A Projected Lloyd's Algorithm for Coverage Control Problems

Yoav Palti* and Daniel Zelazo†

This work presents a new approach for a coverage problem for large areas and a limited number of sensors. We consider the problem where there are not enough sensors to provide complete coverage of the designated area at once, and therefore the sensors should coordinate their coverage in a sequential manner. Furthermore, we require that at each stage of coverage, a sub-region in the area is constantly monitored. Our solution approach is a combination of two algorithms: the first is Lloyd's algorithm, which is well known and used in this field, and the second one is a modified Lloyd's algorithm that we term the *projected Lloyd's Algorithm*. Lastly, we combine Lloyd's algorithm (and its projected version) with a distance based formation controller. We demonstrate our results with numerical simulations.

I Introduction

The mission of monitoring some range (for example, an area) using sensors is widely spread in various fields - it can be used for photographing a large area, receiving or sending electromagnetic signals, tracking targets, or even for designing an algorithm for a robotic vacuum cleaner [1–3]. This research area is known in the scientific community as the *coverage control* problem [4]. Sensor coverage can be generally described as the reflection of how well a given range is monitored by sensors. This field is well explored in the scientific community, especially in the recent years when the fields of distributed algorithms and cooperative systems were developed.

A concept that repeats in many works is finding a set of trajectories (at least one route for every sensor), allowing the mobile sensors to achieve the required coverage goal. For example, when trying to cover an area using two mobile sensors, the algorithm will define a trajectory for each sensor such that the whole area will be covered [5,6]. Another idea that appears in most of the relevant literature is the use of Voronoi diagrams for optimizing the sensor location [6, 7].

When coverage is required for large areas, and specifically when a single sensor cannot provide full coverage of that area, the idea of using a *deployment* of mobile sensors is presented [4, 7]. In [8], the problem is formed using a probabilistic network model and some density function, to model detections of events. Then, an optimization algorithm is proposed to find the optimal deployment such that maximum coverage is obtained with minimal communications cost. In [7], Cortes *et al.* propose an optimization algorithm for maximal coverage using Centroidal Voronoi Tessellations (CVT) [9]. To achieve this partitioning, they are utilizing a continuous gradient controller implementation of the *Lloyd's algorithm* [10], an algorithm that employs a simple iterative method to compute the CVT. The former concept will be utilized and modified in this paper to achieve new coverage capabilities that will be further described.

In this work, we formulate the coverage problem from another angle. We firstly assume that the given range can be at most *partially* covered. However, we also define some sub-range

*Faculty of Aerospace Engineering, Technion - Israel Institute of Technology, Haifa, yoavp10@gmail.com

†Associate Professor, Faculty of Aerospace Engineering, Technion - Israel Institute of Technology, Haifa, dzelazo@technion.ac.il

that must be at least partially covered at all times (for example, surveillance or home-base communications constraint). We present an algorithm for partitioning this range into tiles, such that each tile intersects with this sub-range constraint, and then, utilizing the approach outlined in [7] to achieve coverage of this tile. To achieve this intersection, we present a new, modified version of Lloyd's algorithm - the *projected Lloyd's algorithm*. This new algorithm modifies the original Lloyd's algorithm and utilizes a specific projection function to achieve partitioning answering the sub-range constraint. With this method, we can guarantee complete coverage of the range by sequentially deploying the agents from each tile to tile while ensuring that these tiles all have a non-empty intersection with the sub-range of interest.

However, there are some cases where maintaining spatial properties are also as important as keeping the required range covered. One example is when the communications between the sensors are limited to some range. Another example is when the spatial shape is critical for the mission, as in geolocation missions. There are several solutions for this problem, known generally as the *formation control* problem [11]. In this work, we will present a method to combine a deployment algorithm with a distance based formation controller. Both controllers are gradient-based controllers, therefore the combination strategy will be adding one to each other. The combination is done using some coefficient, and its value determines weather the results are nearing optimal deployment or optimal formation. That said, we will not provide a condition for full coverage when this combination is in use.

The organization of this paper is as follows. Following this chapter, the problem will be formally defined. Then, mathematical background on the main concepts of this paper will be given, followed by the solution algorithms and control strategies. Finally, numerical simulations will be shown to validate our approach.

II Problem Formulation

Let us consider an area $A \subset \mathbb{R}^2$ that we aim to cover with $n \in \mathbb{N}$ *mobile* sensors. Those sensors are defined by the set $S = \{s_1, \ldots, s_n\}$ located at positions $P(t) = \{p_i(t) \in \mathbb{R}^2 \mid i = 1, \ldots, n\}$. The n mobile sensors have integrator dynamics, i.e., $\dot{p}_i(t) = u_i(t)$, and each sensor can cover a region described by the abstract set $C_i\left(p_i(t)\right) \subset \mathbb{R}^2$, as can be seen in Figure 1.

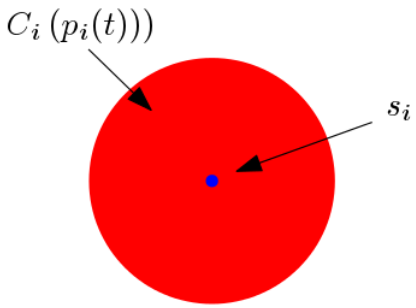


Figure 1: A sensor and its coverage region.

A configuration c of the mobile sensors at time t is the stack of the sensor positions at time t,

$$c(t) = \begin{bmatrix} p_1^T(t) & \cdots & p_n^T(t) \end{bmatrix}^T \in \mathbb{R}^{2n}. \tag{1}$$

Thus the coverage of a configuration is given by

$$D\left(c\left(t\right)\right) = \cup C_i(p_i(t)).$$

We also emphasize in this work that we assume it is not possible to cover the entire area A using a single configuration.

Assumption 1. *For any configuration $c(t)$, the coverage satisfies $D\left(c\left(t\right)\right) \subset A$.*

We also introduce in this work a designated sub-area of the area A that must remain under constant coverage. In this direction, let us consider a sub-area of A, $A_m \subset A$. The area A_m can represent a ground station communication constraint, a surveillance constraint etc., that means that A_m must be partially covered at all times.

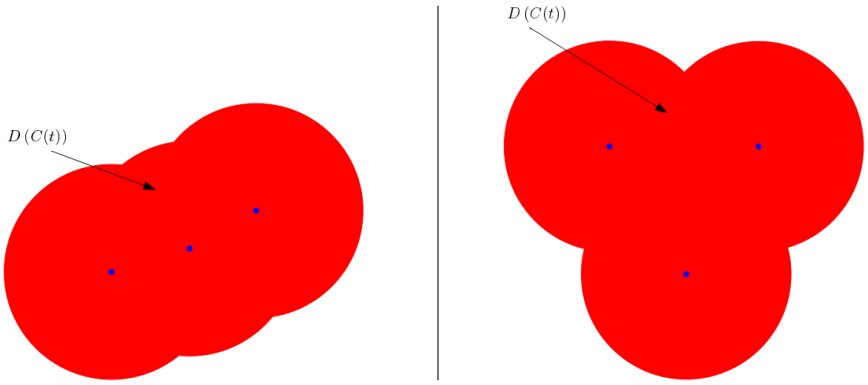


Figure 2: Two configurations built from the same sensors set.

In Figure 2, we can see how a given configuration can cover some area in different ways. However, due to Assumption 1 we know that it isn't possible using a single configuration. Therefore, we shall *partition* the area, as can be seen in Figure 3.

Definition 1. *A* partition *of $A \in \mathbb{R}^2$, $PR(A)$ is a finite set built from n subsets $pr_i \subset \mathbb{R}^2$, $i = 1, \ldots, n$ such that:*

- $pr_i \cap pr_j = \emptyset \; \forall i \neq j$.
- $\cup pr_i = A$.

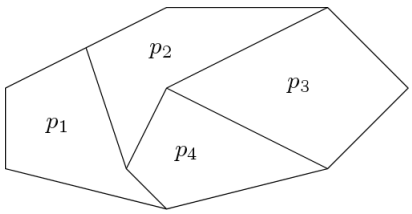


Figure 3: An example of a partition of some area.

This work aims to find a partition $PR(A)$ such that each of it's subsets pr_i intersects with A_m, and that each pr_i is eventually covered during the deployment.

Problem 1. *(a) Find a partition $PR(A)$ such that each of the subsets $pr_i \in PR(A)$ satisfy $pr_i \cap A_m \neq \emptyset$;*

(b) Find a deployment controller for each sensor, such that

$$pr_i \subseteq \lim_{t\to\infty} D(c(t)), \textit{for } i = 1, \ldots, n.$$

In the following section, the basic mathematical concepts used in this work will be introduced, following by a solution for Problem 1.

III Voronoi Diagrams and Distributed Coverage Algorithms

In this chapter we will introduce two main concepts that were used in this work. The first is that of *Voronoi diagrams*, and the later *Lloyd's algorithm*.

III.A Voronoi Diagrams

The basic mathematical concept that we employ in this work is the *Voronoi Diagram* (also known as Voronoi partition or Voronoi tessellation). While being a method to partition an area with some cost function, it is a widely used in the optimal coverage problem [6, 7, 9].

The Voronoi Diagram of a region $\Omega \subset \mathbb{R}^2$ is the set of partitions $\mathcal{V} = \{V_1, \ldots, V_n\}$ generated by the generators $\mathcal{Z} = \{z_1, \ldots, z_n \mid z_i \in \Omega\}$, such that

$$V_i = \{q \in \Omega \mid \|q - z_i\| \leq \|q - z_j\| \, \forall z_i, z_j \in \mathcal{Z}\}, \tag{2}$$

and $\cup_i V_i = \Omega$, where $\|\cdot\|$ denotes the Euclidean distance. A more intuitive and non-formal definition of the Voronoi Diagram is as follows. If we take some area and place points p_i in it, then each partition is the set of all points that are closer to p_i than p_j, when $j \neq i$. An example is the problem where there are post offices in some city, and it is needed to decide which house will be served by which specific post office. Using Voronoi partitioning, we can determine which houses are the nearest to each office and therefore conclude which houses each branch will serve. A visual example can be seen in Figure 4(a).

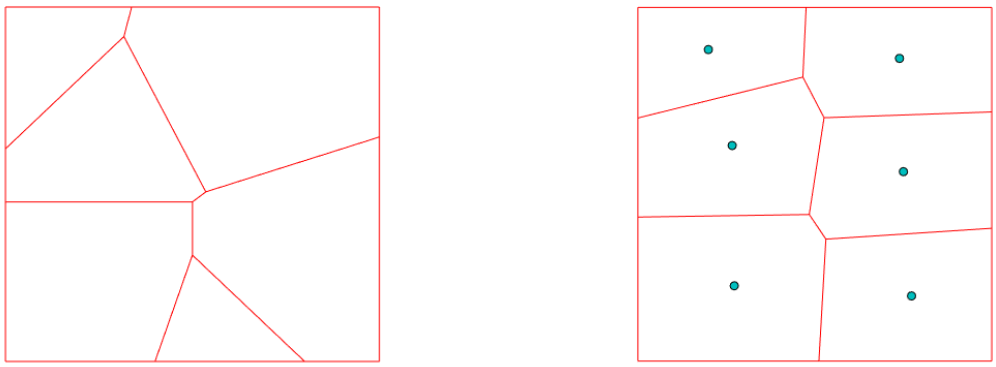
One can also define a density function, ρ_i, for each Voronoi partition V_i. Then, we can define the center of mass for each partition as,

$$z_i^* = \frac{\int_{V_i} y\rho(y)dy}{\int_{V_i} \rho(y)dy}. \tag{3}$$

If a generator $z_i = z_i^* \, \forall \, V_i$, we call this partitioning a *centroidal Voronoi tessellation* (CVT). Such tessellations are useful in terms of location optimization [5, 7, 9]. A centroidal Voronoi tessellation illustration can be seen in Figure 4(b).

While Voroni diagrams are useful for partitioning some existing range, CVT is a technique used for (but not only) for *planning* in an optimal way. If in the Voronoi diagram explanation, it was used to plan the delivery areas for each office, we will use the CVT to find where are the optimal locations to place the post offices.

As can be seen in (2), the calculation of each Voronoi cell for some agent depends on the other agents positions. Therefore, any sort of calculation requires *sharing information*. This can be done in a centralized approach (as it was done for this research), but there exists also decentralized calculations, for example [12]. Using decentralized calculation, each agent only requires information from their neighbours (or any limited amount of agents). This allows calculating huge networks, in dynamic situations where the agents' network can be changed in real time.



(a) Voronoi Diagram.

(b) Centroidal Voronoi Tessellations. The points represent the center of mass, calculated using the density function $\rho = 1$.

Figure 4: Illustration of Voronoi Diagram and a CVT, both from the same initial conditions

III.B Lloyd's Algorithm

As mentioned above, the CVT's are very useful for locational optimization. However, calculating the CVT might be a complicated task. Lloyd proposed a very simple way of calculating the CVT [10], presented in Alogirthm 1.

Algorithm 1 Lloyd's Algorithm

1: Calculate the Voronoi diagram for the current agents positions.
2: Calculate the center of mass for every cell.
3: Move the agents to the center of mass.
4: Repeat until converge.

Cortes et al. [7] proposed a control algorithm based on Lloyd's algorithm. Let us define a polygon $\mathcal{W} \subset \mathbb{R}^2$. The basic deployment problem is finding a position $p \in \mathbb{R}^2$ to maximize the coverage of the polygon $\mathcal{W}$. According to [7], minimizing the function

$$\mathcal{H}(p, \mathcal{W}) = \int_{\mathcal{W}} \|p - q\|^2 \, dq \tag{4}$$

will yield the requested result.

The function $\mathcal{H}(p, \mathcal{W})$ measures the distance between a point p to any point $q \in \mathcal{W}$. In [7], this problem was expanded to the case of n points, such that each point p_i represent the i'th sensor position, and we are looking for the best partition $(\mathcal{W}_1, \ldots, \mathcal{W}_n)$ such that the coverage is maximized. Now, the total cost function will be:

$$\mathcal{H}(P, \mathcal{W}) = \sum_{i=1}^{n} \int_{\mathcal{W}} \|p_i - q\|^2 \, dq. \tag{5}$$

The solution of this problem is the voronoi partition $\mathcal{V} = (V_1, \ldots, V_n)$.

The main challenge now is finding the points that maximize (5). According to [7], when using Voronoi partitioning, (5) can be expressed as:

$$\mathcal{H}(P, \mathcal{V}) = \mathcal{H}_{\mathcal{V}}(P) = \int_{\Omega} \min_{i \in (1,\ldots,n)} \|p_i - q\|^2 \, \phi(q) dq, \tag{6}$$

where $\phi(q)$ represents a measure of information or probability that some event take place over Ω. For the rest of the problem, and for the sake of simplicity, we assume that $\phi(q) = \rho(q) = 1$.

It is possible to define the mass M_{V_i} and the center of mass C_{V_i} of the i'th Voronoi region:

$$M_{V_i} = \int_{V_i} \rho(q)dq,$$
$$C_{V_i} = \frac{1}{M_{V_i}} \int_{V_i} q\rho(q)dq.$$

Cortes et al. [7] showed that at least one of the solutions for the optimization of (6) is the CVT:

$$C_{V_i} = \arg\min_{p_i} \mathcal{H}_{\mathcal{V}}(p). \tag{7}$$

Assuming that a sensor has integrator dynamics, i.e., for each sensor, $\dot{p}_i = u_i$, [7] shows a way to implement Lloyd's algorithm as,

$$u_i = -k_{prop}\left(p_i - C_{V_i}\right), \tag{8}$$

where k_{prop} is a positive constant. Note that the dynamics in (8), which minimizing (4), is a gradient dynamical system, and Cortes et al. [7] proved that the controller (8) is locally asymptotic stable.

IV Proposed Solution

In this chapter, Problem 1 will be addressed. First, we will propose an algorithm to address the special partition requirements, and then a solution for the whole problem will be shown.

IV.A Projected Lloyd's Algorithm

In this section, a solution for Problem 1(a) under Assumption 1 will be given. Recall that in Problem 1(a) it is required that for the partition of A, that $pr_i \cap A_m \neq \emptyset$ is satisfied. To show the solution, we will first define a *projection* onto the area A_m.

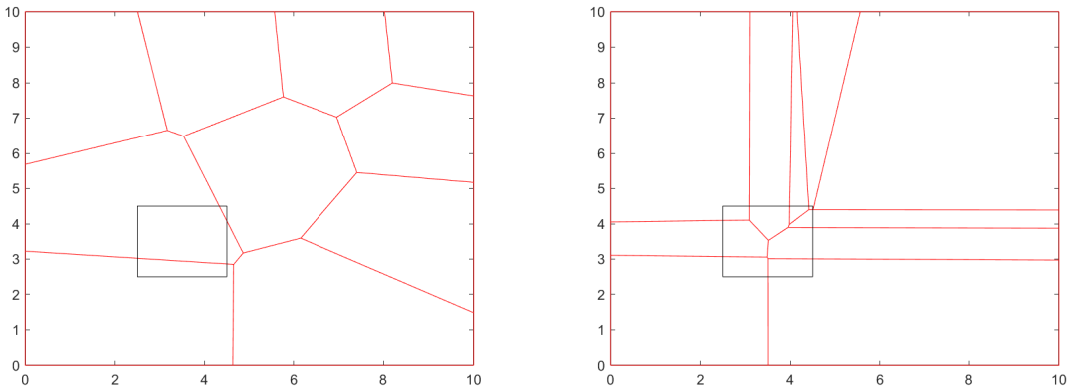
Definition 2. *Consider a set $A_m \subset \mathbb{R}^2$ and a point $x \in \mathbb{R}^2$. The* projection *of x onto A_m is given by*

$$\text{PROJ}_{A_m}(x) = \arg\min_{y \in A_m} \|x - y\|^2.$$

Applying the projection operator in conjunction with Lloyd's algorithm yields the *Projected Lloyd's Algorithm* (PLA), given in Algorithm 2. This combination allows solving Problem 1(a), and it is this work's main contribution.

Algorithm 2 Projected Lloyd's Algorithm

1: Calculate the Voronoi diagram for the current agents positions.
2: Calculate the center of mass for every cell.
3: Project the center of mass of every cell to the area constraint limiting polygon.
4: Move the agents the projected center of mass.
5: Repeat until converge.



(a) CVT calculated using Lloyd's algorithm (b) PLA solution

Figure 5: CVT and PLA solutions for initial conditions. The black box represents A_m.

As one can see, two more steps were added to the original Lloyd's algorithm - where we project the cell center of mass to the polygon that defines the area constraint (steps number 3-4), and then moving the agents to those points. Those step allows us to ensure that at least one agent will be *on* the limiting polygon, thus we will have coverage within the constraint.

In Figure 5, the PLA results can be clearly seen. For the same initial condition, Figure 5(a), it can be clearly seen that only 3 (out of 10 tiles) intersects with A_m. In Figure 5(b), the PLA algorithm was activated and all tiles intersect with A_m.

This algorithm can be also formed in a continuous time form, based on (1),

$$u_i = -k_{prop}\left(p_i - \text{PROJ}_{A_m}\left(C_{V_i}\right)\right). \tag{9}$$

This continuous time controller, which is a modification of the controller proposed in [7], is always converging into a solution where each tile intersects with A_m.

In [7], it was proven that the controller (8) converges asymptotically to a set of critical points. It was done using (6) as a Lyapunov candidate function, and then, using La'Salle's invariance principal, it was shown that the sensors converged into C_{V_i}. We employ this same idea to show the convergence of the projected Lloyd's algorithm.

Theorem 1. *The controller proposed in (9) is locally asymptotic stable.*

Proof. Let us define a Voronoi partition moment of inertia:

$$J_{V,p} = \int_V \|q - p\|^2 \rho(q) dq \tag{10}$$

In [7] it was shown that:

$$\mathcal{H}_{\mathcal{V}}(P) = \sum_{i=1}^{n} J_{V_i,C_{V_i}} + \sum_{i=1}^{n} M_{V_i}\left\|p_i - C_{V_i}\right\|^2$$
$$\frac{\partial \mathcal{H}_{\mathcal{V}}}{\partial p_i} = 2M_{V_i}\left(p_i - C_{V_i}\right).$$

Let us define $\Psi(C_v, A_m) = \text{PROJ}_{A_m}\left(C_{V_i}\right)$. We must notice that C_{V_i} is calculated in advance for each iteration, and therefore it is known.

Let us propose a candidate Lyapunov function:

$$\mathcal{H}_{\mathcal{P}}(P) = \sum_{i=1}^{n} J_{V_i,C_{V_i}} + \sum_{i=1}^{n} M_{V_i} \left\| p_i - \Psi(C_v, A_m) \right\|^2, \tag{11}$$

it follows that [7]:

$$\frac{\partial \mathcal{H}_{\mathcal{P}}}{\partial p_i} = 2M_{V_i} \left(p_i - \Psi(C_v, A_m) \right). \tag{12}$$

According to Lyapunov direct method, to prove asymptotic stability, one must show that

(1) $\mathcal{H}_{\mathcal{P}}(0) = 0$,

(2) $\mathcal{H}_{\mathcal{P}}(\zeta) > 0 \Leftrightarrow \zeta \neq 0$,

(3) $\dot{\mathcal{H}}_{\mathcal{P}}(\zeta) < 0 \Leftrightarrow \zeta \neq 0$.

Conditions 1,2 are direct from the definition of $\dot{\mathcal{H}}_{\mathcal{P}}(P)$. As for condition 3, it is possible to see [7]:

$$\frac{\partial \mathcal{H}_{\mathcal{P}}}{\partial p_i} = -2k_{prop} \sum_{i=1}^{n} M_{V_i} \left\| p_i - \Psi(C_v, A_m) \right\|^2 < 0.$$

□

Note that both the PLA and Lloyd's algorithm converge to some local minima of the cost function. Lloyd's algorithm will converge to the CVT, but the PLA converges to some other tessellation determined by the projection onto the designated area A_m.

IV.B Full Problem Solution Algorithm

The PLA is answering the question of how to partition such that $pr_i \cap A_m \neq \emptyset$. The second part of the problem, which requires a deployment controller, was answered in [7]. It is possible to utilized their solution, proposing a solution algorithm for Problem 1.

Algorithm 3 Problem 1 Solution Algorithm

1: Using some random initial guess, calculate the PLA for the whole area.
2: For each partition (assuming that the agents can actually cover each partition with their coverage radius), calculate the CVT. The initial positions for the CVT calculation is the previous partition CVT.

The above *centralized* algorithm propose a simple method to solve Problem 1. Starting by partitioning the area while maintaining $pr_i \cap A_m \neq \emptyset$, and then calculating the optimal deployment for each partition. Assuming that the sensors can cover each partition using this deployment, the problem is solved. We must notice that in stage 1, the number of the tiles of the PLA solution is decided by the user (and does not take into account $D(c(t))$). Therefore, for the assumption in step 2 to hold, the user should wisely select the number of tiles, such that the agents can provide full coverage to each tile using the optimal deployment calculated in step 2.

IV.C Results

In this sub-section, we will show results for Problem 1, using Algorithm 3. In the following simulations, the initial conditions were identical.

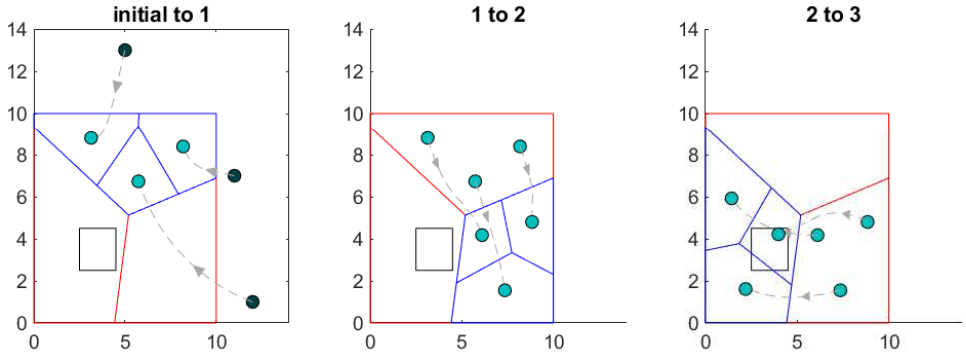


Figure 6: Results without PLA. The box represents A_m.

In Figure 6, we can see a simulation for 3 agents. The area is partitioned into 3 tiles using Lloyd's algorithm (that is CVT), and each tile is covered using the algorithm proposed in [7]. The agents moved between the tiles in arbitrary order, and one can notice that only in the last step there was coverage of A_m.

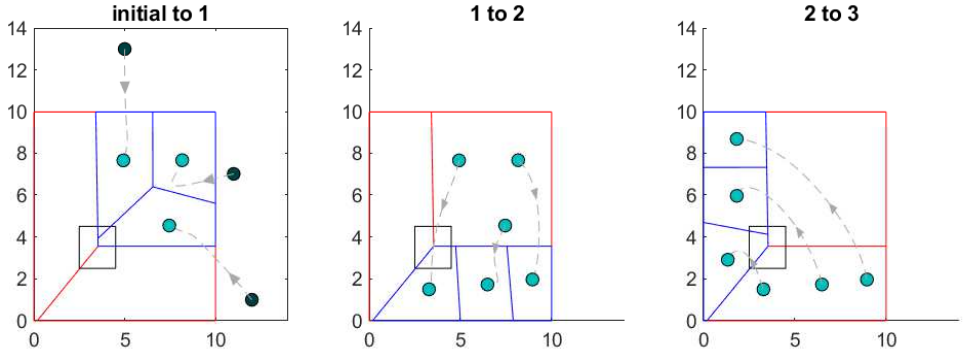


Figure 7: Results with PLA. The box represents A_m.

In Figure 7, we perform another simulation with 3 agents. The area is partitioned into 3 tiles using Projected Lloyd's algorithm, and each tile is covered using the algorithm proposed in [7]. The agents moved between the tiles in arbitrary order, and each tile has some partial coverage of A_m, as expected by the algorithm.

V Combining Lloyd's Algorithm and Distance-Based Formation Control

Sometimes the agents topology should be constrained. For example, if the agent mission is search and rescue using geolocation calculation of emitters, then the spatial structure of the formation plays a major role in the system performance. We refer to this problem as the *formation control* problem [11]. In this chapter, the *Distance-based* formation control will be combined with a deployment controller.

V.A Distance-Based Formation Control

In this section, a short mathematical background will be given, following by this work contribution.

V.A.1 Mathematical Background

A graph $\mathcal{G} = (V, \mathcal{E})$ is a pair of two finite sets - the *vertices* (nodes) set V ($|V| = n$), and edges $\mathcal{E} \subseteq V \times V$. Each vertex can represent a mathematical or physical entity, for example - a sensor. Each edge represents some relation between two entities, for example - communication connectivity or congruent sensing range. Given a configuration as defined in (1), we can define the *connectivity graph* [13]. A pair $\{\mathcal{G}, c\}$ mapping between a graph $\mathcal{G} = (V, \mathcal{E})$ and a configuration c is called a *framework*. In [14], the properties of frameworks are defined, and the most important definition for this work is given - a *minimally infinitesimally rigid framework* (MIR). In [13] the formal definitions are given.
In brief, a framework is *rigid* if every motion creates a new framework that maintains the connectivity graph $\mathcal{G}$ unchanged (in both terms of connections and distance between graph nodes). *minimally rigid framework* is a rigid framework such that the removal of each edge in $\mathcal{G}$ will result in a non-rigid framework. A *infinitesimally rigid framework* is a framework that maintains its rigidity for each infinitesimal movement of the framework nodes. A framework is MIR if it is infinitesimally rigid and minimally rigid.

A useful mathematical tool in this regard is the *rigidity matrix* (complete definition can be found on [15]). For this work, it will be marked as $R(c) \in \mathbb{R}^{|\mathcal{E}| \times 2|\mathcal{V}|}$. For example, *a framework is infinitesimally rigid if and only if* $\mathbf{rk}\,[R] = 2|\mathcal{V}| - 3 = |\mathcal{E}|$ [16]. As a result, MIR frameworks have full row rank.

V.A.2 Distance-Based Formation Controller

Let us define a framework $\{\mathcal{G}, c\}$. The graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a connectivity graph, and we also define the vector $D = \left[d_1, \ldots, d_{|\mathcal{E}|}\right]$ as the required length of each edge of the graph, while the actual length is given by $\left\|p_i - p_j\right\| = \|e_k\|\,,\ e_k = (v_i, v_j) \in \mathcal{E}$. We can now define a *formation potential* of a framework:

$$F(P) = \frac{1}{4}\sum_{k=1}^{\varepsilon}\left(\|e_k\|^2 - d_k^2\right)^2 = \frac{1}{4}\sum_{k=1}^{\varepsilon}\sigma(p_i)_k^2 = \frac{1}{4}\sigma(P)^T\sigma(P), \tag{13}$$

where P is the positions vector of the agents in a configuration c, and $\sigma(P) = [\sigma_1, \ldots \sigma_{\mathcal{E}}]\,,\ \sigma_{p_i} = \left\|e_i^2 - d_i^2\right\|$.

A proposed controller is a gradient-dynamics based controller [15, 17]:

$$\dot{p} = -\nabla F(p) = -R(p)^T R(p)p - R(p)^T D^2, \tag{14}$$

where $D^2 = \left[d_1^2, \ldots, d_{|\mathcal{E}|}^2\right]$The controller can be also defined for each individual agent. If d_{ij} is the required distance between agents i, j (that is the required distance assigned to the edge $e_k = \{v_i, v_j\}$), then

$$\dot{p_i} = -\sum_{i \sim j}\left(\left\|p_i - p_j\right\|^2 - d_{ij}^2\right)\left(p_i - p_j\right). \tag{15}$$

Being a gradient dynamical system, it can be shown that this controller is locally asymptomatic stable if it has full row rank (that is, the framework is MIR) [15, 17].

V.B Combining Distance-Based Formation Control and Lloyd's Algorithm

In this section, a combined deployment and formation controller will be shown. As both (8) and (14) are locally asymptomatic stable gradient descent controllers, it might be possible to combine them. Let us define a coefficient $0 \leq \alpha \leq 1$. The proposed controller for each agent is then

$$u_i = \dot{p_i} = \alpha \left(-k_p \left(p_i - C_{V_i}\right)\right) + (1-\alpha) \left[-\sum_{i \sim j} \left(\left\|p_i - p_j\right\|^2 - d_{ij}^2\right)\left(p_i - p_j\right)\right]. \quad (16)$$

There are few interesting results regarding this controller.

Proposition 1. *Let us assume that there exists a Central Voronoi Tessellation (CVT), with C_{V_i} such that $\forall i \neq j, \left\|C_{V_i} - C_{V_j}\right\|^2 = d_{ij}^2$ (that is C_{V_i} satisfy the formation controller constraint). Then, there exists an equilibrium such that $P_e = C_V$ (where $P_{e_i} = C_{V_i}$ and $C_V = [C_{V_1}, \ldots, C_{V_n}]$), which is locally asymptotic stable with (16) dynamics.*

Proof. First of all, let us show that $P_e = C_V$ is an equilibrium point of (16). Indeed, by assumption, we have $\forall i \neq j, \left\|C_{V_i} - C_{V_j}\right\|^2 = d_{ij}^2$, so $\left\|p_{e_i} - p_{e_j}\right\|^2 = \left\|C_{V_i} - C_{V_j}\right\|^2 = d_{ij}^2$, showing $P_e = C_V$ is an equilibrium point.

Now we are ready to prove that this equilibrium is stable. Let us choose a candidate Lyapunov function for the combined controller, which is a combination of the Lyapunov functions for each controller [7]:

$$V_c\left(P\right) = \alpha \left[\sum_{i=1}^{n} \int_{V_i} \min_{i \in \{1,\ldots,n\}} \left\|q - p_i\right\|^2 \phi(q) dq\right] + (1-\alpha)\left[\frac{1}{4}\sigma(P_e)^T \sigma(P_e)\right], \quad (17)$$

where $J_{V_i, C_{V_i}}, M_{V_i}$ are as defined in [7]. Since the equilibrium is $P_e = C_V$, we will change the variables, so $\zeta = P_e - C_V$, hence:

$$V_c\left(\zeta\right) = \alpha \left[\sum_{i=1}^{n} \int_{V_i} \min_{i \in \{1,\ldots,n\}} \left\|q - \zeta_i - C_{V_i}\right\|^2 \phi(q) dq\right] + (1-\alpha)\left[\frac{1}{4}\sigma^T(\zeta + C_{V_i})\sigma(\zeta + C_{V_i})\right]. \quad (18)$$

According to Lyapunov direct method, to prove asymptotical stability, one have to show:

(1) $V_C(0) = 0$,

(2) $V_C(\zeta) > 0 \Leftrightarrow \zeta \neq 0$,

(3) $\dot{V}_C(\zeta) < 0$.

For condition (1) [7],

$$V_C(0) = \alpha \left[\sum_{i=1}^{n} \int_{V_i} \min_{i \in \{1,\ldots,n\}} \left\|q - C_{V_i}\right\|^2 \phi(q) dq\right] + (1-\alpha)\left[\frac{1}{4}\sigma^T(C_{V_i})\sigma(C_{V_i})\right].$$

While it is obvious that $\sigma(C_{V_i}) = 0$, we have to look into the first term of the Lyapunov function. It is obvious that there is a minimum at C_V, therefore the integral gets him minimal

value over there. Moreover, since the function we are minimizing is for the form $f(x) = x^2$, the minimum is 0. Therefore, condition (1) applies.

For condition (2), it's obvious that both terms are strictly positive for $\zeta \neq 0$. For condition (3), the combined derivative of the combined Lyapunov function is [7]:

$$\dot{V}_C(\zeta) = \alpha \left[-2k_p \sum_{i=1}^{n} M_{V_i} \left\| p_{e_i} - C_{V_i} \right\|^2 \right] + (1-\alpha) \left[-\sigma(P_e)^T R(P_e) R^T(P_e) \sigma(P_e) \right] \leq 0. \quad (19)$$

As we can see, this term is smaller *or equal* to zero. Therefore, using Lyapunov method, this system is stable, however not asymptotically stable. That said, we can notice that the set $S = \{P = C_V\}$ doesn't contain any trajectory apart from the trivial trajectory (when $\zeta = 0$), therefore by LaSalle principal, the system is Locally asymptotic stable. □

Theorem 2. *The dynamic system described by the controller (16) asymptotically converges to a critical point of the function $G(p) = \alpha \mathcal{H}_{\mathcal{V}}(p) + (1-\alpha) F(p)$, where $F(p)$ is the distance-based formation controller (14) potential, and $\mathcal{H}_{\mathcal{V}}(p)$ is the Voronoi-based deployment controller (8) potential function [7].*

Proof. Both the formation control and the deployment controller can be expressed as a gradient dynamics system. It is known that gradient flows converge to the critical points of the potential function. □

Remark 1. *As the combination of controllers provide some interesting results, it is very important to remember that it doesn't necessarily solves Problem 1. The reason is that the formation is changing the deployment, thus the condition for full area coverage might be broken. This work* does not *provide a condition where the formation control holds* and *Problem 1 is fully solved.*

V.C Simulations

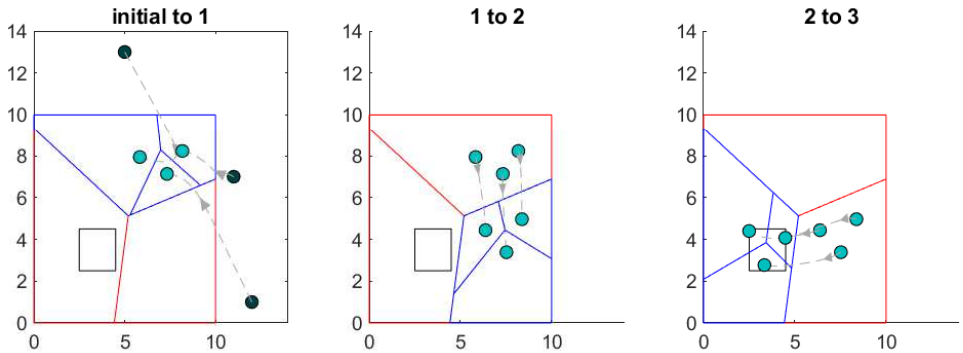


Figure 8: Results of the combination of Lloyd's algorithm and distance-based formation controller. The black box represents A_m.

In Figure 8, we can see a simulation for 3 agents where a triangular formation was defined ($d_{12} = d_{13} = 1,\ d_{23} = 2$), and combined with Lloyd's algorithm, with $\alpha = 0.5$. Comparing to Figure 6, which can be seen as the same simulation where $\alpha = 1$, it is clear that the agents are having a more triangular shape.

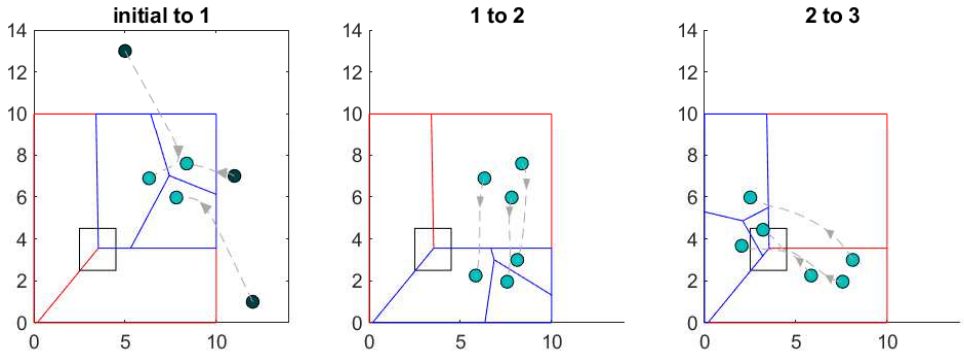


Figure 9: Results of the combination of PLA and distance-based formation controller. The black box represents A_m.

In Figure 9, we can see a simulation for 3 agents where a triangular formation was defined ($d_{12} = d_{13} = 1,\ d_{23} = 2$), and this time combined with PLA, with $\alpha = 0.5$. Comparing to Figure 7, and similar to Figure 8, it is clear that the agents are having a more triangular shape.

VI Concluding Remarks

The deployment problem under different constraints is a well known and researched problem. This work provides a new approach for partial coverage problem. When some sub-range should be covered constantly, this work presents a sequential coverage strategy, jumping between tiles. The Projected Lloyd's Algorithm, presented in this work, guarantees at least partial coverage of this sub range. Another contribution of this work is the combination of Lloyd's algorithm with a distance-based formation controller to achieve some spatial properties.

As the PLA is calculated in a centralized way, future work might include decentralizing its calculation. Future work might also include a condition for formation and deployment solution while maintaining some deployment and/or formation properties, as well as combining other types of formation controllers. Decentralizing the PLA, together with finding this condition, can lead to some interesting results for autonomous, dynamically changed swarms.

References

[1] Nigam, N., Bieniawski, S., Kroo, I., and Vian, J., "Control of multiple UAVs for persistent surveillance: Algorithm and flight test results," *IEEE Transactions on Control Systems Technology*, Vol. 20, No. 5, 2012, pp. 1236–1251.

[2] Palacios-Gasós, J. M., Montijano, E., Sagüés, C., and Llorente, S., "Multi-robot persistent coverage with optimal times," in "2016 IEEE 55th Conference on Decision and Control (CDC)," , 2016, pp. 3511–3517.

[3] Loizou, S. G. and Constantinou, C. C., "Multi-robot coverage on dendritic topologies under communication constraints," in "2016 IEEE 55th Conference on Decision and Control (CDC)," , 2016, pp. 43–48.

[4] Cassandras, C. G. and Li, W., "Sensor Networks and Cooperative Control," *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pp. 4237–4238.

[5] Atinç, G. M., Stipanović, D. M., Voulgaris, P. G., and Karkoub, M., "Supervised coverage control with guaranteed collision avoidance and proximity maintenance," *Proceedings of the IEEE Conference on Decision and Control*, pp. 3463–3468.

[6] Hussein, I. I. and Stipanovic, D. M., "Effective Coverage Control for Mobile Sensor Networks With Guaranteed Collision Avoidance," *IEEE Transactions on Control Systems Technology*, Vol. 15, No. 4, 2007, pp. 642–657.

[7] Cortes, J. and Martinez, S., "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, Vol. 20, No. 2, 2004, pp. 243–255.

[8] Li, W. and Cassandras, C. G., "Distributed cooperative coverage control of sensor networks," *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference, CDC-ECC '05*, Vol. 2005, No. January 2006, 2005, pp. 2542–2547.

[9] Du, Q., Faber, V., and Gunzburger, M., "Centroidal Voronoi Tessellations: Applications and Algorithms," *SIAM Review*, Vol. 41, No. 4, 1999, pp. 637–676.

[10] Lloyd, S., "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, Vol. 28, No. 2, 1982, pp. 129–137.

[11] Oh, K. K., Park, M. C., and Ahn, H. S., "A survey of multi-agent formation control," *Automatica*, Vol. 53, 2015, pp. 424–440.

[12] Adams, J. S., Sun, W., and Chen, Y. Q., *Formations with decentralized Centroidal Voronoi tessellation algorithm*, Vol. 42, IFAC, 2009.

[13] Muhammad, A., Ji, M., and Egerstedt, M., *Applications of Connectivity Graph Processes in Networked Sensing and Control*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 69–82, 2006.

[14] Roth, B., "The Rigidity," *Journal of Mathematical Analysis and Applications*.

[15] Krick, L., Broucke, M. E., and Francis, B. A., "Stabilization of infinitesimally rigid formations of multi-robot networks," *Proceedings of the IEEE Conference on Decision and Control*, pp. 477–482.

[16] Tay, T. S. and Whiteley, W., "Recent Advances in the Generic Ridigity of Structures," *Topologie structurale*, Vol. 9, 1984, pp. 31–38.

[17] Kwang-Kyo Oh and Hyo-Sung Ahn, "Distance-based formation control using Euclidean distance dynamics matrix: General cases," *Proceedings of the 2011 American Control Conference*, pp. 4816–4821.