frontiers in Robotics and AI

ORIGINAL RESEARCH
published: 25 October 2021
doi: 10.3389/frobt.2021.740460

Persistent Object Search and Surveillance Control With Safety Certificates for Drone Networks Based on Control Barrier Functions

Hayato Dan[1], *Takeshi Hatanaka*[1]*, *Junya Yamauchi*[2], *Takumi Shimizu*[1] *and Masayuki Fujita*[2]

[1]School of Engineering, Tokyo Institute of Technology, Tokyo, Japan, [2]Guraduate School of Information Physics and Computing, The University of Tokyo, Tokyo, Japan

In this paper, we address a persistent object search and surveillance mission for drone networks equipped with onboard cameras, and present a safe control strategy based on control barrier functions The mission for the object search and surveillance in this paper is defined with two subtasks, persistent search and object surveillance, which should be flexibly switched depending on the situation. Besides, to ensure actual persistency of the mission, we incorporate two additional specifications, safety (collision avoidance) and energy persistency (battery charging), into the mission. To rigorously describe the subtask of persistent search, we present a novel notion of γ-level persistent search and the performance certificate function as a candidate of a time-varying Control Barrier Function. We then design a constraint-based controller by combining the performance certificate function with other CBFs that individually reflect other specifications. In order to manage conflicts among the specifications, the present controller prioritizes individual specifications in the order of safety, energy persistency, and persistent search/object surveillance. The present controller is finally demonstrated through simulation and experiments on a testbed.

Keywords: search and surveillance, drone networks, safe control, persistency, control barrier functions, distributed control, coverage control

OPEN ACCESS

Edited by:
Ashwin Dani,
University of Connecticut,
United States

Reviewed by:
Elias B. Kosmatopoulos,
Democritus University of Thrace,
Greece
Nikhil Chopra,
University of Maryland, College Park,
United States

***Correspondence:**
Takeshi Hatanaka
hatanaka@sc.e.titech.ac.jp

Specialty section:
This article was submitted to Robotic Control Systems, a section of the journal Frontiers in Robotics and AI

Received: *13 July 2021*
Accepted: *01 October 2021*
Published: *25 October 2021*

Citation:
Dan H, Hatanaka T, Yamauchi J, Shimizu T and Fujita M (2021) Persistent Object Search and Surveillance Control With Safety Certificates for Drone Networks Based on Control Barrier Functions. Front. Robot. AI 8:740460. doi: 10.3389/frobt.2021.740460

1 INTRODUCTION

Environmental monitoring is one of the key applications of networked multi-robot systems, wherein each robot is expected to deploy over the mission space. To this end, the most promising control technology is coverage control that provides distributed control strategies for enhancing efficiency of information acquisition on the environment (Cortés et al., 2005; Martínez et al., 2007; Renzaglia et al., 2012). The recent technological advances in drone technology make it viable to implement coverage control on drone networks, and many successful results have been reported in the literature (Schwager et al., 2011; Bentz et al., 2018; Funada et al., 2019). These publications consider the scene such that drones with onboard cameras looking down the ground to be monitored move around over the ground as illustrated in **Figure 1**.

Specifications for environmental monitoring vary depending on the application scenarios. In this paper, we address a scene where drones are required to surveil a target object on the environment whose location is initially unknown for the drones. In this scenario, the drones need to first search the object, and then to switch the task to surveillance of the object once it is found. In

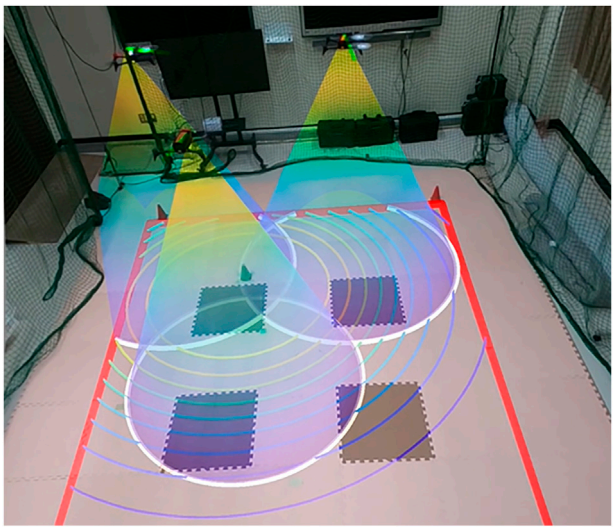

FIGURE 1 | A scene of coverage control with drone networks.

the phase of searching, the drones are expected to take exploratory actions to patrol the mission space while avoiding too much overlaps of fields of view among drones. Avoiding the overlaps is handled by coverage control but most of the coverage control algorithms lead robots to a stationary configuration rather than persistently taking patrolling motion. Consequently, some subregion may remain uncovered and, accordingly, the drones may fail to find the object especially when the number of drones is not many enough to fully cover the environment as in the scene of **Figure 1**. To address the issue, the authors presented persistent coverage control schemes in (Hübel et al., 2008; Sugimoto et al., 2015; Kapoutsis et al., 2019), where a notion of information reliability is introduced and so-called density function is dynamically updated according to the reliability. It is then exemplified that the gradient ascent algorithm with the update of the density function generates persistently patrolling motion over the mission space. A similar concept is also presented in (Wang and Wang, 2017), wherein the concept is termed *awareness*. However, these methodologies do not provide any guarantee on the coverage performance. Meanwhile, Franco et al. (2015) and Palacios-Gasós et al. (2016) address the performance guarantee for the persistent coverage, but a prescribed performance level is not always ensured therein in the presence of the performance decay in time. Kapoutsis et al. (2019) present a persistent coverage scheme not requiring exact models of the environment and robot's coverage capabilities.

In order to ensure persistency of the mission in practice, it is not enough just to make drones take persistent motion and we have to meet a variety of constraints. For example, we need to certify safety during the mission. Specifically, collision avoidance among drones must be a key in ensuring persistency since drones no longer continue the mission if they collide with each other just once. Moreover, drones are normally driven by batteries with limited storage, and battery exhaustion prevents drones from continuing the mission. We thus need to take account of energy persistency, namely we need to control drones so that they return to charging stations before their batteries are exhausted. These issues have been individually addressed e.g. in (Hussein et al., 2007; Zhu and Martínez, 2013; Bentz et al., 2018; Wang et al., 2020), but a more general framework to flexibly integrate a variety of specifications is needed. Meanwhile, a great deal of recent publications have been devoted to Control Barrier Function (CBF) in order to certify the constraint fulfillment, e.g., to ensure safe operation of multi-robot systems (Ames et al., 2017; Notomista et al., 2018). The CBF has also been employed in coverage control, e.g., in (Egerstedt et al., 2018; Funada et al., 2019). Egerstedt et al. (2018) certifies collision avoidance and maintenance of the energy level in the coverage mission based on the inherent flexibility of CBFs that allows one to integrate various specifications. Funada et al. (2019) manages overlaps of fields of view for drone networks using the CBFs. The paper most closely related to the present paper is Santos et al. (2019), wherein the authors investigate coverage control with a time-varying density function similarly to the persistent coverage control. However, the paper does not give any explicit guarantee of the coverage performance.

In this paper, we present a novel persistent object search and surveillance control with safety certificates for drone networks based on CBFs. We first introduce a new concept of γ-level persistent search as a performance metric for the searching mission in the form of a constraint function. We then formulate constraint functions that describe the control goal for the object surveillance and specifications for safety (collision avoidance) and energy persistency (battery charging). We then formulate inequality constraints to be met by the control input, following the manner of CBFs. A constraint-based controller is then presented, including all of the above inequality constraints. The controller with all of the constraints however may result in issues on infeasibility in online optimization required by the controller. We thus present prioritization among the constraints, where we place priority in the order of safety, energy persistency, and persistent search/object surveillance. Based on the designed priority, we present a novel constraint-based controller that ensures feasibility, where the inequality constraints for persistent search and object surveillance are appropriately switched depending on whether the object is detected or not. The controller is moreover shown to be implemented in a partially distributed manner. We then run simulation of the constraint-based control only with the performance certificate for the persistent search. It is revealed there that the present constraint-based controller maintains the γ-level persistent search during the simulation, while the gradient-based controller in (Sugimoto et al., 2015) occasionally fails to meet the level. Finally, we implement the present control algorithm including not only the constrained-based controller but also an object detection algorithm and takeoff from/landing to the charging stations on a testbed with three drones.

The contributions of this paper are summarized as follows: 1) a novel constraint-based controller is presented so that a prescribed

performance level is maintained, differently from the gradient-based persistent coverage algorithm (Hübel et al., 2008; Sugimoto et al., 2015), constraint-based coverage algorithms (Santos et al., 2019), and other related algorithms (Franco et al., 2015; Palacios-Gasós et al., 2016; Wang and Wang, 2017), 2) a novel object search/surveillance problem is formulated, wherein not only the persistent coverage, safety certificates and energy persistency in (Egerstedt et al., 2018; Santos et al., 2019) but also task switches between search and surveillance are integrated, and 3) the algorithm is demonstrated through experiments, where we put the vision data and associated image processing in the loop while other related publications purely examine only robot motion (Schwager et al., 2011; Sugimoto et al., 2015; Egerstedt et al., 2018; Funada et al., 2019; Santos et al., 2019).

A part of the contents in this paper is presented in the conference version (Dan et al., 2020). The incremental contributions relative to (Dan et al., 2020) are: 4) we implement the present partially distributed control architecture on Robot Operating System (ROS), while the experimental setup in (Dan et al., 2020) took a centralized control architecture, 5) owing to the contribution 4), we increase the number of drones from two to three in the experiment, and 6) we newly add simulation to precisely check if the performance is guaranteed in the absence of uncertain factors in real experiments.

2 PRELIMINARY: CONTROL BARRIER FUNCTION

In this section, we present the notion of control barrier functions that play a central role in this paper. Let us consider a control affine system formulated as

$$\dot{p} = f(p) + g(p)u, \tag{1}$$

where $p \in \mathbb{R}^N, u \in \mathcal{U} \subseteq \mathbb{R}^M$, and vector fields f, g are assumed to be Lipschiz continuous. Suppose now that there exists a unique solution $p(t)$ on $[t_0, t_1]$ to (1). A set $\mathcal{S}$ is then said to be *forward invariant* with respect to system (1) if for every $p(t_0) \in \mathcal{S}$, the inclusion $p(t) \in \mathcal{S}$ holds for all $t \in [t_0, t_1]$ (Ames et al., 2017).

Define the Control Barrier Function (CBF) as below.

Definition 1. *Let h: $\mathcal{D} \subset \mathbb{R}^N \to \mathbb{R}$ be a continuously differentiable function and the set $\mathcal{C}$ is defined as $\mathcal{C} := \{p \in \mathbb{R}^N | h(p) \geq 0\}$. Then, h is said to be a CBF for system* (1) *if there exists a locally Lipschitz extended class $\mathcal{K}$ function α such that*

$$\sup_{u \in \mathcal{U}} \left[L_f h(p) + L_g h(p) u + \alpha(h(p)) \right] \geq 0, \tag{2}$$

for all x in the set $\mathcal{C}$, where $L_f h(10)$ and $L_g h(10)$ represent Lie derivative of h in the vector fields f and g, respectively.

It is shown that if h is a CBF, then the set $\mathcal{C}$ is forward invariant (Ames et al., 2017). If the set $\mathcal{C}$ consists of the states that ensure safety, 2) means that there always exists input signal u such that the state p is enforced to be inside of $\mathcal{C}$, namely safety is always ensured as long as the function h that characterizes $\mathcal{C}$ is a CBF.

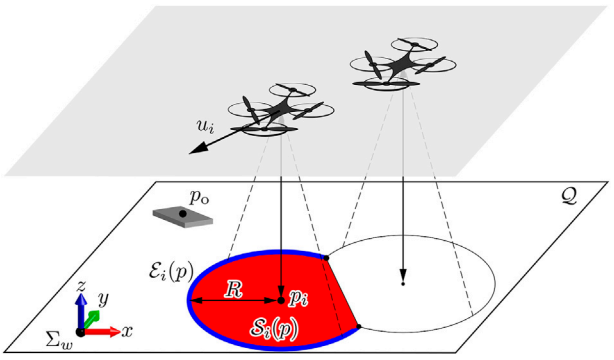

FIGURE 2 | Illustration of the problem setting with the field $\mathcal{Q}$, the world frame Σ_w, drones on the plane (gray plane), drone i's sensing region $\mathcal{S}_i$ (red region), its inner edge $\mathcal{E}_i$ (blue curve), and target object.

We next present an extension of CBF to the case where the set $\mathcal{C}$ is time varying. Consider the following set defined by a continuously differentiable function h: $\mathbb{R}^N \times \mathbb{R}_{\geq t_0} \to \mathbb{R}$,

$$\mathcal{C}(t) := \{p \in \mathbb{R}^N \mid h(p,t) \geq 0\}. \tag{3}$$

It is shown that the forward invariance of the set $\mathcal{C}(t)$ can be ensured with so-called time-varying CBF defined as follows (Lindemann and Dimarogonas, 2019; Notomista and Egerstedt, 2021).

Definition 2. *Given a dynamical system (1) and a set $\mathcal{C}(t)$ defined in* **Eq. 3**, *the function h is time-varying CBF defined on $\mathcal{D} \times \mathbb{R}_{\geq t_0}$ with $\mathcal{C}(t) \subseteq \mathcal{D} \subset \mathbb{R}^N$, if there exists a locally Lipschitz extended class $\mathcal{K}$ function α such that $\forall p \in \mathcal{D}$ and $\forall t \in [t_0, t_1]$,*

$$\sup_{u \in \mathcal{U}} \left[\frac{\partial h}{\partial t} + L_f h(p,t) + L_g h(p,t) u + \alpha(h(p,t)) \right] \geq 0$$

holds.

3 PROBLEM SETTING

Let us consider a 3-D space including n drones to be controlled and a ground modelled by a 2-D plane as illustrated in **Figure 2**. Without loss of generality, we arrange the world frame Σ_w so that its origin is on the ground, and its (x, y)-plane is parallel to the ground. The subset of the (x, y)-coordinates on the ground to be monitored is called *field*, and denoted by a compact set $\mathcal{Q} \subset \mathbb{R}^2$. It is assumed that a target object to be surveilled by the drones may be on the field, and its 2-D position is denoted by $p_o = [x_o \ y_o]^T \in \mathcal{Q}$. We assume no prior knowledge about not only the position p_o but also whether the object exists or not. We then define the persistent object search and surveillance mission by the following two subtasks:

- *Persistent search*: Drones patrol the entire field persistently to search the object.

- *Object surveillance*: Drones keep monitoring the object once the object is found through the persistent search.

These subtasks should be appropriately switched depending on whether the object is detected or not.

Let us denote the set of identifiers of n drones by $\mathcal{I} = \{1, \ldots, n\}$. The x, y, and z coordinates of drone i in Σ_w are denoted by x_i, y_i, and z_i, respectively. In this paper, each drone is assumed to be locally controlled so that the altitude z_i is constant and common among all drones $i \in \mathcal{I}$. We thus mainly focus on the 2-D motion of $p_i = [x_i \ y_i]^T \in \mathcal{Q}$. Each drone $i \in \mathcal{I}$ is assumed to follow the kinematic model:

$$\dot{p}_i = u_i, \ u_i \in \mathcal{U} \subseteq \mathbb{R}^2, \tag{4}$$

where u_i is the velocity input to be designed. Throughout this paper, we assume that p_i is available for control of drone i. Remark that the constant and common altitudes are assumed in order to highlight the main issue to be addressed in this paper. It is actually possible to handle full 3-D motion of the drones, e.g., by taking the formulation of (Funada et al., 2019) at the cost of the computational simplicity.

We next present an external sensor and network models for the drones. Every drone is assumed to be equipped with a single onboard camera that captures the ground. We suppose that the optical axis of each camera is perpendicular to the ground, and that the field of view of camera i is modeled by a circle

$$\mathcal{B}_i(p_i) = \{q \in \mathcal{Q} \mid \|q - p_i\| \leq R\}$$

for a sensing radius $R > 0$. Let us now introduce the Voronoi partition of the field $\mathcal{Q}$ (Cortés et al., 2005), which means the collection of the following sets for all $i \in \mathcal{I}$:

$$\mathcal{V}_i(p) = \left\{q \in \mathcal{Q} \mid \|q - p_i\| \leq \|q - p_j\|, \ \forall j \in \mathcal{I} \setminus \{i\}\right\}.$$

Using the above sets, we define the feasible sensing area $\mathcal{S}_i(p)$ by so-called r-limited Voronoi cell (Martínez et al., 2007) defined by

$$\mathcal{S}_i(p) := \mathcal{B}_i(p_i) \cap \mathcal{V}_i(p_i),$$

where p is the collection of $p_1, p_2, \ldots, p_n$. For convenience of the subsequent discussions, we also define the following set called inner edge of the set $\mathcal{S}_i(p)$ (Sugimoto et al., 2015).

$$\mathcal{E}_i(p) := \{q \in \mathcal{S}_i \mid \|q - p_i\| = R\}.$$

We also assume an inter-drone network such that drone i and j can exchange messages if their distance $\|p_i - p_j\|$ is smaller than or equal to $2R$. It is then well-known that the set $\mathcal{S}_i(p)$ is computable in a distributed fashion (Cortés et al., 2005). We also assume that each drone can detect the object when the object is inside of the sensing area $\mathcal{B}_i(p)$, and define a binary variable

$$\Delta_i := \begin{cases} 1 & p_o \in \mathcal{B}_i(p) \\ 0 & \text{otherwise.} \end{cases}$$

When $\Delta_i = 1$ holds, drone i can compute the position of the object p_o by the detection result and the geometric relation. In real applications, drones need to install an algorithm for detecting the object in the sensing area. See **Section 6** for more details on how to detect the object.

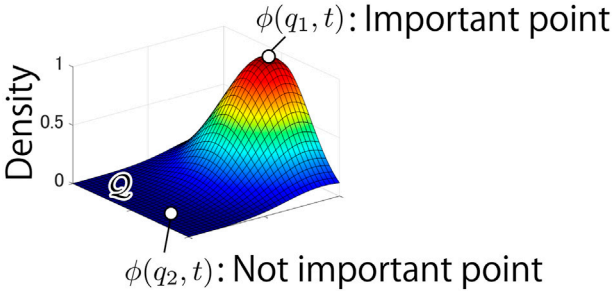

FIGURE 3 | Example of distribution of density function $\phi(q, t)$. The point q_1 is more important point, which need to be monitored than the point q_2.

In this paper, we implicitly assume that the collection of the fields of view $\mathcal{B}_i(p_i)$ for all $i \in \mathcal{I}$ is not wide enough to fully cover the field $\mathcal{Q}$. The goal of persistent search is then to let the drones persistently patrol the field $\mathcal{Q}$, while preventing any subregion in $\mathcal{Q}$ from being uncovered. To address the issue, the authors' antecessors (Hübel et al., 2008; Sugimoto et al., 2015) presented a gradient ascent algorithm-based controller for the following objective function to be maximized:

$$J(p,t) := -\sum_{i=1}^{n} \int_{\mathcal{S}_i} \|q - p_i\|^2 \phi(q,t) dq + b \int_{\mathcal{Q} \setminus \cup_{i=1}^{n} \mathcal{S}_i} \phi(q,t) dq \qquad (b \leq -R^2), \tag{5}$$

The function $\phi: \mathcal{Q} \times \mathbb{R}_{\geq t_0} \to [0,1]$, called density function, enables one to mark the important points in the field, as illustrated in **Figure 3**. The papers (Hübel et al., 2008; Sugimoto et al., 2015) presented a novel update rule of the function ϕ formulated by

$$\frac{d\phi(q,t)}{dt} = \begin{cases} -\underline{\delta}\phi(q,t), & \text{if } q \in \cup_{i=1}^{n} \mathcal{S}_i \\ \bar{\delta}(1 - \phi(q,t)), & \text{otherwise.} \end{cases} \quad (\bar{\delta}, \underline{\delta} > 0). \tag{6}$$

Eq. 6 means that importance of point q monitored by at least one drone decays while that of point q such that $q \notin \mathcal{B}_i(p_i) \ \forall i \in \mathcal{I}$ increases. In view of the nature of the gradient-based coverage, it tends to deliver robots to positions with high density, drones are expected to repeatedly visit all of uncovered regions, which is close to the objective of the persistent search mission in this paper. The control algorithm with **Eq. 6** is actually demonstrated through experiments in (Sugimoto et al., 2015). However, the gradient-based controller does not ensure any guarantee on the performance quantified by $J(p, t)$.

In order to certify the search performance, we formally define the objective of the persistent search as below.

Definition 3. *Let a function* $h_J: \mathcal{Q}^n \times \mathbb{R}_{\geq t_0} \to \mathbb{R}$ *be*

$$h_J(p,t) := J(p,t) - \gamma,$$

where γ *is a negative real constant. The drones are then said to achieve* γ*-level persistent search, if*

$$h_J(p(t),t) \geq 0 \tag{7}$$

holds for all $t \geq t_0$ with a given initial time t_0.

Remark that a similar concept is also investigated in (Franco et al., 2015; Palacios-Gasós et al., 2016). It is an extension of the concept to the time-varying objective function.

Let us next consider the object surveillance that should be performed only when Δ_i takes the value of 1. Define the function $h_{i,\mathrm{sur}}\colon \mathcal{Q} \to \mathbb{R}$ as:

$$h_{i,\mathrm{sur}}(p_i) = -\|p_i - p_o\|^2 + d_{\mathrm{sur}}^2.$$

Assuming $R > d_{\mathrm{sur}} > 0$, the object must be inside of the field of view $\mathcal{B}_i$ if

$$h_{i,\mathrm{sur}}(p_i) \geq 0 \tag{8}$$

holds. It is also fully expected that **Eq. 8** holds at the time when Δ_i switches from 0 to 1. The goal of the object surveillance is thus to keep meeting (8) during the period with $\Delta_i = 1$.

Besides the above subtasks, we need to meet the following specifications in order to ensure persistency in real operations.

- *Safety*: Drones avoid collisions with each other.
- *Energy persistency*: Drones return to their charging stations before their batteries run out.

If either of the above two would not be satisfied, drones would no longer continue the search and surveillance mission. In this sense, we should place a higher priority on these specifications than 7) and (8). Remark that the subsequent formulations follow the manner of (Egerstedt et al., 2018; Santos et al., 2019).

In order to formulate the specification for safety, let us first define the function:

$$h_{i,\mathrm{avd}}(p) = \|p_i - p_{i,\mathrm{near}}\|^2 - d_{\mathrm{avd}}^2,$$

where $p_{i,\mathrm{near}}$ denotes the position of the drone nearest to drone i within the radius $2R$, and d_{avd} is selected so that $d_{\mathrm{avd}} > 0$. Then, drone i keeps the distance from all other drones greater than d_{avd} if

$$h_{i,\mathrm{avd}}(p) \geq 0. \tag{9}$$

holds. Accordingly, collisions are avoided as long as d_{avd} is selected to be large enough and 9) is satisfied.

We finally formulate the condition for energy persistency. To this end, the state of charge for drone i, denoted by E_i, is assumed to obey

$$\dot{E}_i = -K_{\mathrm{chg}} \quad (K_{\mathrm{chg}} > 0).$$

We then assume that there is a minimum energy level $E_{\min}$, that is, $E_i \geq E_{\min}$ must hold during the mission. Also, charging stations are assumed to be located on the ground, where the center of the station assigned to drone i is denoted by $\hat{p}_i$. For simplicity, we leave the landing and takeoff motion out of consideration, and assume that the battery is recharged as long as $\|p_i - \hat{p}_i\| \leq d_{\mathrm{chg}}$. Let us now define the function

$$h_{i,\mathrm{chg}}(p_i,E_i) = E_i - E_{\min} - \frac{K_{\mathrm{chg}}}{k_{\mathrm{chg}}}(\|p_i - \hat{p}_i\| - d_{\mathrm{chg}}) \quad (k_{\mathrm{chg}} > 0).$$

Note that the positive constant k_{chg} should be selected so that $(K_{\mathrm{chg}}/k_{\mathrm{chg}})(\|p_i - \hat{p}_i\| - d_{\mathrm{chg}})$ is greater than the battery needed for returning to the station from the position p_i. Then, if the condition

$$h_{i,\mathrm{chg}}(p_i,E_i) \geq 0, \tag{10}$$

is always satisfied, the state of charge for drone i is never exhausted before arriving at the station.

In summary, two subtasks, persistent search and object surveillance, and two specifications, safety and energy persistency, are formulated in the form of the constraint functions (7)–(10), respectively. The control goal for the persistent object search and surveillance mission is to design the control inputs that satisfy the inequalities (7)–(10).

4 CONSTRAINT-BASED CONTROLLER

In this section, we present a constraint-based controller to meet (7)–(10) that are possibly conflicting with each other. To this end, we first focus on **Eq. 7** for the γ-level persistent search in Definition 3.

Now, the time derivative of the function h_J along with the trajectories of system 4) is given as

$$\dot{h}_J = \frac{\partial J(p,t)}{\partial t} + \sum_{i=1}^{n}\left(\frac{\partial J}{\partial p_i}\right)^T u_i.$$

The first term in the right hand side of the equation is rewritten as below according to (Diaz-Mercado et al., 2017) and (6).

$$\begin{aligned}\frac{\partial J(p,t)}{\partial t} &= -\sum_{i=1}^{n}\int_{\mathcal{S}_i}\|q-p_i\|^2\frac{\partial\phi(q,t)}{\partial t}dq + b\int_{\mathcal{Q}\setminus\cup_{i=1}^{n}\mathcal{S}_i}\frac{\partial\phi(q,t)}{\partial t}dq \\ &= \sum_{i=1}^{n}\underline{\delta}\int_{\mathcal{S}_i}\|q-p_i\|^2\phi(q,t)dq + b\int_{\mathcal{Q}\setminus\cup_{i=1}^{n}\mathcal{S}_i}\bar{\delta}(1-\phi(q,t))dq.\end{aligned} \tag{11}$$

The second term can be expressed as

$$\begin{aligned}b\int_{\mathcal{Q}\setminus\cup_{i=1}^{n}\mathcal{S}_i}\bar{\delta}(1-\phi(q,t))dq = \sum_{i=1}^{n} b\Bigg\{\frac{1}{n}\int_{\mathcal{Q}}\bar{\delta}(1-\phi(q,t))dq \\ -\int_{\mathcal{S}_i}\bar{\delta}(1-\phi(q,t))dq\Bigg\}.\end{aligned} \tag{12}$$

In the same way as (12), h_J is also rewritten as

$$\begin{aligned}h_J = \sum_{i=1}^{n}\Bigg\{-\frac{\gamma}{n} - \int_{\mathcal{S}_i}\|q-p_i\|^2\phi(q,t)dq + \frac{b}{n}\int_{\mathcal{Q}}\phi(q,t)dq \\ -b\int_{\mathcal{S}_i}\phi(q,t)dq\Bigg\}.\end{aligned} \tag{13}$$

Combining (11)–(13), we find that

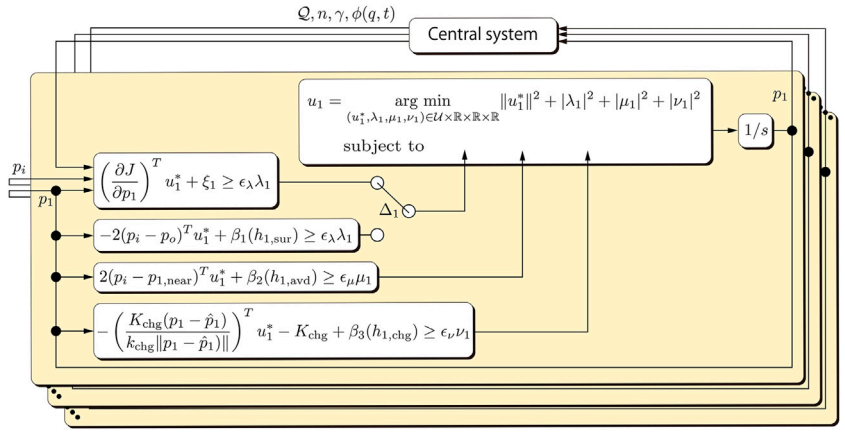

FIGURE 4 | Block diagram of the partially distributed control architecture.

$$\dot{h}_J + kh_J = \frac{\partial J(p,t)}{\partial t} + \sum_{i=1}^{n}\left(\frac{\partial J}{\partial p_i}\right)^T u_i + kh_J = \sum_{i=1}^{n}\left\{\left(\frac{\partial J(p,t)}{\partial p_i}\right)^T u_i + \xi_i(p,t)\right\} \quad (k>0),$$

where

$$\xi_i(p,t) := (\underline{\delta} - k)\int_{\mathcal{S}_i}\|q - p_i\|^2\phi(q,t)dq + b\left\{\frac{1}{n}\int_{\mathcal{Q}}(\bar{\delta} + (k-\bar{\delta})\phi(q,t))dq - \int_{\mathcal{S}_i}(\bar{\delta} + (k-\bar{\delta})\phi(q,t))dq\right\} - \frac{k}{n}\gamma.$$

Assume that there exists a controller for each agent: $u_i = K_i(p,t)$: $\mathcal{Q}^n \times \mathbb{R}_{\geq t_0} \to \mathcal{U}$ that is locally Lipschitz in $p \in \mathcal{Q}^n$, continuous in $t \in [t_0, t_1]$, and satisfies

$$K_i(p,t) \in \mathcal{K}_i(p,t) := \left\{u_i \in \mathcal{U} \middle| \left(\frac{\partial J(p,t)}{\partial p_i}\right)^T u_i + \xi_i(p,t) \geq 0\right\},$$

$\forall p \in \mathcal{Q}^n$ and $\forall t \in [t_0, t_1]$. This means that the function h_J is a time-varying CBF defined on $\mathcal{Q}^n \times \mathbb{R}_{\geq t_0}$ with extended class $\mathcal{K}$ function $\beta_0(s) = ks$. Lemma 1 in (Notomista and Egerstedt, 2021) then ensures that the controller guarantees forward invariance of the set

$$\mathcal{C}_0(t) := \{p \in \mathcal{Q}^n \mid h_J(p,t) \geq 0\}.$$

Then, the definition of the forward invariance means γ-level persistent search for any initial condition inside of the set $\mathcal{C}_0(0)$. In the case of $\mathcal{U} = \mathbb{R}^2$, $\mathcal{K}_i(p,t) = \varnothing$ happens only if $\frac{\partial J(p,t)}{\partial p_i} = 0$. The gradient is now equivalent to the control law in (Sugimoto et al., 2015), wherein $\frac{\partial J(p,t)}{\partial p_i} = 0$ means that the robot stops at a point. Through extensive simulations and experiments, we have never observed such a scene and it is fully expected that $\mathcal{K}_i(p,t) \neq \varnothing$ in practice, which is demonstrated through simulation in **Section 5**.

Remark also that the above discussions require that the initial state is selected in the set $\mathcal{C}_0(0)$, and do not ensure recovery of the level from an initial condition outside of $\mathcal{C}_0(0)$, namely $h_J(p,t) \geq 0$ for some $t \geq t_0$ from an initial condition with $h_J(p(t_0), t_0) < 0$. In the case of time-invariant CBFs, the recovery is rigorously proved in (Ames et al., 2017). The result is not trivially extended to the time-varying CBFs. It is however exemplified in Dan et al. (2020) that the recovery is achieved even for the time-varying case in practice.

Let us next consider the satisfaction of **Eqs 8–10**. It is known that $h_{i,\text{sur}}$, $h_{i,\text{avd}}$ and $h_{i,\text{chg}}$ are all CBFs (Egerstedt et al., 2018; Notomista et al., 2018; Notomista and Egerstedt, 2021). According to Definition 1, we thus formulate the inequality constraints for ensuring (8)–(10) as:

$$\left(\frac{\partial h_{i,\text{sur}}}{\partial p_i}\right)^T u_i + \beta_1(h_{i,\text{sur}}(p_i)) = -2(p_i - p_o)^T u_i + \beta_1(h_{i,\text{sur}}(p_i)) \geq 0. \tag{14}$$

$$\left(\frac{\partial h_{i,\text{avd}}}{\partial p_i}\right)^T u_i + \beta_2(h_{i,\text{avd}}(p_i)) = 2(p_i - p_{i,\text{near}})^T u_i + \beta_2(h_{i,\text{avd}}(p_i)) \geq 0, \tag{15}$$

$$\left(\frac{\partial h_{i,\text{chg}}}{\partial p_i}\right)^T u_i - K_{\text{chg}} + \beta_3(h_{i,\text{chg}}(p_i, E_i)) = -\left(\frac{K_{\text{chg}}(p_i - \hat{p}_i)}{k_{\text{chg}}\|p_i - \hat{p}_i\|}\right)^T u_i - K_{\text{chg}} + \beta_3(h_{i,\text{chg}}(p_i, E_i)) \geq 0, \tag{16}$$

with locally Lipschitz extended class $\mathcal{K}$ functions β_1, β_2, β_3 respectively. By definition of CBFs, if we take the controller $u_i = K_i(p,t)$ such that

$$K_i(p,t) \in \mathcal{K}_i(p,t) := \left\{u_i \in \mathcal{U}^n \middle| \left(\frac{\partial J(p,t)}{\partial p_i}\right)^T u_i^* + \xi_i(p,t) \geq 0, (14), (15), (16)\right\},$$

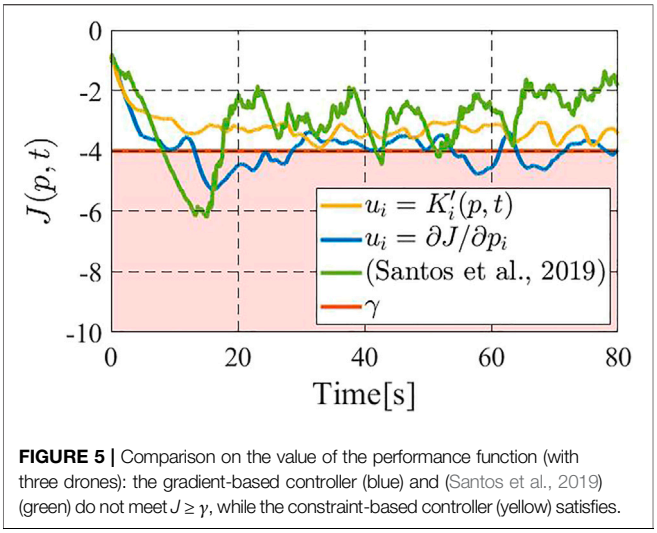

FIGURE 5 | Comparison on the value of the performance function (with three drones): the gradient-based controller (blue) and (Santos et al., 2019) (green) do not meet $J \geq \gamma$, while the constraint-based controller (yellow) satisfies.

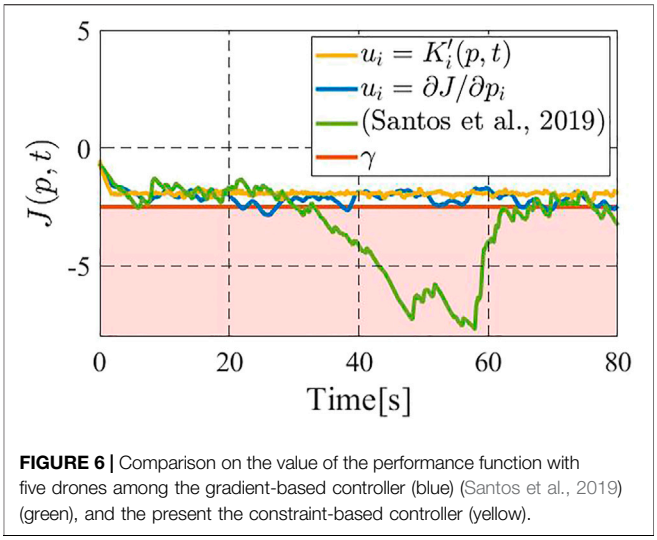

FIGURE 6 | Comparison on the value of the performance function with five drones among the gradient-based controller (blue) (Santos et al., 2019) (green), and the present the constraint-based controller (yellow).

all of **Eqs 7–10** are satisfied. However, due to the conflicts among the specifications, the controller set $\mathcal{K}_i(p,t)$ can be empty.

To address the above issue, we prioritize the specifications, which can be realized by relaxing some of the constraints. It is now immediate to see that 7) and 8) are never met in practice if the safety constraint 9) or energy constraint (10) is violated. Accordingly to the insight, we propose the following controller $u_i = K_i(p, t)$:

$$K_i(p,t) = \underset{(u_i^*,\lambda_i,\mu_i,\nu_i)\in\mathcal{U}\times\mathbb{R}\times\mathbb{R}\times\mathbb{R}}{\arg\min} \|u_i^*\|^2 + |\lambda_i|^2 + |\mu_i|^2 + |\nu_i|^2 \tag{17a}$$

$$\text{s.t.} \quad (1-\Delta_i)\left\{\left(\frac{\partial J(p,t)}{\partial p_i}\right)^T u_i^* + \xi_i(p,t)\right\} + \Delta_i\left\{-2(p_i - p_o)^T u_i^* + \beta_1(h_{i,\text{sur}}(p_i))\right\} \geq \epsilon_\lambda \lambda_i, \tag{17b}$$

$$2(p_i - p_{i,\text{near}})^T u_i^* + \beta_2(h_{i,\text{avd}}(p_i)) \geq \epsilon_\mu \mu_i, \tag{17c}$$

$$-\left(\frac{K_{\text{chg}}(p_i - \hat{p}_i)}{k_{\text{chg}}\|p_i - \hat{p}_i\|}\right)^T u_i^* - K_{\text{chg}} + \beta_3(h_{i,\text{chg}}(p_i, E_i)) \geq \epsilon_\nu \nu_i, \tag{17d}$$

where the weights ϵ_λ, ϵ_μ, and ϵ_ν are non-negative scalars. The slack variables λ_i, μ_i, ν_i allow the violations of the associated constraints, and the corresponding weights adjust the penalty on the individual constraint violations. When one of the weights takes a value smaller than other weights, then the controller tries to satisfy the corresponding constraint more strictly than others. When the weight is equal to zero, then the controller treats the constraint as a hard constraint. In this paper, we arrange the weights so that $\epsilon_\lambda \gg \epsilon_\mu, \epsilon_\nu$ in order to prioritize safety and energy persistency over the control goals of the subtasks. If the weights ϵ_λ, ϵ_ν, ϵ_μ are all positive or only one of ϵ_μ and ϵ_ν is equal to zero, then the optimization problem in (17) is ensured to be feasible as long as (9) and (10) are satisfied at the initial time t_0.

We finally show that the present controller is implementable in a (partially) distributed manner. The gradient $\partial J(p, t)/\partial p_i$ in **Eq. 17b** is known to be rewritten as follows (Cortés et al., 2005):

$$\frac{\partial J(p,t)}{\partial p_i} = 2\text{mass}(\mathcal{S}_i(p))(\text{cent}(\mathcal{S}_i(p)) - p_i) - (R^2 + b)\int_{\mathcal{E}_i} \frac{q - p_i}{\|q - p_i\|}\phi(q,t)dq,$$

where

$$\text{mass}(\mathcal{S}_i(p)) := \int_{\mathcal{S}_i} \phi(q,t)dq,$$

$$\text{cent}(\mathcal{S}_i(p)) := \frac{1}{\text{mass}(\mathcal{S}_i(p))}\int_{\mathcal{S}_i} q\phi(q,t)dq.$$

As mentioned before, the sets $\mathcal{S}_i$ and $\mathcal{E}_i$ can be locally computed under the network assumed in **Section 3**. In addition, 17) consists only of local variables/parameters excluding $\mathcal{Q}$, n, γ, and $\phi(q, t)$ in $\mathcal{S}_i$. In other words, if the field $\mathcal{Q}$, the number of drones n, and desired performance level γ are shared by the drones, and the density function $\phi(q, t)$ in $\mathcal{S}_i$ would be given, each drone i can locally solve the optimization problem **Eq. 17**. It should be now noted that, as assumed in (Hübel et al., 2008; Sugimoto et al., 2015), the density update 6) must be inherently executed by a central system since each drone hardly knows if other drones visited each $q \in \mathcal{Q}$ in the past. The overall control architecture is then illustrated in **Figure 4**. The comprehensive algorithm for drone i including landing/takeoff motion and object detection is informally described as **Algorithm 1**, where $E_{\max}$ means the battery level at which drones stop charging, and $E_{\min}$ is the level at which each drone starts landing.

Remark 1. The computation in the density update (6) left to the central computer is almost scalable with respect to the number of drones, while solving the optimization problems (17) for all i at a central computer is not scalable. It is thus fully expected that the present partially distributed architecture works even for large-scale drone networks.

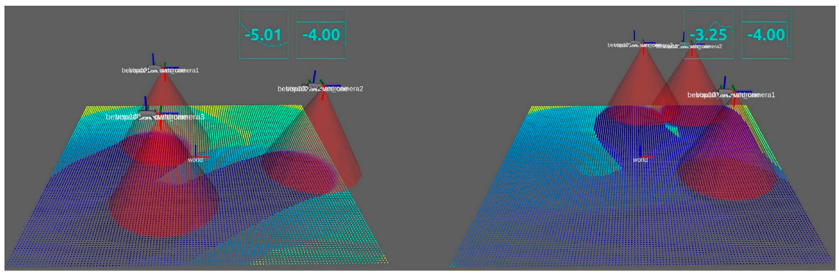

FIGURE 7 | Snapshots at time $t = 17$ s: the constraint-based controller **(right)** almost fills the entire field with blue, while some regions remain yellow for the gradient-based controller **(left)**.

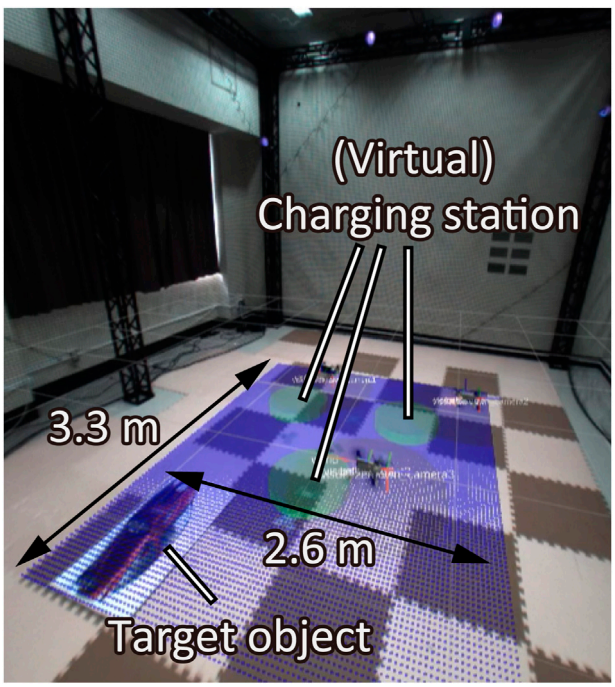

FIGURE 8 | Experiment room: Overview of the environment.

Nevertheless, some readers may have a concern about using a central computer itself. In many practical applications, however, the communication infrastructure between drones and a central system is established so that a person at the monitoring center monitors the data acquired by the drones. Thus, assuming the computational supports from the central computer must be reasonable in such application scenarios.

Remark 2. Santos et al. (2019) addressed coverage control with a time varying density function using time-varying CBFs, which is close to the present approach. The contribution of this paper relative to (Santos et al., 2019) is as follows. The controller presented in (Santos et al., 2019) is designed based on the distance between the current robot position and the centroid of the Voronoi cell. However, the relation between the metric and the coverage performance quantified by the objective function is not always obvious. On the other hand, the presented controller The switches between subtasks are also not investigated in (Santos et al., 2019).

Algorithm 1. Algorithm for drone i.

loop
  Get local information and own state.
  if Drone is on the ground. **then**
    if $E_i \geq E_{\max}$ **then**
      Start the take-off action.
    else
      Charge the battery.
    end if
  else if $E_i \leq E_{\min} \cap \|p_i - \hat{p}_i\| \leq d_{\text{chg}}$ **then**
    Start landing action.
  else
    Determine $\Delta_i \in \{0, 1\}$ by an object detector.
    Input $u_i = K_i(p, t)$.
  end if
end loop

5 SIMULATION

In this section, we focus only on the persistent search mission while ignoring other objectives, object surveillance, safety, and energy persistency. We then verify through simulation that the constrained-based controller achieves the performance specified by the parameter γ. To this end, we employ the simplified version of the controller (17):

$$K_i'(p,t) = \arg\min_{u_i^* \in \mathbb{R}^2} \|u_i^*\|^2 \tag{18a}$$

$$\text{s.t.} \left(\frac{\partial J(p,t)}{\partial p_i}\right)^T u_i^* + \xi_i(p,t) \geq 0. \tag{18b}$$

In the simulation, the field is set to $\mathcal{Q} = [-2, 2]\text{m} \times [-2, 2]\text{m}$. We also let $n = 3$ and the initial positions be selected as $p_1 = [-1\ 0]$

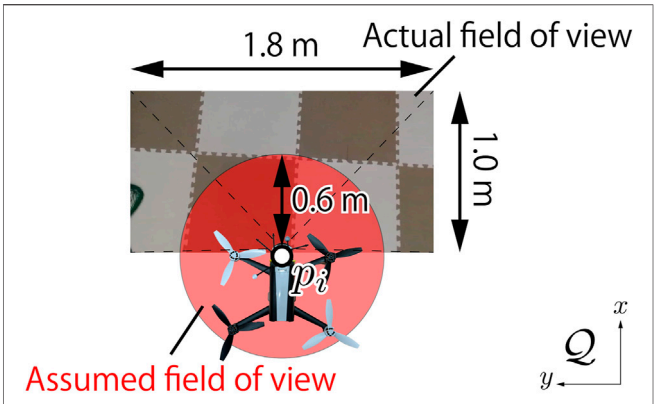

FIGURE 9 | Definition of the field of view for a drone.

^{T}m, $p_2 = [1\ 0]^T$m, and $p_3 = [1\ 1]^T$m. The altitude z_i and radius R of every drone are set to 1.2 and 0.6 m mimicking the experimental testbed that will be presented in the next section. Under the setting, we run the constraint-based controller 18) with $\gamma = -4.0$, and compare the performance with the gradient-based controller (Sugimoto et al., 2015), namely $u_i = \kappa \partial J(p, t)/\partial p_i$ with $\kappa = 5.0$ and (Santos et al., 2019). In all of the controllers, we take $\bar{\delta} = 0.05$, $\underline{\delta} = 1.0$ and $b = -1.0$. Remark now that (Santos et al., 2019) does not consider limitation of the sensing radius, but we impose the same limitation as the other two methods by just changing Voronoi cells to r-limited ones in order to fairly compare the methods. The gradients of the centroids of the r-limited Voronoi cells needed for implementing (Santos et al., 2019) are numerically computed.

Figure 5 shows the time responses of the performance function J for the above two methods, where the blue line shows the performance by the gradient-based controller (Sugimoto et al., 2015), the green line that by (Santos et al., 2019) the yellow line that by the constraint-based controller (18), and the red line illustrates the prescribed performance level $\gamma = -4.0$. We see that the gradient-based controller (Sugimoto et al., 2015) and (Santos et al., 2019) occasionally fail to meet the desired performance level, namely the value of performance function J goes below γ. On the other hand, the constraint-based controller 18) successfully keeps the performance above the level $\gamma = -4.0$. **Figure 6** illustrates the results for $n = 5$, wherein we take $\gamma = -2.5$ to highlight the differences between the present controller and the other two. It is immediate to see that the above insights from **Figure 5** are also applied to this case. It is now to be noted that, if we remove the density update 6) from consideration, the controller in (Santos et al., 2019) is itself fully distributed, while the present constraint-based controller still needs partial support from a central computer. However, in the present scene, 6) needs to be executed in a central computed regardless of the control algorithm, as mentioned in **Section 4**.

Figure 7 shows the snapshot of simulation in **Figure 5** at $t = 17$ s, where the left and right figure correspond to the gradient-based controller (Sugimoto et al., 2015) and the constraint-based controller (18), respectively. The color map on the field illustrates the value of the density function $\phi(q, t)$, where the yellow region has high density while the dark blue means low density. We immediately see from the definition of J in **Eq. 5** that low density is directly linked with a good search performance. In the left, some areas remain yellow while, in the right, the entire area is almost filled with blue. It is thus concluded that the constraint-based controller 18) achieves a better performance than the gradient-based controller.

Remark that if we take a larger gain κ, then the gradient-based controller tends to achieve a better performance and may even meet the prescribed performance level. Even through that, the performance level is not rigorously ensured and, more importantly, it is hard to know an appropriate gain for given environment and parameters in advance. Of course, taking a too large feedback gain may result in unstable motion in real implementation.

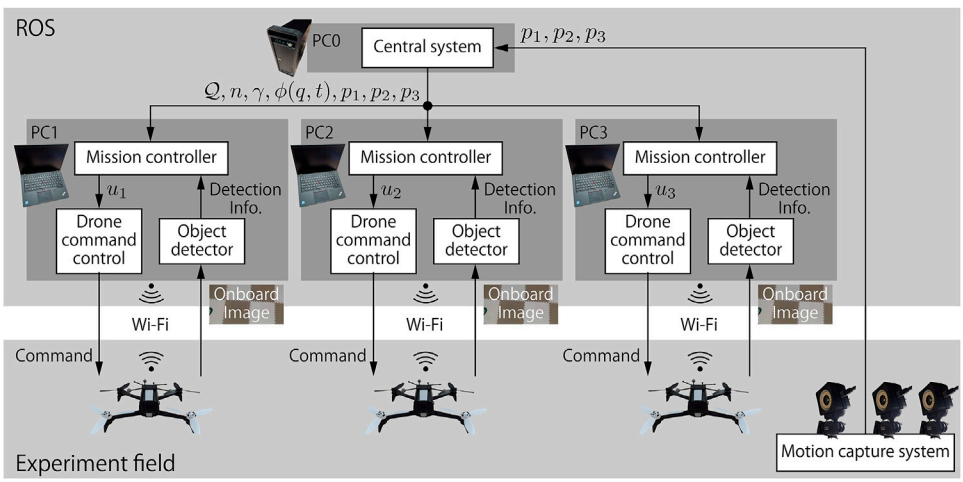

FIGURE 10 | Experiment systems: Schematic description of the system.

TABLE 1 | Parameter setting.

b	−1.0	γ	−2.0	$E_{\min}$	1,500	λ	0.05
K	0.3	d_{sur}	0.1	k_{chg}	0.15	ϵ_μ	0
δ	0.05	d_{avd}	0.5	d_{chg}	0.3	ϵ_ν	0.01
$\underline{\delta}$	1.0	K_{chg}	25	$E_{\max}$	4,000		

It is finally to be noted that the optimization problem in the controller has never been infeasible, namely the gradient $\frac{\partial J(p,t)}{\partial p_i}$ has never been equal to 0 throughout the simulation. Due to pathological cases, the function h_J has not been rigorously proved to be always a time-varying CBF, but it would not matter in practice.

6 EXPERIMENT

In this section, we demonstrate Algorithm 1 through experiments on a testbed. We set the field $\mathcal{Q}$ as a 3.3 m × 2.6 m ground plane as shown in **Figure 8**. We place a picture of a car on the field as the object to be surveilled. We also employ three Parrot Bebop2 drones ($n = 3$), whose onboard cameras capture the ground plane. We set the virtual charging stations, in which we suppose that drones can charge their batteries.

A local controller for each drone is designed so that its altitude is maintained to be 1.2 m and the body is parallel to the ground. When a drone takes the above desirable states, the field of view of the camera is given by about 1.8 m × 1.2 m rectangle as illustrated in **Figure 9**. In order to compensate the gap from the circular field

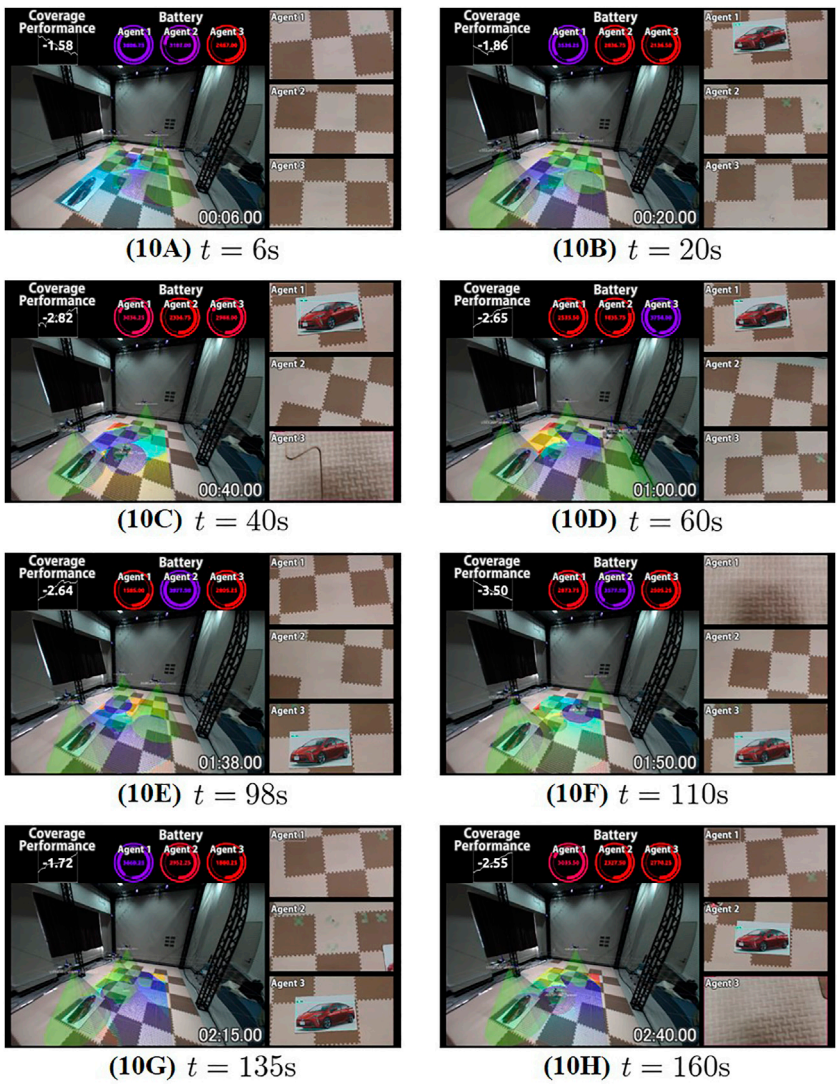

FIGURE 11 | Snapshots of the experiment, where the plane $\mathcal{Q}$ (rectangle), charging station (green cylinder) the shifted field of views (green cone) are overlaid. The current value of the performance function J, the each state of charge, and the onboard camera views of the drones are also tiled. Note that the shifted field of view and actual camera view do not match perfectly due to the differences shown in **Figure 9**. In **(C)**, **(F)**, and **(H)**, the drones land on the ground for charging. The object (picture of car) is monitored by one of the drones in all scene except **(A)**.

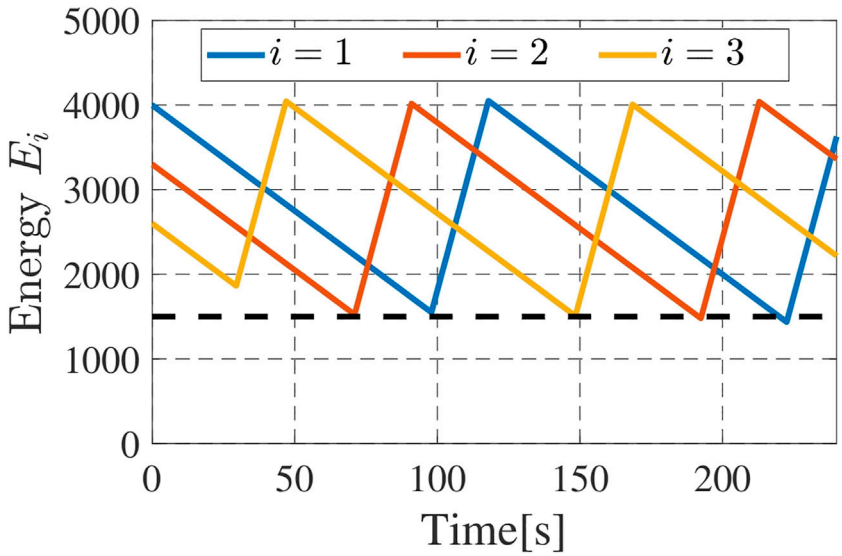

FIGURE 12 | Time series of E_i. Each drone recharges its battery before E_i reaches the minimum limit.

of view assumed in the previous sections, we set the red circle in **Figure 9** with radius 0.6 m inside of the rectangle while accepting conservatism. Also, the optical axis of the camera is not perpendicular to the body, which differs from the model in **Figure 2**. In order to fill the gap, the center of the circle is shifted from that of the rectangle. This shift does not matter in practice since the object position is also shifted in the sequel. Generalization of the algorithm that does not require such remedies is left as a future work.

The schematic of the testbed is illustrated in **Figure 10** which consists of a desktop computer, three laptops, and a motion capture system (OptiTrack) as well as drones. The motion capture measures the positions of drones at every 4.17 ms (240 fps). The desktop computer (PC0) receives all drones' positions from the motion capture system, updates the value of density function $\phi(q, t)$, and publishes the positions and the field information such as field size ($\mathcal{Q}$), the number of drones (n), performance target γ, and the current value of $\phi(q, t)$, to each laptop. Each laptop (PC1–3) implements the distributed controller $K_i(p, t)$, and outputs the velocity command u_i ($i = 1, 2, 3$) to be sent to each drone. The laptops are connected to individual drones by Wi-Fi communication. Each laptop receives the onboard camera images from the drone in real time. It then detects the object by using the `tensorflow object detector` (https://github.com/osrf/tensorflow_object_detector). The object position is computed by the detection result and the geometric relation, and then shifted to compensate the gap between the rectangle and red circle in **Figure 9**. The laptop then calculates the inputs u_i based on the information published by PC0 and the detected object position by *Python* script. The quadratic program in **Eq. 17** is solved in the script using `CVXOPT`. The input is converted into a suitable format for communication and sent to the drone. Note that each distributed controller needs the positions of not all drones but only the neighboring drones within the radius $2R = 1.2$ m. To mimic the real distributed computation, each laptop deletes drones' positions not within the radius, and does not use the information at all in the program.

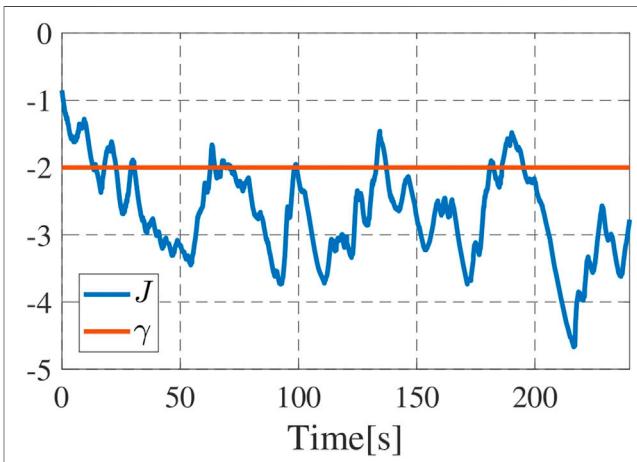

FIGURE 13 | Time series of J, where the blue line denotes the function J and red line does its target level γ.

The weight of constraints are given as follows:

$$\epsilon_\lambda > \epsilon_\nu > \epsilon_\mu = 0.$$

This means that the primary constraint is safety, namely the collision avoidance, and it is treated as a hard constraint. The secondary is the battery charging, and tertiary is the subtasks: the persistent search and object surveillance, which are treated as soft constraints. For the safety reason, we restrict the speed of drones by setting input space $\mathcal{U}$ as $[-0.3\ 0.3]$m/s $\times$ $[-0.3\ 0.3]$m/s. The other parameters needed for implementing **Algorithm 1** are listed in **Table 1**.

The snapshots of the experiment are shown in **Figure 11**. When the object is not detected and all drones' batteries have enough states

of charge, drones run the persistent search and move around over the plane $\mathcal{Q}$ (**Figure 11A**). In **Figure 11B**, drone 1 successfully detects the object. Accordingly, it switches to the subtask of the object surveillance, and stays above the object. Meanwhile, other drones continue the persistent search. We also see from **Figure 11C** that, when the battery level of drone 3 is low, it returns to and lands on the charging station. After charging, he restarts the subtasks (**Figure 11D**). Through the experiment, every drone autonomously repeats these actions depending on the situation (**Figures 11E–H**). It is to be emphasized that the drones never crash against each other through the experiment owing to the primary constraint.

Let us next confirm the function of the secondary constraint for the energy persistency. The time series of the (virtual) states of charge are shown in **Figure 12**. We see from the figure that the drones successfully return to the charging station, and recharge the battery before their batteries reach the minimum limit $E_{\min}$ shown by dashed line with slight exception at around $t = 225$s. Finally, **Figure 13** shows the time series data of the value of the function J. We see that the drones frequently failed to satisfy the performance level γ. This is fully reasonable since the collision avoidance, energy persistency and object surveillance are prioritized over the subtasks of persistent search. We see that the performance level is high in the early stage, where all drones engage in the persistent search as seen in **Figure 11A**. The performance decreases at around $t = 20$s since a drone switches to object surveillance (**Figure 11B**). The performance further decays around $t = 30$–40s since a drone goes back to the charging station and only one drone engages in persistent search (**Figure 11C**). Once the drone restarts persistent search (**Figure 11D**), the performance improves during $t = 60$–80 but it again decays at around $t = 80$s since another drone returns to the station. It is thus concluded that the present prioritization works as expected, and the present algorithm autonomously completes the overall mission.

7 CONCLUSION

In this paper, we have investigated a persistent object search and surveillance mission with safety certificates for drone networks. To address the issue, the control goals for the persistent object search and surveillance together with certificates for safety and energy persistency have been rigorously formulated in the form of constraint functions. To design a controller that fulfills the constraints, we have derived inequality constraints to be met by the control input, following the manner of CBFs. We then have presented a constraint-based controller that appropriately prioritizes constraints to manage conflicts among specifications. The simulation study has revealed that the constraint-based controller certifies a prescribed performance level for the searching mission, differently from the authors' antecessor and other related publications. The present algorithm has also been demonstrated through experiments. In the experiment, it has been confirmed that safety and energy persistency are successfully guaranteed by the controller even in the presence of a variety of uncertain factors in the real physical world, not in the ideal mathematical models. We have also observed through experiments that the present prioritization of the specifications works as expected, namely drones prioritize safety and energy persistency at the cost of the control goals for persistent object search and surveillance.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/**Supplementary Material**, further inquiries can be directed to the corresponding author.

AUTHOR CONTRIBUTIONS

All authors listed have made a substantial, direct, and intellectual contribution to the work and approved it for publication.

FUNDING

Japan Society for the Promotion of Science Grant-in-Aid for Scientific Research (C) 21K04104

ACKNOWLEDGMENTS

The authors would like to thank Tokyo Tech Academy for Super Smart Society for their supports to build the experimental testbed.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/frobt.2021.740460/full#supplementary-material

Supplementary Material | Experimental movie of the present constraint-based controller.

REFERENCES

Ames, A. D., Xu, X., Grizzle, J. W., and Tabuada, P. (2017). Control Barrier Function Based Quadratic Programs for Safety Critical Systems. *IEEE Trans. Automat. Contr.* 62, 3861–3876. doi:10.1109/TAC.2016.2638961

Bentz, W., Hoang, T., Bayasgalan, E., and Panagou, D. (2018). Complete 3-D Dynamic Coverage in Energy-Constrained Multi-UAV Sensor Networks. *Auton. Robot* 42, 825–851. doi:10.1007/s10514-017-9661-x

Cortés, J., Martínez, S., and Bullo, F. (2005). Spatially-distributed Coverage Optimization and Control with Limited-Range Interactions. *Esaim: Cocv* 11, 691–719. doi:10.1051/cocv:2005024

Dan, H., Yamauchi, J., Hatanaka, T., and Fujita, M. (2020). "Control Barrier Function-Based Persistent Coverage with Performance Guarantee and Application to Object Search Scenario," in Proceedings of IEEE Conference on Control Technology and Applications, Montreal, QC, August 24–26, 2020, 640–647. doi:10.1109/CCTA41146.2020.9206273

Diaz-Mercado, Y., Lee, S. G., and Egerstedt, M. (2017). "Human-Swarm Interactions via Coverage of Time-Varying Densities," in *Trends in Control*

and Decision-Making for Human–Robot Collaboration Systems. Editors Y. Wang and F. Zhang (Cham, Switzerland: Springer International Publishing), 357–385. doi:10.1007/978-3-319-40533-9_15

Egerstedt, M., Pauli, J. N., Notomista, G., and Hutchinson, S. (2018). Robot Ecology: Constraint-Based Control Design for Long Duration Autonomy. *Annu. Rev. Control.* 46, 1–7. doi:10.1016/j.arcontrol.2018.09.006

Franco, C., Stipanović, D. M., López-Nicolás, G., Sagüés, C., and Llorente, S. (2015). Persistent Coverage Control for a Team of Agents with Collision Avoidance. *Eur. J. Control.* 22, 30–45. doi:10.1016/j.ejcon.2014.12.001

Funada, R., Santos, M., Yamauchi, J., Hatanaka, T., Fujita, M., and Egerstedt, M. (2019). "Visual Coverage Control for Teams of Quadcopters via Control Barrier Functions," in Proceedings of 2019 International Conference on Robotics and Automation, Montreal, QC, May 20–24, 2019, 3010–3016. doi:10.1109/ICRA.2019.8793477

Hübel, N., Hirche, S., Gusrialdi, A., Hatanaka, T., Fujita, M., and Sawodny, O. (2008). Coverage Control with Information Decay in Dynamic Environments. *IFAC Proc. Volumes* 41, 4180–4185. doi:10.3182/20080706-5-kr-1001.00703

Hussein, I. I., Stipanovic, D. M., and Yue Wang, Y. (2007). "Reliable Coverage Control Using Heterogeneous Vehicles," in Proceedings of 2007 46th IEEE Conference on Decision and Control, New Orleans, LA, December 12–14, 2007, 6142–6147. doi:10.1109/CDC.2007.4434510

Kapoutsis, A. C., Chatzichristofis, S. A., and Kosmatopoulos, E. B. (2019). A Distributed, Plug-N-Play Algorithm for Multi-Robot Applications with A Priori Non-computable Objective Functions. *Int. J. Robotics Res.* 38, 813–832. doi:10.1177/0278364919845054

Lindemann, L., and Dimarogonas, D. V. (2019). Control Barrier Functions for Multi-Agent Systems under Conflicting Local Signal Temporal Logic Tasks. *IEEE Control. Syst. Lett.* 3, 757–762. doi:10.1109/LCSYS.2019.2917975

Martínez, S., Cortés, J., and Bullo, F. (2007). Motion Coordination with Distributed Information. *IEEE Control. Syst.* 27, 75–88. doi:10.1109/MCS.2007.384124

Notomista, G., and Egerstedt, M. (2021). Persistification of Robotic Tasks. *IEEE Trans. Contr. Syst. Technol.* 29, 756–767. doi:10.1109/TCST.2020.2978913

Notomista, G., Ruf, S. F., and Egerstedt, M. (2018). Persistification of Robotic Tasks Using Control Barrier Functions. *IEEE Robot. Autom. Lett.* 3, 758–763. doi:10.1109/LRA.2018.2789848

Palacios-Gasós, J. M., Montijano, E., Sagüés, C., and Llorente, S. (2016). Distributed Coverage Estimation and Control for Multirobot Persistent Tasks. *IEEE Trans. Robot.* 32, 1444–1460. doi:10.1109/TRO.2016.2602383

Renzaglia, A., Doitsidis, L., Martinelli, A., and Kosmatopoulos, E. B. (2012). Multi-robot Three-Dimensional Coverage of Unknown Areas. *Int. J. Robotics Res.* 31, 738–752. doi:10.1177/0278364912439332

Santos, M., Mayya, S., Notomista, G., and Egerstedt, M. (2019). "Decentralized Minimum-Energy Coverage Control for Time-Varying Density Functions," in Proceedings of 2019 International Symposium on Multi-Robot and Multi-Agent Systems, New Brunswick, NJ, August 22–23, 2019, 155–161. doi:10.1109/MRS.2019.8901076

Schwager, M., Julian, B. J., Angermann, M., and Rus, D. (2011). Eyes in the Sky: Decentralized Control for the Deployment of Robotic Camera Networks. *Proc. IEEE* 99, 1541–1561. doi:10.1109/JPROC.2011.2158377

Sugimoto, K., Hatanaka, T., Fujita, M., and Huebel, N. (2015). "Experimental Study on Persistent Coverage Control with Information Decay," in Proceedings of 2015 54th Annual Conference of the Society of Instrument and Control Engineers of Japan, Hangzhou, China, July 28–30, 2015, 164–169. doi:10.1109/SICE.2015.7285343

Wang, Y.-W., Zhao, M.-J., Yang, W., Zhou, N., and Cassandras, C. G. (2020). Collision-free Trajectory Design for 2D Persistent Monitoring Using Second-Order Agents. *IEEE Trans. Control. Netw. Syst.* 7, 545–557. doi:10.1109/TCNS.2019.2954970

Wang, Y., and Wang, L. (2017). "Awareness Coverage Control in Unknown Environments Using Heterogeneous Multi-Robot Systems," in *Cooperative Control of Multi-Agent Systems*. Editors Y. Wang, E. Garcia, D. Casbeer, and F. Zhang (Hoboken, NJ: John Wiley & Sons), 265–290. doi:10.1002/9781119266235.ch10

Zhu, M., and Martínez, S. (2013). Distributed Coverage Games for Energy-Aware mobile Sensor Networks. *SIAM J. Control. Optim.* 51, 1–27. doi:10.1137/100784163

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

The reviewer NC declared a past co-authorship with one of the authors MF to the handling Editor.