IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 13 December 2023, accepted 26 December 2023, date of publication 8 January 2024,
date of current version 12 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3351052

RESEARCH ARTICLE

Adaptive Centroidal Voronoi Tessellation With Agent Dropout and Reinsertion for Multi-Agent Non-Convex Area Coverage

KANGNEOUNG LEE[1], (Student Member, IEEE), AND KIJU LEE[1,2], (Member, IEEE)
[1]Department of Mechanical Engineering, Texas A&M University, College Station, TX 77843, USA
[2]Department of Engineering Technology and Industrial Distribution, Texas A&M University, College Station, TX 77843, USA

Corresponding author: Kiju Lee (kiju.lee@tamu.edu)

This work was supported by the National Institute of Food and Agriculture, United States Department of Agriculture (USDA-NIFA), under Award 2021-67021-35959.

ABSTRACT Voronoi diagrams are widely used for area partitioning and coverage control. Nevertheless, their utilization in non-convex domains often necessitates additional computational procedures, such as diffeomorphism application, geodesic distance calculations, or the integration of local markers. Extending these techniques across diverse non-convex domains proves challenging. This paper introduces the adaptive centroidal Voronoi tessellation (αCVT) algorithm, which combines iterative centroidal Voronoi tessellation (iCVT) with an innovative agent dropout and reinsertion strategy. This integration aims to enhance area coverage control in non-convex domains while maintaining adaptability across varied environments without the need for complex computational processes. The efficacy of this approach is validated through simulations involving non-convex domains with disjoint target areas, obstacles, and shape constraints for both homogeneous and heterogeneous agents. Additionally, the αCVT algorithm is extended for real-time coverage control scenarios. Performance metrics are employed to assess the distribution of partitioned Voronoi regions and the overall coverage of the target areas. Results demonstrate improved performance compared to methods that do not incorporate the agent dropout and reinsertion strategy.

INDEX TERMS Area partitioning, non-convex area coverage control, multi-agent system, Voronoi tessellation.

I. INTRODUCTION

In coverage control problems involving a multi-agent system, the entire area is usually partitioned into subareas allocated to individual agents. In an ideal scenario for homogeneous agents, the target domain is divided equally among them. However, subarea assignments for heterogeneous agents may vary based on their coverage capabilities. Consideration must also be given to areas with different priorities within the target domain. Several existing methods for multi-agent coverage control include ergodic dynamics-based methods [1], [2], [3], K-means clustering [4], and methods based on Voronoi diagrams. Ergodic dynamics-based methods employ the Fourier Transform to calculate the ergodic metric, representing coverage performance [1], [2], [3]. K-means clustering-based methods rely on the agents' positions relative to the target locations in the domain [4].

The Voronoi diagram has found extensive use in mobile sensing networks [5], [6], [7], [8], [9], exploration strategies [10], [11], [12], [13], [14], and task allocation methodologies [15], [16], [17], [18]. Many existing works on Voronoi-based coverage control focused on convex domains. A coverage control law employing homogeneous agents was derived for convex domains in [5]. Another work demonstrated how homogeneous agents could perform adaptive information sampling for coverage control [12].

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

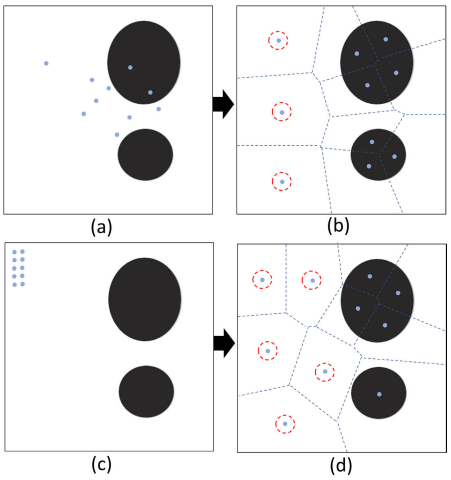

FIGURE 1. *i*CVT used for covering two disjoint targets shown in black in a bounded open environment: (a) initial positions of 10 agents, (b) the partitioning result, (c) different initial positions of 10 agents, and (d) the partitioning result.

In [6], an anisotropic sensor model was integrated into a coverage control scheme. Furthermore, previous research has considered heterogeneous agents in convex domains, considering variations in sensing capabilities [7], [9], [19], agents' sizes [20], and speeds [21].

Coverage control strategies for non-convex domains have also been explored. In [22], researchers approximated a non-convex domain to a convex environment through diffeomorphism. Another study [23] integrated diffeomorphism with a time- and space-differentiable density function, representing the relative importance of points in a given domain. Utilizing geodesic distance instead of Euclidean distance in Voronoi partitioning was highlighted in a different study [8], where this method extended Voronoi regions into non-visible areas obscured by obstacles. Additionally, multi-robot exploration in non-convex domains was tackled in [13], employing local Voronoi decomposition and markers to indicate explored areas. Combining Voronoi tessellation with deep reinforcement learning was explored in [14], where the upper-level layer determined agents' waypoints using Voronoi tessellation and the lower-level layer calculated path and control inputs for each agent. Another study [24] adopted a local search algorithm, discretizing continuous domains by sampling multiple points. Several studies introduced the concept of density functions for coverage control in non-convex environments. For instance, in [25], agents' positions locally altered the environment's density function. Similarly, a time-varying density function in [26] enabled the agents to explore invisible regions behind obstacles. This work employed an adaptive static coverage scheme similar to the convex weighted Voronoi diagram introduced in [27].

The existing Voronoi-based methods designed for non-convex domains exhibit several limitations. Firstly, implementing coverage control via local Voronoi decomposition necessitates markers to signify whether specific areas have been explored or not [10], [13]. Secondly, these methods typically entail high computational complexity. For instance, the geodesic-based approach requires computing hyperbolic arcs for each agent, resulting in increased computational demands [8]. Algorithms that integrate diffeomorphism techniques [22], [23] may encounter an ill-conditioned Jacobian matrix [22]. Additionally, the utilization of deep reinforcement learning, as explored in [14], demands substantial time and resources for the training process. Most notably, Voronoi-based methods, as highlighted in [1], exhibit poor performance when dealing with domains that contain disjoint target areas.

In Fig. 1, two disjoint target areas depicted in black are subject to coverage by mobile agents represented as blue dots. This environment serves as an illustrative example akin to the scenario presented in [2] and [3]. We employed a strictly positive density function ρ to establish the priority of the target area(s) across the domain. Unlike several previous works [5], [12], [19], [20], [21], the density function in this example is not smooth. The iterative centroidal Voronoi tessellation (*i*CVT), the classic Lloyd algorithm [28] widely adopted for coverage control, was applied for two different initial positions of the agents. With the arbitrary initial positions displayed in Fig. 1a, the *i*CVT algorithm resulted in three agents located outside the target areas, four within the larger target, and the remaining three in the smaller one (Fig. 1b). When agents were initially positioned at the upper-left corner (Fig. 1c), five agents were placed outside the target areas, four in the larger target, and one in the smaller one (Fig. 1d). This example highlights two key limitations of *i*CVT. Firstly, the method does not guarantee the placement of all agents within the target areas. Secondly, the coverage outcomes are significantly influenced by the initial positions of the agents.

This paper presents an innovative Voronoi-based partitioning algorithm named adaptive centroidal Voronoi tessellation (αCVT), which combines the *i*CVT and projection algorithms with a novel *agent dropout and reinsertion* strategy. The aim is to compel agents to optimize coverage control performance based on the proposed metrics that assess the distribution of individual Voronoi regions and coverage of target areas. This study focuses on two types of non-convex environments: i) those featuring obstacles and ii) environments with disjoint targets. In scenarios with obstacles, these impediments disrupt agents' coverage and mobility. Consequently, mobile agents must navigate to the centroids of assigned Voronoi regions while circumventing obstacles to maximize coverage. Environments comprising disjoint target areas necessitate agents to position themselves within these defined target regions. It is important to note that the presented algorithm operates under the assumption of a centralized system, aligning with similar centralized approaches as reviewed in previous relevant works [15], [16].

The performance evaluation of the αCVT algorithm encompassed four distinct non-convex area coverage cases: 1) homogeneous agents tasked with covering disjoint target

areas, 2) homogeneous agents operating within a target domain obstructed by obstacles, 3) homogeneous agents tackling a target domain with shape constraints, and 4) heterogeneous agents deployed in non-convex target domains. To demonstrate the algorithm's applicability in real-time coverage control scenarios utilizing mobile agents, we adopted a control strategy previously utilized in [3], [11], [14], and [24]. This strategy involves a two-layer control algorithm. The upper layer computes goal points for the agents, while the lower layer generates the desired path and controls their motion. To assess the effectiveness of this real-time coverage control scheme, we conducted simulations using ground robots that emulate mobile agents, implemented through Unity and Robot Operating System (ROS).

The core innovation of the presented work is in the agent dropout/reinsertion technique and its versatility in addressing diverse area coverage problems. Unlike other methodologies requiring environment-specific control laws derived on a case-by-case basis, this method eliminates the need for such mathematical derivations. Notably, the agent dropout/reinsertion technique enhances performance without imposing additional computational burdens, distinguishing itself from methods like geodesic disc [8], diffeomorphism [22], [23], and control policy from deep reinforcement learning [14]. Furthermore, most prior Voronoi-based approaches have predominantly focused on environments featuring obstacles, often overlooking scenarios with multiple disjoint targets. In contrast, the proposed algorithm demonstrates effectiveness in addressing both types of non-convex environments.

The rest of the paper consists of the following. Section II first summarizes the *i*CVT [5], the projection algorithm [11], and the density definition of the non-convex environment [22]. This section then introduces the αCVT algorithm integrating the agent dropout and reinsertion process. Section III describes the evaluation settings and delineates the attained results from the experiments. Section IV details the extension of the αCVT algorithm for real-time coverage control scenarios involving mobile agents. Finally, Section V provides a conclusive summary and discussion regarding potential avenues for future research.

II. ADAPTIVE CENTROIDAL VORONOI TESSELLATION

This section presents the area partitioning method combining the *i*CVT and the projection algorithm with the concept of agent dropout and reinsertion, referred to as αCVT. We begin by reviewing the *i*CVT, the projection algorithm, and the density definition of the non-convex environment.

A. REVIEW OF ITERATIVE CVT

We first briefly review the *i*CVT (a.k.a. Lloyd algorithm) described in [5]. In a closed region S of a $2D$ convex domain $\mathbb{R}^2$, let $A = \{a_1, a_2, \cdots, a_k\}$ be a set of agents located in the interior of S, and $P = \{p_1, p_2, \cdots, p_k\}$ be a set of agents' positions. Voronoi partition of S based on P is performed as follows:

$$V_i = \{s \in S \mid ||s - p_i|| \leq ||s - p_j||, \forall j \neq i\} \tag{1}$$

where V_i is the Voronoi region of the i^{th} agent, s is a point in S, and $||\cdot||$ denotes the Euclidean norm. In [5], the objective function for locational optimization is given by

$$H(P) = \sum_{i=1}^{k} \int_{V_i} ||s - p_i||^2 \rho(s) ds \tag{2}$$

where k is the total number of agent, and ρ is a density function. The elements of P determine the value of the objective function.

The mass (M_{V_i}), centroid (C_{V_i}), and polar moment of inertia about p_i (J_{V_i,p_i}) corresponding to the i^{th} Voronoi region are then obtained as

$$M_{V_i} = \int_{V_i} \rho(s) ds \tag{3a}$$

$$C_{V_i} = \frac{1}{M_{V_i}} \int_{V_i} s\rho(s) ds \tag{3b}$$

$$J_{V_i,p_i} = \int_{V_i} ||s - p_i||^2 \rho(s) ds \tag{3c}$$

Using the parallel axis theorem, (3c) can be expressed as

$$J_{V_i,p_i} = J_{V_i,C_{V_i}} + M_{V_i} ||p_i - C_{V_i}||^2 \tag{4}$$

where $J_{V_i,C_{V_i}}$ is the polar moment of inertia of the Voronoi region V_i about its centroid. Then, $H(P)$ in (2) and its derivative with respect to p_i are given by

$$H(P) = \sum_{i=1}^{k} J_{V_i,C_{V_i}} + \sum_{i=1}^{k} M_{V_i} ||p_i - C_{V_i}||^2 \tag{5}$$

$$\frac{\partial H(P)}{\partial p_i} = 2M_{V_i}(p_i - C_{V_i}) \tag{6}$$

Therefore, $H(P)$ is minimized when all the agents are located at the centroids of their Voronoi regions. This statement is formulated as [5]

$$C_{V_i} = argmin_{p_i}(H(P)), \quad \forall a_i \in A \tag{7}$$

Since C_{V_i} is the centroid of V_i, it is always located within V_i. As described in [5], the iterative scheme can guarantee global convergence if the domain is convex and unchanged. The detailed steps are as follows: 1) calculate Voronoi regions (V_i) corresponding to each p_i for all agents; 2) calculate M_{V_i} and C_{V_i} corresponding to each V_i; and 3) set the agents' location to their centroids (C_{V_i}), and return to the step 1). This process is *i*CVT, leading to finding the agents' final positions with global convergence.

B. REVIEW OF PROJECTION ALGORITHM AND DENSITY FUNCTION

If the environment contains obstacles, the target position of an agent cannot be within the obstacle regions. In that case, this agent's target position must be relocated. The projection

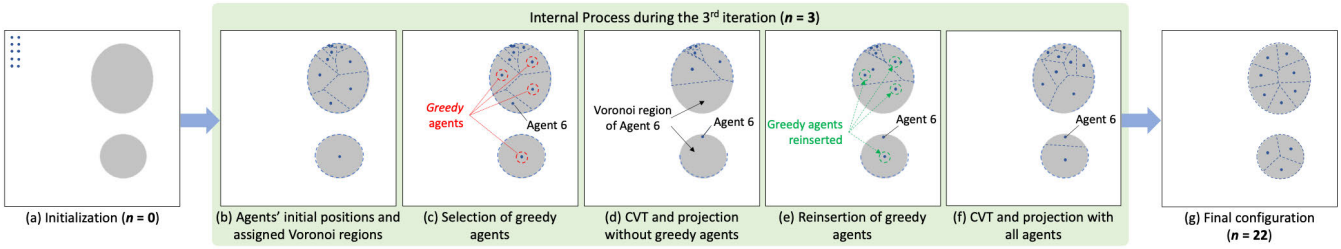

FIGURE 2. Procedure of αCVT-based coverage control using 10 homogeneous agents for two disjoint targets: (a) the initial configuration, (b) - (f) step-by-step procedure detailing the agent dropout and reinsertion processes, and (g) the final configuration after 22 iterations.

algorithm used in [11] moves such target position to the closest non-obstacle point by using

$$t_i^{real} = argmin_{s\in S_f} ||s - t_i^{virt}||, \quad t_i^{virt} = C_{V_i} \tag{8}$$

where t_i^{virt} and t_i^{real} are the virtual and real target positions for the i^{th} agent, respectively. S_f is the obstacle-free space in domain S. In [11], the closest non-obstacle point is identified by searching the environment. In our presented work, we adopted a counterclockwise direction for searching. Also, the target position of the agent is relocated to the closest point with $\rho > 0$ if the current target position has $\rho = 0$.

In [22], a non-convex environment with obstacles was defined using a density function, assigning $\rho = 0$ for all points within the obstacle regions and strictly positive values for the rest. We slightly modified this definition by setting $\rho = 0$ for all points outside of the coverage targets as well as the obstacle regions. This modification allows all agents to be located within the target areas when combined with the projection algorithm. This definition also simplifies the CVT process by excluding non-target areas and obstacle regions from calculating M_{V_i}, and C_{V_i} in (3).

C. AGENT DROPOUT AND REINSERTION TECHNIQUE

The agent dropout and reinsertion technique drew inspiration from the dropout method introduced for neural networks [29]. This approach involves randomly deselecting nodes within neural networks. A similar tactic was employed in a large neighborhood search method [30], where the removal and reinsertion tasks among multiple agents were driven by a cost function in conjunction with a random process. The method presented in this paper differs from these prior approaches by employing a distinct strategy to determine which agent to drop out, rather than relying on a random selection process. This dropout selection strategy is based on the masses of individual Voronoi regions. Specifically, for homogeneous agents, those with a relatively higher M_{V_i} compared to others are temporarily dropped out and identified as '*greedy*' agents. Following the dropout phase, the CVT is performed for the remaining agents.

The pseudocode in Algorithm 1 outlines the agent dropout process. To begin, the algorithm identifies greedy agents in Line 3 using a threshold value, τ_{drop}. This threshold is important in determining these greedy agents. When all

Algorithm 1 AgentDropout Function

1: $A_{normal} = \phi, A_{greedy} = \phi$
2: **for** $a_i \in A$ **do**
3: **if** $M_{V_i} \geq \tau_{drop}$ **then**
4: $A_{greedy} \leftarrow a_i$
5: **else**
6: $A_{normal} \leftarrow a_i$
7: **end if**
8: **end for**
9: Determine $V_i, \forall a_i \in A_{normal}$ from Eq. 1
10: Calculate $p_i, \forall a_i \in A_{normal}$ from Eq. 7 and Projection

agents show equal masses of their Voronoi regions, all are categorized as greedy agents if $\tau_{drop} = \overline{M}_V$, where $\overline{M}_V$ represents the average mass of the Voronoi regions, as defined in (9). Therefore, τ_{drop} should be greater than $\overline{M}_V$. In our selection, we opted for $\tau_{drop} = \frac{k}{k-1}\overline{M}_V$, where k denotes the number of agents. Should M_{V_i} for all agents fall below τ_{drop}, the algorithm terminates the agent dropout and reinsertion process.

D. ADAPTIVE CVT ALGORITHM

We propose a performance metric tailored for homogeneous agents, quantifying the average deviation of the individual masses allocated to k agents. This metric, denoted as I, is calculated as:

$$I = \frac{1}{k}\left(\sum_{i=1}^{k} \left|\frac{M_{V_i}}{\overline{M}_V} - 1\right|\right), \quad \overline{M}_V = \frac{1}{k}\sum_{i=1}^{k} M_{V_i} \tag{9}$$

The objective is to achieve evenly partitioned target areas among agents, thereby minimizing I. When all agents possess equal masses of their Voronoi regions (i.e., $M_{V_i} = \overline{M}_V$ for all $i = 1, \cdots, k$), the metric I equals zeros. This metric also facilitates assigning individual agents at the centroids of the assigned Voronoi regions, aligning with the objective of CVT. To determine termination for the agent dropout process, we define ΔI using:

$$\Delta I = I_{t-1} - I_t \tag{10}$$

Here, a low pass filter is applied to smooth the trajectory of ΔI.

Algorithm 2 Adaptive CVT Using Agent Dropout and Reinsertion

```
1:  Initialize p_i, ∀i
2:  Determine V_i, ∀i from Eq. 1
3:  Update p_i, ∀i from Eq. 7 and Projection
4:  DropoutOff = False
5:  while do
6:      if DropoutOff == False then
7:          AgentDropout()
8:      end if
9:      Determine V_i, ∀i from Eq. 1
10:     Calculate p_i, ∀i from Eq. 7 and Projection
11:     Calculate ΔI from Eq. 9 and 10
12:     if ΔI < ϵ then
13:         DropoutOff = True
14:     end if
15: end while
```

Algorithm 2 outlines the αCVT algorithm integrating *i*CVT, projection, and the agent dropout and reinsertion process. When the 'DropoutOff' flag is 'False,' the algorithm invokes the 'AgentDropout' function at Line 7. This function utilizes Algorithm 1 to identify greedy agents. Subsequently, CVT is executed without these identified greedy agents (from Line 9 to 10 in Algorithm 1). Following the 'AgentDropout' function call, the algorithm reinserts the previously identified greedy agents and proceeds to perform CVT with all agents (from Line 9 to 10 in Algorithm 2). The termination condition for the dropout process relies on $\Delta I < \epsilon$, where ΔI signifies the change in the performance metric. If this change falls below the threshold ϵ, it indicates negligible improvement in the metric, prompting the dropout process to terminate. The value of ϵ must be greater than 0 to ensure the termination of the agent dropout function; in our case, we selected $\epsilon = 0.002$.

Fig. 2 illustrates the internal process of the proposed algorithm with the homogeneous agents and two disjoint targets. Within the figure, non-target regions are shown in white ($\rho = 0$), and the two target areas are represented in gray ($\rho = 0.235$). Initially, Fig. 2a shows the simulation's starting configuration. The agent dropout and reinsertion algorithm remains continuously active until its termination. The following sub-figures (Fig. 2b-f) show the internal process during the third iteration of the algorithm. Fig. 2b displays the initial positions of agents and their corresponding Voronoi regions at the onset of this iteration. Subsequently, in Fig. 2c, the algorithm identifies and selects four greedy agents based on the specified τ_{drop} value. For instance, Agent 6 initially occupies the upper target area, as depicted in the figure. Following the removal of these greedy agents, the algorithm conducts CVT in conjunction with the projection algorithm (Fig. 2d). Consequently, Agent 6 relocates to the lower target area. Reinsertion of the previously removed greedy agents takes place (Fig. 2e), returning them to their original positions. Post-reinsertion, the algorithm conducts another CVT with projection (Fig. 2f). Once the dropout and reinsertion process is deactivated, the algorithm exclusively performs CVT with the projection algorithm until reaching the final configuration (Fig. 2g). The step-by-step process is summarized below:

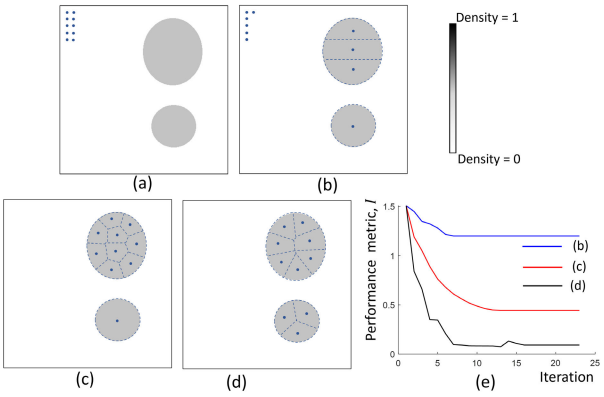

FIGURE 3. Two disjoint target areas with $\rho = 0.235$: (a) initial state with 10 homogeneous agents, and the results from (b) *i*CVT, (c) *i*CVT+Projection [11], (d) αCVT, and (e) comparisons of *I* values over iterations.

1) Perform initial CVT and projection for all agents to be located within the target areas in the domain.
2) Determine greedy agents and drop them out.
3) Perform CVT and projection for the remaining agents.
4) Reinsert the greedy agents.
5) Repeat CVT and projection.
6) If the dropout process terminates, return to 5); otherwise, return to 2).

III. ALGORITHM APPLICATIONS AND EVALUATIONS

This section applies the algorithm to particular applications and assesses the performance of αCVT in different experimental simulation settings, comparing it against existing methods. The simulations were executed within ROS, where the agents were represented as dots moving freely within a 2D environment. The αCVT algorithm is compared with '*i*CVT+Projection' [11], which employs an iterative scheme of executing CVT and the projection algorithm at each iteration. In Section III-B, further comparisons were conducted, extending the evaluation by contrasting the proposed αCVT method with other recent works presented in [25], [26], and [31]. We note that the size of the 2D domains in these evaluations was fixed at 200 × 200 pixels, standardizing the evaluation criteria and facilitating direct comparisons between the different methodologies assessed within this section.

A. HOMOGENEOUS AGENTS FOR DISJOINT TARGETS

We conducted three sets of experiments with homogeneous agents for disjoint targets in an open, bounded environment where the agents can freely move around. Open, non-target space with $\rho = 0$ is shown in white, and target areas are shaded in grayscale, where a darker shade indicates a higher

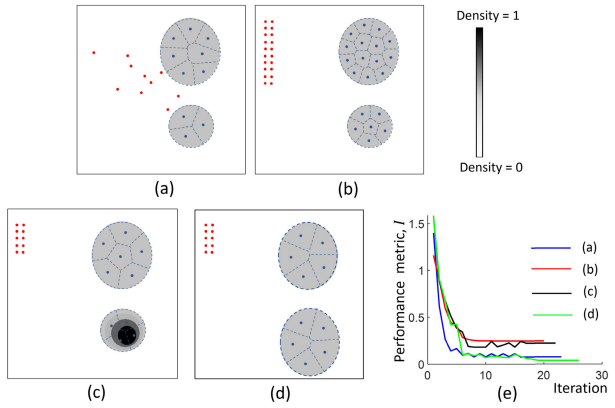

FIGURE 4. Simulation results (blue dots) for (a) different initial positions (red dots) of 10 homogeneous agents, (b) 20 homogeneous agents, (c) a different target priority density distribution, (d) two identical disjoint targets, and (e) the change of performance metric I over iterations.

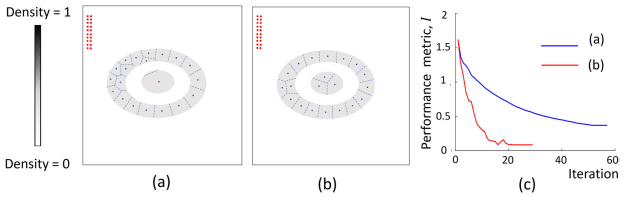

FIGURE 5. Simulation results for a bullseye-shaped target: (a) using *i*CVT+Projection [11] and (b) using αCVT, (c) change of performance metric I over iterations.

density ($0 < \rho \leq 1$). The same color scheme is applied to all simulations.

The *first* experimental scenario involved ten agents for two disjoint target areas with $\rho = 0.235$. Fig. 3 shows (a) the simulation setting and (b-d) results from *i*CVT, *i*CVT+Projection, and αCVT, respectively. Ten agents were initially located near the upper-left corner (Fig. 3a). When using the *i*CVT method, only four agents were placed at the target areas, and six agents remained at the initial locations without the assigned Voronoi regions (Fig. 3b). The *i*CVT+Projection method placed all agents within the two target areas, where nine were aggregated in the larger target and only one in the smaller one (Fig. 3c). The proposed αCVT method resulted in all agents located in the two targets with evenly distributed Voronoi regions (Fig. 3d). We measured the changes of I over iterations for the three cases (Fig. 3e). Using *i*CVT shown in (b), I converged to 1.2; using *i*CVT+Projection shown in (c), it converged to 0.44; using αCVT shown in (d), I reached as low as 0.046. The convergence speed using αCVT was also the fastest.

The *second* set of experiments adopted αCVT for the following conditions: different starting positions of the agents (Fig. 4a), a different number of agents (Fig. 4b), two targets with different densities (Fig. 4c), and two identical targets (Fig. 4d). In all four cases, αCVT showed reliable performance by effectively assigning evenly partitioned Voronoi regions to all agents. Corresponding performances measured by I over iterations are shown in Fig. 4e. The I

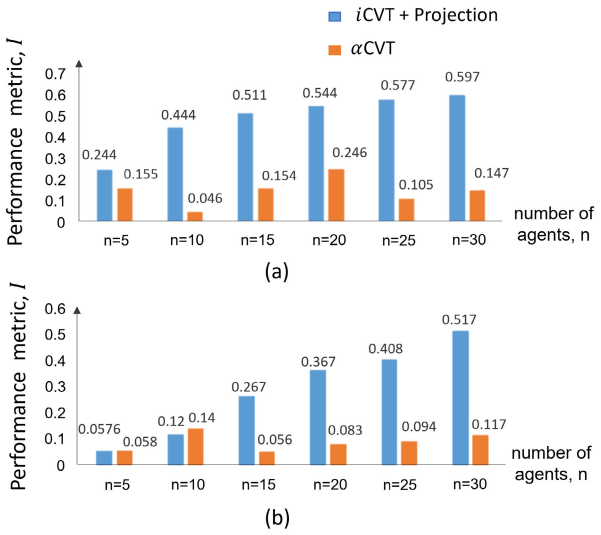

FIGURE 6. The performance results measured by I over increasing n at the final state with different agent numbers for (a) the domain including two disjoint target areas and (b) the bullseye-shaped target.

value converged to 0.076 in (a), 0.246 in (b), 0.22 in (c), and 0.039 in (d).

The *third* set of experiments considered 20 homogeneous agents for a bullseye-shaped target in an open, bounded environment (Fig. 5), benchmarking the environment introduced in [3]. We employed (a) *i*CVT+Projection and (b) αCVT for comparison. Both algorithms placed all agents within the target areas, but αCVT showed a significantly improved distribution of the individually assigned Voronoi regions. Moreover, αCVT resulted in a faster convergence than the *i*CVT+Projection algorithm, as shown in Fig. 5c. The I value converged to 0.083 using αCVT, while *i*CVT+Projection resulted in 0.367.

In all three sets of experiments, I converged to specific values. Convergence of I implies the convergence of Voronoi regions and the agents' positions because I is based on the mass of individual Voronoi regions M_{V_i} and the agents' positions determine the Voronoi regions. Therefore, the convergence of the proposed algorithm is achieved when the agent dropout process terminates. Appendix A-A proves this convergence statement. For the two environments employed in Figs. 3 and 5, we also examined the scalability of the two algorithms (*i*CVT+Projection and αCVT) by evaluating I over an increasing number of agents, $n = 5, 10, \cdots, 30$. As shown in Fig. 6, I from the results using *i*CVT+Projection (blue bars) increased as n and was significantly higher than I from the αCVT results (orange bars) except for when $n = 5$ and $n = 10$ in Fig. 6b (the bullseye-shaped target). When $n = 5$, $I = 0.0576$ for *i*CVT+Projection and $I = 0.058$ for αCVT; when $n = 10$, $I = 0.12$ for *i*CVT+Projection and $I = 0.14$ for αCVT. When a small number of agents are considered, *i*CVT+Projection allows them to easily adjust their positions because the Voronoi regions determined by Euclidean norm for individual agents are relatively large. As the number of agents increases, the Voronoi regions

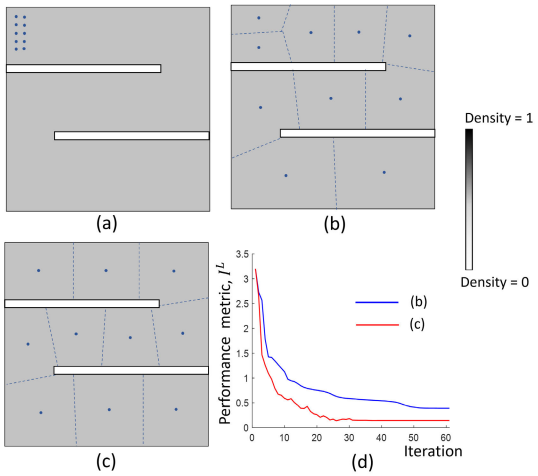

FIGURE 7. Simulation results of non-convex domain including obstacles: (a) initial positions of 10 homogeneous agents in the non-convex domain [8], (b) result of *i*CVT+Projection [11], (c) result of αCVT, and (d) change of performance metric over iterations.

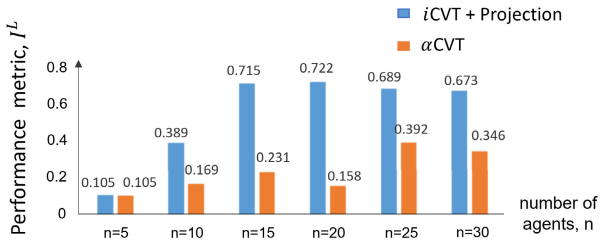

FIGURE 8. Performance metrics at the final state with different agent numbers of non-convex domain including obstacles.

become compact and the agents have less room to adjust their positions. Therefore, the agents using *i*CVT+Projection can be stuck at a local minimum. On the other hand, the agent dropout and reinsertion process in αCVT improved the overall distribution of the coverage regions.

B. HOMOGENEOUS AGENTS FOR TARGET DOMAIN WITH OBSTACLES

To demonstrate the algorithm performance in a non-convex domain involving obstacles restricting the agents' mobility and coverage capabilities, we adopted the environment introduced in [8] (Fig. 7). The agents cannot traverse the obstacles, and the individual agents' coverage is affected by the obstacles. Application-specific modifications to the CVT algorithm and the definition of I are applied.

1) ALGORITHM MODIFICATION

We considered the line of sight when calculating the Voronoi region as follows [13]:

$$L_{i,s} = \{s \in S_f | \forall w \in \{\alpha p_i + (1-\alpha)s\}, w \notin S_o\} \quad (11)$$

where $\alpha \in [0, 1]$, $L_{i,s}$ is the line of sight of i^{th} agent, and S_f and S_o are the free space and the obstacle regions in domain S, respectively. The term $\alpha p_i + (1-\alpha)s$ depicts the straight line connecting the agent's current position p_i to a point s, and w represents all points on the line. If the line connecting p_i to s intersects the obstacle, $s \notin L_{i,s}$. Otherwise, $s \in L_{i,s}$. The Voronoi region, considering the visibility, is redefined as follows:

$$V_i^L = \{s \in S \mid ||s - p_i|| \leq ||s - p_j||, \forall j \neq i\} \cap L_{i,s} \quad (12)$$

where V_i^L is a visible Voronoi region of i^{th} agent.

For the performance metrics to capture the partitioning as well as the coverage performances, we modified (9) to

$$I^L = I + I_c \quad (13)$$

where

$$I_c = \frac{1}{k\overline{M}_V^L}\left(\int_S \rho(s)ds - \sum_{i=1}^{k} M_{V_i^L}\right), \ \overline{M}_V^L = \frac{1}{k}\sum_{i=1}^{k} M_{V_i^L} \quad (14)$$

The first term I in (13) remains the same as (9). The second term I_c is newly added to calculate the normalized mass of uncovered (or invisible) areas, as defined in (14). The term $\int_S \rho(s)ds$ captures the total mass of S. The uncovered areas are the spaces where $\rho > 0$, but not included in any visible Voronoi regions. Therefore, the proposed algorithm works towards reducing the uncovered areas. When the agents cover the entire target area after completing the iteration, $I_c = 0$. To consider visibility, M_{V_i} in τ_{drop}, Algorithms 1 and 2 was replaced with $M_{V_i}^L$.

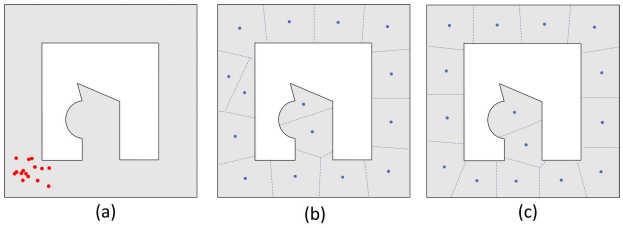

FIGURE 9. Simulation results to compare αCVT with [25]. (a) Initial positions of agents in [25]. (b) Final configuration of the coverage control in [25], and (c) the final configuration of αCVT.

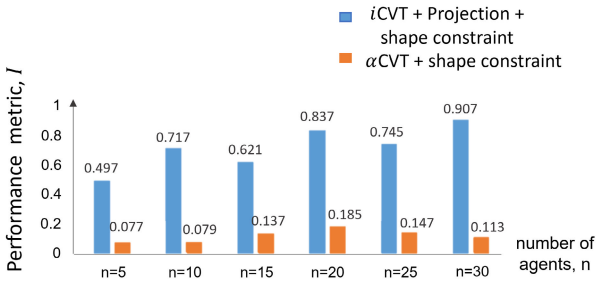

FIGURE 10. Simulation results to compare αCVT with [26]: (a) initial positions of agents in [26], (b) final configuration of the coverage control in [26], and (c) final configuration of αCVT.

2) SIMULATION RESULTS

We first considered an environment with two walls, as shown in Fig. 7a. Figs. 7b,c compare the performances of *i*CVT+Projection and αCVT. While both methods resulted in full coverage of the target area after completing the

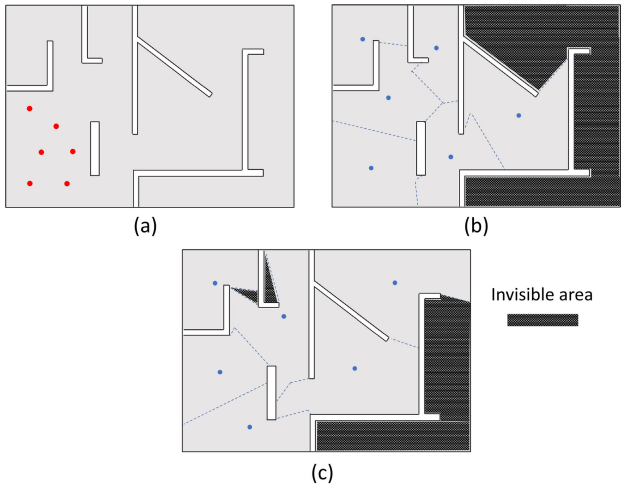

FIGURE 11. Simulation results to compare αCVT with [31]: (a) initial positions of agents in [31], (b) final configuration of the coverage control in [31], and (c) final configuration of αCVT.

iterations, αCVT resulted in a better distribution than iCVT. Fig. 7d plots I^L over iterations for the two algorithms. αCVT showed faster convergence with lower I^L (0.169) than the I^L of iCVT+Projection (0.389). The scalability of the two algorithms for this environment was also tested, and the results are shown in Fig. 8. Overall, the αCVT algorithm outperformed iCVT+Projection. The I^L values of αCVT for this environment were generally higher than the environments introduced in Section III-A, especially when $n = 25$ (0.392) and 30 (0.346). An increase in the I^L values is attributed to the long pathways some agents had to take due to the obstacles. In this case, tuning the τ_{drop} value can further improve the performance. For example, when we selected $\tau_{drop} = 1.025\overline{M}_V^L$, I^L values were decreased to 0.269 and 0.213 for $n = 25$ and 30, respectively. However, such a case-by-case tuning process requires significant time and thus makes the algorithms non-adaptability. With $\tau_{drop} = \frac{k}{k-1}\overline{M}_V^L$ for the environment with obstacles or $\frac{k}{k-1}\overline{M}_V$ for disjoint targets across all simulations (where k is the number of agents), the αCVT method still resulted in a significantly better performance than iCVT+Projection. As described in Proposition 2 in Appendix A-B, the convergence of the proposed algorithm is achieved under specific conditions.

We also compared the performance of αCVT with other recent coverage control methods presented in [25], [26], and [31]. The algorithm in [25] and [31] only concerned non-convex environments with obstacles. The other [26] performs the coverage and rendezvous controls in the non-convex environment with obstacles. For comparison between αCVT and the algorithm in [25], we replicated the same non-convex environment used in [25] as shown in Fig. 9a with the agents' initial positions (red dots). The final results of the coverage control are shown in Figs. 9b and c, respectively. The Voronoi regions in Fig. 9b were regenerated from the final positions of the agents in [25]. The final configuration using αCVT achieved slightly better

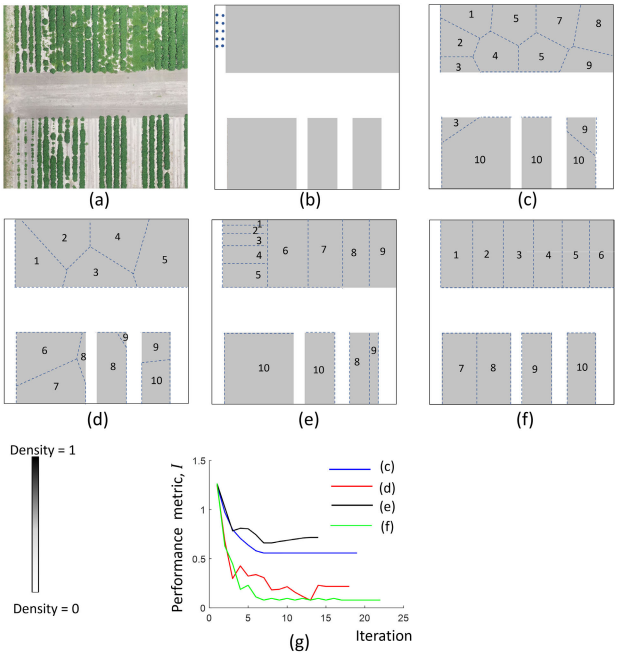

FIGURE 12. Simulation results from a non-convex domain with a shape constraint: (a) aerial view of the agriculture field, (b) corresponding density map, (c) result of iCVT+Projection [11], (d) result of αCVT, (e) result of iCVT+Projection [11] under a shape constraint, and (f) result of αCVT under a shape constraint. (g) change of performance metric over iterations.

performance in terms of the distribution of Voronoi regions, resulting in $I^L = 0.0901$, compared to $I^L = 0.164$ using the comparison algorithm. Comparison with the algorithm in [26] also adopted the same environment and initial settings as in this reference (Fig. 10a). The final configurations from the two algorithms were very similar, as shown in Figs. 10b,c, resulting in $I^L = 0.061$ using the algorithm in [26] and $I^L = 0.044$ using αCVT. Lastly, the initial settings and the environment of [31] were replicated (Fig. 11a) for comparative evaluations. The final results using the two algorithms are shown in Figs. 11b,c. The performance metric values obtained from [31] and αCVT were 0.778 and 0.532, respectively. αCVT covered a larger area than [31] in this experimental setting.

C. HOMOGENEOUS AGENTS FOR SHAPE CONSTRAINED DOMAIN

This scenario concerns the partitioning of a non-convex domain that has a specific shape constraint. One example of such cases may include an agricultural field with crops planted in rows, as shown in Fig. 12a. If the area partitioning is performed for ground robots to move along the crop rows, each agent must be assigned to a well-defined rectangular region. If aerial robots are involved, such a constraint may not be applied. Some previous works have applied Voronoi-based area partitioning to agricultural fields [17], [18], but the fields were considered unconstrained convex environments. In this simulation environment, we consider a non-convex domain with multiple disjoint targets with geometric constraints.

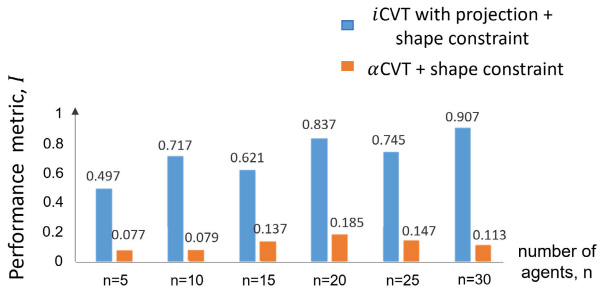

FIGURE 13. Performance metrics at the final state with different agent numbers of non-convex domain with a shape constraint.

To do so, we propose application-specific modifications to the presented algorithms.

1) ALGORITHM MODIFICATION

To achieve the desired geometry-constrained partitioning, we modify (1) as follows:

$$V_i^G = \{s \in S \mid ||T_c(s-p_i)^T|| + \lambda \ell_{p_i}^s \leq ||T_c(s-p_j)^T|| + \lambda \ell_{p_j}^s,\ \forall j \neq i\},\quad T_c = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix},\ 0 \leq a, b \leq 1 \quad (15)$$

where T_c is a variation of the transformation matrix introduced in [6], $\ell_{p_i}^s$ is a total length of line, where $\rho = 0$ between p_i and s, and λ is a penalty factor. The form of (15) is similar to the power diagram in [32], while the sign in front of $\lambda \ell_{p_i}^s$ is '+'. Therefore, $\lambda \ell_{p_i}^s$ acts as a penalty when determining the Voronoi regions. The terms a and b are the weight factors on horizontal and vertical distances, respectively, when the Voronoi partition is performed. If $a = 0$, the horizontal distance is ignored for the Voronoi partition; otherwise, if $b = 0$, the vertical distance is disregarded. To consider geometric constraints, M_{V_i} in τ_{drop}, Algorithms 1 and 2 was replaced with $M_{V_i}^G$ where $M_{V_i}^G$ is the mass of V_i^G. In this work, we used $a = 1$, $b = 0$, and $\lambda = 2$. The performance metric I is used for this application since the environment is a non-convex domain with multiple disjoint targets with geometric constraints.

2) SIMULATION RESULTS

We considered a cotton field located in Corpus Christi, Texas, USA (Fig. 12a). Vision processing may be applied to convert this image into a simplified target domain, as shown in Fig. 12b. Figs. 12c,d show the results of ten homogeneous agents using *i*CVT+Projection and αCVT, respectively, *without* any geometric constraint; Figs. 12e,f show the results of the two algorithms *with* the proper geometric constraint. Using (15), both algorithms resulted in rectangular-shaped Voronoi regions, while αCVT resulted in a better distribution of the target areas. Fig. 12g shows the I values over iterations. An interesting observation is that the proposed algorithm with shape constraints (Fig. 12f) shows better coverage performance than without shape constraints. In the case of (f) in Fig. 12g, I converged to 0.0788. Fig. 13 shows I over

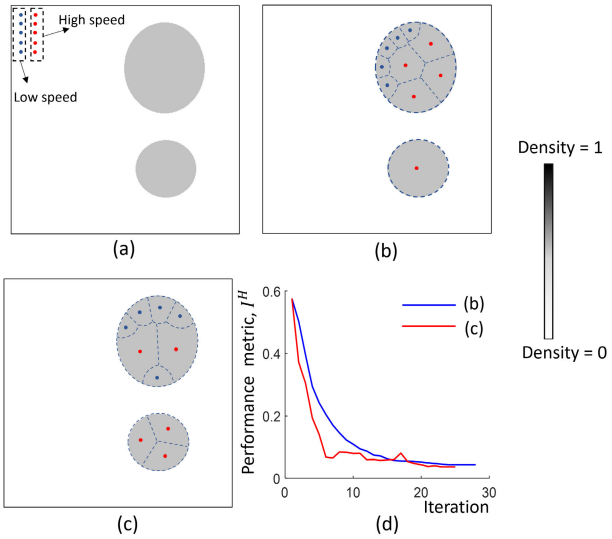

FIGURE 14. Simulation results of the presented algorithm with heterogeneous agents and the non-convex domain including disjoint target areas: (a) initial positions of 10 heterogeneous agents, (b) result of *i*CVT+Projection [11], and (c) result of αCVT. (d) change of performance metric over iterations.

$n = 5, 10, \cdots, 30$. The αCVT algorithm shows significantly lower I values than *i*CVT+Projection in all cases.

The convergence of the proposed algorithm was experimentally shown in Fig. 12g. Mathematical proof in the case when a geometric constraint is applied cannot be clearly described. Nevertheless, all simulations using αCVT converged as shown in Fig. 12g.

D. HETEROGENEOUS AGENTS

This example involves heterogeneous mobile agents with coverage capacities differentiated by their maximum speeds. When the agents have different maximum speeds, the temporal cost can be used for Voronoi partitioning [21]. We extend this temporal cost to consider the scalability of the coverage areas.

1) ALGORITHM MODIFICATION

First, the maximum speed of agents is converted into the reachable distance within a specific time duration, such that

$$R_i = v_{i,\max} \cdot t \quad (16)$$

where $v_{i,\max}$ is the maximum speed of the i^{th} agent and t is the time duration for travel. If the maximum speeds of agents are fixed, we can set a high value of t for large coverage areas and a low value of t for small ones. In other words, t serves as a scaling factor. We applied the same value of t to all agents to maintain synchrony and consistency.

The Voronoi partition with R_i can be redefined as

$$V_i^H = \{s \in S \mid ||\frac{s-p_i}{R_i}|| \leq ||\frac{s-p_j}{R_j}||, \forall j \neq i\} \cap L_{i,s} \quad (17)$$

where $L_{i,s}$ is the same in (11). If the environment is an open space without obstacles, $L_{i,s}$ is disregarded. The above

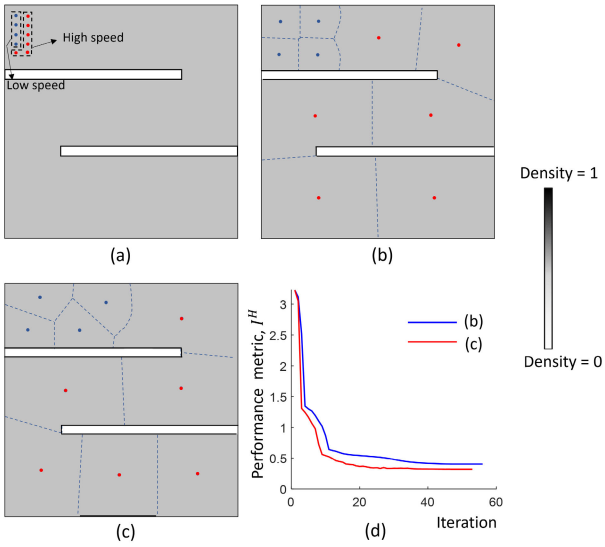

FIGURE 15. Simulation results of the proposed algorithm with heterogeneous agents and the non-convex domain including obstacles: (a) initial positions of 10 heterogeneous agents, (b) result of *i*CVT+Projection [11], and (c) result of αCVT. (d) change of performance metric over iterations.

equation follows the characteristics of the Voronoi partition described in [21]. The performance metric in (9) is also modified as follows to consider the reachable distance:

$$I^H = \frac{1}{k}\left(\sum_{i=1}^{k}\left|\frac{M_{V_i}^h}{\overline{M}_V^h} - 1\right|\right) + I_c \tag{18}$$

where

$$\overline{M}_V^h = \frac{1}{k}\sum_{i=1}^{k} M_{V_i}^h; \quad M_{V_i}^h = \frac{M_{V_i}^H}{\pi R_i^2}$$

The modified mass $M_{V_i}^h$ considers the areas defined by πR_i^2. Therefore, $M_{V_i}^h$ becomes small for a large value of R_i, and vice versa. We also note that I_c in (18) is defined in (14). If an open environment without any obstacle is considered, $I_c = 0$. To consider heterogeneity, M_{V_i} in τ_{drop}, Algorithms 1 and 2 was replaced with $M_{V_i}^h$.

2) SIMULATION RESULTS

Fig. 14 shows the simulation result of the proposed algorithm for a domain with two disjoint targets employing two types of agents, $R_i = 8$ for $i = 1, \cdots, 5$, and $R_i = 16$ for $i = 6, \cdots, 10$. Given the initial status shown in Fig. 14a, the partitioning results of *i*CVT+Projection and αCVT are shown in Figs. 14b and c, respectively. The high-speed agents occupy larger areas, and the low-speed ones occupy smaller areas. In Fig. 14b, one high-speed agent is located at the lower target area. On the other hand, the result of the proposed algorithm in Fig. 14c shows that three high-speed agents are located at the lower target area. The performance metrics of *i*CVT+Projection and αCVT are 0.0437 and 0.0367, respectively. Fig. 15 shows the simulation result of the proposed algorithm considering heterogeneity for a domain

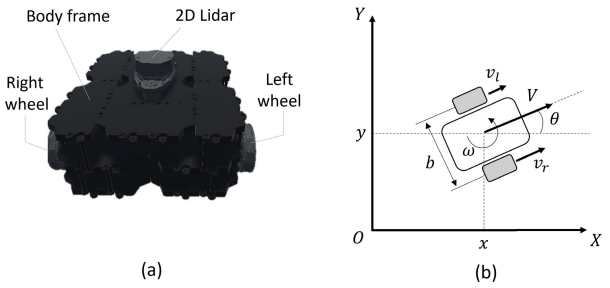

FIGURE 16. (a) Turtlebot3 model adopted for simulation, and (b) kinematic representation of a two-wheeled robot.

with obstacles. Given the initial state shown in Fig. 15a, two types of agents were deployed to cover the target area. We set $R_i = 24$ for $i = 1, \cdots, 4$, and $R_i = 48$ for $i = 5, \cdots, 10$. Figs. 15b and c show the result of *i*CVT+Projection and αCVT, respectively. in both cases, the high-speed agents traveled much farther and had larger coverage areas, and the low-speed ones were distributed near the initial positions with smaller coverage areas. As shown in Fig. 15d, αCVT outperforms *i*CVT+Projection. The performance metrics of *i*CVT+Projection and αCVT converge to 0.402 and 0.318, respectively.

Voronoi regions, considering the different maximum speeds of the agents, result in circular boundaries (see Section II in [21]). However, CVT with this heterogeneity converges if the agents move to the centroid of their Voronoi regions (Proposition 1 in [21]). Therefore, the different maximum speeds of the agents do not influence the convergence of the proposed algorithm, and Propositions 1 and 2 in the Appendix are still valid.

IV. EXTENSION TO REAL-TIME COVERAGE CONTROL

This section applies αCVT to real-time coverage control problems using mobile agents in Unity-ROS simulations. The size of the environment was $20 \times 20m^2$. We assume that all agents are connected to a centralized computer that runs the coverage control algorithm and that there is no communication loss or sensing limit.

A. MOBILE ROBOT IN SIMULATION

We imported the Unity model of Turtlebot3 [33] as a mobile agent (Fig. 16a). Each robot was equipped with a 2D Lidar to detect obstacles and wheel encoders to calculate wheel odometry. The associated kinematics is represented in Fig. 16b and can be expressed as [34],

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}V\\ \omega\end{bmatrix} \tag{19}$$

$$\omega = \frac{v_r - v_l}{b}; \quad V = \frac{v_r + v_l}{2} \tag{20}$$

where x and y represent the robot's position, θ is the heading angle of the robot in the global coordinate, b is the distance between the left and right wheel, V is the linear velocity, and ω is the angular velocity of the robot. The terms v_r

and v_l indicate linear velocities of the right and left wheels, respectively. V and ω are the control inputs obtained from an upper-level controller. V and ω are achieved by controlling v_r and v_l. The simulations based on this kinematic model were performed in Unity.

B. REAL-TIME COVERAGE CONTROL

Algorithm 3 αCVT for Real-Time Coverage Control

```
1:  DropoutOff = False
2:  while do
3:      receive agent pose p_i from each i^th agent
4:      while for 1 second do
5:          if DropoutOff == False then
6:              AgentDropout()
7:          end if
8:          Determine V_i, ∀i from Eq. 1
9:          Calculate p_{i,goal}, ∀i from Eq. 7 and Projection
10:         Calculate △I from Eq. 9 and 10
11:         if △I < ε_drop then
12:             DropoutOff = True
13:         end if
14:     end while
15:     Send p_{i,goal} to each i^th agent
16: end while
```

Algorithm 3 extends the presented method for real-time coverage control. It is similar to Algorithm 2, except for real-time updating of the agents' poses (Line 3) and goals (Line 15). After receiving the real-time pose (Line 3), this algorithm calculates the goal for each agent within a specific time duration. The time duration is necessary to calculate meaningful goals for the agents. We used 1 second for this, but it can be adjusted as needed. The calculated goals (Line 15) are sent to individual agents, and each agent performs path planning and speed control to reach the goal.

Many previous studies derived a control law by adopting the gradient descent or ascent method from the Voronoi regions [5], [7], [8], [20], [21], [22], [25], [26]. Different strategies introduced in [3], [11], and [24] may also be considered for path planning and motion control for this study. We adopted the A* algorithm [35] for path planning and velocity control with reciprocal velocity obstacle (RVO) [36] for collision avoidance. The advantage of this strategy is its ability to avoid static and dynamic obstacles. A* algorithm calculates the desired path to the goal point. For a collision-free path from the static obstacles, A* requires a pre-determined map. The desired path from A* is inputted to the RVO algorithm to calculate the command velocity. The command velocity aims to enable the agent to follow the desired path, and the RVO algorithm handles static and dynamic obstacles in the multi-agent system.

To determine the command velocity, the algorithm generates the velocity samples (v_s) within the velocity range, considering the kinematic constraint of the agent [36],

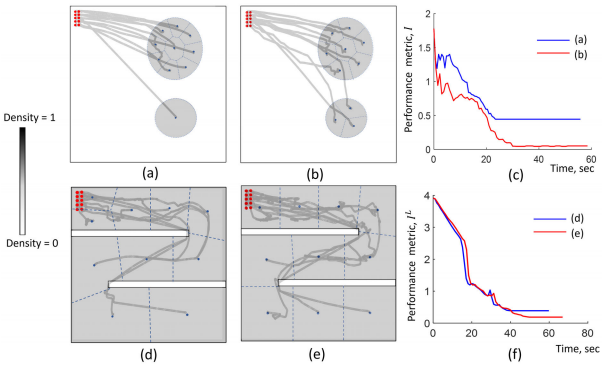

FIGURE 17. Results of real-time coverage control using Algorithm 3 and homogeneous mobile agents: (a) result of *i*CVT+Projection [11], and (b) result of αCVT in a non-convex domain including disjoint target areas. (c) change of performance metric over time-lapse in the non-convex domain with disjoint targets. (d) result of *i*CVT+Projection [11], and (e) result of αCVT in a non-convex domain with obstacles. (f) change of performance metric over time-lapse in the non-convex domain with obstacles.

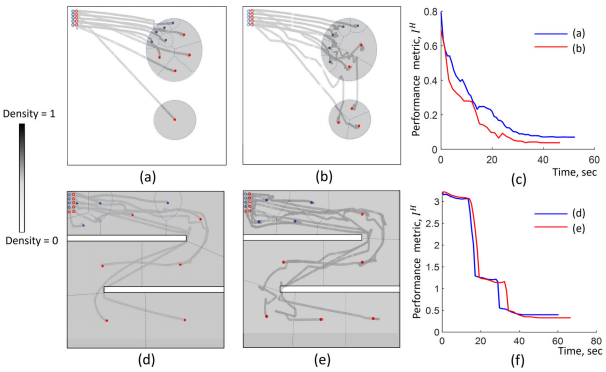

FIGURE 18. Results of real-time coverage control using Algorithm 3 and heterogeneous mobile agents with different maximum speeds: (a) result of *i*CVT+Projection [11], and (b) result of αCVT in a non-convex domain including disjoint target areas. (c) change of performance metric over time-lapse in the non-convex domain with disjoint targets. (d) result of *i*CVT+Projection [11], and (e) result of αCVT in a non-convex domain with obstacles. (f) change of performance metric over time-lapse in the non-convex domain with obstacles.

such that

$$V_{set} = \{v_s \mid ||v_s|| < v_{max} \wedge ||v_s - v_{cur}|| < a_{max}\Delta t\} \quad (21)$$

where v_{cur}, v_{max}, and a_{max} are the current velocity, maximum velocity limit, and maximum acceleration of the agent, respectively, and Δt is the sampling time. The algorithm performs forward simulations using V_{set} to check the time to collision and trajectory deviation from the desired path, and then, it calculates the penalty of each velocity sample [36]. Finally, the velocity sample with the lowest penalty is selected as a command velocity. One difference of our application from [36] is that the control inputs for the mobile agent are linear and angular velocities considering the kinematics in (20). Therefore, the velocity control algorithm in our work generates linear and angular velocity samples and calculates the penalty considering both. Finally, the algorithm returns the linear and angular velocity commands simultaneously. The algorithm was embedded for each agent in Unity.

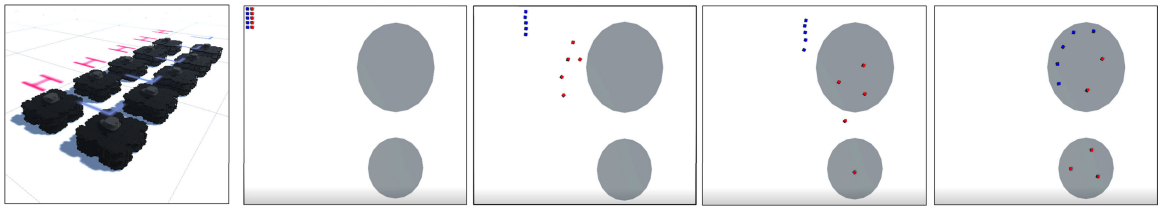
(a) Robot model (0:00) (b) Top view (0:00) (c) Top view (0:15) (d) Top view (0:25) (e) Top view (0:40)

FIGURE 19. Ten robots employed for simulation (a) and time-lapse sequences of real-time coverage control for the domain shown in Fig. 18b (b-e).

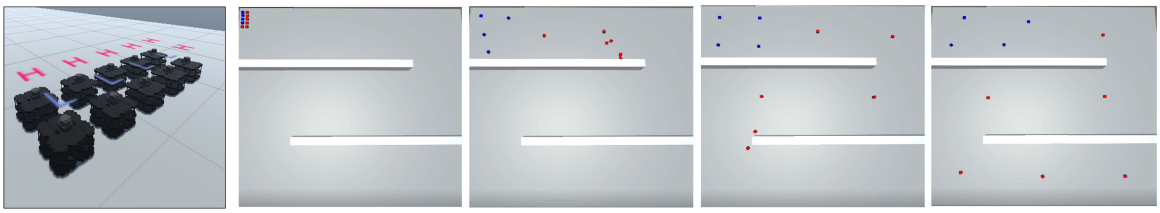
(a) Robot model (0:00) (b) Top view (0:00) (c) Top view (0:15) (d) Top view (0:40) (e) Top view (1:00)

FIGURE 20. Ten robots employed for simulation (a) and time-lapse sequences of real-time coverage control for the domain shown in Fig. 18e (b-e).

C. EVALUATION

We evaluated the performance of real-time coverage control using Algorithm 3 in two experimental settings and two types of environments for each setting.

We first considered homogeneous agents in two previously considered environments, an open environment with two disjoint targets and one with walled obstacles (Fig. 17). We set $v_{i,max} = 0.4\ m/s$ for the mobile agents with two disjoint targets. The gray lines indicate the trajectories of individual agents. Figs. 17a,b show the result of real-time coverage control for two disjoint targets using *i*CVT+Projection and αCVT, respectively, and Fig. 17c compares the I values over time for the two algorithms. *i*CVT+Projection resulted in $I = 0.444$ while αCVT resulted in $I = 0.051$. Figs. 17d,e show the result for the environment, including obstacles using *i*CVT+Projection and αCVT, respectively. We set $v_{i,max} = 0.8\ m/s$ for all mobile agents. Fig. 17f shows similar convergence behavior of I using both methods. After the convergence, $I = 0.387$ using *i*CVT+Projection and $I = 0.187$ using αCVT.

Second, we adopted the agents with two different maximum speeds. For the environment involving two disjoint targets, we set $v_{i,max} = 0.4\ m/s$ for $i = 1, \cdots, 5$ (blue dots in Fig. 18a,b) and $v_{i,max} = 0.8\ m/s$ for $i = 6, \cdots, 10$ (red dots in Fig. 18a,b). Figs. 18a,b show the results using *i*CVT+Projection and αCVT, respectively. Fig. 18c shows the change of performance metrics over time for both algorithms. The performance metric I^H converged to 0.0701 using *i*CVT+Projection and 0.0375 using αCVT. Figs. 18d,e show the results for the environment, including obstacles. We used $v_{i,max} = 0.4m/s$ for $i = 1, \cdots, 4$ (blue dots in Fig. 18d,e) and $v_{i,max} = 0.8m/s$ for $i = 5, \cdots, 10$ (red dots in Fig. 18d,e). The performance metrics I^H converged to 0.4003 using *i*CVT+Projection and 0.3305 using αCVT (Fig. 18f). Fig. 19 and Fig. 20 show the time-lapse sequences of real-time coverage control for these two examples. The speed of blue agents is $0.4m/s$, while the speed of red agents is $0.8m/s$.

V. CONCLUSION AND DISCUSSION

We introduced the αCVT algorithm, combining a novel agent dropout and reinsertion strategy with *i*CVT+Projection for efficient coverage control in 2D non-convex domains. Our evaluation involved simulations across various experimental settings encompassing both homogeneous and heterogeneous agents. These scenarios included disjoint target areas and environments obstructed by obstacles. The presented αCVT algorithm consistently showcased significant enhancements in Voronoi region distribution and overall coverage compared to *i*CVT+Projection in all evaluated cases. Fine-tuning τ_{drop} for each case could potentially further enhance performance. However, this fine-tuning process introduces a trade-off, requiring a trial-and-error-based approach, thereby sacrificing some level of adaptability.

The proposed algorithm can be applied for exploration tasks that mandate agents to move within the designated Voronoi regions continuously. A potential comparison can be drawn with the ergodic dynamics-based exploration method [2]. Of particular interest to our study is its potential application in agriculture, as depicted in Fig. 12. In an agricultural setting, a human user can designate high-interest areas within the field for monitoring crop conditions based on an aerial view. These specified high-interest regions can subsequently undergo coverage utilizing the algorithm presented in this study. Ground robots can employ localized path-planning strategies to cover the allocated regions effectively. This adaptation indicates the algorithm's versatility and potential in real-world applications.

The presented work assumes a centralized system architecture. We plan to expand the current work by developing a decentralized version of αCVT, tailored for implementation within decentralized multi-robot systems. In this envisioned framework, each agent would derive its control inputs by leveraging information received from neighboring agents. This information exchange would account for factors such as sensing capability [6], [7], [9], [19] and collision-avoidance considerations [20], [26]. By transitioning towards a decentralized approach, the aim is to empower agents within multi-robot systems to collaborate effectively and autonomously. This adaptation holds promise for enhancing adaptability and scalability in real-world applications, where decentralized agent coordination is pivotal for optimal performance.

APPENDIX A
PROPOSITIONS AND PROOFS

A. PROPOSITION 1

Once the agent dropout process terminates, the algorithm converges to one of the local minima in the non-convex environment with disjoint targets.

Proof: The proposed algorithm (Algorithm 2) only performs the projection algorithm and CVT if 'AgentDropout' is deactivated by the termination criterion ($\Delta I < \epsilon$). The projection algorithm makes all positions of agents located in the closest target areas (where $\rho > 0$) by (8). If the position calculated by CVT is already located in the target area, the projection algorithm is ignored. Therefore, all agents are located in the target areas in every iteration. Also, the proposed algorithm follows the density definition in [22]. It means that the outside of target areas is the infeasible area (where $\rho = 0$). The infeasible area is excluded from calculating the mass and centroid of the Voronoi region since $\rho = 0$. After all the agents are located in the disjoint targets, the centroid of each agent's Voronoi regions is located in the disjoint target where the agent is located by the density definition since the infeasible areas are excluded [22]. Therefore, the agents converge to stationary positions using CVT in each disjoint target (see Section IV in [22]).

B. PROPOSITION 2

The algorithm converges to one of the local minima in the non-convex environment with obstacles if the following specific conditions are satisfied: 1) 'AgentDropout' is terminated; 2) If the boundary between two visible Voronoi regions exists, the Euclidean distances from the boundary to the agents of the two visible Voronoi regions are equal. 3) there is no uncovered area.

Proof: By Condition 1), the proposed algorithm (Algorithm 2) only performs CVT and the projection algorithm. If there are uncovered areas, the boundary between the visible Voronoi region and the uncovered area can change as the agent moves [37]. If there is no uncovered area, the boundary changes between agents and the uncovered area can be ignored (see Section II in [37]). If Conditions 2) and 3) are satisfied, the problem of the non-convex environment with obstacles can be solved by CVT since the cost function for locational optimization does not require the constraints of the uncovered area and visibility [37]. Moreover, the projection algorithm does not prevent convergence as described in the proof of Proposition 1 since the role of the projection algorithm is to find a feasible target position only if the target position calculated by CVT is set on the infeasible area (where $\rho = 0$). Therefore, The algorithm converges to one of the local minima.

Condition 2) may seem obvious and redundant when considering the Voronoi region in the convex environment since the boundary between the two Voronoi regions is formed at the points that have the same Euclidean distances from two agents by (1). However, the obstacles in the non-convex environment sometimes block the visibility of the agent, and the boundary of the visible Voronoi region can be formed regardless of the Euclidean distance [37]. The convergence of the proposed algorithm was experimentally shown in Fig. 7d since the performance metrics in Fig. 7d converge to specific values. It means the convergence of Voronoi regions and agents' positions as mentioned in Section III-A. However, One limitation is that the proposed algorithm can not guarantee convergence for all cases even though the algorithm works toward reducing the uncovered area by (13). For example, if the number of agents is too small to cover the whole area of the environment with obstacles, then Condition 3) is not satisfied.

REFERENCES

[1] A. Mavrommati, E. Tzorakoleftherakis, I. Abraham, and T. D. Murphey, "Real-time area coverage and target localization using receding-horizon ergodic exploration," *IEEE Trans. Robot.*, vol. 34, no. 1, pp. 62–80, Feb. 2018.

[2] A. Prabhakar, I. Abraham, A. Taylor, M. Schlafly, K. Popovic, G. Diniz, B. Teich, B. Simidchieva, S. Clark, and T. Murphey, "Ergodic specifications for flexible swarm control: From user commands to persistent adaptation," in *Robotics: Science and Systems*, Jun. 2020. [Online]. Available: https://par.nsf.gov/biblio/10175963

[3] J. Meyer, A. Prabhakar, A. Pinosky, I. Abraham, A. Taylor, M. Schlafly, K. Popovic, G. Diniz, B. Teich, B. Simidchieva, S. Clark, and T. Murphey, "Scale-invariant specifications for human-swarm systems," 2022, *arXiv:2212.03106*.

[4] D. Yu, H. Xu, C. L. P. Chen, W. Bai, and Z. Wang, "Dynamic coverage control based on K-means," *IEEE Trans. Ind. Electron.*, vol. 69, no. 5, pp. 5333–5341, May 2022.

[5] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. Robot. Autom.*, vol. 20, no. 2, pp. 243–255, Apr. 2004.

[6] A. Gusrialdi, S. Hirche, T. Hatanaka, and M. Fujita, "Voronoi based coverage control with anisotropic sensors," in *Proc. Amer. Control Conf.*, Jun. 2008, pp. 736–741.

[7] Y. Stergiopoulos and A. Tzes, "Cooperative positioning/orientation control of mobile heterogeneous anisotropic sensor networks for area coverage," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 1106–1111.

[8] Y. Stergiopoulos, M. Thanou, and A. Tzes, "Distributed collaborative coverage-control schemes for non-convex domains," *IEEE Trans. Autom. Control*, vol. 60, no. 9, pp. 2422–2427, Sep. 2015.

[9] M. Santos, Y. Diaz-Mercado, and M. Egerstedt, "Coverage control for multirobot teams with heterogeneous sensing capabilities," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 919–925, Apr. 2018.

[10] J. G. M. Fu, T. Bandyopadhyay, and M. H. Ang, "Local Voronoi decomposition for multi-agent task allocation," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 1935–1940.
[11] A. Breitenmoser, M. Schwager, J.-C. Metzger, R. Siegwart, and D. Rus, "Voronoi coverage of non-convex environments with a group of networked robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 4982–4989.
[12] S. Kemna, J. G. Rogers, C. Nieto-Granda, S. Young, and G. S. Sukhatme, "Multi-robot coordination through dynamic Voronoi partitioning for informative adaptive sampling in communication-constrained environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 2124–2130.
[13] G. A. Cardona, D. Yanguas-Rojas, M. F. Arevalo-Castiblanco, and E. Mojica-Nava, "Ant-based multi-robot exploration in non-convex space without global-connectivity constraints," in *Proc. 18th Eur. Control Conf. (ECC)*, Jun. 2019, pp. 2065–2070.
[14] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 14413–14423, Dec. 2020.
[15] B. Kumar, L. Sharma, and S.-L. Wu, "Job allocation schemes for mobile service robots in hospitals," in *Proc. IEEE Int. Conf. Bioinf. Biomed. (BIBM)*, Dec. 2018, pp. 1323–1326.
[16] G. Oliveira, P. D. M. Plentz, and J. T. Carvalho, "Multi-constrained Voronoi-based task allocator for smart-warehouses," in *Proc. IEEE 15th Int. Symp. Appl. Comput. Intell. Informat. (SACI)*, May 2021, pp. 515–520.
[17] A. Barrientos, J. Colorado, J. D. Cerro, A. Martinez, C. Rossi, D. Sanz, and J. Valente, "Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots," *J. Field Robot.*, vol. 28, no. 5, pp. 667–689, Sep. 2011, doi: 10.1002/rob.20403.
[18] J. Kim and H. I. Son, "A Voronoi diagram-based workspace partition for weak cooperation of multi-robot system in orchard," *IEEE Access*, vol. 8, pp. 20676–20686, 2020.
[19] K. R. Guruprasad and D. Ghose, "Heterogeneous locational optimisation using a generalised Voronoi partition," *Int. J. Control*, vol. 86, no. 6, pp. 977–993, Jun. 2013, doi: 10.1080/00207179.2013.768775.
[20] O. Arslan and D. E. Koditschek, "Voronoi-based coverage control of heterogeneous disk-shaped robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 4259–4266.
[21] S. Kim, M. Santos, L. Guerrero-Bonilla, A. Yezzi, and M. Egerstedt, "Coverage control of mobile robots with different maximum speeds for time-sensitive applications," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 3001–3007, Apr. 2022.
[22] C. H. Caicedo-Nunez and M. Zefran, "A coverage algorithm for a class of non-convex regions," in *Proc. 47th IEEE Conf. Decis. Control*, Dec. 2008, pp. 4244–4249.
[23] X. Xu and Y. Diaz-Mercado, "Multi-robot control using coverage over time-varying non-convex domains," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 4536–4542.
[24] A. S. Yengejeh, A. B. Asghar, and S. L. Smith, "Distributed multirobot coverage control of nonconvex environments with guarantees," *IEEE Trans. Control Netw. Syst.*, vol. 10, no. 2, pp. 796–808, Jun. 2023.
[25] H. F. Parapari, F. Abdollahi, and M. B. Menhaj, "Coverage control in non-convex environment considering unknown non-convex obstacles," in *Proc. 2nd RSI/ISM Int. Conf. Robot. Mechatronics (ICRoM)*, Oct. 2014, pp. 119–124.
[26] M. Boldrer, L. Palopoli, and D. Fontanelli, "A unified Lloyd-based framework for multi-agent collective behaviours," *Robot. Auto. Syst.*, vol. 156, Oct. 2022, Art. no. 104207. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092188902200118X
[27] M. Boldrer, L. Palopoli, and D. Fontanelli, "Lloyd-based approach for robots navigation in human-shared environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 6982–6989.
[28] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. IT-28, no. 2, pp. 129–137, Mar. 1982.
[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.
[30] S. Ropke and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows," *Transp. Sci.*, vol. 40, no. 4, pp. 455–472, Nov. 2006.
[31] K. Giannousakis and A. Tzes, "Distributed control of mobile robots in an environment with static obstacles," *IET Cyber-Syst. Robot.*, vol. 3, no. 2, pp. 128–139, Jun. 2021, doi: 10.1049/csy2.12018.
[32] F. Aurenhammer, "Power diagrams: Properties, algorithms and applications," *SIAM J. Comput.*, vol. 16, no. 1, pp. 78–96, Feb. 1987, doi: 10.1137/0216006.
[33] Unity Technologies. *Navigation 2 SLAM Example*. Accessed: May 3, 2023. [Online]. Available: https://github.com/Unity-Technologies/Robotics-Nav2-SLAM-Example
[34] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*, 2nd ed. Cham, Switzerland: Springer, 2016.
[35] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SCS-4, no. 2, pp. 100–107, Mar. 1968.
[36] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2008, pp. 1928–1935.
[37] J.-S. Marier, C.-A. Rabbath, and N. Léchevin, "Visibility-limited coverage control using nonsmooth optimization," in *Proc. Amer. Control Conf. (ACC)*, Jun. 2012, pp. 6029–6034.

KANGNEOUNG LEE (Student Member, IEEE) received the B.S. and M.S. degrees in mechanical engineering from Sungkyunkwan University, South Korea, in 2015 and 2017, respectively. He is currently pursuing the Ph.D. degree in mechanical engineering with Texas A&M University. From 2017 to 2020, he was a Research Engineer with Mando Corporation, South Korea. His research interests include computer vision, path planning, and control of mobile robots.

KIJU LEE (Member, IEEE) received the B.S.E. degree in electrical and electronics engineering from Chung-Ang University, Seoul, South Korea, in 2002, and the M.S.E. and Ph.D. degrees in mechanical engineering from Johns Hopkins University, Baltimore, MD, USA, in 2006 and 2008, respectively. From 2008 to 2019, she was with the Department of Mechanical and Aerospace Engineering, Case Western Reserve University. Since 2019, she has been a joint Faculty Member with the Department of Engineering Technology and Industrial Distribution and the Department of Mechanical Engineering, Texas A&M University, College Station, TX, USA. She is the Director of the Adaptive Robotics and Technology (ART) Laboratory, through which she runs various research projects focusing on swarm robotics, novel robotic mechanism design, and human–robot interaction.