

ARTICLE TEMPLATE

Angle-Aware Coverage with Camera Rotational Motion Control

Zhiyuan Lu^a, Muhammad Hanif^{ib}, Takumi Shimizu^{ib}, and Takeshi Hatanaka^{ib}^aTokyo Institute of Technology, 2-12-1 Ookayama, Meguro-ku, Tokyo, 152-8550, Japan

ARTICLE HISTORY

Compiled April 23, 2024

ABSTRACT

This paper presents a novel control strategy for drone networks to improve the quality of 3D structures reconstructed from aerial images by drones. Unlike the existing coverage control strategies for this purpose, our proposed approach simultaneously controls both the camera orientation and drone translational motion, enabling more comprehensive perspectives and enhancing the map's overall quality. Subsequently, we present a novel problem formulation, including a new performance function to evaluate the drone positions and camera orientations. We then design a QP-based controller with a control barrier-like function for a constraint on the decay rate of the objective function. The present problem formulation poses a new challenge, requiring significantly greater computational efforts than the case involving only translational motion control. We approach this issue technologically, namely by introducing JAX, utilizing just-in-time (JIT) compilation and Graphical Processing Unit (GPU) acceleration. We finally conduct extensive verifications through simulation in ROS (Robot Operating System) and show the real-time feasibility of the controller and the superiority of the present controller to the conventional method.

KEYWORDS

Coverage control, control barrier function, drone, multi-agent systems, just-in-time compilation, GPU acceleration

1. Introduction

The advancement of 3D map reconstruction technology has played a pivotal role in supporting various sectors, including building information modeling [1,2], precision agriculture [3,4], and construction site inspection [5]. The success of the Structure from Motion (SfM) algorithm [6] has been instrumental in efficiently reconstructing a 3D model of a target object by analyzing a collection of images captured by a camera. Notably, the past decades have witnessed the introduction of numerous sensors and platforms for capturing high-quality images, with unmanned aerial vehicles (UAVs) emerging as particularly prevalent platforms capable of autonomous image acquisition [7]. Therefore, ensuring the efficient coverage of viewpoints by UAVs becomes a crucial requirement for achieving high-quality map reconstruction using the SfM algorithm.

To ensure efficient coverage of viewpoints, at least two crucial factors must be considered. Firstly, achieving comprehensive observation of every part of the scene from multiple angles is vital to enhance the reliability and accuracy of 3D reconstruction [7].

This necessitates sufficient overlap between viewpoints, compelling UAVs to explore a 6D configuration encompassing both 3D position and 3D angle orientation. However, in practice, the roll angle is often neglected, simplifying the search space to a 5D configuration for camera poses [8,9]. Secondly, addressing the one-time visit problem is essential [10]. This involves systematically capturing images across the target field, ensuring that each 5D point within the target field is observed and visited.

Addressing the aforementioned requirement, classical coverage control algorithms have traditionally been widely adopted to tackle the scenario outlined above [11]. Moreover, various studies have also explored coverage control strategies in environments with diverse shapes [12–17]. However, a notable limitation in the existing literature is that they do not consider coverage in a 5D search space, a crucial aspect in the context of UAV-based 3D map reconstruction. Furthermore, many of these studies tend to guide robots into static configurations, falling short of covering each point within the target field.

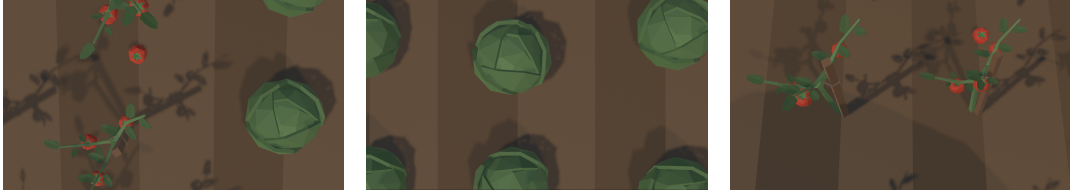
Alternatively, there has been a study on persistent coverage control algorithms, enabling UAVs to monitor the environment persistently [18–21]. However, once more, these studies predominantly focus on 2D coverage, overlooking the necessity to observe points from multiple angles. Furthermore, persistent coverage strategies may not address the specific requirement of a one-time visit, as specified earlier.

Recent work in angle-aware coverage control [10] marks the initial attempt at achieving 5D coverage with multiple UAVs. This approach also provides a solution to the one-time visit problem. By integrating the concept of capturing images from various angles within the coverage control framework, significant improvements in 3D map reconstruction quality have been demonstrated, as shown in [22]. However, it is essential to note that even in [10], the drones’ camera orientations remained fixed, suggesting room for further improvements. This could be achieved by considering dynamic camera control of the UAV using gimbal mechanisms. In an earlier exploration into camera angle control, documented in a previous study [23,24], the focus was primarily on adjusting the camera’s angle while keeping the camera position static; therefore, it could not solve the one-time visit problem.

In the context of this paper, our objective is to extend the existing angle-aware coverage control framework [10] by integrating camera rotational motion control via gimbal mechanisms. This enhancement allows for simultaneous control of both camera orientation and drone motion within the coverage control framework. To achieve this, we present a new formulation of the problem that includes a new performance function to assess the camera orientations and drone positions. We then design a QP-based controller [25] with a constraint using a control barrier-like function on the decay rate of the objective function. This new problem formulation is significantly more computationally demanding than the case of coverage with translational motion control only. We address this challenge by implementing JAX [26], employing JIT compilation and GPU acceleration. Finally, using simulation in the ROS, we conduct thorough verifications demonstrating the controller’s real-time viability and superiority over the traditional approach.



(a) 3D Map Reconstruction Scene



(b) Image captured by each drone

Figure 1. Collaborative 3D map reconstruction using drone networks scene.

2. Problem settings

2.1. Drones, Virtual Field, and Geometry

We consider a scenario involving n drones, each represented by the index set $\mathcal{I} := \{1, \dots, n\}$, operating in three-dimensional Euclidean space. All the drones are regarded as rigid bodies, and we define two frames of reference: the world frame Σ_w and the body frame Σ_i , which is fixed on the camera of the i -th drone as illustrated in Fig 2. The x , y , and z coordinates of the origin of Σ_i relative to Σ_w are denoted by x_i , y_i , and z_i , respectively. The orientation of Σ_i relative to Σ_w is denoted by $R_i \in SO(3)$. Each drone i is equipped with an onboard camera that can be adjusted both horizontally, denoted by φ_i , and vertically, denoted by ϕ_i , using a gimbal system. The horizontal angle φ_i ranges from 0 to 2π , while the vertical angle ϕ_i ranges from 0 to $\pi/2$, effectively forming a hemisphere of observable angles for the camera.

During operation, all drones are assumed to maintain a constant altitude $z_i = z_c$. Consequently, we exclude the altitude component from the drone's position description, defining a state vector of drone i , $p_i := [x_i \ y_i \ \varphi_i \ \phi_i]^T \in \mathcal{P} \times [0, 2\pi) \times [0, \pi/2] \subset \mathbb{R}^4$, where $\mathcal{P} \subset \mathbb{R}^2$ represents a compact subset of a plane. Furthermore, each drone within the set \mathcal{I} follows the subsequent dynamics:

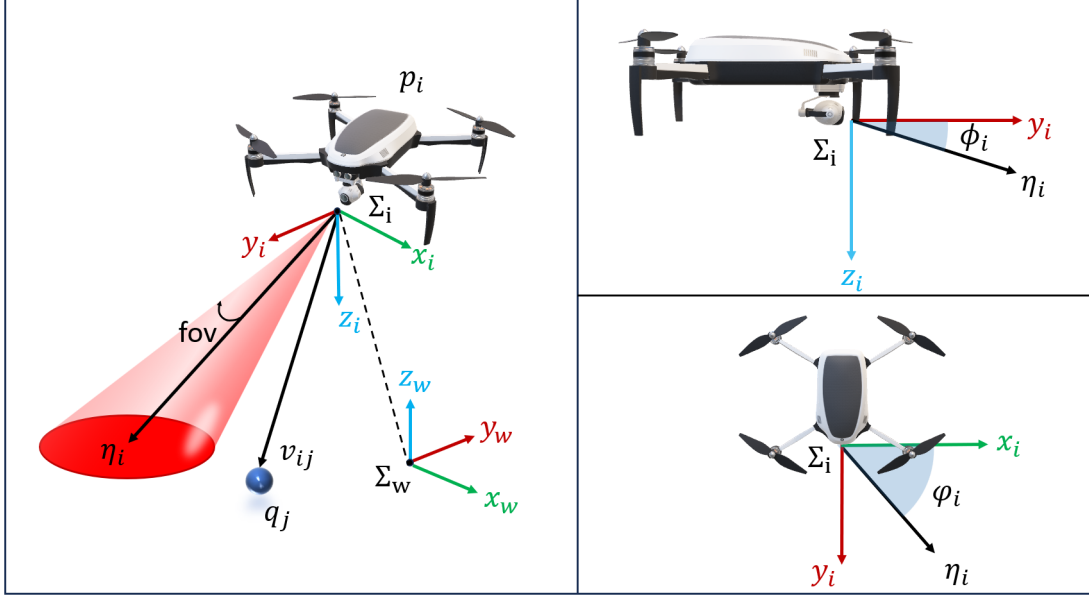


Figure 2. Gimbal coordinate reference.

$$\dot{p}_i = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\phi}_i \\ \dot{\phi}_i \end{bmatrix} = \begin{bmatrix} u_i^x \\ u_i^y \\ u_i^\varphi \\ u_i^\phi \end{bmatrix} = u_i. \quad (1)$$

Here, u_i^x and u_i^y represent the linear velocity input for drone i , while u_i^φ and u_i^ϕ correspond to the angular velocity input for adjusting the gimbal angles of drone i .

Our main objective is to reconstruct 3D structure about a given target field. In this paper, we model the set of position coordinates in Σ_w of all points in the target field to be observed as $\mathcal{B} \subset \mathbb{R}^3$, encompassing the highest and lowest points and any objects within. Now, it is widely known that every point in \mathcal{B} should be observed by rich viewing angles in order to obtain a high-quality 3D structure. To reflect this objective, we define the horizontal angle $\theta_h \in [-\pi, \pi)$ and the vertical angle $\theta_v \in (0, \pi/2]$ to represent the angles from which we observe specific target points. The points to be observed are then characterized by five variables consisting of not only the position coordinates $[x \ y \ z]^T \in \mathcal{B}$ but also the viewing angles θ_h and θ_v (See Fig 3). Accordingly, we consider a coverage control problem over the 5D virtual space, $\mathcal{Q}_c \subset \mathbb{R}^5$, which encompasses all observation variables: $(x, y, z, \theta_h, \theta_v)$. Hence, the primary goal is how we can effectively control drone $p_i \in \mathcal{P}$ to monitor every point $q \in \mathcal{Q}_c$. In the sequel, we discretize the 5D virtual space \mathcal{Q}_c into m cells and a representative point of the j -th cell is denoted by $q_j = [x_j \ y_j \ z_j \ \theta_j^h \ \theta_j^v] \in \mathcal{Q}_c$. The collection of $q_j, j = 1, 2, \dots, m$ is denoted by \mathcal{Q} .

Let us now introduce geometry associated with the present control problem. Consider drone i having the state $p_i := [x_i \ y_i \ \varphi_i \ \phi_i]^T$. The optical axis of the camera is then described in Σ_i as $\eta_i = [\cos(\varphi_i) \cdot \cos(\phi_i), \sin(\varphi_i) \cdot \cos(\phi_i), \sin(\phi_i)]^\top$. On the other

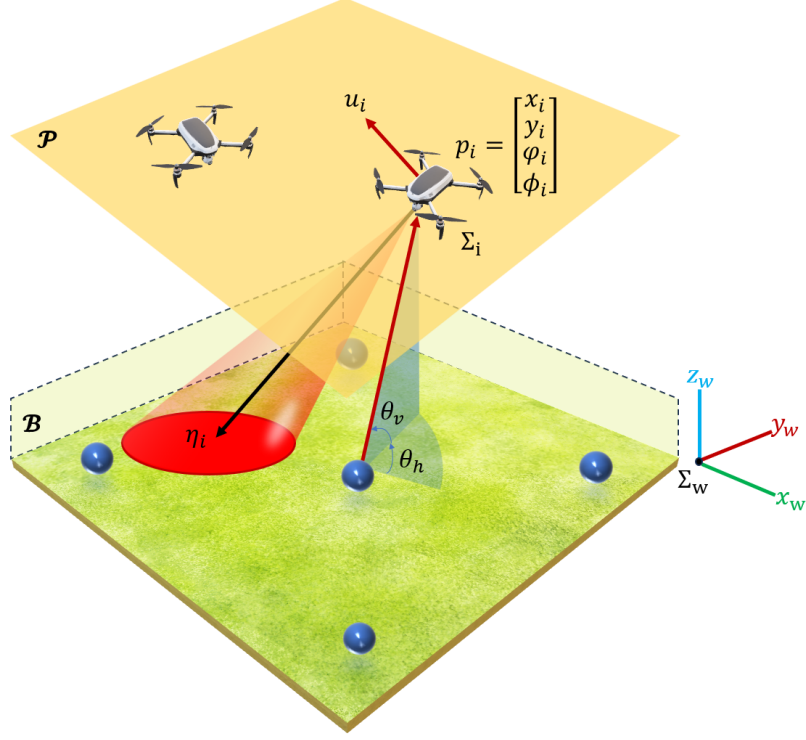


Figure 3. Illustration of the angle aware coverage problem with camera orientation control.

hand, the position vector in q_j (the first three elements) is described as

$$v_{ij} = R_i^T v_{ij}^w, \quad v_{ij}^w = \begin{bmatrix} x_j - x_i \\ y_j - y_i \\ z_j - z_c \end{bmatrix} \quad (2)$$

in Σ_i . Also, the vector specified by the angles θ_j^h and θ_j^v is represented in Σ_w as

$$v_j = \begin{bmatrix} \cos(\theta_h) \cdot \cos(\theta_v) \\ \sin(\theta_h) \cdot \cos(\theta_v) \\ \sin(\theta_v) \end{bmatrix}. \quad (3)$$

Remark 1. In the context of drone camera capabilities, it is essential to address certain limitations that may affect their performance. While some drone cameras offer pitch, yaw, and roll rotation capability, the structural limitations of many gimbals often restrict the range of yaw rotation angles, often preventing a complete 360-degree rotation. To address this, controlling the camera's yaw angle typically involves coordination with the drone's body rotation. However, in this paper, we do not focus on the drone's rotational behavior, and for simplicity, we consider it as part of the gimbal's angle control. Moreover, it's important to note that camera sensors generally have rectangular designs, leading to a rectangular beam-shaped observable area for the drone. Nevertheless, for the sake of simplicity and without loss of generality, we assume a circular shape for the drone's observable area.

2.2. Performance function and importance index

In this subsection, we present a performance function, denoted as $h : \mathcal{P} \times [0, 2\pi) \times [0, \pi/2] \times \mathcal{Q}_c \rightarrow [0, 1]$. This function characterizes a drone's coverage capability at state p_i concerning a specific point $q_j \in \mathcal{Q}$. A higher value of the performance function corresponds to a more effective coverage capability. The effective coverage of point q_j by a drone positioned at state p_i depends on two key factors:

- (1) The geometric position of point q_j must fall within the drone's camera's field of view.
- (2) The relative geometric relationship between the drone's position and the observed point q_j must satisfy the angle (θ_h, θ_v) of q_j .

The first condition can then be described as follows:

$$h_1(p_i, q_j) = (\text{fov} - \arccos(\eta_i \cdot v_{ij} / \|v_{ij}\|))^2 \geq 0, \quad (4)$$

where fov is the half-angle of the camera's field of view. The second condition can be described in a similar way:

$$h_2(p_i, q_j) = (\arccos(v_j \cdot v_{ij}^w / \|v_{ij}^w\|))^2 \geq 0. \quad (5)$$

In the subsequent controller design, we employ the following performance function combining h_1 and h_2 with appropriate tuning of the parameters σ_1 and σ_2 :

$$h(p_i, q_j) := \exp\left(-\frac{h_1(p_i, q_j)}{2\sigma_1^2}\right) \exp\left(-\frac{h_2(p_i, q_j)}{2\sigma_2^2}\right). \quad (6)$$

Remark that the parameter σ_1 should be tuned so that the performance function is sufficiently close to 0 for any point q outside of the field of view. On the other hand, we have not thought of any systematic way to tune σ_2 , and tune empirically this parameter in the subsequent simulation. We would like to leave the issue to future work.

Let us next introduce the importance index $\psi_j \in [0, \infty)$ assigned to each point $q_j \in \mathcal{Q}$. From the definition of the performance function h , a large value of $h(p_i, q_j)$ for some i means that the drone already samples a good image on q_j and then we can reduce the importance of the point q_j . Based on this observation, similarly to [10], we propose the following update law for the importance index ψ_j :

$$\dot{\psi}_j = -\delta \max_{i \in \mathcal{I}} h(p_i, q_j) \psi_j \quad (\delta > 0). \quad (7)$$

In terms of the controller design, it is preferable that the performance function meets these two properties:

- Restricting the performance function within the range from 0 to 1 in order to govern the decay rate of ψ_j in (7).
- Ensuring a continuous non-zero gradient for the performance function in order to ensure the existence of the input u_i that increases the function b_i .

An example that meets the above properties is the Gaussian function. This is why we utilize it as the performance function.

2.3. Objective function

We are now prepared to present the objective function that needs to be minimized, defined as

$$J := \sum_{j=1}^m \psi_j. \quad (8)$$

This equation represents the integral of the density function across the entire region. When the density function approaches values close to 0 for individual points, it signifies the effective capture of images for those points. As J approaches 0, it indicates that the region's coverage is nearing completion, which is crucial for capturing images suitable for reconstructing the 3D structure. Consequently, the primary objective is to steer the drones in a manner that drives J towards convergence to zero. However, due to the monotonically decreasing property of ψ_j in (7), it is trivial to achieve $J \rightarrow 0$ itself. We thus impose another specification $\dot{J} \leq -\gamma$ for a positive constant γ to specify efficiency of the task completion. This covers the primary objective $J \rightarrow 0$. In summary, the problem to be addressed in this paper is to determine the velocity input u_i so that $\dot{J} \leq -\gamma$ is satisfied for a given parameter $\gamma > 0$.

3. QP-based Controller Design

In this section, we propose a controller based on quadratic programming, which enforces the constraint $\dot{J} \leq -\gamma$ using the concept of control barrier functions. To this end, we begin by computing the time derivative of J :

$$\begin{aligned} \dot{J} &= \sum_{j=1}^m \dot{\psi}_j(q) = - \sum_{j=1}^m \delta \max_{i \in \mathcal{I}} h(p_i, q_j) \psi_j \\ &= - \sum_{i=1}^n \int_{j \in \mathcal{V}_i(p)} \delta h(p_i, q_j) \psi_j = - \sum_{i=1}^n I_i, \end{aligned} \quad (9)$$

where

$$I_i := \int_{j \in \mathcal{V}_i(p)} \delta h(p_i, q_j) \psi_j \quad (10)$$

corresponds to the contribution by drone i to reduce J in (8). The set $\mathcal{V}_i(p)$, which depends on $p := (p_i)_{i \in \mathcal{I}}$, is a Voronoi-like partition of the set $\mathcal{M} := \{1, 2, \dots, m\}$ defined as

$$\mathcal{V}_i(p) := \{j \in \mathcal{M} \mid h(p_i, q_j) \leq h(p_k, q_j) \ \forall k \in \mathcal{I}\}. \quad (11)$$

By defining $b_{i,I} = I_i - \gamma/n$ as a candidate for the control barrier function, with a given $\gamma > 0$, we aim to compel the drone to minimize the objective function J at a specified rate of decrease, denoted by γ . This criterion is met when the control input u_i is chosen such that the following inequality holds:

$$\left(\frac{\partial b_{i,I}}{\partial \psi_j}\right) \dot{\psi}_j + \left(\frac{\partial b_{i,I}}{\partial p_i}\right)^T u_i + \alpha_1(b_{i,I}) \geq 0, \quad (12)$$

where α_1 is a locally Lipschitz extended class \mathcal{K} function. A continuous function $\alpha : (-b, a) \rightarrow (-\infty, \infty)$ is said to belong to extended class \mathcal{K} for some $a, b > 0$ if it is strictly increasing and $\alpha(0) = 0$ [27].

Furthermore, to adhere to the gimbal input limitations, we introduced a secondary control barrier function constraint, restricting the vertical angle ϕ_i of the drone's gimbal within the range of 0 to $\pi/2$. While functionally equivalent vertical angles exist between π and $\pi/2$ as those between 0 and $\pi/2$, the physical structure of the gimbal imposes constraints on this range. Failing to enforce these constraints during optimization may result in commands that surpass the $\pi/2$ threshold, which not only prevents gimbal from executing the command, but also results in a reduction in horizontal rotation. By maintaining the vertical angle within the specified limits, we ensure optimal performance and prevent unintended consequences associated with exceeding the gimbal's mechanical boundaries.

Therefore, to ensure the vertical angle ϕ_i of the gimbal varies between ϕ_{\min} and ϕ_{\max} , we define a control barrier function. This function, denoted as $b_{i,\phi}$, ensures that the condition

$$b_{i,\phi} = (\phi_{\max} - \phi_{\min})^2 - \left(\phi_i - \frac{\phi_{\min} + \phi_{\max}}{2}\right)^2 \geq 0 \quad (13)$$

is satisfied at all times, securing the appropriate vertical angle constraints throughout the optimization process. This criterion is met when the control input u_i is chosen such that the following inequality holds:

$$\left(\frac{\partial b_{i,\phi}}{\partial p_i}\right)^T u_i + \alpha_2(b_{i,\phi}) \geq 0, \quad (14)$$

where α_2 is a locally Lipschitz extended class \mathcal{K} function.

Now, we are ready to present the QP-based controller as follows:

$$(u_i^*, w_i^*) = \arg \min_{(u_i, w_i) \in \mathcal{U} \times \mathbb{R}} \epsilon \|u_i\|^2 + |w_i|^2 \quad (15a)$$

$$\text{s.t.} \quad \dot{b}_{i,I} + \alpha_1(b_{i,I}) \geq w_i, \quad (15b)$$

$$\dot{b}_{i,\phi} + \alpha_2(b_{i,\phi}) \geq 0, \quad (15c)$$

where ϵ is the penalty variable and w_i is the slack variable.

Theorem 3.1. Suppose that no $q_j (j \in \mathcal{M})$ is located on the boundary of $\mathcal{V}_i(p)$. When $\alpha_1 : \mathbb{R} \rightarrow \mathbb{R}$ and $\alpha_2 : \mathbb{R} \rightarrow \mathbb{R}$ are set as a linear function such that $\alpha_1(b_{i,I}) = a_1 b_{i,I}$ and $\alpha_2(b_{i,\phi}) = a_2 b_{i,\phi}$, where $a_1 > 0$ and $a_2 > 0$ is a positive scalar, the problem is equivalently reformulated as :

$$(u_i^*, w_i^*) = \arg \min_{(u_i, w_i) \in \mathcal{U} \times \mathbb{R}} \epsilon \|u_i\|^2 + |w_i|^2 \quad (16a)$$

$$s.t. \quad \xi_{1i}^T u_i + \xi_{2i} \geq w_i, \quad (16b)$$

$$\chi_{1i}^T u_i + \chi_{2i} \geq 0, \quad (16c)$$

where

$$\xi_{1i} := \int_{j \in \mathcal{V}_i(p)} \delta \frac{\partial h(p_i, q_j)}{\partial p_i} h(p_i, q_j) \psi_j, \quad (17a)$$

$$\xi_{2i} := -\frac{a_1 \gamma}{n} + \int_{j \in \mathcal{V}_i(p)} (-\delta^2 h^2(p_i, q_j) + a \delta h(p_i, q_j)) \psi_j, \quad (17b)$$

$$\chi_{1i} := \begin{bmatrix} 0 \\ 0 \\ 0 \\ 2 \left(\phi_i - \frac{\phi_{\min} + \phi_{\max}}{2} \right) \end{bmatrix}, \quad (18a)$$

$$\chi_{2i} := a_2 \left((\phi_{\max} - \phi_{\min})^2 - \left(\phi_i - \frac{\phi_{\min} + \phi_{\max}}{2} \right)^2 \right). \quad (18b)$$

Please refer to Appendix A for the detail proof of the controller (16).

Remark 2. The optimization problem (15) is always feasible because the constraint is softened by the slack variable w_i . However, the existence of the solution to (15) does not mean that the original specification $I_i \geq \gamma/n$ is met. For a too large γ , the specification might not be satisfied in the transient in the presence of the velocity limit of the drone. It is desirable that γ is appropriately determined so that $I_i \geq \gamma/n$ is violated only when the coverage is almost completed, namely $J \approx 0$. However, a systematic way to determine such γ for a given velocity limit is left for future work.

4. Computation acceleration

In contrast to conventional two-dimensional coverage control, dealing with coverage in a five-dimensional space significantly increases the computational load. Specifically, the number of cells, m , drastically increases in the five-dimensional case, which affects various processes including calculation of the performance function, updates of the importance index, Voronoi-like region partitioning, and, more significantly, gradient computation of the performance function. In particular, the gradient computation in terms of the five variables x_i , y_i , φ_i and ϕ_i takes up most of the computational time. [10] presented a computationally efficient implementation of the controller with slight

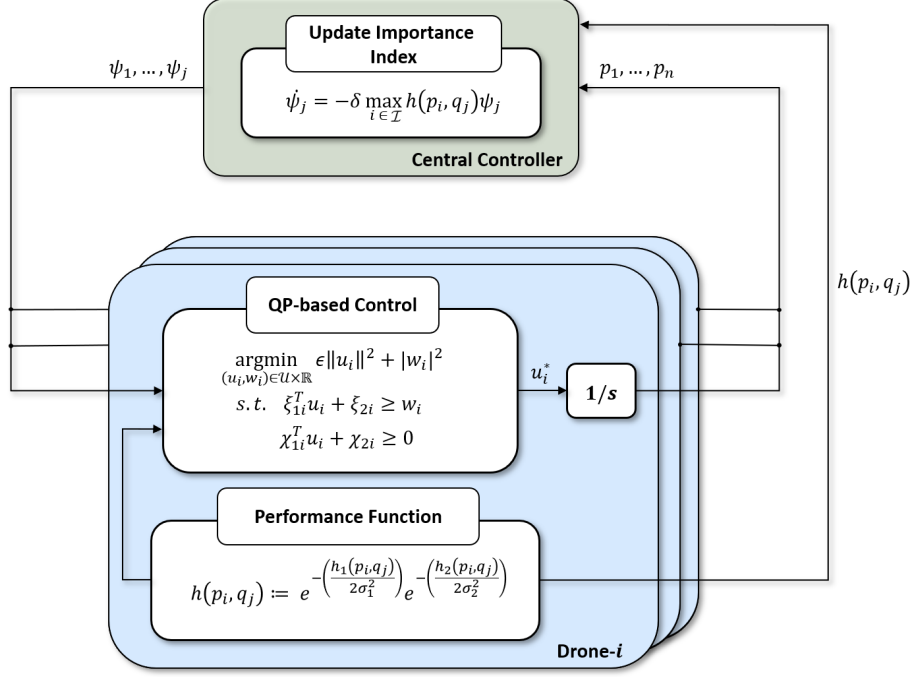


Figure 4. Controller architecture

approximations in computation, where the position of the drone monitoring a point from a specified view angle was uniquely determined, enabling the mapping of the five-dimensional target field to a two-dimensional space with slight approximation. On the other hand, in this paper, we assume that the camera orientation can be controlled. In this case, the camera state monitoring a point from a specified view angle is not uniquely determined, and the mapping from five-dimensional field to a lower-dimensional space cannot be defined. Therefore, the same approach could not be applied to the present problem. We thus approach this problem technologically, namely we introduce a JAX library accelerating the processing through just-in-time (JIT) compilation and GPU acceleration.

4.1. JIT compilation

As same as in [10], our program is primarily written in Python, and we use NumPy library for matrix calculations. Since Python is an interpreted language, the code is not compiled before execution. This leads to a significant performance loss when running large-scale computations in Python compared to statically compiled code. The complex calculations involved in computing the gradient of the performance function exacerbate this loss, making real-time calculations challenging. We thus replace NumPy by JAX's built-in NumPy, allowing to compile functions of pure matrix computations.

We use JIT in two types of ROS nodes in our program: central controller and drone controller, where the importance index is updated in the central controller, while the computation performance function, dividing the Voronoi-like partition and the gradient computation is done in the drone controller, as Fig. 4 shows. The first two rows of table 1 compare the impact on the computation speed of whether JIT is used or not. For drone controller, which is more computationally intensive compared to

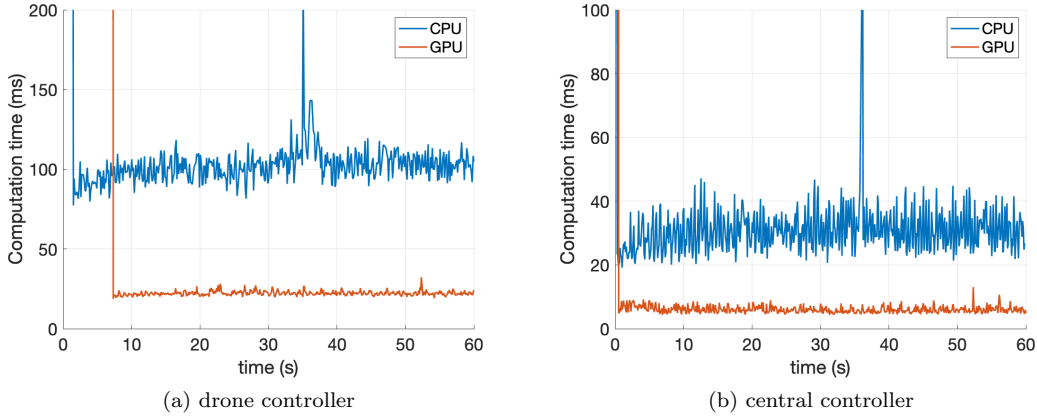


Figure 5. Comparisons of CPU and GPU computational time.

central controllers, computation times over 1s are unacceptable for real-time control. Meanwhile, after accelerating the function using JIT, the computational speed gains a significant increase even with the same computer, enabling it to reach a computational frequency of around 10Hz.

Table 1. Average computation time (ms) of one step

	Central controller	Drone controller
CPU	212	1206
JIT, CPU	31	102
JIT, GPU	6	22

4.2. GPU acceleration

Since CPU performance is very difficult to improve and the computation time using CPU can be greatly altered by the size of the matrix, we utilize JAX’s GPU acceleration feature to do matrix calculations on GPU.

We conducted a timing comparison on a laptop running Ubuntu 20.04.3 LTS, comparing the time it takes to perform computations using the CPU and GPU. The laptop is equipped with a 20-core Intel i7-12800H CPU and a Nvidia RTX A3000 GPU. Fig 5 shows how the computation time consumed at each step changes over time when using CPU and GPU. We note that the first step of the computation takes a very long time in comparison, which is caused by the JIT compilation during the initial execution of functions. When using the GPU, compilation took longer, but after compilation was complete, computation using the GPU took roughly 1/5 the time of the CPU, and the time spent was relatively stable in comparison, while the time spent on the CPU fluctuated considerably.

During CPU-based computations, the CPU usage is around 60%, and during GPU-based computations, the GPU usage is reported only 24%. As indicated in the table 1, employing GPU acceleration for matrix computations results in significant time savings, particularly for large matrices, despite the overhead of data transfer between the CPU and GPU memory. It is to be expected that after increasing the size of the matrix, the computation time of the CPU and GPU will produce a bigger gap.

5. Controller Evaluation and Verification

Table 2. Parameter setting

a_1	a_2	σ_1	σ_2	ϵ	γ	δ
5.0	1.0	0.13	0.18	0.0001	0.05	5.0

In this section, we demonstrate the proposed controller through simulation in the ROS Noetic environment.

The monitored space was set as a cube shape with the range of $[-1, 1]\text{m} \times [-1, 1]\text{m} \times [0, 0.5]\text{m}$. The viewing angle space is set to $\theta_h \in [-\pi, \pi)$, and $\theta_v \in [\pi/6, \pi/2]$. We set the number of drones n to 3 and their initial positions are uniformly distributed to $p_1 = [1.0, 0.2]^T\text{m}$, $p_2 = [-1.0, -0.2]^T\text{m}$ and $p_3 = [0.0, 0.5]^T\text{m}$. Their flight altitude is fixed to $h_c = 1.0\text{m}$ after takeoff. To simulate a drone equipped with a camera with a focal length of $x[\text{mm}]$, we set the field of view of the drone to $\pi/6$. The initial angles of the gimbal for the three drones are all set to $\varphi_i = 0$ and $\phi_i = \pi/2$, indicating that the cameras were initially pointed vertically downwards. The horizontal angles of the drone gimbal φ_i were limited to $[0, 2\pi)$, while the vertical angles ϕ_i were limited to $(0, \pi/2]$. The field is divided into $m = 1.5 \times 10^7$ small cells of size $0.02\text{m} \times 0.02\text{m} \times 0.1\text{m} \times \pi/30\text{rad} \times \pi/30\text{rad}$. Other parameters are presented as table 2. The quadratic programming is solved by CVXOPT [28].

All the drones start the coverage from the same moment $t = 0\text{s}$, and their positions and camera orientations are presented by a series of screenshots as Fig. 6. The value of the importance function is a five-dimensional matrix that is difficult to visualize. Therefore we take the average of the two dimensions θ_h and θ_v , transform it into a matrix containing xyz , and visualize it as a point cloud. The color of the points represents the importance of the position, purple means not yet covered, and red means well covered. The region changed from purple, gradually to red, representing a good completion of the coverage. We observe that drones tend to cover within the task region and keep downwards at the beginning, and start moving out of the task region and turning the gimbal to cover more points at a later stage.

The evolution of the objective function J is shown in Fig. 7, indicating a linear decreasing trend for the majority of the time, which implies that the decrease of the objective function meets the requirement of the target γ . After the coverage is completed to a certain extent, the coverage performance of γ becomes difficult to achieve and the constraint is violated, so the decline of the objective function gradually slows down.

Let us next compare the performance of the present controller with that in [10] without considering the camera orientation control. Since these controllers employ different objective functions, they cannot be used as metrics of the comparison. To fairly compare these controllers, we need to prepare another metric that quantifies their performances. In this paper, we revert to the original use case to establish quantifiable metrics related to the actual 3D model quality. In practical applications, drones perceive the environment by taking photos or videos at a limited frequency, rather than through continuous performance functions. We mimic this behavior by attempting to update the drone’s coverage of the target region at a finite frequency. Specifically, we set a shooting rate of images for the drone, representing the number of photos it takes

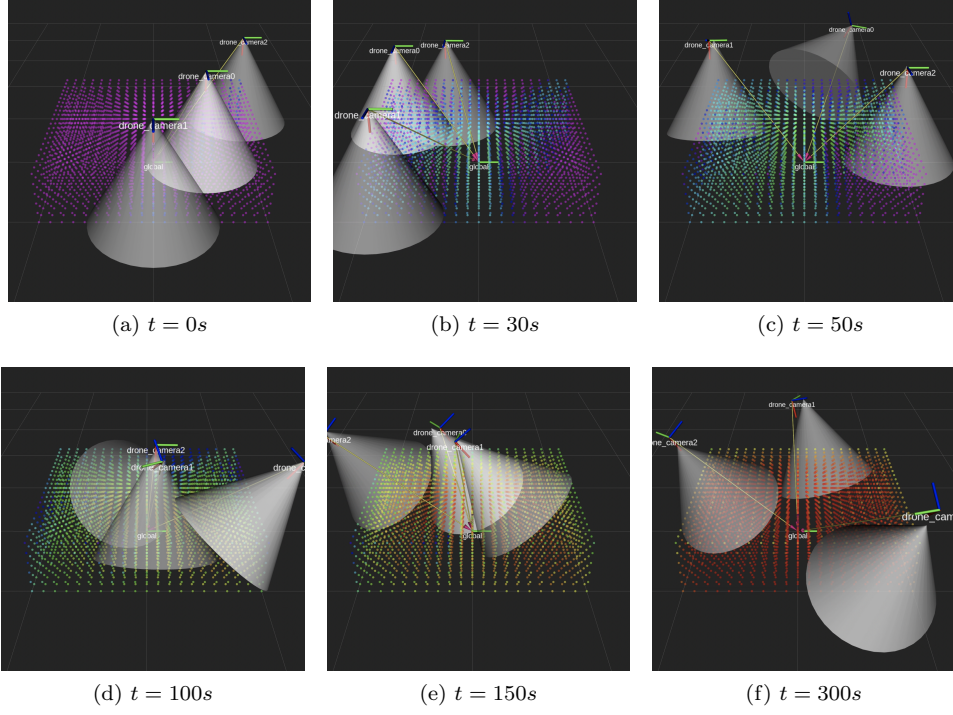


Figure 6. Snapshots of the simulation.

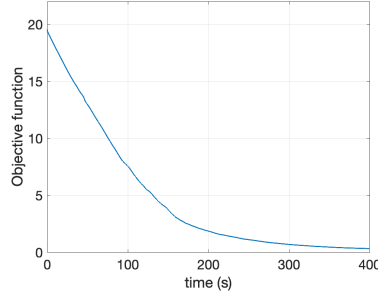


Figure 7. Time evolution of the objective function.

per second. During each shoot, we record which points are covered by the drone. When a point has met the standard of covering, it is marked as “covered”. This process is similar to updating the objective function, but the difference lies in using Boolean values instead of continuous values. By tracking the reduction in the number of uncovered points, we can evaluate how well a controller contributes to the reconstruction of the 3D model. However, we still need to establish criteria to determine whether a point is considered covered. Similar to the performance function, we also consider two aspects. A point is considered covered at a specific moment if and only if both of the following conditions are met:

- (1) The point lies within the field of view of the drone’s camera at that moment.
- (2) The angle between the line connecting the drone to that point and the observed angle of that point differs by less than a certain threshold.

We set the threshold value to $\pi/16$ and recorded the covered points for both of the

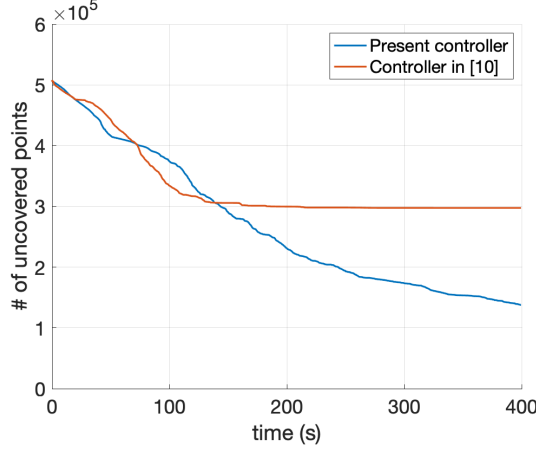


Figure 8. Comparison of angle-aware coverage and with camera rotational motion control

present controller and [10] at a burst rate of 5 Hz. The recorded results of decreased objective functions are presented in Fig. 8. Since [10] cannot rotate the camera, the coverage was completed at around $t = 130$ s and then stopped moving. Due to the limited range of viewing angles, the number of uncovered points remain high even after finishing the coverage. Meanwhile, when we use the present controller with camera orientation control, the number decreases to about 1/2 of [10]. This result demonstrates the benefit of controlling the camera orientations as well as the drone positions.

6. Conclusion

In this paper, we presented a novel angle-aware coverage with camera rotational motion control. To this end, we presented a novel problem formulation including a novel performance function that integrates the camera orientations. The real-time viability of the present QP-based controller was also demonstrated with the help of JIT and GPU computing. Moreover, we verified that the present controller achieves a better coverage performance than the original algorithm without camera control [10].

Future work should be directed to the hardware experiments and the performance comparison in terms of the accuracy of the reconstructed 3D structure.

Acknowledgment

This research was partially funded by Japan Society for the Promotion of Science (JSPS) KAKENHI under grant 21K04104. Additionally, we extend our thanks to Fabrizio Dabbene for his invaluable guidance and to Martina Mammarella for their dedicated contributions, both instrumental in the success of our research.

Disclosure Statement

No potential conflict of interest was reported by the author(s).

Appendix A. Proof of Theorem 3.1

The constraint in QP-based controller (15) consists of term $\dot{b}_{i,I}$, $\dot{b}_{i,\phi}$, $\alpha_1(b_{i,I})$, and $\alpha_2(b_{i,\phi})$. The term $\dot{b}_{i,I}$ and $\dot{b}_{i,\phi}$ can be calculated as follows:

$$\begin{aligned}\dot{b}_{i,I} &= \dot{I}_i = \int_{j \in \mathcal{V}_i(p)} \left(\delta \left(\frac{\partial h(p_i, q_j)}{\partial p_i} \right) \psi_j + \delta h(p_i, q_j) \dot{\psi}_j \right) \\ &= \int_{j \in \mathcal{V}_i(p)} \left(\delta \left(\frac{\partial h(p_i, q_j)}{\partial p_i} \right) \psi_j - \delta^2 h^2(p_i, q_j) \psi_j \right),\end{aligned}\quad (\text{A1})$$

$$\dot{b}_{i,\phi} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{\partial b_{i,\phi}}{\partial \phi_i} \end{bmatrix}^T = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 2 \left(\phi_i - \frac{\phi_{\min} + \phi_{\max}}{2} \right) \end{bmatrix}^T. \quad (\text{A2})$$

Then, remarking $\alpha_1(b_{i,I}) = a_1 b_{i,I}$ and $\alpha_2(b_{i,\phi}) = a_2 b_{i,\phi}$, we obtain

$$\alpha_1(b_{i,I}) = a_1 I_i - a_1 \gamma / n = -\frac{a_1 \gamma}{n} + \left(\int_{j \in \mathcal{V}_i(p)} a_1 \delta h(p_i, q_j) \psi_j \right), \quad (\text{A3})$$

$$\alpha_2(b_{i,\phi}) = a_2 \left((\phi_{\max} - \phi_{\min})^2 - \left(\phi_i - \frac{\phi_{\min} + \phi_{\max}}{2} \right)^2 \right). \quad (\text{A4})$$

By substituting I_i with (10) and equation (A1),(A3) to constraint (15b), we can get that $\dot{b}_{i,I} + \alpha_1(b_{i,I}) = \xi_{1i} u_i + \xi_{2i}$ holds true. Similarly, by substituting equation (A2),(A4) to constraint (15c), we can also get that $\dot{b}_{i,\phi} + \alpha_2(b_{i,\phi}) = \chi_{1i} u_i + \chi_{2i}$ holds true. Thus, the controller (16) is equivalent to (15).

References

- [1] Zheng X, Wang F, Li Z. A multi-uav cooperative route planning methodology for 3d fine-resolution building model reconstruction. *ISPRS journal of photogrammetry and remote sensing*. 2018;146:483–494.
- [2] Macher H, Landes T, Grussenmeyer P. From point clouds to building information models: 3d semi-automatic reconstruction of indoors of existing buildings. *Applied Sciences*. 2017; 7(10):1030.
- [3] Comba L, Biglia A, Aimonino DR, et al. Unsupervised detection of vineyards by 3d point-cloud uav photogrammetry for precision agriculture. *Computers and electronics in agriculture*. 2018;155:84–95.
- [4] Dong J, Burnham JG, Boots B, et al. 4d crop monitoring: Spatio-temporal reconstruction for agriculture. In: 2017 IEEE International Conference on Robotics and Automation (ICRA); IEEE; 2017. p. 3878–3885.
- [5] Xue J, Hou X, Zeng Y. Review of image-based 3d reconstruction of building for automated construction progress monitoring. *Applied Sciences*. 2021;11(17):7840.

- [6] Schonberger JL, Frahm JM. Structure-from-motion revisited. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 4104–4113.
- [7] Maboudi M, Homaei M, Song S, et al. A review on viewpoints and path planning for uav-based 3d reconstruction. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*. 2023;.
- [8] Liu Y, Cui R, Xie K, et al. Aerial path planning for online real-time exploration and offline high-quality reconstruction of large-scale urban scenes. *ACM Transactions on Graphics (TOG)*. 2021;40(6):1–16.
- [9] Smith N, Moehrle N, Goesele M, et al. Aerial path planning for urban scene reconstruction: A continuous optimization method and benchmark. *ACM Transactions on Graphics (TOG)*. 2018;.
- [10] Shimizu T, Yamashita S, Hatanaka T, et al. Angle-aware coverage control for 3-d map reconstruction with drone networks. *IEEE Control Systems Letters*. 2021;6:1831–1836.
- [11] Cortes J, Martinez S, Bullo F. Spatially-distributed coverage optimization and control with limited-range interactions. *ESAIM: Control, Optimisation and Calculus of Variations*. 2005;11(4):691–719.
- [12] Kantaros Y, Thanou M, Tzes A. Visibility-oriented coverage control of mobile robotic networks on non-convex regions. In: 2014 IEEE International Conference on Robotics and Automation (ICRA); IEEE; 2014. p. 1126–1131.
- [13] Kantaros Y, Thanou M, Tzes A. Distributed coverage control for concave areas by a heterogeneous robot–swarm with visibility sensing constraints. *Automatica*. 2015;53:195–207.
- [14] Schwager M, McLurkin J, Rus D. Distributed coverage control with sensory feedback for networked robots. In: *robotics: science and systems*; 2006. p. 49–56.
- [15] Schwager M, Rus D, Slotine JJ. Decentralized, adaptive coverage control for networked robots. *The International Journal of Robotics Research*. 2009;28(3):357–375.
- [16] Breitenmoser A, Schwager M, Metzger JC, et al. Voronoi coverage of non-convex environments with a group of networked robots. In: 2010 IEEE international conference on robotics and automation; IEEE; 2010. p. 4982–4989.
- [17] Stergiopoulos Y, Thanou M, Tzes A. Distributed collaborative coverage-control schemes for non-convex domains. *IEEE Transactions on Automatic Control*. 2015;60(9):2422–2427.
- [18] Palacios-Gasós JM, Montijano E, Sagüés C, et al. Distributed coverage estimation and control for multirobot persistent tasks. *IEEE transactions on Robotics*. 2016;32(6):1444–1460.
- [19] Sugimoto K, Hatanaka T, Fujita M, et al. Experimental study on persistent coverage control with information decay. In: 2015 54th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE); IEEE; 2015. p. 164–169.
- [20] Dan H, Yamauchi J, Hatanaka T, et al. Control barrier function-based persistent coverage with performance guarantee and application to object search scenario. In: 2020 IEEE Conference on Control Technology and Applications (CCTA); IEEE; 2020. p. 640–647.
- [21] Dan H, Hatanaka T, Yamauchi J, et al. Persistent object search and surveillance control with safety certificates for drone networks based on control barrier functions. *Frontiers in Robotics and AI*. 2021;8:740460.
- [22] Suenaga M, Shimizu T, Hatanaka T, et al. Experimental study on angle-aware coverage control with application to 3-d visual map reconstruction. In: 2022 IEEE Conference on Control Technology and Applications (CCTA); IEEE; 2022. p. 327–333.
- [23] Schwager M, Julian BJ, Angermann M, et al. Eyes in the sky: Decentralized control for the deployment of robotic camera networks. *Proceedings of the IEEE*. 2011;99(9):1541–1561.
- [24] Hatanaka T, Funada R, Fujita M. Visual surveillance of human activities via gradient-based coverage control on matrix manifolds. *IEEE Transactions on Control Systems Technology*. 2019;28(6):2220–2234.
- [25] Ames AD, Xu X, Grizzle JW, et al. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*. 2016;62(8):3861–3876.
- [26] Bradbury J, Frostig R, Hawkins P, et al. JAX: composable transformations of

- Python+NumPy programs ; 2018. Available from: <http://github.com/google/jax>.
- [27] Ames AD, Xu X, Grizzle JW, et al. Control barrier function based quadratic programs for safety critical systems. IEEE Transactions on Automatic Control. 2017;62(8):3861–3876.
 - [28] Andersen MS, Dahl J, Vandenberghe L, et al. Cvxopt: A python package for convex optimization. Available at cvxopt.org. 2013;54.