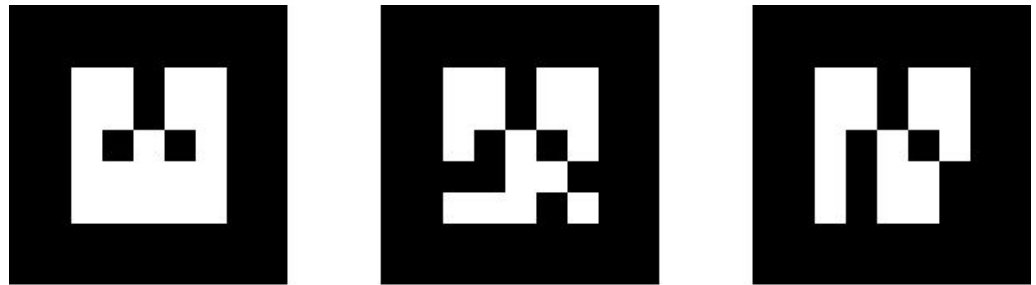


Figure 3. ARTag marker with various IDs. Left ID = 0, Middle ID = 1, and Right ID = 2.

The connection to each Raspberry Pi Mouse V3 is made by connecting both the robot and the PC to the same network where the former has its IP address. Upon calling the IP address on the PC side, one will be able to designate a robot ID and have complete control of the robot's parameters, such as the motors.

The camera used as a tracking system in this experiment is the Intel RealSense Depth Camera D435, a stereo depth camera that is attached to the ceiling and used as an overhead camera. This camera is ideal for use with ROS (Robot Operating System) since the software package required for ARTag (Augmented Reality Tag) tracking is already included in the ROS system. According to our experimental setup, the optimal distance range for this camera is 0.3 m to 3 m, which works perfectly. Nevertheless, its wide field of view makes it suitable for use in applications related to robotics or augmented reality. The physical port of this camera is the latest version, USB-C TO USB 3.1 Gen 1.

In the actual experiment, the robots work in a flat area measuring 2×2 m within the experimental area. The cameras are mounted on the ceiling as a global overhead camera tracking system to determine the positions of all robots and generate feedback loops to control the positions of the robots. The experiments require a personal computer with Ubuntu and ROS pre-installed.

In our case, we are using Ubuntu 20.04 with ROS Noetic, as several libraries we use for Python programming require Python3, and ROS Noetic supports it. For the robot, we have the Ubuntu 18.04 server and ROS Melodic installed. The important thing is that all robots and the local PC must be connected to the same network. In short, they must all be connected to the same Wi-Fi. Figure 4 shows an image of the experimental setup. The USB camera is attached to the lab ceiling using a universal camera ceiling mount. The camera is raised approximately 2.3 m above the ground and has a downward field of view at approximately a 90-degree angle to the ground. To track AR markers, you must first install both the ROS package for your camera and the AR Track Alvar package. The structure of this system is shown in Figure 5. In our experiments, the control inputs of the robot are continuously updated according to the position of the robot. The proposed control algorithm calculates the control inputs and sends them to each robot. As a result, the robot must be in the camera's field of view at all stages.

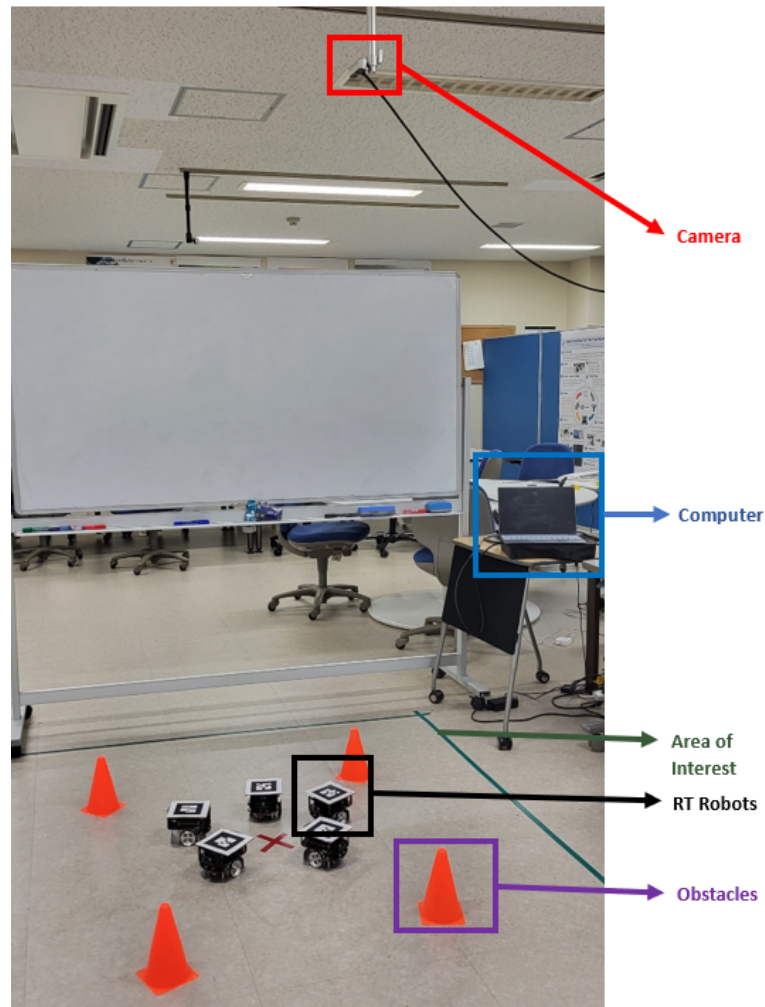


Figure 4. The experimental setup.

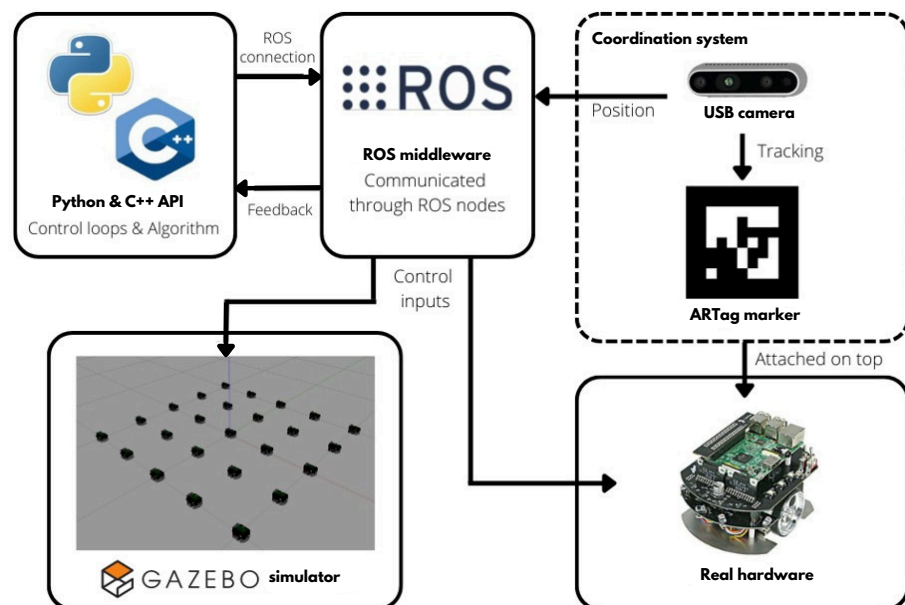


Figure 5. System architecture.

4.2. Software

In this section, the software utilized in our system, including the middleware that controls the robots, the simulator for simulation, and the global coordination system, is described.

In general, robots are equipped with a variety of sensors and actuators. Instead of having a large, difficult-to-maintain, monolithic code base, middleware is required to construct a distributed system for each component, particularly in situations in which the robot needs to communicate with either another robot or more external hardware. ROS is open-source robotics middleware designed for low-level operation and management of robotic assets. ROS is built around the concepts of messages, nodes, services, and communication topics. Nodes act as processes that perform computations and communicate with other nodes via messages. When information of interest is requested, a topic or subscription is published and then a message is sent. ROS provides complete support for two client libraries and programming languages: Python (rospy) and C++ (roscpp). The client libraries are structured into pre-configured software modules that are referred to as packages. Because of its platform portability, modularity, and capacity to handle concurrent resource management, ROS is the software of choice.

The simulation test is a necessary step in the experimentation process. Simulators make it possible to build and test the algorithms on the machines themselves before putting them through their paces on the actual robot. The Gazebo simulator is popular simulation software that works with ROS to simulate the dynamics and physics of real robots [31]. The robot's 3D URDF (Unified Robotics Description Format) model is required for simulation in Gazebo. The Raspberry Pi mouse URDF model is a replica of the real robot, which can be seen in Figure 6.

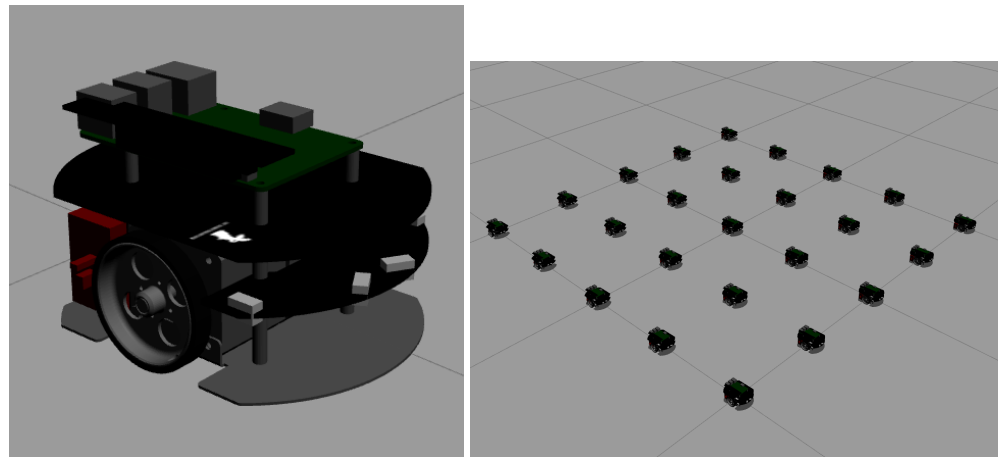


Figure 6. Raspberry Pi Mouse V3 model (left), Multiple Raspberry Pi robots in Gazebo (right).

5. Structure of the Proposed Framework

This section presents a proposed framework designed to enhance the performance and reliability of mobile robots. The proposed framework integrates several key phases to achieve effective coverage control and obstacle avoidance, ensuring that agents operate safely and efficiently in real time. Figure 7 shows the system flowchart of the proposed framework.

The first step of the framework is to gather the current coordinates and goal points for each agent in the system. This starts with collecting accurate positional data from each robot, which serves as the basis for subsequent operations. Goal points are then established to guide the agents to their desired destinations. Proper setup of these coordinates and goals is essential to ensure that the agents can navigate effectively and accomplish their tasks.

After the agents' positions and goals are established, the framework deals with static obstacles within the environment. This phase includes detecting and representing obstacles to ensure that they are considered in the navigation process. By integrating these obstacles

into the system, the framework can modify agent movements to avoid collisions and maintain safety. Effective obstacle processing ensures that the agents can navigate around fixed barriers without compromising their coverage objectives.

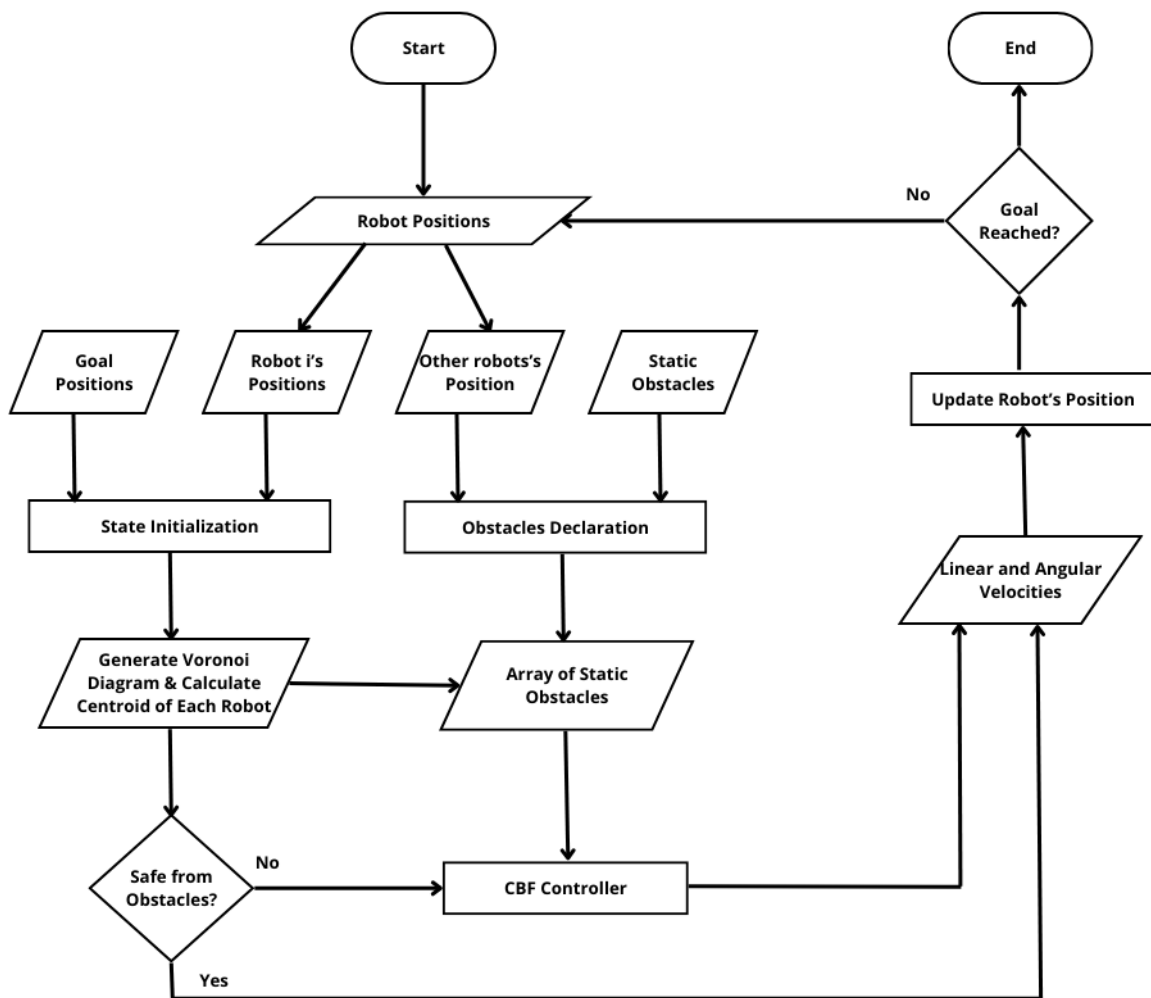


Figure 7. System flowchart of proposed framework.

To ensure the best possible coverage and coordination, the framework creates a Voronoi diagram based on the positions of the agents. This diagram divides the environment into regions of influence for each agent, which helps to optimize coverage. The center of each Voronoi cell is calculated to determine the central point for each agent’s area of responsibility. This information is essential for guiding agents to the best positions and ensuring efficient coverage of the environment. This coverage is provided by CVT.

A critical component of the framework is the CBF, which enforces safety constraints and prevents collisions during coverage operations. The CBF imposes movement constraints to ensure agents avoid obstacles and maintain safe distances from each other. By using CBFs, the framework addresses the complex challenge of collision avoidance, ensuring that agents adhere to safety requirements in complex environments.

In the final phase, the framework calculates and applies the necessary linear and angular velocities to control the agents’ movements. These velocities are derived from the previous phases and are essential for executing the planned trajectories. This step translates the theoretical commands into actionable movements, enabling real-time navigation and effective coverage.

6. Case Study

For the verification of the proposed CBF-based controller, a safety critical control scenario is considered in the coverage problem. Specifically, each agent has a safety range that should not overlap with that of other agents.

To demonstrate the effectiveness and efficiency of obstacle avoidance for multi-agent coverage, a comparison between the results under the Artificial Potential Field (APF) method and the proposed CBF-based controller is conducted in a Gazebo simulator. Figure 8 shows the simulation results under the CVT-APF controller and Figure 9 shows the simulation results under the CVT-CBF controller. In the CVT-APF controller, collision-free motion cannot be guaranteed which is illustrated by Figure 8c,d, whereas, in the CVT-CBF controller, collision-free motion is always guaranteed as shown in Figure 9.

The trajectories of all robots controlled by the CVT-APF controller and the minimum safe distance graphs between all robot pairs are shown in Figures 10a and 11a. The moving trajectories of all robots controlled by the CVT-CBF controller and the minimum safe distance graphs between all robot pairs are shown in Figures 10b and 11b. This comparison between the CVT-APF controller and the CVT-CBF controller highlights the effectiveness and advantages of our proposed approach.

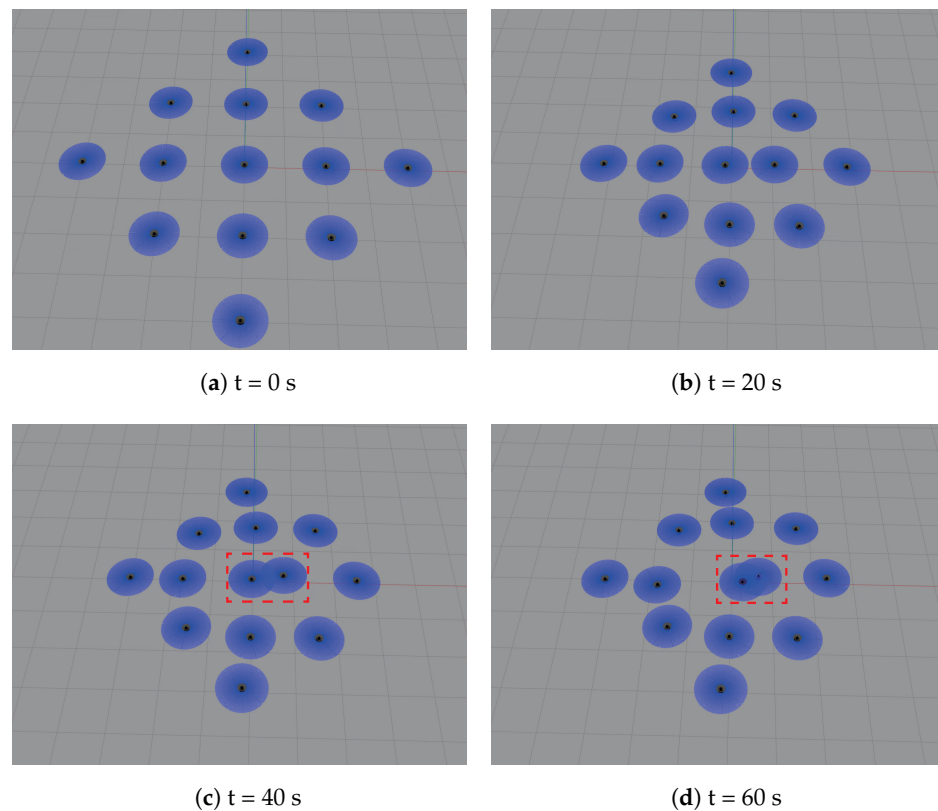


Figure 8. Simulation results under CVT-APF controller.

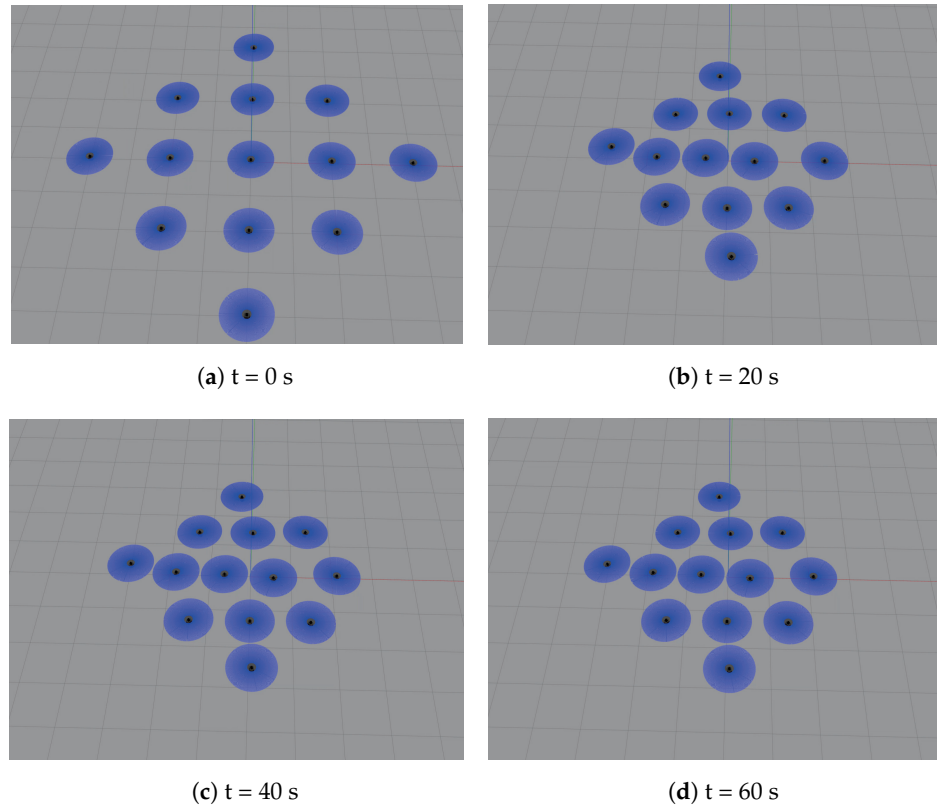


Figure 9. Simulation results under CVT-CBF controller.

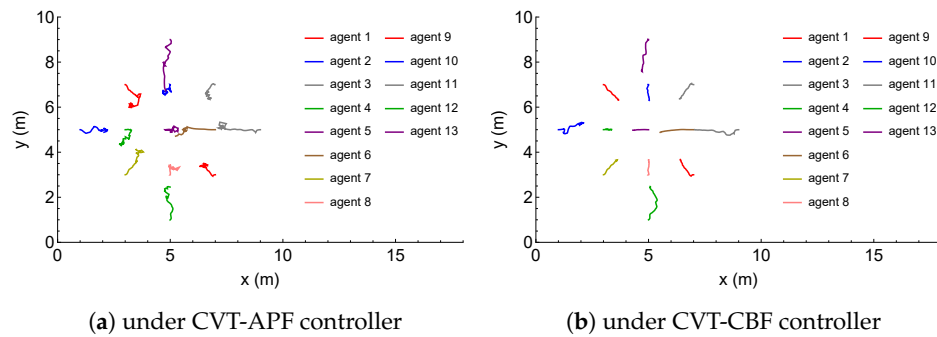


Figure 10. Trajectory path of all the robots

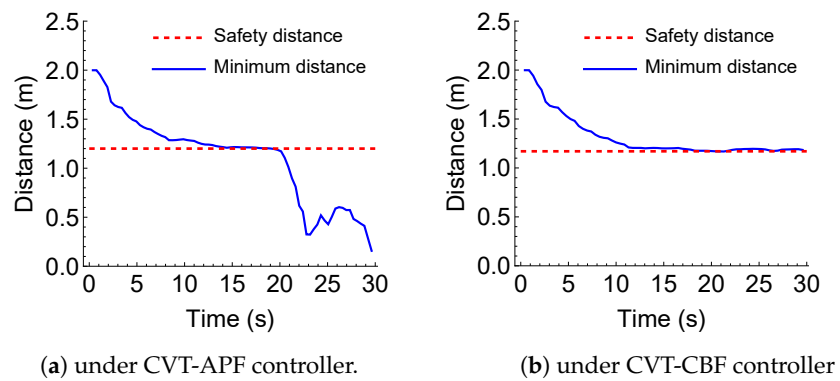


Figure 11. Minimum distance between all pairs of the robots.

7. Simulation Results

In this section, the proposed controller is verified under the ROS/Gazebo simulator. The purpose of the simulation is to show multiple robots distributed in a designated area of interest while demonstrating obstacle avoidance between robots.

The Gazebo simulator which is a good demonstration displays how real robots function in a simulated environment. The Gazebo simulator enables developers to model complex environments and integrate with the widely used ROS framework. Simulating in Gazebo is a cost-effective way to iterate and refine robotic systems, reducing the need for expensive physical prototypes and environments. Overall, Gazebo's capabilities make it a preferred choice for robotics simulation and development.

The robot's setup in the Gazebo simulator is shown in Figure 12. The initial position and the goal position of the robots are given in the Gazebo simulator. The simulator has four robots and nine obstacles for verification. The simulation was successful with all four robots moving from their initial positions without colliding with any obstacles or with each other. The snapshots of the simulation result with RT robots taken at $t = 0$ s, $t = 20$ s, $t = 40$ s, and $t = 60$ s for four robots and nine obstacles are depicted in Figure 13. The initial position could be randomly positioned but, for this experiment, it is set up as shown in Figure 13a. The simulation was successfully achieved with all four of the robots going from initial positions without colliding with the obstacles which is shown in Figure 13d.

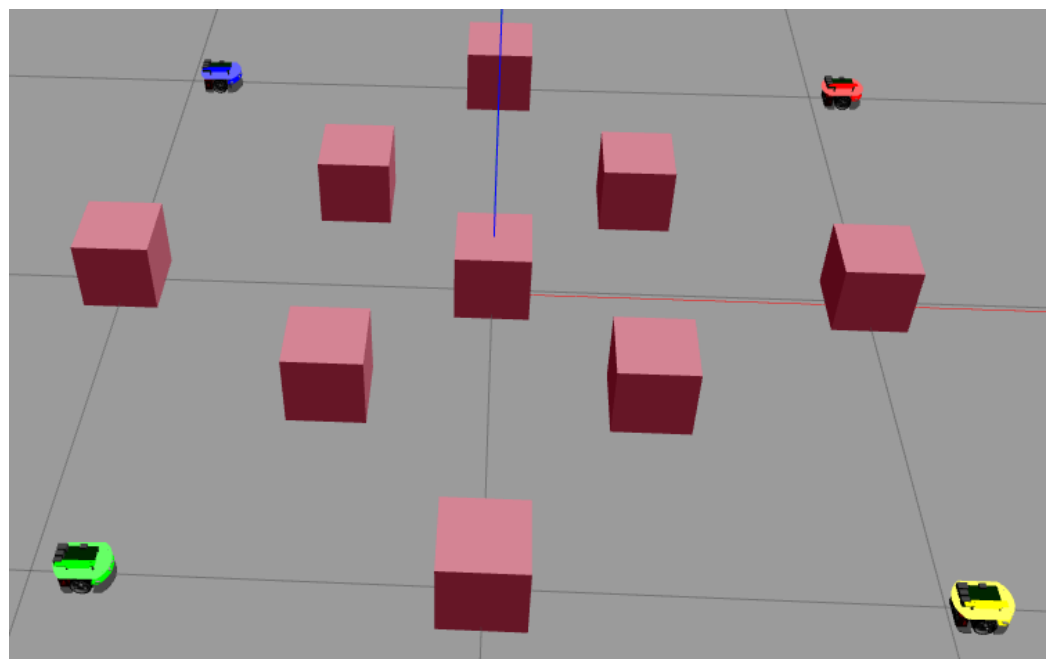


Figure 12. Experimental setup in Gazebo simulator for four robots and nine obstacles.

The trajectory path of all the robots and the minimum safety distance graph between all the pairs of the robots in the Gazebo simulator is depicted in Figure 14. The black dots in the trajectory path represent obstacles. The red dots represent the initial positions of the robots. The safety distance of each robot in the Gazebo simulator is 0.4 m.

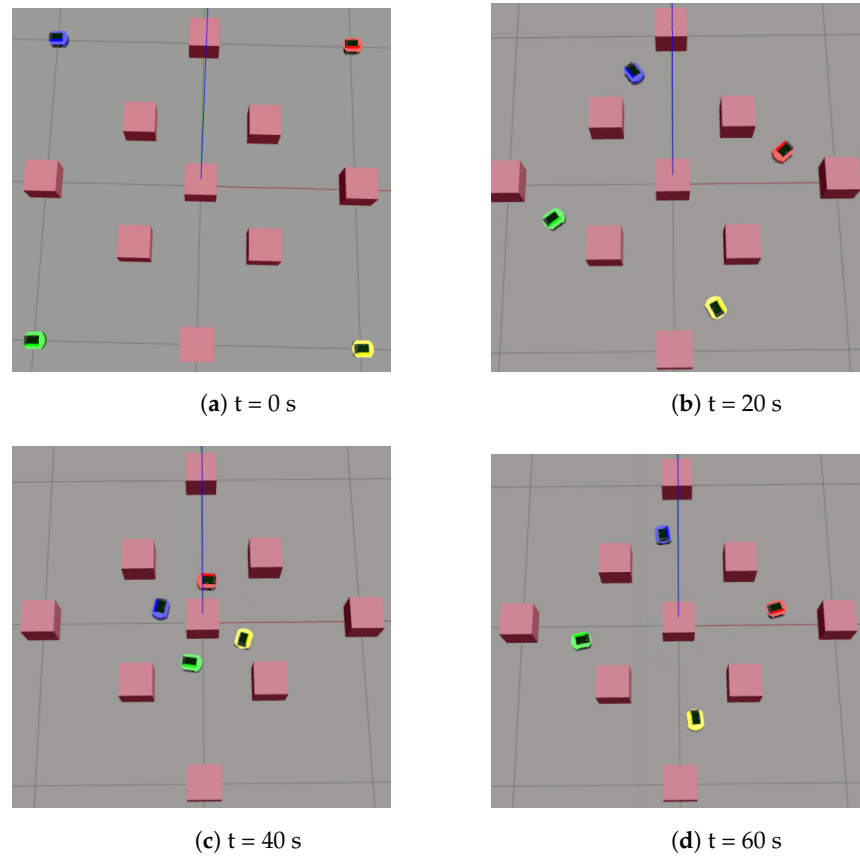


Figure 13. Simulation results with RT robots.

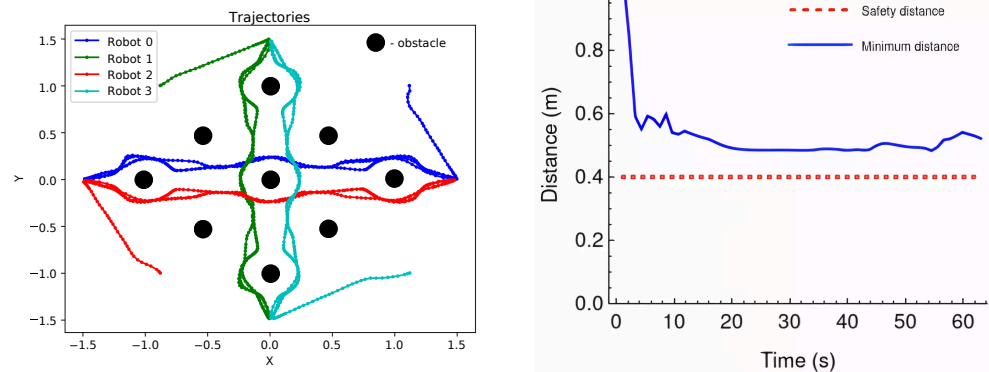


Figure 14. Trajectories of all the robots with four obstacles (left), the minimum distance between all pairs of the robots (right).

8. Experimental Results

To test the proposed controller, an experiment is designed with four robots and nine obstacles. The robot setup in the experimental scenario is shown in Figure 15. The experiment is conducted with four RT robots and nine orange cones which represent the obstacles.

Snapshots of the experiment result with RT robots taken at $t = 0$ s, $t = 20$ s, $t = 40$ s, and $t = 60$ s for four robots and nine obstacles are depicted in Figure 16. The initial position could be randomly positioned but, for this experiment, it is set up as shown in Figure 16a. The experiment was successfully achieved with all four robots going from initial positions without colliding with the obstacles which is shown in Figure 16d. The trajectory plot for all four RT robots and the minimum safety distance between all pairs of the RT robots are

shown in Figure 17. The black dots in the trajectory path represent obstacles. The red dots represent the initial positions of the robots. The safety distance of each robot in the Gazebo simulator is 0.10 m.

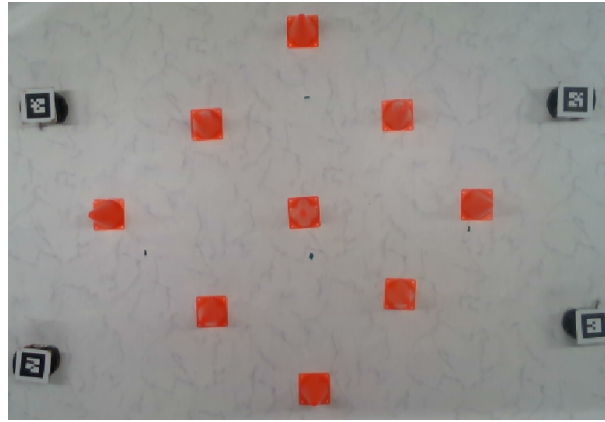


Figure 15. Experimental setup with RT robots.

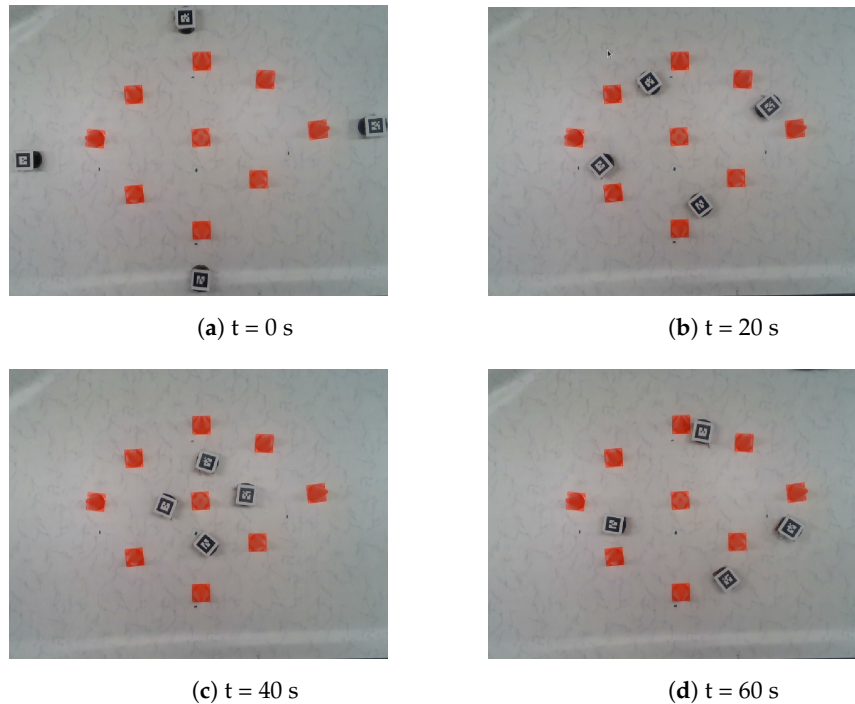


Figure 16. Experimental results with RT robots.

The simulations and real experiments may differ in terms of system dynamics. Simulations typically rely on mathematical models that represent the behavior of the system. However, real-world systems can exhibit unanticipated complexities, nonlinear behaviors, or dynamic interactions that are not fully captured by the simulation models. These differences can affect the determination of safety distances. The simulations often involve simplifications and assumptions to make the calculations more tractable. Real experiments involve physical measurements, which can have inherent uncertainties and errors. These can affect the determination of safety distances.

As the RT mobile robots are differential-drive robots, they can be modeled as unicycles

$$\dot{x}_i = v_i \cos \theta_i, \quad \dot{y}_i = v_i \sin \theta_i, \quad \dot{\theta}_i = \omega_i, \quad (22)$$

where (x_i, y_i) is the position of robot i , θ_i its heading, and v_i, ω_i are the translation and angular velocities. In comparison, the coverage algorithm provides desired motions in terms of \dot{p}_i and we map these onto v_i, ω_i through

$$v_i = \|\dot{p}_i\|, \quad (23)$$

$$\omega_i = [-\sin \theta_i, \cos \theta_i] \dot{p}_i / \|\dot{p}_i\|, \quad (24)$$

so that the amount of lateral motion in the given vector \dot{p}_i is proportional to the angular velocity of the i -th robot.

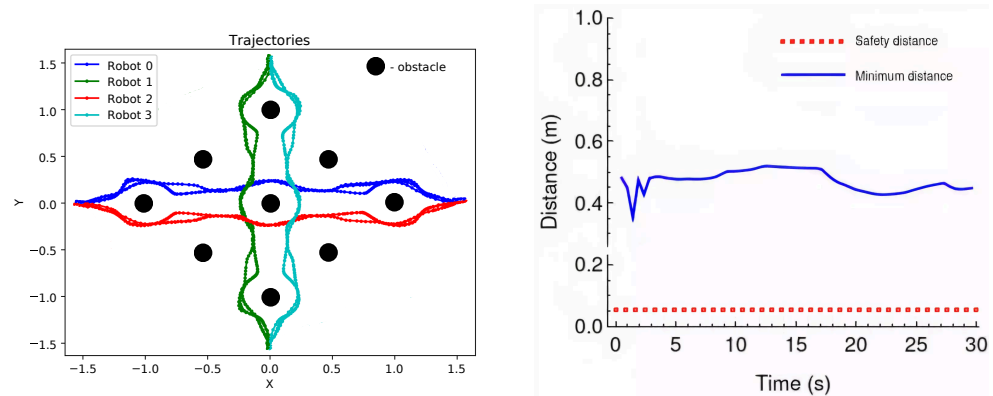


Figure 17. Trajectory plot for all five RT robots (left), minimum distance between all pairs of the RT robots (right).

9. Conclusions

This paper introduces safe coverage control algorithms for multi-agent systems that utilize CVT and CBFs. The main goal of this paper has been achieved by integrating the concept of CVT and CBFs with obstacles, testing the controller under simulation, and verifying the results with RT mobile robots.

The proposed approach in this paper has presented the integration of guaranteeing safety provided by CBF with the spatial optimization capabilities of CVT. This integration achieves a synergy that addresses safety and spatial optimization simultaneously in multi-agent systems, where both are crucial. Simulator setups have been built to test the validity of the proposed controller. The proposed controller has been verified through experiments in the ROS Gazebo simulator and with mobile RT robots. The use of CVT- and CBF-based controllers ensures the optimal distribution of robots and obstacle avoidance.

In developing and verifying the proposed algorithm for safe coverage control in multi-agent systems, it is crucial to consider the following constraints and assumptions. Firstly, the communication constraint is addressed by taking into account latency in a setup that involves less than 10 robots. However, it is noted that, as the number of robots increases, latency may become more noticeable. Secondly, the sensing constraint revolves around the accurate positioning data required by the algorithm. While the algorithm assumes ideal coverage, variations in accuracy and a limited field of view could impact the algorithm's performance in practical scenarios. These considerations are essential for understanding the potential limitations and practical implications of the algorithm in real-world applications.

These findings provide a better understanding of the integration of CVT and CBF in the scope of coverage control with obstacle avoidance for multi-agent systems. Throughout the simulations carried out in ROS/Gazebo and experiments with RT robots, CVT and CBF have displayed their role in determining safe coverage while taking into account the desired trajectory. In particular, the ROS/Gazebo simulator gave us a clear picture of how CVT and CBF could be implemented. The experiment with RT robots has shown how a collision-free coverage system is achieved in practice despite the size limitations.

As we scale up the system, it is important to consider some potential challenges and limitations. Firstly, our current approach relies on a centralized framework, which may pose scalability challenges as the number of agents increases. The current centralized approach can strain computational resources and cause delays, making real-time decision making challenging. To tackle this issue, a shift from a centralized to a decentralized or distributed control architecture can be implemented. A decentralized approach can distribute the computational load among agents, reducing the dependence on a single central unit and improving scalability. Secondly, our current system uses a centralized camera setup for position tracking with ARTags, but it has limitations when managing a large number of robots due to the limited field of view. To address this, we are considering decentralized solutions commonly used in field robotics. These systems use onboard sensors like LiDAR, GPS, IMUs, and local cameras for tracking and navigation without relying on a central camera system, which can improve coverage and reliability.

In our future work, we plan to explore additional aspects to develop a more comprehensive approach to multi-agent systems. This will include scenarios where the obstacles are equidistant from agents, which may require the design of an extra layer in the control scheme. Furthermore, we will conduct research on the non-convex shape of obstacles in the combined CVT and CBF framework, and transition from a centralized framework to a decentralized framework to enhance the overall efficiency and versatility of the systems which can lead to interesting findings and challenges.

Author Contributions: Methodology, M.S.; Software, F.C.S. and J.S.; Validation, J.S.; Investigation, F.C.S. and M.S.; Writing—original draft, F.C.S. and M.S.; Visualization, J.S.; Supervision, M.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

- Ávila-Martínez, E.J.; Barajas-Ramírez, J.G. Flocking motion in swarms with limited sensing radius and heterogeneous input constraints. *J. Frankl. Inst.* **2021**, *358*, 2346–2366. [[CrossRef](#)]
- Lee, S. Coverage Technology of Autonomous Mobile Mapping Robots. In *Autonomous Mobile Mapping Robots*; IntechOpen: 2023. Available online: <https://www.intechopen.com/chapters/84996> (accessed on 29 June 2024).
- Kim, K.H.; Kim, J.H.; Ko, Y.J.; Lee, H.E. Body-attachable multifunctional electronic skins for bio-signal monitoring and therapeutic applications. *Soft. Sci.* **2024**, *4*, 24. [[CrossRef](#)]
- Jan, A.A.; Kim, S.; Kim, S. A skin-wearable and self-powered laminated pressure sensor based on triboelectric nanogenerator for monitoring human motion. *Soft. Sci.* **2024**, *4*, 23. [[CrossRef](#)]
- Liu, K.; Yang, P.; Wang, R.; Jiao, L.; Li, T.; Zhang, J. Observer-Based Adaptive Fuzzy Finite-Time Attitude Control for Quadrotor UAVs. *IEEE Trans. Aerosp. Electron. Syst.* **2023**, *59*, 8637–8654. [[CrossRef](#)]
- Liu, K.; Yang, P.; Jiao, L.; Wang, R.; Yuan, Z.; Li, T. Observer-Based Adaptive Finite-Time Neural Control for Constrained Nonlinear Systems With Actuator Saturation Compensation. *IEEE Trans. Instrum. Meas.* **2024**, *73*, 7502516. [[CrossRef](#)]
- Liu, K.; Wang, X.; Wang, R.; Sun, G.; Wang, X. Antisaturation Finite-Time Attitude Tracking Control Based Observer for a Quadrotor. *IEEE Trans. Circuits Syst. II Express Briefs* **2021**, *68*, 2047–2051. [[CrossRef](#)]
- Zhiyang, J.; Hui, Z.; Ying, T.; Xiang, C. Coverage control of mobile sensor networks with directional sensing. *Math. Biosci. Eng.* **2022**, *19*, 2913–2934.
- Lee, S.G.; Diaz-Mercado, Y.; Egerstedt, M. Multirobot Control Using Time-Varying Density Functions. *IEEE Trans. Robot.* **2015**, *31*, 489–493. [[CrossRef](#)]
- Chism, I.; Plante, D.; Miah, M.S. Area Coverage Optimization using Networked Mobile Robots with State Estimation. In Proceedings of the International FLAIRS Conference Proceedings, Clearwater Beach, FL, USA, 14–17. May 2023; Volume 36.
- Chevet, T.; Maniu, C.S.; Vlad, C.; Zhang, Y. Guaranteed Voronoi-based Deployment for Multi-Agent Systems under Uncertain Measurements. In Proceedings of the 2019 18th European Control Conference (ECC), Naples, Italy, 25–28 June 2019; pp. 4016–4021.
- Benevento, A.; Santos, M.; Notarstefano, G.; Paynabar, K.; Bloch, M.; Egerstedt, M. Multi-Robot Coordination for Estimation and Coverage of Unknown Spatial Fields. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 7740–7746.

13. Li, S.; Li, B.; Yu, J.; Zhang, L.; Zhang, A.; Cai, K. Probabilistic Threshold k-ANN Query Method Based on Uncertain Voronoi Diagram in Internet of Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 3592–3602. [[CrossRef](#)]
14. Zhu, H.; Brito, B.; Alonso-Mora, J. Decentralized probabilistic multi-robot collision avoidance using buffered uncertainty-aware Voronoi cells. *Auton. Robot.* **2022**, *46*, 401–420. [[CrossRef](#)]
15. Sharma, Y.R.; Ratnoo, A. Designing Safe Lane-Change Maneuvers Using an Explicit Path Planner. In Proceedings of the IEEE 17th International Conference on Automation Science and Engineering (CASE), Lyon, France, 23–27 August 2021; pp. 1920–1926.
16. Pierson, A.; Schwarting, W.; Karaman, S.; Rus, D. Weighted Buffered Voronoi Cells for Distributed Semi-Cooperative Behavior. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 5611–5617.
17. Lin, H.Y.; Peng, X.Z. Autonomous Quadrotor Navigation with Vision Based Obstacle Avoidance and Path Planning. *IEEE Access* **2021**, *9*, 102450–102459. [[CrossRef](#)]
18. Ahmed, S.; Qiu, B.; Ahmad, F.; Kong, C.W.; Xin, H. A State-of-the-Art Analysis of Obstacle Avoidance Methods from the Perspective of an Agricultural Sprayer UAV's Operation Scenario. *Agronomy* **2021**, *11*, 1069. [[CrossRef](#)]
19. Nouman, B.; Saadi, B.; Gabriel, D.; Sherali, Z. An obstacle avoidance approach for UAV path planning. *Simul. Model. Pract. Theory* **2023**, *129*, 102815.
20. Ulrich, I.; Borenstein, J. Vfh+: Reliable obstacle avoidance for fast mobile robots. In Proceedings of the 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146), Leuven, Belgium, 20–20 May 1998; Volume 2, pp. 1572–1577.
21. Srinivasan, D.; Cheu, R.L.; Poh, Y.P.; Chwee, A.K. Development of an intelligent technique for traffic network incident detection. *Eng. Appl. Artif. Intell.* **2000**, *13*, 311–322. [[CrossRef](#)]
22. Munoz-Vazquez, A.J.; Parra-Vega, V.; Sanchez-Orta, A. Adaptive Fuzzy Velocity Field Control for Navigation of Nonholonomic Mobile Robots. *J. Intell. Robot. Syst.* **2021**, *101*, 38. [[CrossRef](#)]
23. Holliday, T.; Hill, B.; Wold, J. Modeling and Prototyping a Modular, Low-Cost Collision Avoidance System for UAVs. In Proceedings of the IEEE Aerospace Conference (50100), Big Sky, MT, USA, 6–13 March 2021; pp. 1–15.
24. Ames, A.D.; Coogan, S.; Egerstedt, M.; Notomista, G.; Sreenath, K.; Tabuada, P. Control barrier functions: Theory and applications. In Proceedings of the 18th European Control Conference (ECC), Naples, Italy, 25–28 June 2019; pp. 3420–3431.
25. Singletary, A.; Klingebiel, K.; Bourne, J.; Browning, A.; Tokumaru, P.; Ames, A. Comparative Analysis of Control Barrier Functions and Artificial Potential Fields for Obstacle Avoidance. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 8129–8136.
26. Kostak, M.; Slaby, A. Designing a Simple Fiducial Marker for Localization in Spatial Scenes Using Neural Networks. *Sensors* **2021**, *21*, 5407. [[CrossRef](#)] [[PubMed](#)]
27. Santos, M.; Madhushani, U.; Benevento, A.; Leonard, N.E. Multi-robot Learning and Coverage of Unknown Spatial Fields. In Proceedings of the International Symposium on Multi-Robot and Multi-Agent Systems (MRS), Cambridge, UK, 4–5 November 2021; pp. 137–145.
28. Rossi, F.; Bandyopadhyay, S.; Wolf, M.; Pavone, M. Review of Multi-Agent Algorithms for Collective Behavior: A Structural Taxonomy. *IFAC-PapersOnLine* **2018**, *51*, 112–117. [[CrossRef](#)]
29. Thirugnanam, A.; Zeng, J.; Sreenath, K. Nonsmooth Control Barrier Functions for Obstacle Avoidance between Convex Regions. *arXiv* **2023**, arXiv:2306.13259.
30. Wang, S.; Laskar, Z.; Melekhov, I.; Li, X.; Kannala, J. Continual Learning for Image-Based Camera Localization. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, BC, Canada, 11–17 October 2021; pp. 3252–3262.
31. Cohen, M.; Belta, C. Safety-Critical Control. In *Adaptive and Learning-Based Control of Safety-Critical Systems*; Synthesis Lectures on Computer Science; Springer: Cham, Switzerland, 2023; pp. 29–56.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.