ORIGINAL ARTICLE

Open Access

Distributed multi-robot sweep coverage for a region with unknown workload distribution

Muqing Cao¹, Kun Cao¹, Xiuxian Li², Shenghai Yuan¹, Yang Lyu¹, Thien-Minh Nguyen¹ and Lihua Xie^{1*}

Abstract

This paper considers the scenario where multiple robots collaboratively cover a region in which the exact distribution of workload is unknown prior to the operation. The workload distribution is not uniform in the region, meaning that the time required to cover a unit area varies at different locations of the region. In our approach, we divide the target region into multiple horizontal stripes, and the robots sweep the current stripe while partitioning the next stripe concurrently. We propose a distributed workload partition algorithm and prove that the operation time on each stripe converges to the minimum under the discrete-time update law. We conduct comprehensive simulation studies and compare our method with the existing methods to verify the theoretical results and the advantage of the proposed method. Flight experiments on mini drones are also conducted to demonstrate the practicality of the proposed algorithm.

Keywords: Multi-robot systems, Coverage control, Distributed control

1 Introduction

Multi-robot systems are attracting increasing attention from researchers due to their potential in many applications. Some common application scenarios for multi-robot systems are environmental monitoring [1], mine sweeping [2] and maintenance inspection [3]. These tasks are generally formulated as multi-robot coverage problems in literature.

Multi-robot coverage problems can be generally classified into coverage path planning (CPP) [4] and coverage control problems [5]. In multi-robot CPP problems, obstacle-free paths are designed to achieve coverage of an area or volume by the accumulated sensor footprints of the robots [6]. Depending on the CPP tasks, the efficiency metrics may be defined in terms of coverage percentage

[7], time to completion [8], etc. In coverage control problems, distributed control laws for robots are designed to maximize certain coverage criteria, such as coverage frequency [9] or a self-defined coverage function [10]. Sweep coverage was initially addressed as a coverage control problem with the objective of maximizing the detection rate of any events during a periodic coverage mission of a region [11]. In recent years, some literature address the multi-robot sweep coverage problem as a one-time coverage task, which is similar to the CPP problems, with the objective of maximizing coverage percentage [12] or minimizing time of operation [13]. Many existing multi-robot CPP literature employ region or workload partition techniques to minimize total time of operation. In [14], a target region is firstly partitioned using Boustrophedon cellular decomposition [15], then a simple back-and-forth movement is used by the robots to cover each cell. In [16], the authors propose a line formation strategy for multiple unmanned aerial vehicles (UAVs) to conduct target search over a region. The UAVs conduct parallel sweeps until they reach the boundary of the region, where they make

*Correspondence: elhxie@e.ntu.edu.sg

A video showing the main algorithm, simulation and experiment results can be downloaded as supplementary media material. A full-length video can be viewed directly at <https://youtu.be/QFXr0ZWNCtE>.

¹School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore

Full list of author information is available at the end of the article

turns together and move to the next unswept region. Both works try to allocate areas of similar size to the robots and plan paths that minimize repeated coverage by the robots, i.e. travelling to locations that have been covered before.

In our work, the robots are tasked with a collaborative one-time coverage of a region where the workload is not uniformly distributed. A real-world example of a coverage mission with non-uniform workload distribution is the structural inspection, where a region with higher structural load is more prone to small cracks and needs to be inspected in greater detail, hence a robot has to move across the region with lower speed to capture higher-resolution images. In this case, it is not efficient to simply divide the region into areas of equal size and assign one to each robot. Furthermore, we consider that the exact distribution of workload is unknown prior to the operation, hence there is a need for online partitioning of the region to ensure fair sharing of workload among all the robots. Despite the abundance of multi-robot CPP literature, most of them rely on the assumption of uniform workload distribution in the region, without which their algorithms cannot guarantee optimal operation time. Hence their results have limited applicability in our scenario.

On the other hand, we can find more relevant works from the coverage control literature. While most of the existing sweep coverage control literature consider the region to be covered as uniform [12], in [17] and [13], the authors propose a multi-agent sweep coverage algorithm that considers uneven workload distribution in the region. The target region is firstly divided into multiple horizontal stripes, on each stripe, a workload partition problem is solved and each agent is assigned a sub-stripe. All agents take part in the partition process through inter-agent communication. When an agent finishes sweeping a sub-stripe, it has to travel to the next sub-stripe before starting sweeping again. The authors have proved that using the continuous-time algorithm, the workload assigned to each agent on each stripe converges to equal value. However, when the inter-stripe travelling time is taken into consideration, the solution is sub-optimal as the time taken on each stripe is not minimized. Furthermore, the convergence of the discrete-time algorithm with a prescribed sampling period (instead of a sufficiently small one assumed in [13]) is not proven. For application to the physical systems, the performance of the discrete-time algorithm needs to be guaranteed.

In this work, we introduce a distributed multi-robot sweep coverage algorithm that partitions workload according to the combined sweeping time of a stripe and travelling time between the stripes. Compared with the relevant works [13, 17], our algorithm guarantees the convergence to the time-optimal workload assignment on

each stripe, and we provide the proof in the discrete-time formulation without any impractical assumption on sampling period. The efficiency of the proposed approach is verified through simulation studies and comparisons with the existing approaches. Finally, we conduct an indoor experiment using multiple miniature UAVs to verify the practicality of the algorithm. In summary, the main contributions of this work are:

- A distributed multi-robot sweep coverage algorithm that minimizes operation time per stripe by considering both the sweeping time and the transition time between the stripes;
- Novel theoretical analysis to prove the convergence of the discrete time-varying system by invoking the property of Lipschitz continuous gradient;
- Comprehensive simulation studies and comparisons with the existing methods;
- Real-world experiments to demonstrate the feasibility of the propose approach;

The remaining of the paper is organized as follows: Section 2 provides the problem formulation of the multi-robot sweep coverage problem. Section 3 describes the proposed approach and provides the convergence analysis of the proposed algorithm. Section 4 discusses the simulation results. Section 5 describes the experimental setup and discusses the results. Section 6 concludes the paper.

2 Problem formulation

We consider a network of n robots $\mathcal{N} = \{1, 2, \dots, n\}$ working on sweeping a region D enclosed by vertical bounds $y = 0$, $y = l_y$ and horizontal bounds $x = f_1(y)$ and $x = f_2(y)$, as shown in Fig. 1. f_1 and f_2 can be continuous functions or piece-wise linear functions which are more common in a practical scenario. Each robot is equipped with a sweeping device for conducting sweep coverage and a sensor for obtaining workload information from the

environment. The entire region can be divided into several horizontal stripes according to the effective coverage width of the on-board sweeping device. The robots then sweep the region by moving horizontally along the stripes. With an effective sweeping width of d , the total number of stripes is $q = \lceil l_y/d \rceil$, where $\lceil \cdot \rceil$ denotes a ceiling function. The workload amount at position x along the x -axis at the j -th stripe is denoted as $\omega^j(x)$, which can be computed by

$$\omega^j(x) = \int_{(j-1)d}^{jd} \rho(x, y) dy, \quad (1)$$

where $\rho(x, y)$ denotes the workload density at a particular position, bounded by $\underline{\rho} \leq \rho(x, y) \leq \bar{\rho}$, $\forall (x, y) \in D$ and $\rho(x, y) = 0$, $\forall (x, y) \notin D$. The exact value of $\omega^j(x)$ at any location is unknown prior to the sweeping operation, but is known to be bounded by $\underline{\omega} \leq \omega^j(x) \leq \bar{\omega}$ for $\min_{y \in ((j-1)d, jd)} f_1(y) \leq x \leq \max_{y \in ((j-1)d, jd)} f_2(y)$ and $j = 1, 2, \dots, q$.

Within the stripe j , the region assigned to a robot $i \in \mathcal{N}$ is called the i -th sub-stripe, and is defined by two partition marks x_{i-1}^j and x_i^j . Two neighbouring robots' sub-stripes on the same stripe share a common partition mark, as shown in Fig. 1. The first and the last partition marks are fixed at the borders of the region, i.e. $x_0^j = \min_{y \in ((j-1)d, jd)} f_1(y)$ and $x_n^j = \max_{y \in ((j-1)d, jd)} f_2(y)$. When a robot is at stripe j , it is able to acquire the workload information of its subsequent sub-stripe $j+1$ using its on-board sensor. Using a discrete-time formulation, the workload for the i -th robot at the $(j+1)$ -th stripe is

$$m_i^{j+1}(k) = \int_{x_{i-1}^{j+1}(k)}^{x_i^{j+1}(k)} \omega^{j+1}(x) dx, \quad (2)$$

where $x_i^{j+1}(k)$ denotes the i -th partition mark position at the stripe $j+1$ at sampling time k . The amount of workload swept per unit time for each robot is w_s , hence the time to sweep the workload at a particular location x of stripe j is $\frac{\omega^j(x)}{w_s}$, and the allowable moving speed of a robot at this location is $\frac{w_s}{\omega^j(x)}$.

At the end of the sweeping operation of stripe j , each robot has to move from its finishing position to the starting position of its subsequent sub-stripe $j+1$. The movements include a vertical displacement equal to the width of a stripe, and a horizontal displacement equal to the distance between the partition mark of the stripe j and that of the stripe $j+1$. We define the horizontal displacement on stripe $j+1$ as the $(j+1)$ -th transition movement. In order to reduce the amount of transition movement, we define the sweeping direction for the robots as follows: at odd stripes ($j = 1, 3, \dots$), robot i starts sweeping from the $(i-1)$ -th partition mark, x_{i-1}^j and moves towards the i -th partition mark, x_i^j ; at even stripes ($j = 2, 4, \dots$),

robot i starts sweeping from x_i^j and moves towards x_{i-1}^j . Hence, the total distance moved by robot i in the $(j+1)$ -th transition movement, denoted as δ_i^{j+1} , can be calculated as

$$\delta_i^{j+1}(k) = \begin{cases} |x_i^{j+1}(k) - x_i^j|, & j = 1, 3, \dots \\ |x_{i-1}^{j+1}(k) - x_{i-1}^j|, & j = 2, 4, \dots \end{cases} \quad (3)$$

An illustration of the transition movements and sweeping directions can be seen in Fig. 1. During the transition movement, the robots move with a speed of v_t .

We define the $(j+1)$ -th operation cycle for robot i as the sum of the time spent in the $(j+1)$ -th transition movement and the time spent in sweeping the $(j+1)$ -th stripe:

$$\psi_i^{j+1}(k) = \frac{m_i^{j+1}(k)}{w_s} + \frac{\delta_i^{j+1}(k)}{v_t}. \quad (4)$$

In a discrete-time formulation, the total time spent by robot i on stripe $j+1$ is $t_i^{j+1}(k) = \lceil \psi_i^{j+1}(k) \rceil$.

3 Proposed approach

3.1 The multi-robot sweep coverage algorithm

The proposed multi-robot sweep coverage algorithm is illustrated in Algorithm 1 and described in detail here. During the j -th operation cycle, at sampling time k , robot i is able to conduct transition or sweeping actions at stripe j (line 4 to 13), and partition the next stripe $j+1$ by updating the i -th partition mark, $x_i^{j+1}(k)$ (line 14). With the updated partition marks, it also updates its incoming workload $m_i^{j+1}(k)$ and its operation cycle $\psi_i^{j+1}(k)$ (line 15), and communicates with the neighbour to obtain the neighbour's updated operation cycle $\psi_{i+1}^{j+1}(k)$, for the next update cycle (line 16). The above actions continue until the last robot ends its current (j) -th operation cycle (line 3). The robots that have finished sweeping their sub-stripes will stop sweeping but continue to update the partition marks until the last robot finishes its sub-stripe. Then, all robots move up vertically to reach stripe $j+1$ (line 21), before starting a new operation cycle concurrently. The partition marks at the new stripe are initialized with the same values as the final partition marks at the previous stripe (line 2).

Motivated by the consensus algorithm [13, 18], the following discrete-time update law for line 14 is used: for robot $i = 1, 2, \dots, n-1$

$$x_i^{j+1}(k) = x_i^{j+1}(k-1) + \kappa \left(\psi_{i+1}^{j+1}(k-1) - \psi_i^{j+1}(k-1) \right), \quad (5)$$

where κ is a positive real number to be designed. We denote a vector of partition marks as $\mathbf{x}^j(k) = [x_1^j(k), x_2^j(k), \dots, x_{n-1}^j(k)]^\top$, and denote a vector of

Algorithm 1 Distributed Sweep Coverage Algorithm for Robot i

```

1: while true do
2:   Initialize  $x_{i-1}^{j+1}(0) = x_{i-1}^j$ ,  $x_i^{j+1}(0) = x_i^j$ ,  $k = 0$ 
3:   while not all robots have finished sweeping do
4:     if not startedSweeping then
5:       if position of robot within tolerance then
6:         Conduct sweeping action at stripe  $j$ 
7:         startedSweeping=true
8:       else
9:         Conduct transition movement at stripe  $j$ 
10:      end if
11:    else if not finishedSweeping then
12:      Conduct sweeping action
13:    end if
14:    Update  $x_i^{j+1}(k)$  using (5) and send to robot  $i + 1$ 
15:    Update  $m_i^{j+1}(k)$  using (2) and  $\psi_i^{j+1}(k)$  using (4)
16:    Receive  $\psi_{i+1}^{j+1}(k)$  from robot  $i + 1$ 
17:  end while
18:  if this is the last stripe  $j == q$  then
19:    End operation
20:  else
21:    Move vertically to the next stripe,  $j = j + 1$ 
22:  end if
23: end while

```

operation cycles as $\psi^j(k) = [\psi_1^j(k), \psi_2^j(k), \dots, \psi_n^j(k)]^\top$, hence the update law can be represented in vector form as

$$x^{j+1}(k) - x^{j+1}(k-1) = -\kappa \tilde{L} \psi^{j+1}(k-1), \quad (6)$$

where $\tilde{L} \in \mathbb{R}^{(n-1) \times n}$ is the first $n-1$ rows of the Laplacian matrix L that denotes the communication network of the robots.

Remark 1 The update law (6) defines the communication network of the robots to be a directed chain. However,

two-way communication between the neighbouring robots is required for this algorithm, as each robot needs to receive information from and send information to its neighbour $i + 1$.

Assumption 1 The transition speed of the robots is always larger than their moving speed during a sweeping operation, i.e. $v_t > \frac{w_s}{\omega}$.

Assumption 1 is practical and intuitive, as a robot should move faster across a region if it does not need to sweep the region. This ensures that $\psi_i^j(k)$ is a strictly increasing function with respect to $x_i^j(k)$, because

$$\begin{aligned}
 \frac{\partial \psi_i^j(k)}{\partial x_i^j(k)} &= \frac{1}{w_s} \frac{\partial m_i^j(k)}{\partial x_i^j(k)} + \frac{1}{v_t} \frac{\partial \delta_i^j(k)}{\partial x_i^j(k)} \\
 &= \frac{\omega^j(x_i^j(k))}{w_s} + \frac{1}{v_t} \frac{|x_i^j(k)|}{x_i^j(k)} \\
 &\geq \frac{\omega}{w_s} - \frac{1}{v_t} \\
 &> 0.
 \end{aligned}$$

Remark 2 It may seem inefficient to require all the robots to sweep the same stripe concurrently, as this causes robots to travel between sub-stripes without performing useful coverage. However, as we will show in Table 1 in Section 4, due to more balanced workload assignment, our approach outperforms the existing methods that aims to reduce repeated paths.

3.2 Convergence analysis

Since all robots start a new operation cycle concurrently, the time spent by the robots in a stripe is the same and equal to the longest time among the robots $\max_{i \in \mathcal{N}} t_i^j$. We would like to minimize the total time spent by the robots in sweeping the region, therefore, the objective is to minimize $\max_{i \in \mathcal{N}} t_i^j$ for each stripe j . A sufficient condition for

Table 1 Average completion time for four approaches

n	Cell length	Average Time(s)			
		Proposed	Equal Workload	Cell Decomp	Line Formation
4	10	1392.3	1412.2	1464.9	1448.7
	15	2097.1	2127.7	2211.6	2177.7
	20	2805.5	2846.8	2971.7	2911.0
	10	1134.9	1156.0	1205.5	1214.7
5	15	1700.6	1733.3	1825.6	1827.6
	20	2260.2	2303.8	2446.6	2434.1
	10	960.7	976.4	1026.1	1020.0
6	15	1430.6	1462.0	1533.3	1508.9
	20	1877.6	1922.2	2029.8	1994.2

The best performance in each row is shown in boldface

minimizing $\max_{i \in \mathcal{N}} l_i^j$ is to minimize $\max_{i \in \mathcal{N}} \psi_i^j$. Let $\mathbb{S} \in \mathbb{R}^{n-1}$ denote a set of all possible partition mark positions, i.e. $\mathbb{S} = \{[x_1, x_2, \dots, x_{n-1}]^\top \mid 0 < x_i < l_x, i = 1, 2, \dots, n-1, x_1 < x_2 < \dots < x_{n-1}\}$, then the optimisation objective becomes

$$\min_{\mathcal{X} \in \mathbb{S}} \max_{i \in \mathcal{N}} \psi_i^j. \quad (7)$$

We use the following lemma to show that the objective is achieved when all robots share equal operation cycle.

Lemma 1 *The optimisation objective (7) is achieved if and only if all robots have equal operation cycle for stripe j , i.e.*

$$\psi_i^j(k) = \psi_l^j(k), \forall i, l \in \mathcal{N}, i \neq l. \quad (8)$$

Proof Since the value of $\psi_i^j(k)$ is only dependent on $\mathcal{X}^j(k)$, in the following proof we use the expression $\psi_i^j(\mathcal{X}^j)$ to denote the value of the operation cycle under a particular set of partition marks configuration $\mathcal{X}^j \in \mathbb{S}$.

(Necessity) We prove this by contradiction. Without loss of generality, considering an example where $\min_{\mathcal{X}^j \in \mathbb{S}} \max_{i \in \mathcal{N}} \psi_i^j = \psi_1^j(\tilde{\mathcal{X}}^j)$, but $\psi_1^j(\tilde{\mathcal{X}}^j) > \psi_2^j(\tilde{\mathcal{X}}^j)$. In this case, $\psi_1^j(\tilde{\mathcal{X}}^j)$ can be further reduced by shifting the partition mark $\tilde{x}_1^j$ towards zero, due to Assumption 1. There exists $\tilde{\mathcal{X}}^{j-} = [\tilde{x}_1^{j-}, \tilde{x}_2^j, \dots, \tilde{x}_{n-1}^j]^\top$, where $\tilde{x}_1^{j-} < \tilde{x}_1^j$, such that $\psi_1^j(\tilde{\mathcal{X}}^{j-}) < \psi_1^j(\tilde{\mathcal{X}}^j)$ and $\psi_1^j(\tilde{\mathcal{X}}^{j-}) = \max_{i \in \mathcal{N}} \psi_i^j(\tilde{\mathcal{X}}^{j-})$. This means $\min_{\mathcal{X}^j \in \mathbb{S}} \max_{i \in \mathcal{N}} \psi_i^j \neq \psi_1^j(\tilde{\mathcal{X}}^j)$ and hence contradicts with the statement that the objective (7) is achieved.

(Sufficiency) Consider we have a set of partition mark positions $\tilde{\mathcal{X}}^j \in \mathbb{S}$ that satisfies $\psi_1^j(\tilde{\mathcal{X}}^j) = \psi_2^j(\tilde{\mathcal{X}}^j) = \dots = \psi_n^j(\tilde{\mathcal{X}}^j) = \max_{i \in \mathcal{N}} \psi_i^j(\tilde{\mathcal{X}}^j)$. Assume there exists another set of partition mark positions $\mathcal{X}_{\min}^j \neq \tilde{\mathcal{X}}^j \in \mathbb{S}$ such that $\max_{i \in \mathcal{N}} \psi_i^j(\mathcal{X}_{\min}^j) < \max_{i \in \mathcal{N}} \psi_i^j(\tilde{\mathcal{X}}^j)$. If $\mathcal{X}_{\min}^j$ exists, $\tilde{\mathcal{X}}^j$ can be adjusted to reach $\mathcal{X}_{\min}^j$. However, any movement of the partition marks in $\tilde{\mathcal{X}}^j$ will result in an increased $\max_{i \in \mathcal{N}} \psi_i^j(\tilde{\mathcal{X}}^j)$. For example, increasing $\tilde{x}_1^j$ to $\tilde{x}_1^{j+}$ will result in $\psi_1^j(\tilde{\mathcal{X}}^{j+}) > \max_{i \in \mathcal{N}} \psi_i^j(\tilde{\mathcal{X}}^j)$, while reducing $\tilde{x}_1^j$ to $\tilde{x}_1^{j-}$ will result in $\psi_2^j(\tilde{\mathcal{X}}^{j+}) > \max_{i \in \mathcal{N}} \psi_i^j(\tilde{\mathcal{X}}^j)$. This contradicts with the existence of $\mathcal{X}_{\min}^j$. Hence $\max_{i \in \mathcal{N}} \psi_i^j(\tilde{\mathcal{X}}^j)$ is the minimum. $\square$

From Lemma 1, we know we can achieve our objective (7) by showing consensus on the operation cycle $\psi_i^j(k)$. To evaluate the task assignment at the j -th stripe, we introduce the following Lyapunov function

$$H^j(k) = \|\Delta \Psi^j(k)\|^2, \quad (9)$$

where $\Delta \Psi^j(k) = \Psi^j(k) - \mathbf{1}_n \bar{\psi}^j(k)$, $\bar{\psi}^j(k) = \sum_{i \in \mathcal{N}} \psi_i^j(k)/n$ and $\mathbf{1}_n = (1, 1, \dots, 1)^\top$. We introduce the following lemma before the main theorem.

Lemma 2 *The Lyapunov function $H^j(k)$ is differentiable with respect to $\mathcal{X}^j(k)$ and $\nabla_{\mathcal{X}^j} H^j(k)$ is η -Lipschitz continuous for all $k > 0$, where $\eta = \frac{\sqrt{n-1}}{w_s^2} (4\bar{\omega}^2 + 4v\bar{\omega} + v^2)$ for $n = 2$ and $\eta = \frac{2\sqrt{n-1}}{w_s^2} (4\bar{\omega}^2 + 4v\bar{\omega} + \frac{2n-3}{n}v^2)$ for $n \geq 3$.*

Proof

$$\begin{aligned} \nabla_{\mathcal{X}^j} H^j(k) &= \frac{\partial H^j(k)}{\partial \Psi^j(k)} \frac{\partial \Psi^j(k)}{\partial \mathcal{X}^j(k)} \\ &= \frac{2}{w_s} \left(\Omega(k)^\top + vA(k)^\top \right) \Delta \Psi^j(k), \end{aligned} \quad (10)$$

where $\Omega(k) =$

$$\begin{bmatrix} \text{diag} \left(\left\{ \omega^j(x_i^j(k)) \right\}_{i=1}^{n-1} \right) \\ \mathbf{0}_{1 \times (n-1)} \end{bmatrix} - \begin{bmatrix} \mathbf{0}_{1 \times (n-1)} \\ \text{diag} \left(\left\{ \omega^j(x_i^j(k)) \right\}_{i=1}^{n-1} \right) \end{bmatrix},$$

$v = \frac{w_s}{v_t} \cdot A(k) \in \mathbb{R}^{n \times (n-1)}$ is defined differently at different stripes. For odd stripes, $A(k)$ is a matrix with main diagonal elements equal to ones or negative ones and off-diagonal elements equal to zeros, i.e. $A(k) \in \{A \in \mathbb{R}^{n \times (n-1)} \mid a_{ii} \in \{1, -1\}, i = 1, 2, \dots, n-1, a_{ip} = 0, \forall i \neq p\}$. For even stripes, $A(k)$ is a matrix with sub-diagonal elements equal to ones or negative ones and all other elements equal to zeros, i.e. $A(k) \in \{A \in \mathbb{R}^{n \times (n-1)} \mid a_{ip} = \{1, -1\}, i = 2, 3, \dots, n, p = i-1, a_{ih} = 0, \forall h \neq i-1\}$. Clearly, $H^j(k)$ is differentiable for all $k > 0$. We also have

$$\begin{aligned} \nabla_{\mathcal{X}^j}^2 H^j(k) &= \frac{\partial \nabla_{\mathcal{X}^j} H^j(k)}{\partial \Psi^j(k)} \frac{\partial \Psi^j(k)}{\partial \mathcal{X}^j(k)} \\ &= \frac{2}{w_s^2} (\Omega(k) + vA(k))^\top \left(I_n - \frac{1}{n} \mathbf{1}_n \mathbf{1}_n^\top \right) (\Omega(k) + vA(k)). \end{aligned}$$

Clearly, $\nabla_{\mathcal{X}^j}^2 H^j(k)$ is a positive semi-definite matrix. Using matrix properties we can obtain $\|\nabla_{\mathcal{X}^j}^2 H^j(k)\| \leq \sqrt{n-1} \|\nabla_{\mathcal{X}^j}^2 H^j(k)\|_\infty \leq \eta$, where $\eta = \frac{\sqrt{n-1}}{w_s^2} (4\bar{\omega}^2 + 4v\bar{\omega} + v^2)$ for $n = 2$ and $\eta = \frac{2\sqrt{n-1}}{w_s^2} (4\bar{\omega}^2 + 4v\bar{\omega} + (2 - \frac{3}{n})v^2)$ for $n \geq 3$. Hence $\nabla_{\mathcal{X}^j} H^j(k)$ is η -Lipschitz continuous for an $\eta > 0$. $\square$

Assumption 2 $\lambda_2 > 0$, where

$$\lambda_2 = \frac{1}{2} \min_{\substack{\mathcal{X}^j \in \mathbb{S} \\ 1 \leq j \leq q}} \lambda_2 \left((\Omega(k) + vA(k)) \tilde{L} + \tilde{L}^\top (\Omega(k) + vA(k))^\top \right),$$

and $\lambda_2(\cdot)$ returns the second smallest eigenvalue.

Remark 3 If the $\omega^j(x)$ belongs to a continuous space, it is unlikely to check whether Assumption 2 holds. However, in a practical scenario, values of $\omega^j(x)$ are chosen from a set of discrete values, there will be finite possible values of λ_2 to be checked. Hence, before the operation, Assumption 2 can be checked and $\underline{\lambda}_2$ can be obtained.

Having shown that the Lyapunov function $H^j(k)$ has Lipschitz continuous gradient, we are able to provide the condition for the consensus.

Theorem 1 Under the proposed multi-robot sweep coverage algorithm and the update law (5), the optimisation objective (7) is achieved for the stripe j when $k \rightarrow \infty$, under the condition

$$0 < \kappa < \frac{4\lambda_2}{w_s \eta \lambda_L}, \quad (11)$$

where λ_L is the largest eigenvalue of the symmetric matrix $\tilde{L}^\top \tilde{L}$, and η takes the value stated in Lemma 2.

Proof Using Theorem 2.1.5 from [19] and Lemma 2 we can establish the following inequality

$$\begin{aligned} H^j(k+1) - H^j(k) &\leq \nabla_{\mathbf{x}} H^j(k)^\top (\mathbf{x}^j(k+1) - \mathbf{x}^j(k)) \\ &\quad + \frac{\eta}{2} \|\mathbf{x}^j(k+1) - \mathbf{x}^j(k)\|^2 \\ &= -\frac{2\kappa}{w_s} \Delta \boldsymbol{\psi}^j(k)^\top (\Omega(k) + \nu A(k)) \tilde{L} \boldsymbol{\psi}^j(k) \\ &\quad + \frac{\eta \kappa^2}{2} \|\tilde{L} \boldsymbol{\psi}^j(k)\|^2 \\ &= \Delta \boldsymbol{\psi}^j(k)^\top \left(-\frac{2\kappa}{w_s} (\Omega(k) + \nu A(k)) \tilde{L} \right. \\ &\quad \left. + \frac{1}{2} \kappa^2 \eta \tilde{L}^\top \tilde{L} \right) \Delta \boldsymbol{\psi}^j(k) \\ &\leq \left(-\frac{2\kappa}{w_s} \underline{\lambda}_2 + \frac{1}{2} \eta \kappa^2 \lambda_L \right) \|\Delta \boldsymbol{\psi}^j(k)\|^2, \end{aligned} \quad (12)$$

where (6) is used in the second step, $\tilde{L} \boldsymbol{\psi}^j(k) = \tilde{L} \Delta \boldsymbol{\psi}^j(k)$ is used in the third step due to $\tilde{L} \mathbf{1}_n = 0$. Therefore, given a positive real κ that satisfies $\kappa < \frac{4\lambda_2}{w_s \eta \lambda_L}$, we can ensure $H^j(k+1) - H^j(k) \leq 0$. Furthermore, $H^j(k+1) - H^j(k) = 0$ only when $\|\Delta \boldsymbol{\psi}^j(k)\|^2 = 0$, i.e. $\boldsymbol{\psi}_1^j(k) = \boldsymbol{\psi}_2^j(k) = \dots = \boldsymbol{\psi}_n^j(k) = \tilde{\boldsymbol{\psi}}^j(k)$, hence we can conclude that $\lim_{k \rightarrow \infty} H^j(k) = 0$ and the distributed multi-robots system converges to equal operation cycle at the j -th stripe. Then, from Lemma 1, we know that the system will achieve the optimisation objective (7) when $k \rightarrow \infty$. $\square$

Theorem 1 shows that with a proper gain κ , the workload assignment converges to the time-optimal solution regardless of the sampling period of the system. Compared to the relevant work [13], which only shows the

convergence when the sampling period tends to zero, our approach provides a practical guide for choosing an appropriate gain value κ in a practical situation where the sampling period is limited by the communication rate between the robots.

4 Simulation

In this section we discuss the results obtained from the simulation. We design a target region confined within a 10×10 grid map, as shown in Fig. 1. Each grid cell has a height of 3, which is equal to the effective sweeping width d . The length of the grid cell can be varied to adjust the size of the region. The horizontal boundaries f_1 and f_2 are defined by random 7-th order polynomial functions confined within the first and last column of grid cells respectively. The workload density is discretized according to the grids and randomly chosen from the set $\{1, 2, 3\}$. For each robot, the parameters are $w_s = 1$ unit workload/second, $v_t = 2$ unit distance/second. Before the operation, we assume that we have enough information of the first stripe of the region, so that the partition marks for the first stripe are pre-assigned to have equal workload m_i^1 , $i \in \mathcal{N}$ among all robots. Sets of simulation are conducted with varying number of robots and grid cell lengths. We select a range of gain values $\kappa = \{0.001, 0.01, 0.02, 0.05\}$ and plot the operation cycles of the robots in a simulation run where $n = 5$ and grid cell length is 20. As shown in Fig. 2, as time increases, the operation cycles of the robots converge to the same value, which is consistent with Theorem 1. A higher gain value results in faster convergence. We note that for this simulation run, the values of gain do not satisfy the condition (11), but convergence can still be achieved. Therefore, the condition (11) is conservative and in practice, we may use a larger gain value to achieve faster convergence.

To further validate the efficiency of the proposed approach, we compare it with the approach that aims to achieve equal workload per stripe [13] and two approaches from the CPP literature: the cellular decomposition approach with back-and-forth moving pattern [14], and the line formation approach [16]. Illustrations of the two CPP approaches are shown in Fig. 3. Both CPP approaches were not proposed to handle regions with varying workload density, but may still be applied in this scenario by setting the robots' speeds according to the workload amount at a location. The workload swept per second and the transition speed of the robots are fixed for all methods. Each set of simulation consists of 100 randomly generated region profiles. Time is taken from the time when the first robot starts sweeping to the time when the last robot finishes sweeping. In all approaches we ignore the time that the robots move to their starting points.

Fig. 2 Plot of operation cycle over time for a particular stripe. $n = 5$, cell length is 20

Table 1 shows the results of average time to completion with different numbers of robots involved and different grid cell lengths. We can observe that the proposed approach performs the best (uses the least time) in all simulated cases. The equal-workload approach uses 1% to 2% more time than the proposed approach, as the

transition time between the stripes is not minimized. The time increase is more significant when the stripe length increases, due to longer distance between the partition marks and hence more time spent on the transition movements. The CPP approaches generally use 3% to 7% more time than the proposed approach. The average time taken

Fig. 3 Illustration of the existing CPP methods to be compared with. The figures show 4 robots sweeping a region confined within 10×5 cells. The blue dots indicate the starting positions of the robots and the dark green lines indicate the paths of the robots. Both approaches aim to distribute the coverage area among the robots fairly, hence all robots travel the same number of horizontal sweeping paths. In this case, the number of sweeping paths is 12 which is more than the total number of stripes 10, hence each robot's actual sweeping width is $10/12 = \frac{5}{6}d$ and the moving speed is faster than when the full effective sweeping width is used

by both CPP approaches are similar in all cases, because both of them aim to equally partition area and reduce repeated coverage. In the case of non-uniform workload distribution, the total time consumption is dependent on the highest workload amount assigned to any partitioned area, which is random due to the random workload density assignment. Although the proposed approach may result in more repeated paths (where the transition path and sweeping path coincide) than the CPP approaches, the overall performance is better due to more balanced workload sharing among the robots through online partition process. In practice, higher transition speed will further improve the performance of the proposed approach, as the robots will spend less time in the repeated paths.

Figure 4 shows the box plots of the completion time for 100 simulation runs for $n = 4, 5, 6$. We can observe that regardless of the number of robots, the distributions of the completion time for the proposed approach are always narrower than the existing CPP approaches. This means we can expect less variation in the time consumed when dealing with regions of similar sizes. The CPP approaches are less capable of handling the scenarios where the workload is concentrated on a particular stripe, causing the operation time to be prolonged due to a single slowest robot. The time distribution of the equal-workload approach appears to be the same as the proposed approach but shifted slightly higher, because both approaches follow a stripe-by-stripe partition process, but the time taken using the equal-workload approach is always slightly longer than that using the proposed approach in the same simulation run.

5 Experiments

Experiments are conducted using three Ryze Tech Tello mini drones to simulate a facade inspection scenario. In this scenario we have a rectangular facade of size $3.6 \text{ m} \times 1.2 \text{ m}$, which can be divided into 6×12 cells with

Fig. 4 Box plot showing distribution of completion time for $n = 4, 5, 6$, cell length is 20

Fig. 5 System overview for the i -th drone in the experiments

0.2 m height and 0.3 m length. Each cell is either made of concrete ($\omega^j(x) = 1.0$ unit) or glass ($\omega^j(x) = 1.3$ unit). Figure 5 shows the software and hardware overview of the i -th drone in the experiments. Due to weight and size constraints of the mini drones, we cannot install on-board computers to the drones. Instead, we use three laptops to form a distributed system. The software is implemented in Robot Operating System (ROS) framework. Each laptop runs two ROS nodes, one executes the proposed algorithm and the other provides the interface with the mini drone. The codes for the algorithm nodes are the same among all laptops. The communication between the algorithm nodes is established through a Wi-Fi network. The distribution of $\omega^j(x)$ in the region is represented as a set of parameters that can be accessed by the algorithm nodes. Drone parameters are $\kappa = 0.02$, $w_s = 0.12$ unit/s, $v_t = 0.2$ m/s. Each drone knows its initial relative positions to the neighbours, and it estimates its position using the on-board optical flow sensor.

We make slight modifications to Algorithm 1 to adapt to the practical scenario. The main algorithm node is implemented with two independent loops, a faster loop (2 hz) that updates the partition marks (line 14 to 16) and a slower loop (0.4 hz) that computes the position change commands for the drones during the sweeping and transition action (line 4 to 13). Due to the constraints of the inner-loop dynamics and the software limit of Tello mini drones, the drones cannot respond to a position change command arbitrarily fast [20]. In our experiments, sending position change commands at higher rates does not improve the position tracking performance. On the other hand, the partition marks updates can be executed at a faster rate to achieve faster convergence.

Multiple flight tests are conducted and one of them is shown in the supplementary video¹. We can observe that the partition marks for the stripes are adjusted such that the operation cycles converge, as shown in Fig. 6. Hence, the algorithm is effective even with a small area. Figure 7 shows the final partition result on the target region and the flight trajectories of the drones. Despite the position inaccuracies, the target region is covered to a large extent.

¹<https://youtu.be/QFXr0ZWNCtE>

Therefore, the proposed approach is demonstrated to be practical.

6 Conclusion

In this paper, we proposed a distributed multi-robot sweep coverage algorithm that can be applied to regions with unknown workload distribution. The discrete-time update law considers both the sweeping time of a stripe and the transition time to the stripe. By constructing a Lyapunov function and invoking the property of Lipschitz continuous gradient, we proved that the multi-robot time-varying system achieves consensus and the workload assignment on each stripe converges to the time-optimal assignment. Simulation under different settings was conducted to verify the theoretical results. Comparisons with the existing techniques show that the proposed approach allows the robots to complete the coverage task in a shorter time. Experiments simulating a facade inspection scenario was conducted to demonstrate the practicality of the proposed approach.

There are certain limitations of the current approach that will be addressed in the future works. Firstly, the current approach relies on accurate localization of the robots in the environment to conduct sensing and coverage. Future work will consider coverage subject to localization

error or using relative localization techniques. Secondly, the algorithm can be improved to consider limited sensing range of the robots, in which case the robots are not able to acquire the workload information of their subsequent sub-stripes entirely. Thirdly, flight experiments can be conducted with more capable multirotors that can conduct on-board sensing, computation and communication.

Authors' contributions

MC contributes to the theoretical formulation, software implementation and experiments of this work. KC, XL and LX contribute to the theoretical formulation of this work. SY, YL and T-MN contribute to the experiments of this work. The author(s) read and approved the final manuscript.

Funding

The authors declare no funding for this work.

Availability of data and materials

The data of this work can be made available to the interested researchers upon request.

Code availability

The code of this work is not available to the research community.

Declarations

Competing interests

The authors declare no conflicts of interest for this work.

Author details

¹School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore. ²Department of Control Science and Engineering and Institute for Advanced Study, Tongji University, Shanghai 200092, China.

Received: 8 July 2021 Accepted: 1 October 2021

Published online: 02 December 2021

References

1. M. S. Couceiro, D. Portugal, J. F. Ferreira, R. P. Rocha, in *2019 IEEE/SICE International Symposium on System Integration (SII)*. SEMFIRE: Towards a new generation of forestry maintenance multi-robot systems, (2019), pp. 270–276. <https://doi.org/10.1109/SII.2019.8700403>
2. A. J. Healey, in *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No. 01CH37228)*. Application of formation control for multi-vehicle robotic minesweeping, vol. 2, (2001), pp. 1497–1502 vol.2. <https://doi.org/10.1109/CDC.2001.981106>
3. S. Bernardini, F. Jovan, Z. Jiang, S. Watson, A. Weightman, P. Moradi, T. Richardson, R. Sadeghian, S. Sareh, in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '20)*. A Multi-Robot Platform for the Autonomous Operation and Maintenance of Offshore Wind Farms (International Foundation for Autonomous Agents and Multiagent Systems, Richland, pp. 1696–1700
4. E. Galceran, M. Carreras, A survey on coverage path planning for robotics. *Robot. Auton. Syst.* **61**(12), 1258–1276 (2013)
5. A. V. Savkin, T. M. Cheng, Z. Xi, F. Javed, A. S. Matveev, H. Nguyen, *Introduction*. John Wiley & Sons, Ltd, (2015), pp. 1–7. <https://doi.org/10.1002/9781119058052.ch1>
6. T. M. Cabreira, L. B. Brisolara, P. R. Ferreira Jr, Survey on coverage path planning with unmanned aerial vehicles. *Drones*, **3**(1), 4 (2019)
7. L. Doitsidis, S. Weiss, A. Renzaglia, M. W. Achtelik, E. Kosmatopoulos, R. Siegwart, D. Scaramuzza, Optimal surveillance coverage for teams of micro aerial vehicles in gps-denied environments using onboard vision. *Auton. Robot.* **33**(1–2), 173–188 (2012)
8. G. S. Avellar, G. A. Pereira, L. C. Pimenta, P. Iscold, Multi-uav routing for area coverage and remote sensing with minimum time. *Sensors*, **15**(11), 27783–27803 (2015)
9. L. Wu, Y. Xiong, M. Wu, Y. He, J. She, A task assignment method for sweep coverage optimization based on crowdsensing. *IEEE Internet Things J.* **6**(6), 10686–10699 (2019)

10. J. D. Kelly, D. M. Lofaro, D. Sofge, in *2020 17th International Conference on Ubiquitous Robots (UR)*. Persistent Area Coverage for Swarms Utilizing Deployment Entropy with Potential Fields, (2020), pp. 479–486. <https://doi.org/10.1109/UR49135.2020.9144917>
11. D. W. Gage, Command Control for Many-Robot Systems. AUVS-92, Nineteenth Ann AUVS Tech. Symp. **10**(June), 15 (1992)
12. M. Shi, K. Qin, J. Liu, Cooperative multi-agent sweep coverage control for unknown areas of irregular shape. *IET Control Theory Appl.* **12**(14), 1983–1994 (2018)
13. C. Zhai, Sweep coverage of discrete time multi-robot networks with general topologies. *Kybernetika.* **50**(1), 19–31 (2014)
14. I. Rekleitis, A. P. New, E. S. Rankin, H. Choset, Efficient boustrophedon multi-robot coverage: an algorithmic approach. *Ann. Math. Artif. Intell.* **52**(2-4), 109–142 (2008)
15. H. Choset, P. Pignon, in *Field and Service Robotics*, ed. by A. Zelinsky. Coverage Path Planning: The Boustrophedon Cellular Decomposition (Springer, London, 1998), pp. 203–209
16. P. Vincent, I. Rubin, in *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC '04)*. A Framework and Analysis for Cooperative Search Using UAV Swarms (Association for Computing Machinery, New York, 2004), pp. 79–86. <https://doi.org/10.1145/967900.967919>
17. C. Zhai, Y. Hong, Decentralized sweep coverage algorithm for multi-agent systems with workload uncertainties. *Automatica.* **49**(7), 2154–2159 (2013). <https://doi.org/10.1016/j.automatica.2013.03.017>
18. R. Olfati-Saber, J. A. Fax, R. M. Murray, Consensus and cooperation in networked multi-agent systems. *Proc. IEEE.* **95**(1), 215–233 (2007)
19. Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*. (Springer US, Boston, 2004). https://doi.org/10.1007/978-1-4419-8853-9_2
20. G. Cai, B. M. Chen, T. H. Lee, *Unmanned Rotorcraft Systems*. (Springer London, London, 2011), pp. 161–178. https://doi.org/10.1007/978-0-85729-635-1_8

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.