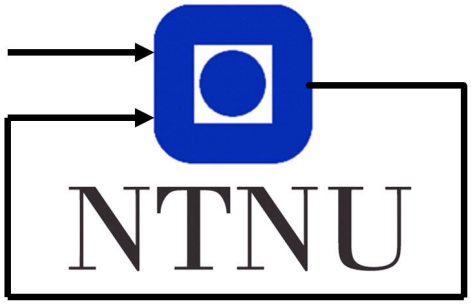
Efficient area coverage using a network of drones

Erlend Lone, 10046

December 17, 2020



Department of Engineering Cybernetics

Abstract

This report investigates different approaches for solving the area coverage problem using a network of drones. It is assumed that the drones have a priori knowledge about the map and should construct a mesh network to be able to detect and localize first responders in disaster situations. The drones are small and have limited computational resources. It is therefore assumed that the network communicates with a powerful base station that does all the computationally expensive tasks. To ensure that the base can get information from all drones, it is of importance that the network stays connected at all times. This means that the distance between drones cannot be greater than a given threshold, and there must exist a communication path from each drone to the base.

Four main techniques for multi-agent area coverage that have been proposed by different researchers are described, and one of them is simulated using Python. These four techniques are area partitioning and coordinate calculation, Voronoi diagrams, virtual potential fields, and the gradient ascent method. The three latter ones are all used extensively for controlling multi-robot systems. The implementation is inspired by an algorithm that takes into account how frequently some event happens in the mission space and maximizes the probability that the event is detected.

This project has been carried out in collaboration with the INGENIOUS project funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 833435 and the Norwegian Research Council project "Autonomous Underwater Fleets" under the grant agreement No 302435.

Contents

List of Acronyms

iNGENIOUS Next-GENeration IoT sOlutions for the Universal Supply chain

IoT Internet of Things

EU European Union

NGIT Next Generation Integrated Toolkit

SINTEF Stiftelsen for industriell og teknisk forskning

MIN Micro Indoor droNe

FR First responder

LiPo Lithium Polymer

PA Power amplifier

USB Universal Serial Bus

CVT Centroidal Voronoi tessellation

VFA Virtual force algorithm

1 Introduction

iNGENIOUS[1] is an EU project that aims to assist first responders (FRs) in disaster scenes and emergency situations. The project consists of developing, integrating, testing, and validating the Next Generation Integrated Toolkit (NGIT) for collaborative response. The use of smart uniforms, boots and helmets, augmented reality, data intelligence and drone swarms are just some of the tools and services the project wants to implement in what is called The First Responder of the Future. Out of several tasks SINTEF has been assigned as part of the iNGENIOUS project, the construction of the Micro Indoor droNe (MIN) platform is considered the main one. The MINs will form a swarm of aerial vehicles which will support search and rescue operations in dangerous areas. The goal is that the MINs should create a positioning network, helping the FRs localizing themselves indoor.

The MIN that is to be used to perform the task discussed above, is called Crazyflie 2.1, depicted in fig. 1. This is a small, lightweight drone developed and manufactured by Bitcraze. There are several ways to control the Crazyflie 2.1. For the prototyping phase, SINTEF controls the Crazyflies by Python scripts running on a laptop that send commands to the drones through the Crazyradio PA. Crazyradio PA is a long-range open USB radio dongle based on the nRF24LU1+ from Nordic Semiconductor. As the network of MINs should be deployed to disaster areas, it might not be possible to use a base station that can communicate reliably with the drones. Therefore, the long-term goal is that the control of the network is implemented through the embedded software running on each single drone.

Figure 1: Crazyflie 2.1.[2]

[1] https://ingenious-first-responders.eu/ingenious-project/
[2] https://www.bitcraze.io/products/crazyflie-2-1

1.1 Motivation

The technological advances within data processing, batteries and drone technology in general enable the development of collaborating swarms of small, lightweight drones. The increased energy density in LiPo batteries leads to increased flight time, which is the main limitation for multirotor drones. Despite the vast improvements within processing power, most commercially available computers do not provide enough computational power to make all the data processing on-board [6]. Because of this, the implementation described in section 3.2 assumes that a base station with strong computational resources is used.

1.2 Related work

The problem of deploying multiple sensors for effective area coverage has been studied in a variety of different ways over the years. The paradigm of multi-robot area coverage was introduced in [12], and the material presented later in the report will investigate what that article defines as *blanket coverage*. This means that the objective is to reach a static configuration that maximizes the detection probability of targets in a given area.

Voronoi diagrams partition an area into several regions with respect to a set of points called *generators*. Both [10] and [11] construct Voronoi diagrams with respect to each robot's position. [11] lets one agent move at the time, constrained that the movement of that agent will increase the total area covered by the network. [10], on the other hand, formulates a sensory function defined on the mission space, describing its importance density. Locally optimal coverage is then achieved when the position of each robot is at the centroid of its Voronoi region.

The concept of virtual potential forces was first presented in [8] and has been widely used in obstacle avoidance and local navigation. [9], [13] and [19] are all examples where this technique has been utilized for deployment of sensor nodes.

Formulating the area coverage as an optimization problem to maximize detection probability has also been studied thoroughly. In [14] Li and Cassandras introduced the idea of combining a gradient based update scheme of the state of the sensors, while implementing a routing protocol so that all sensor nodes could communicate with a base station. This work was further expanded to include minimization of communication to preserve energy in [16], and connectivity preservation in [17].

1.3 Problem definition

The overall problem discussed in this report is how a network of drones efficiently can cover a known map while ensuring that the drone network is connected at all times. The coverage part of this problem is reminiscent of the well-studied *facility location problem* [4], [5], which aims to optimize the placement of facilities to minimize the weighted Euclidean distance between the facilities and points that can be interpreted as users. Due to the connectivity constraint, the problem that is researched in this report differs somewhat from the facility location problem.

Let $\mathcal{M}$ be the map of the area of interest, C_i denote the area covered by drone i and $\mathcal{L}$ be the Laplacian matrix of the graph that is constructed by the network. λ_1 and λ_2 are the lowest and second lowest eigenvalue of $\mathcal{L}$. The problem can then be formulated as

$$\begin{aligned} \max \ & \mathcal{M} \cap \bigcup_i C_i \\ \text{s.t.} \quad & \lambda_1(\mathcal{L}) = 0 \\ & \lambda_2(\mathcal{L}) > 0 \end{aligned} \tag{1}$$

It is assumed that the map $\mathcal{M}$ is provided by other robots or drones and therefore known to the drone network prior to the entry. A powerful base station is assumed used to handle the computationally expensive tasks, and the connectivity constraint ensures that all the drones can send information to the base station through the network.

Throughout this report, what is defined to be drones in the above formulation is referred to as agents, robots, sensors, and nodes. The map to be covered is also called mission space, sensing area, sensing field and region under surveillance.

1.4 Report outline

Section 2 presents relevant background theory so that the reader should have a decent understanding of what is described later in the report, without having prior knowledge of the topics presented. Section 3 delves into four solutions of how the area coverage problem can be solved. Section 4 presents the performance of the implemented approach, while section 5 concludes the work done in this project and discusses several interesting paths to investigate in the future.

2 Background theory

This section presents theory that is used in the approaches that are described in section 3. For a more in-depth explanation of this theory, the reader is referred to the given references.

2.1 Graph theory

A graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of n vertices and $\mathcal{E}$ is a set of edges such that $\mathcal{E} = \{v_i, v_j\}$ with $i \neq j$, $i, j = 1, \ldots, n$. The graph is *undirected* if $(v_i, v_j) \in \mathcal{E}$ and $(v_j, v_i) \in \mathcal{E}$. If $x, y \in \mathcal{V}$ and $(x, y) \in \mathcal{E}$ then the nodes x, y are said to be *connected*. A path of length r is a sequence of $r + 1$ distinct and adjacent vertices. If there is a path between any pair of vertices, then the graph is *connected*.

The *adjacency matrix* of a graph $\mathcal{G}$ is a symmetric $[n \times n]$ matrix where all elements are either 0 or 1. $\mathcal{A}(i, j) = 1$ if $(v_i, v_j) \in \mathcal{E}$ and $\mathcal{A}(i, j) = 0$ if $(v_i, v_j) \notin \mathcal{E}$. The *degree matrix* of $\mathcal{G}$ is a diagonal $[n \times n]$ such that $\mathcal{D}(i, i) = d(v_i)$, where $d(v_i)$ is the number of vertices adjacent to v_i.

Using $\mathcal{A}$ and $\mathcal{D}$ one can define what is known as the *Laplacian matrix* as $\mathcal{L} = \mathcal{D} - \mathcal{A}$. This matrix is symmetric and positive semi-definite and can be used to analyze whether the graph is connected or not. The graph is connected if the smallest eigenvalue of $\mathcal{L}$ is $\lambda_1 = 0$, and the second smallest eigenvalue is $\lambda_2 > 0$.

2.2 Voronoi diagrams

Let $\Omega \subseteq \mathbb{R}^n$ and $P = \{p_1, \ldots, p_k\}$ where all p_i are distinct points in Ω and referred to as *generators*. The Voronoi *region* corresponding to point p_i is then defined as:

$$V_i = \left\{ q \in \Omega \mid \|q - p_i\|_2 \leq \|q - p_j\|_2 \quad \text{for } j = 1, \ldots, k, j \neq i \right\} \tag{2}$$

Each V_i is a polyhedron and $V_i \cap V_j = \emptyset$ for $i \neq j$. The Voronoi *diagram* of Ω is then the set $\{V_i\}_{i=1}^{k}$. Optimal placement of resources is just one of a variety of applications where Voronoi diagrams are useful [5].

Voronoi regions are sometimes referred to as Voronoi *cells*, and Voronoi diagrams are also called Voronoi *tessellation*, Voronoi *decomposition* or Voronoi *partition*. If the centroid of a Voronoi region is used as generator, the tessellation is called Centroidal Voronoi tessellation (CVT).

2.3 Virtual potential fields

The material presented here is found in [7]. The use of virtual potential forces for obstacle avoidance is common in mobile robotics. This is achieved by constructing the potential field U so that each robot is repelled from the

other robots and obstacles in the mission space. Each robot is then subjected to a force

$$\mathbf{F} = -\nabla U \tag{3}$$

The potential field is divided into two components. U_o describes the field that is due to the presence of obstacles in the environment, and U_r represent the field that results from other robots. U can therefore be rewritten as $U = U_o + U_r$, which results in $\mathbf{F} = \mathbf{F}_o + \mathbf{F}_r$.

U_o is constructed in a similar manner as the electrostatic potential between two electrically charged particles. Let k_o be a constant describing the strength of the field and $\mathbf{x}_i$ and $\mathbf{x}$ denoting the position of the obstacle i and the position of the robot, respectively. Then $r_i = \|\mathbf{x}_i - \mathbf{x}\|_2$ will be the Euclidean distance between the robot and obstacle i. The potential field resulting from all observable obstacles can then be defined as

$$U_o = k_o \sum_i \frac{1}{r_i} \tag{4}$$

The potential field due to other robots can be constructed in the same way but summing over all visible robots j instead of summing over all visible obstacles.

$$U_r = -k_r \sum_j \frac{1}{r_j} \text{ [sic]} \tag{5}$$

Using the definitions of the potential fields, the virtual force field that the visible obstacles exerts on each robot is given as

$$\mathbf{F}_o = -\nabla U_o = -\frac{\mathrm{d}U_o}{\mathrm{d}\mathbf{x}} = -\sum_i \frac{\mathrm{d}U_o}{\mathrm{d}r_i} \cdot \frac{\mathrm{d}r_i}{\mathrm{d}\mathbf{x}} \tag{6}$$

Which results in

$$\mathbf{F}_o = -k_o \sum_i \frac{1}{r_i^2} \cdot \frac{\mathbf{r}_i}{r_i} \tag{7}$$

and the force exerted by the other robots

$$\mathbf{F}_r = -k_r \sum_j \frac{1}{r_j^2} \cdot \frac{\mathbf{r}_j}{r_j} \tag{8}$$

Combining eqs. (7) and (8) gives

$$\mathbf{F} = -k_o \sum_i \frac{1}{r_i^2} \cdot \frac{\mathbf{r}_i}{r_i} - k_r \sum_j \frac{1}{r_j^2} \cdot \frac{\mathbf{r}_j}{r_j} \tag{9}$$

2.4 Gradient descent/ascent method

When finding the optimal solution $f(x^*)$ of a convex objective function $f(x)$, it is natural to search in the direction of the gradient of $f(x)$, denoted $\nabla f(x)$. The search direction is decided by whether the goal is to maximize or minimize $f(x)$. The maximum of $f(x)$ can be found by the *gradient ascent method*, while the minimum of $f(x)$ can be found by the *gradient descent method*. The only difference between these two methods is whether the search direction is along $+\nabla f(x)$ or $-\nabla f(x)$. If the objective function is non-convex, gradient descent/ascent might converge to a stationary point that is not the global minimum/maximum.

Given a starting point x for the unconstrained maximization problem $\max_{x \in \mathbb{R}^n} f(x)$, the working steps of the gradient ascent method is to first evaluate $\nabla f(x)$, then choose step size $\alpha > 0$ either by exact or backtracking line search [1] and update x according to $x = x + \alpha \nabla f(x)$. This procedure is repeated until a stopping criterion is satisfied, which usually is on the form $\|\nabla f(x)\|_2 \leq \eta$, where η is a small, positive number.

3 Methods

3.1 Literature review

The work on this thesis started out with a literature review to get a good understanding of what research that has been done previously on the area coverage problem. Not only did the literature review provide information about how different researchers have attacked the problem, but it also provided insight into how the problem stated in section 1.3 could be solved most efficiently. Four different approaches will now be presented, as well as explanations why the methods described in the first three articles were not pursued.

3.1.1 Efficient Deployment Algorithms for Ensuring Coverage and Connectivity of Wireless Sensor Networks

Efficient Deployment Algorithms for Ensuring Coverage and Connectivity of Wireless Sensor Networks by You-Chiun Wang, Chun-Chi Hu and Yu-Chee Tseng [15] solves the deployment of sensor nodes by partitioning the sensing area into *small* and *large* regions and finds the coordinates of the sensor nodes. It is assumed that each sensor node has circular sensing and communication ranges centered at the node's position and with radius r_s and r_c, respectively. The placement of the sensor nodes depends on the relationship between r_s and r_c and is different for small and large regions. Small regions are defined as areas where the distance between individual obstacles and between obstacles and the boundary of the sensing area is less than $\sqrt{3}r_{min}$, where $r_{min} = \min(r_s, r_c)$. Large areas are simply areas that are not small.

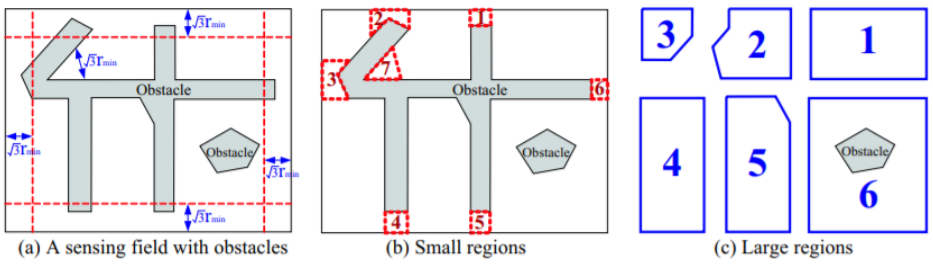


Figure 2: Partition of sensing field in [15].

For small regions, the bisector of the region is found by triangulation, and sensors are placed along this bisector, a distance of r_{min} apart. Note that if the bisector intersects with a corner, a sensor is placed in the corner as well. This procedure ensures both connectivity and coverage, as shown in fig. 3.

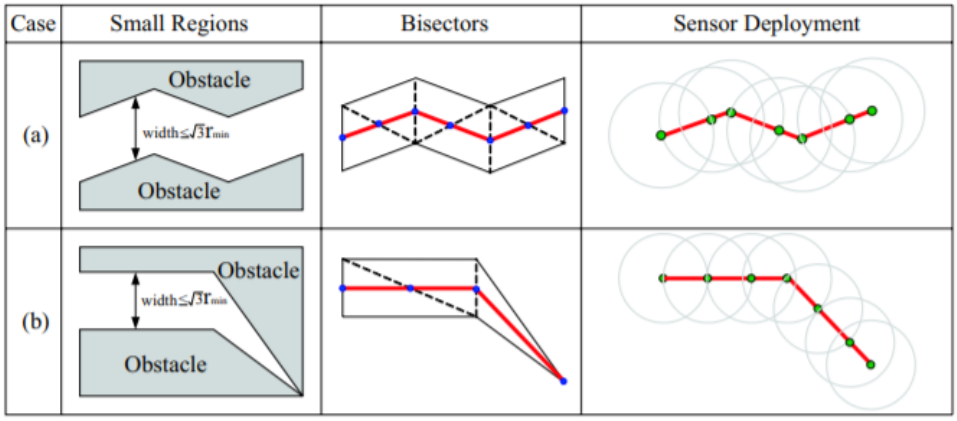


Figure 3: Coverage of *small* regions in [15].

For the large regions, the first sensor is placed at an arbitrary point, and then the coordinates where six potential neighbors should be placed is calculated according to table 1. Each N_i in table 1 are placed in a queue Q and the algorithm enters a loop. For each iteration, a point (x, y) is dequeued from Q, and a sensor is placed at (x, y) if the point is inside the region, and not inside an obstacle. The last step of the iteration is to calculate the six potential neighbors of the sensor placed at (x, y) and add them to Q if their positions are feasible. This procedure repeats until Q is empty.

Neighbor	$r_c \leq \sqrt{3}r_s$	$r_c > \sqrt{3}r_s$
N_1	$(x + r_c, y)$	$\left(x + \sqrt{3}r_s, y\right)$
N_2	$\left(x + \frac{r_c}{2}, y - \sqrt{r_s^2 - \frac{r_c^2}{4}} - r_s\right)$	$\left(x + \frac{\sqrt{3}r_s}{2}, y - \frac{3r_s}{2}\right)$
N_3	$\left(x - \frac{r_c}{2}, y - \sqrt{r_s^2 - \frac{r_c^2}{4}} - r_s\right)$	$\left(x - \frac{\sqrt{3}r_s}{2}, y - \frac{3r_s}{2}\right)$
N_4	$(x - r_c, y)$	$\left(x - \sqrt{3}r_s, y\right)$
N_5	$\left(x - \frac{r_c}{2}, y + \sqrt{r_s^2 - \frac{r_c^2}{4}} + r_s\right)$	$\left(x - \frac{\sqrt{3}r_s}{2}, y + \frac{3r_s}{2}\right)$
N_6	$\left(x + \frac{r_c}{2}, y + \sqrt{r_s^2 - \frac{r_c^2}{4}} + r_s\right)$	$\left(x + \frac{\sqrt{3}r_s}{2}, y + \frac{3r_s}{2}\right)$

Table 1: Coordinates for the six potential placements of the next sensors [15].

When all the regions in fig. 2 are covered, the final configuration of the sensor network will be as shown in fig. 4. It should be noted that even though the final configuration of the network will be connected, there is nothing that ensures that it is connected throughout the execution.

The procedure described above was implemented at an early stage of this project. The connectivity constraint specified in section 1.3 was not considered, and the implementation was mostly done for illustration purposes. This was a useful experience, as it became apparent that it would be hard to design a control scheme that ensured connectivity throughout the deployment. Because of this, and that the problem solved in [15] is difficult to express mathematically, lead to the conclusion to not pursue this approach for the implementation part.

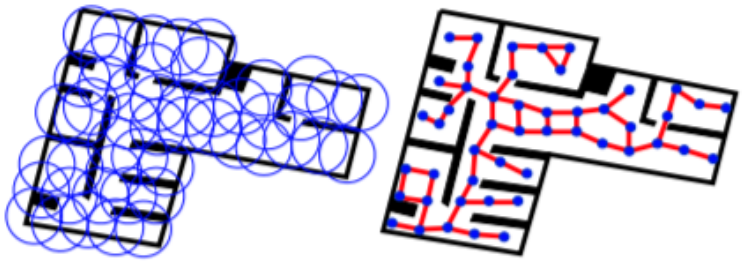


Figure 4: Final coverage of sensing field [15].

3.1.2 Sensor Deployment and Target Localization Based on Virtual Forces

A virtual force algorithm (VFA) is presented by Yi Zou and Krishnendu Chakrabarty in *Sensor Deployment and Target Localization Based on Virtual Forces* [19]. The pseudocode is shown in fig. 5. The algorithm assumes that there is a given number of sensors placed randomly in a sensing field and aims to maximize the covered area. The sensors do not move during the execution of the VFA. The algorithm generates trajectories between the initial and final positions. After the algorithm is finished, each sensor follows its generated path. The sensing area is modeled as a circle centered at the sensor's position. Zou and Chakrabarty presents both a binary (detected/not detected) and a probabilistic model of the sensing capabilities. The probabilistic model is more realistic in regard to how the sensors will behave in real life, but to keep things simple, only the binary model will be taken into account here.

[19] uses a cluster-based network architecture, which means that there is a cluster head in the network that has more computational resources than the sensor nodes. This cluster head is the one that executes the VFA and generates the paths of the sensor nodes. It is further assumed that all sensor nodes can communicate with the cluster head from the initial deployment, and that the sensor nodes only send messages to the cluster head indicating whether a target is detected/not detected. The cluster head will then query more information if necessary.

Procedure *Virtual_Force_Algorithm* (Grid, $\{s_1, s_2, \cdots, s_k\}$)

Set *loops* = 0;
Set *MaxLoops* =**MAX_LOOPS**;
While ($loops < MaxLoops$)
  /* coverage evaluation */
  For $P(x, y)$ in Grid, $x \in [1, width], y \in [1, height]$
    For $s_i \in \{s_1, s_2, \cdots, s_k\}$
      Calculate $c_{xy}(s_i, P)$ from the sensor model using $(d(s_i, P), c_{th}, d_{th}, \alpha, \beta)$;
    End
    If coverage requirements are met
      Break from **While** loop;
    End
  End
  /* virtual forces among sensors */
  For $s_i \in \{s_1, s_2, \cdots, s_k\}$
    Calculate $\vec{F}_{ij}$ using $d(s_i, s_j)$, d_{th}, w_A, w_R;
    Calculate $\vec{F}_{iA}$ using $d(s_i, PA_1, \cdots, PA_{n_P})$, d_{th};
    Calculate $\vec{F}_{iR}$ using $d(s_i, OA_1, \cdots, OA_{n_O})$, d_{th};
    $\vec{F}_i = \sum \vec{F}_{ij} + \vec{F}_{iR} + \vec{F}_{iA}, j \in [1, k], j \neq i$;
  End
  /* move sensors virtually */
  For $s_i \in \{s_1, s_2, \cdots, s_k\}$
    $\vec{F}_i(s_i)$ virtually moves s_i to its next position;
  End
  Set *loops* = *loops* + 1;
End

Figure 5: Pseudocode of the virtual force algorithm proposed in [19]. The input parameters are a grid representation of the sensing field and the initial positions of the sensor nodes.

The sensor nodes will be subject to repelling forces from obstacles and other sensor nodes that are within a given threshold. What [19] does differently than what is described in section 2.3, is that nodes that are in areas with low coverage will exert attractive forces on the sensor nodes that are in more clustered areas. This leads to a uniform coverage. The sensors will also experience attractive forces from areas that are defined of higher importance to cover.

Let the sensor node i be placed at s_i, P be a point in the sensor field and r be the radius of the sensing range of node i. Then the binary sensing model is expressed as

$$c_{xy}(s_i) = \begin{cases} 1, \text{ if } \|s_i - P\| < r \\ 0, \text{ otherwise} \end{cases} \tag{10}$$

The force $\vec{F}_i$ denotes the force that sensor i experiences. This will be the sum of the force exerted on sensor i by the other sensors, the total repulsive force from obstacles and the total attractive forces. This can be written as

$$\vec{F}_i = \sum_{j=1,\ j\neq i}^{k} \vec{F}_{ij} + \vec{F}_{iR} + \vec{F}_{iA} \tag{11}$$

To express the force between node i and j, polar coordinates are used, where the format is $(\|\vec{F}_{ij}\|, \angle\vec{F}_{ij})$. The Euclidean distance between sensor i and j is denoted d_{ij} and the distance d_{th} (threshold) is the minimum separation distance of the nodes. ω_A and ω_R are measures for the attractive and repulsive forces, respectively. α_{ij} is the angle of the line between node i and j.

$$\vec{F}_{ij} = \begin{cases} \left(w_A\left(d_{ij} - d_{th}\right), \alpha_{ij}\right), & \text{if } d_{ij} > d_{th} \\ 0, & \text{if } d_{ij} = d_{th} \\ \left(w_R \frac{1}{d_{ij}}, \alpha_{ij} + \pi\right), & \text{otherwise} \end{cases} \tag{12}$$

Figure 6 shows four sensors and the forces that act on S_1. S_2 is attracting S_1, S_3 is repelling S_1, and S_1 is not experiencing any force from S_4. F_1 is not drawn in fig. 6, but will in this case be $F_1 = F_{12} + F_{13}$.

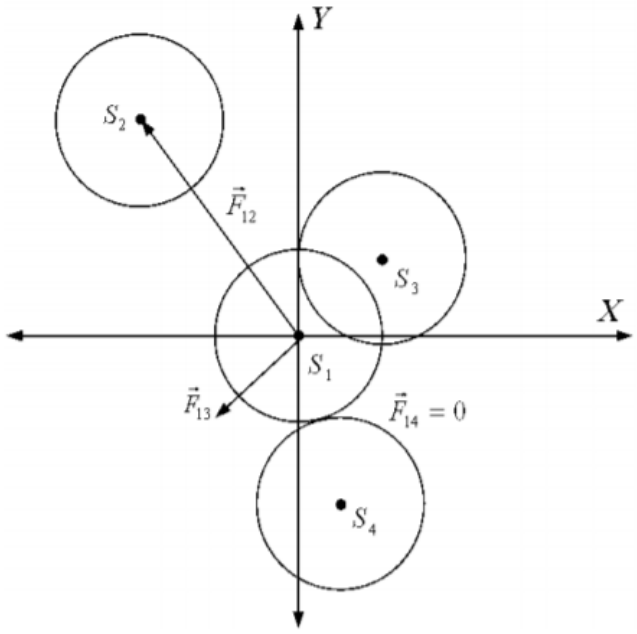


Figure 6: Virtual forces acting on S_1, as described in [19].

With the initial deployment as shown in fig. 7 and using the binary

sensing model eq. (10), Zou and Chakrabarty achieved the coverage that is shown in fig. 8.

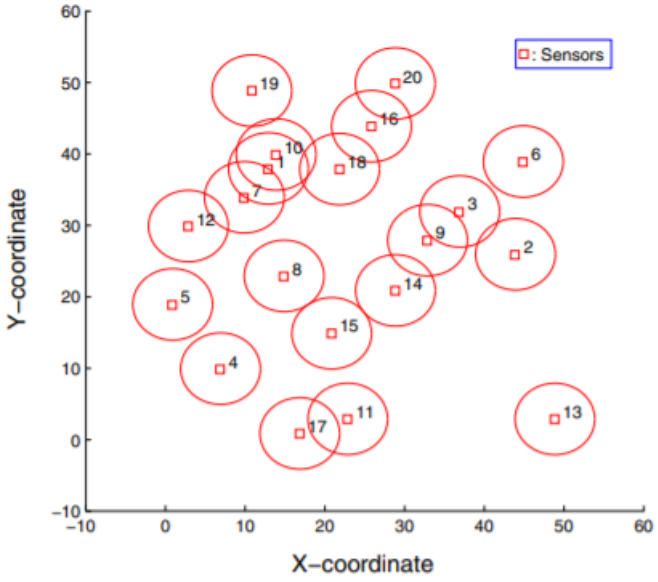


Figure 7: Initial deployment of sensors in [19].

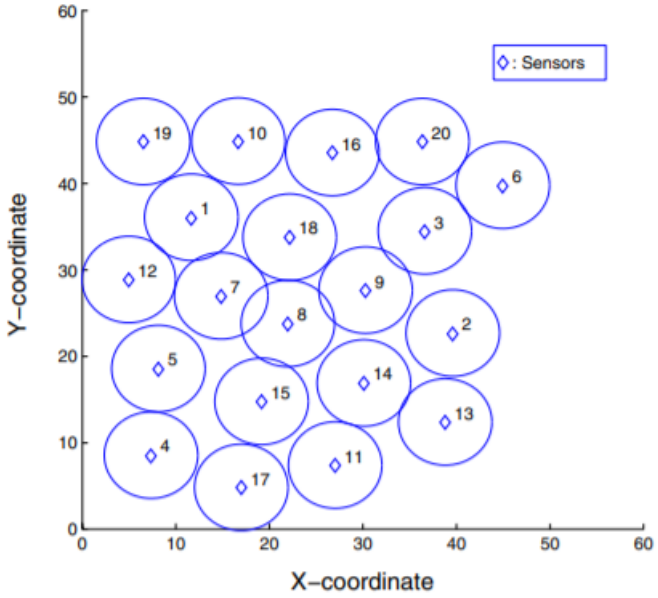


Figure 8: Final deployment of sensors in [19].

Looking at the final configuration of the sensor network, it would be tempting to implement the VFA to solve the problem defined in section 1.3. The VFA assumes that each sensor in the cluster can communicate directly with the cluster head. This imply that the sensor nodes will have a communication range that is far beyond what is applicable for the problem that this project tries to solve. Even though it was of great interest early in this project to implement the VFA, the above-mentioned drawbacks made it clear that it would be better to look for other approaches to implement.

3.1.3 Voronoi–based Coverage Optimization for Mobile Networks with Limited Sensing Range — A Directional Search Approach

In *Voronoi–based Coverage Optimization for Mobile Networks with Limited Sensing Range — A Directional Search Approach* by John Stergiopoulos and Anthony Tzes [11], a given number of agents are placed arbitrarily in the region under surveillance Ω. Ω is then partitioned using Voronoi tessellation, where the positions of the agents are used as generators (section 2.2). Two different algorithms are applied as control policies for coverage optimization, and their performance is compared and analyzed. The trajectories and final positions the two algorithms produce are shown in figs. 9 and 10.

The first algorithm selects, at each time step, an agent that performs an optimization problem for finding the direction within its Voronoi region that increases the total area covered by the network the most. This approach guarantees that the total area covered by the sensor network will increase for each time step, as well as guaranteeing that it increases as fast as possible.

The second approach moves the agents toward the centroid of their R-limited Voronoi region. This region is defined as $V_i \cap C_i$, where V_i is the Voronoi region of agent i and C_i is the uniform circular sensing region centered at the agents position. As the agents move, the Voronoi region for the different agents will necessarily also change, until the sensor network reaches its final state. It is worth noting that this approach follows a scheme called CVT, and is known to not reach optimum coverage, but instead optimize some symmetry criterion.

The work done in [11] is interesting and it would be appealing to implement the discussed algorithms. With that being said, the algorithms do not take a connectivity constraint into account, and each agent is assumed to compute its Voronoi region and keep track of the positions of the agents in its neighboring Voronoi regions. It is uncertain whether the Crazyflie 2.1 is capable of this, and as the algorithm proposed in the article described in section 3.1.4 is more clearly defined, implementing the approach in [11] was not pursued.

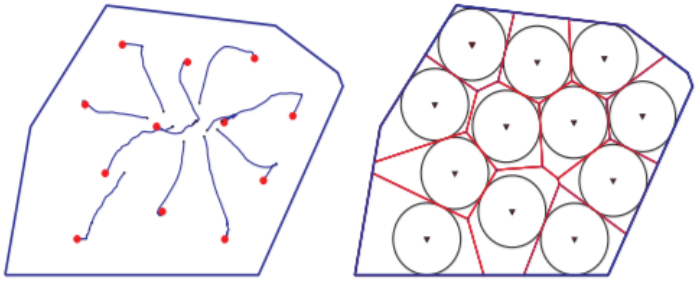


Figure 9: Trajectories and final positions of the agents using the first algorithm presented in [11].

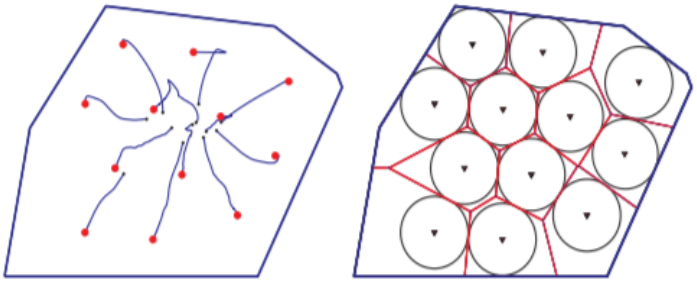


Figure 10: Trajectories and final positions of the agents using the second algorithm presented in [11].

3.1.4 Distributed Coverage Control and Data Collection With Mobile Sensor Networks

The fourth and final article that will be presented in this section is *Distributed Coverage Control and Data Collection With Mobile Sensor Networks* by Minyi Zhong and Christos G. Cassandras [18]. As the implementation described in section 3.2 is very similar to the algorithm presented by Zhong and Cassandras, this article will be described in more detail compared to the three preceding ones.

Distributed Coverage Control and Data Collection With Mobile Sensor Networks builds upon the work done by Li and Cassandras in [2] and [14]. [2] gives an overview of what a sensor network is, described from a system and control theory perspective. [14] on the other hand, looks at the network as a data collection network, with multiple data sources and a single base station, and designs a gradient based algorithm to maximize the probability that the network detects random events. The gradient based coverage control scheme used in *Distributed Coverage Control and Data Collection With Mobile Sensor Networks* was first presented in [3], and uses the routing information from each node to a base station. The routing information

is conserved by a wireless routing algorithm that runs in parallel with the optimization process.

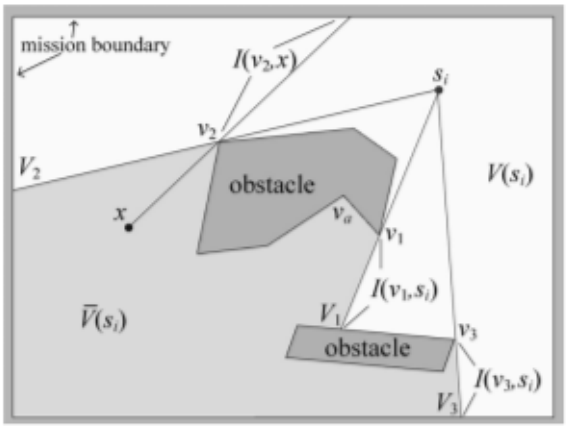


Figure 11: Mission space as defined in [18].

The authors states that a sensor network has to perform three tasks simultaneously. These tasks are *coverage control*, *data source detection* and *data collection*, where the two first-mentioned ones are most relevant for the problem defined in section 1.3. How these two tasks are solved in *Distributed Coverage Control and Data Collection With Mobile Sensor Networks* are presented below.

The *mission space* is defined as a non-self-intersecting polygon $\Omega \subset \mathbb{R}^2$. $R(x)$ is a function that describes the event density, and maps $x \in \Omega$ to $\mathbb{R}$. Further, $R(x) \geq 0 \ \forall x \in \Omega$ and $\int_\Omega R(x)dx < \infty$.

Obstacles in the mission space are also modeled as non-self-intersecting polygons, denoted $P_j \subset \Omega, j = 1, \ldots, m$, where m is the number of obstacles. The interior of each obstacle is infeasible for the sensors and denoted $\overset{\circ}{P}_j$. Hence, the feasible part of the mission space can be formulated as $F = \Omega \backslash \left(\overset{\circ}{P}_1 \cup \cdots \cup \overset{\circ}{P}_m \right)$. As the infeasible points are not of interest, the event density is defined as $R(x) = 0$ for $x \notin F$. Let N sensors be deployed, then the location vector for the sensor network is given as $\mathbf{s} = (s_1, \ldots, s_N)$ where $s_i \in F$, $i = 1, \ldots, N$. If there are no obstacles present and the mission area is convex, the sensing capabilities of each sensor in the network is modelled as

$$p_i(x, s_i) = p_{0i} e^{-\lambda_i ||x - s_i||} \tag{13}$$

where $p_{0i} \in (0, 1]$ and λ_i is a positive constant.

∂ is often used as prefix to describe the boundary of a topological set. Therefore, the boundary of the feasible set F can be written as $\partial F = \partial\Omega \cup P_1 \cdots P_m$. T is defined as the set of vertices of ∂F. A *reflex*

vertex is a vertex v in T where two edges inside F intersect at v and form an angle $\theta > \pi$.

For a point $x \in F$ to be visible from $y \in F$, all points on the line between x and y have to be inside F. This is expressed as $(\alpha x + (1-\alpha)y) \in F$ $\forall \alpha \in [0,1]$. The set of all such lines associated with x is defined as the *visibility polygon* at x and expressed as $V(x) \subset F$. Leading directly from this definition, the *invisibility polygon* at x is defined as $\overline{V}(x) = F \backslash V(x)$. V_1, V_2 and V_3 in fig. 11 are called *impact points*. If v is a reflex vertex and the point x is visible from v and inside the feasible part of the mission space, then the ray between v and ∂F can be expressed as

$$I(v,x) = \{q \in V(v) : q = \lambda v + (1-\lambda)x,\ \lambda > 1\} \tag{14}$$

The sensors are modeled as

$$\hat{p}_i(x, s_i) = \begin{cases} p_i(x, s_i) \text{ if } x \in V(s_i) \\ \tilde{p}_i(x, s_i) \text{ if } x \in \overline{V}(s_i) \end{cases} \tag{15}$$

Further, it is specified that $\tilde{p}_i(x, s_i) \leq p_i(x, s_i)$ since the article also takes into account that the sensors can detect events through obstacles. In this report it is assumed that all obstacles are non-transparent, hence $\tilde{p}_i(x, s_i) = 0$.

The coverage optimization problem is constructed by assuming that the probability of each sensor detecting an event is independent of the probability that any of the other sensors detect the event. $P(x, \mathbf{s})$ is used to denote this and is given by

$$P(x, \mathbf{s}) = 1 - \prod_{i=1}^{N} [1 - \hat{p}_i(x, s_i)] \tag{16}$$

Using $P(x, \mathbf{s})$ together with the event density function $R(x)$ the optimization problem is given as

$$\begin{aligned} &\max_{\mathbf{s}} \int_{\Omega} R(x) P(x, \mathbf{s}) dx \\ &\text{s.t.} \quad s_i \in F,\ i = 1, \ldots, N \end{aligned} \tag{17}$$

For simplifying notation, the objective function is defined as $H(\mathbf{s})$, and the fact that the events that occur outside the feasible area F are not of interest, simplifies the integration area. $H(\mathbf{s})$ can therefore be written as

$$H(\mathbf{s}) = \int_{F} R(x) P(x, \mathbf{s}) dx \tag{18}$$

The derivative of eq. (18) is derived in [2], and is shown to be

$$\frac{\partial H(\mathbf{s})}{\partial s_i} = \int_{\Omega_i} R(x) \prod_{k \in \mathcal{B}_i} \left[1 - p_k\left(x, s_k\right)\right] \frac{dp_i\left(x, s_i\right)}{dd_i(x)} \frac{s_i - x}{d_i(x)} dx \tag{19}$$

In eq. (19) $d_i(x)$ is the Euclidean distance between x and s_i, and Ω_i is the circle centered at s_i, with a radius of δ, which is the sensing radius of the node. $\mathcal{B}_i$ is a set containing the neighbors of node i, given by $\mathcal{B}_i = \{k : \|s_i - s_k\|_2 < 2\delta,\, k = 1, \ldots, N,\, k \neq i\}$.

The gradient ascent method described in section 2.4 is used to update the state of node i at the next time step, using η_k as step length

$$s_i^{k+1} = s_i^k + \eta_k \frac{\partial H(s)}{\partial s_i^k} \tag{20}$$

When it comes to network connectivity preservation, Zhong and Cassandras define $c_1(s_i, s_j)$ and $c_2(s_i, s_j)$ which are boolean variables that specifies whether two sensor nodes are within communication range and line-of-sight. With C denoting the communication radius, these variables are given as

$$c_1(s_i, s_j) = \begin{cases} 1, \ \|s_i - s_j\|_2 \leq C \\ 0, \text{ otherwise} \end{cases} \tag{21}$$

$$c_2(s_i, s_j) = \begin{cases} 1, \ \alpha s_i + (1 - \alpha) s_j \in F \ \forall \alpha \in [0, 1] \\ 0, \text{ otherwise} \end{cases} \tag{22}$$

The link between two nodes is defined as *strong* if both $c_1(s_i, s_j)$ and $c_2(s_i, s_j)$ equal 1. Indicating whether or not a link is strong can therefore be defined as $c(s_i, s_j) = c_1(s_i, s_j) \cdot c_2(s_i, s_j)$.

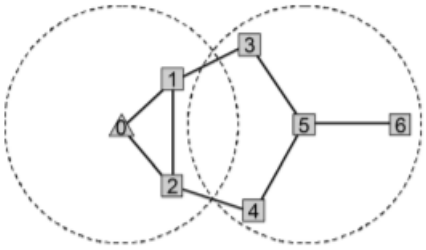


Figure 12: Graphical representation of *strong* links defined in [18]. The communication range for node 0 (base station) and node 5 are shown as dotted circles.

As the goal of the sensor network is stated, and how the connectivity of the sensors is defined, Zhong and Cassandras proceed to present how the network itself is represented. The graph $\mathcal{G}(\mathbf{s}) = (\mathcal{N}, \mathcal{E}(\mathbf{s}))$ is used to

model the sensor network, where $\mathcal{N} = 0, 1, \ldots, N$ is a set containing all the node indices. The base station is assigned the index 0. The edges of the graph are all edges between strongly connected nodes, defined as $\mathcal{E}(\mathbf{s}) = \{(i,j) : i,j \in \mathcal{N}, i \neq j, c(s_i, s_j) = 1\}$. Instead of using the Laplacian matrix $\mathcal{L}$ of the graph to preserve connectivity, Zhong and Cassandras aim to keep the set of all loop-free paths from node i to the base station non-empty. This is stated as $\overline{\Pi}_i \neq \emptyset$, $\forall i \in \mathcal{N}$.

The article further assumes that there is a routing algorithm that runs in parallel with the optimization procedure. The goal of the routing algorithm is to ensure that there exists a set of paths that enables node i to send data to the base station. This set is denoted as $\hat{\Pi}_i$, and the set of paths in $\hat{\Pi}_i$ that consists of only strong links is defined as $\Pi_i = \overline{\Pi}_i \cap \hat{\Pi}_i$. Path k in Π_i is denoted $\pi_{i,k}$ and contains an ordered set of the node indices in this path. The indices are ordered by their hop counts from node i to the base station. The jth element in path k is denoted $\pi^j_{i,k}$. It is also assumed that the routing algorithm keeps track of the indices of the nodes that are one step further away from the base station than what i is, as well as the indices of the nodes that are one step closer to the base. These two sets are called the *upstream* $\mathcal{U}_i$ and *downstream* $\mathcal{D}_i$ of node i, and given as $\mathcal{U}_i = \cup_{j,k}\omega_i(\pi_{j,k})$, where

$$\omega_i(\pi_{j,k}) = \begin{cases} \pi^{l-1}_{j,k}, & \text{if} \quad i \in \pi_{j,k}, i \neq j \quad \text{and} \quad i = \pi^l_{j,k} \\ \emptyset, & \text{otherwise} \end{cases} \tag{23}$$

and $\mathcal{D}_i = \{j : i \in U_i, j \in \{0, ..., N\}\}$, respectively.

The last things to define before presenting the algorithm that is used are the projection of $x \in \mathbb{R}^2$ on a set $\mathcal{S} \subset \mathbb{R}^2$ as $P_{\mathcal{S}}(x) = \arg\min_{y \in \mathcal{S}} \|x - y\|_2$ and $\mathcal{X}(s) = \left\{x : x \in \mathbb{R}^2, c(x,s) = 1\right\}$ is the region in which all the points can form a strong link with s. The algorithm Zhong and Cassandras proposes is displayed in fig. 13.

Theorem 1 in in the article states "Assuming only one node performs a state update at any given time and $\Pi_j \neq \emptyset$ for all $j \in 1, \ldots, N$ before the state update, an iteration of Algorithm 1 preserves the connectivity of $\mathcal{G}(\mathbf{s})$.".

Figure 14 shows the coverage that the authors of [18] are able to achieve when ensuring that each node has at least one other node inside its communication range and in its field of view.

Algorithm 1:

When node i makes a location update at t_k, using (9), it takes the following steps:

1) Using (9), generate a candidate of the next location: $\hat{s}_i$.
2) If for all $j \in \mathcal{D}_i$, $c(\hat{s}_i, s_j) = 0$, go to Step 3; else, go to Step 5.
3) If there exists a node $j \neq i$ such that $s_j \in \mathcal{X}(\hat{s}_i)$ and there exists a path $\pi_{j,l} \in \Pi_j$, such that $i \notin \pi_{j,l}$, go to Step 5; else, go to Step 4.
4) Select some $j \in \mathcal{D}_i$ and project $\hat{s}_i$ on $\mathcal{X}(s_j)$. Redefine the result $P_{\mathcal{X}(s_j)}(\hat{s}_i)$ as $\hat{s}_i$.
5) If for all $j \in \mathcal{U}_i$, $c(\hat{s}_i, s_j) = 1$, go to Step 7; else, go to Step 6.
6) If for all $j \in \mathcal{U}_i$ such that $c(\hat{s}_i, s_j) = 0$ there exists a path $\pi_{j,l} \in \Pi_j$, such that either $i \notin \pi_{j,l}$ or $i \in \pi_{j,l}$ and $c(\hat{s}_i, s_r) = 1$ with $r = \omega_i(\pi_{j,l})$, go to Step 7; else $s_i(k+1) = s_i(k)$ and skip Step 7.
7) $s_i(k+1) = \hat{s}_i$.

Figure 13: Algorithm 1 defined in [18].

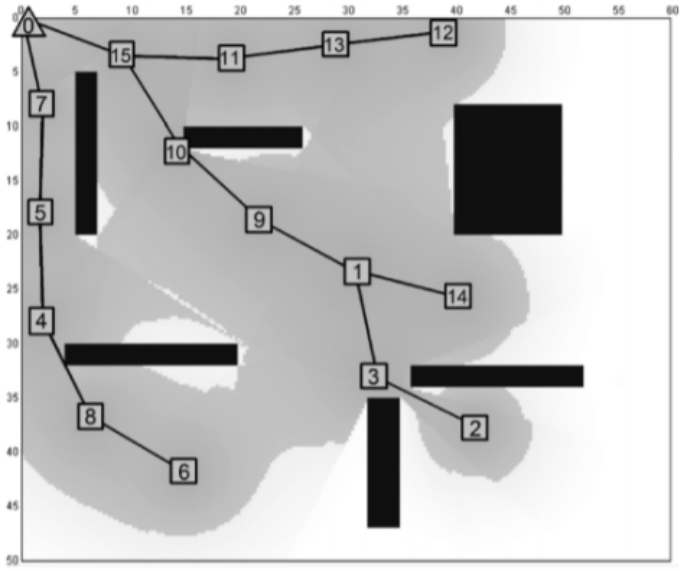


Figure 14: Covered area achieved in [18].

3.2 Implementation

A similar approach that Zhong and Cassandras proposed in [18] is implemented using Python, and resulted from further developing the code provided by Magnus Berdal[3]. It should be noted, as stated in [12], that the algorithm in fig. 13 only ensures that the network reaches a local optimum. This will also be the case for the implementation described below. The source code is found at https://github.com/erllon/Project-thesis.git.

The Python package Shapely is used for constructing the mission space and the obstacles, as well as analyzing different properties with these geometric objects. Scipy.spatial is used for spatial operations such as Delaunay triangulation and computing distances between points. For numerical integration, the package Quadpy is used. Matplotlib.pyplot is utilized for visualization of the network's final configuration and how the covered area develops during the deployment.

Each agent has a uniform circular sensing area with radius 4, and a communication radius of 8. The relationship between the sensing and communication radii is designed this way to improve the visualization but is unfortunately not achievable for the drones described in section 1. The laser sensors that are mounted on the Crazyfies used by SINTEF can measure distances up 4 meters and the communication range of the Crazyflies is about 2 meters, depending on the environment.

As defined in eq. (15), the sensing capability of sensor i is modeled as $p_{0i}e^{-\lambda_i||x-s_i||}$ for all visible points from s_i. It is assumed that the sensors on all agents are equal, meaning $p_{0i} = 1$ and $\lambda_i = 10^{-5}$ for all i. Similar to [18], the base station is placed in a corner and all the agents will initially be located there as well. The event density is uniform, hence $R(x) = 1$.

Compared to the work done in [18], there is no routing algorithm running in parallel with the optimization procedure. When each agent is updating its position, it performs the following steps, which is run until the convergence criterion is fulfilled for all agents.

1) Use eq. (20) to generate a new position candidate

2) Check if the position candidate is feasible and get the objective function value at that point.

3) If the position candidate is a feasible point, the objective function value at the position candidate is greater than at the current position and the position candidate is inside the communication range of at least one other agent, go to step 4), else, decrease the step size and go to step 1).

4) Move to the position candidate and check if some convergence criterion is fulfilled.

[3] https://github.com/mBerdal/master_project_coverage.git

4 Results

The results that the abovementioned implementation produced are presented below. Two step size reduction factors are used for deciding a new position candidate for the agents when the conditions in step 3) fail. Changing the optimization tolerance did not make much difference in performance.

The base station is located in the lower left corner and is represented by a red triangle. As there is not used any routing algorithm, the red lines indicate what [18] defines as *strong* connections. For some network configurations this will be equal to the communication paths, but not always, as it is not guaranteed that this will produce loop-free paths.

4.1 Reduction factor of 0.5

Figures 15 and 16 shows the initial and final positions of the drones in the network when the step size was reduced by one-half each iteration. The network of three drones ended up covering an area of 150.5 out of a theoretically possible area of 150.8. The five drones in fig. 16 covers a total area of 216.5 but could theoretically cover an area of 251.3. As there is some overlapping in the sensor regions for this network, and that two of the drones are positioned too close to an obstacle to fully utilize their sensing capabilities, the ratio between covered area and theoretically possible covered area is considerably lower for the five-drone-network.

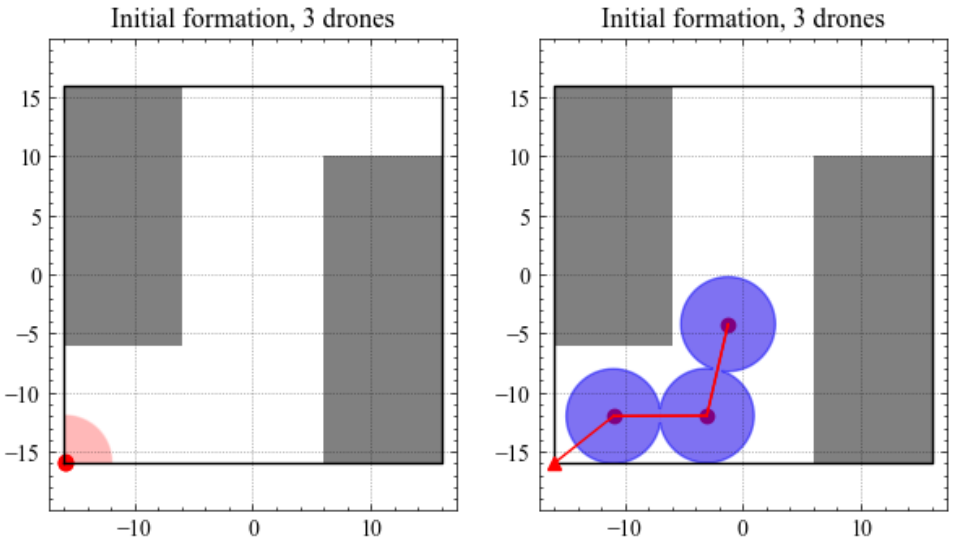


Figure 15: Initial and final positions using a step size reduction factor of 0.5.

How the amount of covered area develops over time is shown in fig. 17. The network of three drones converges to its final configuration in 42 iterations, and when deploying five drones, the network converges in 85 iterations.

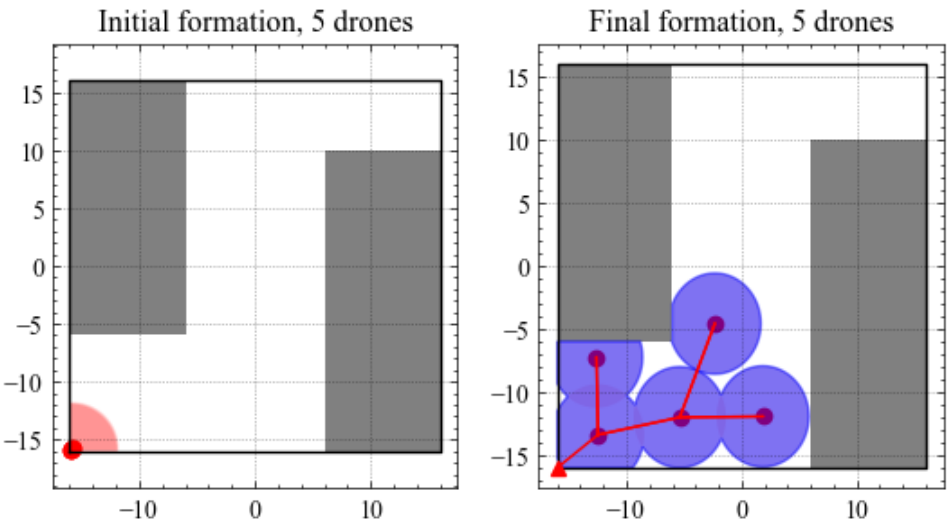


Figure 16: Initial and final positions using a step size reduction factor of 0.5.

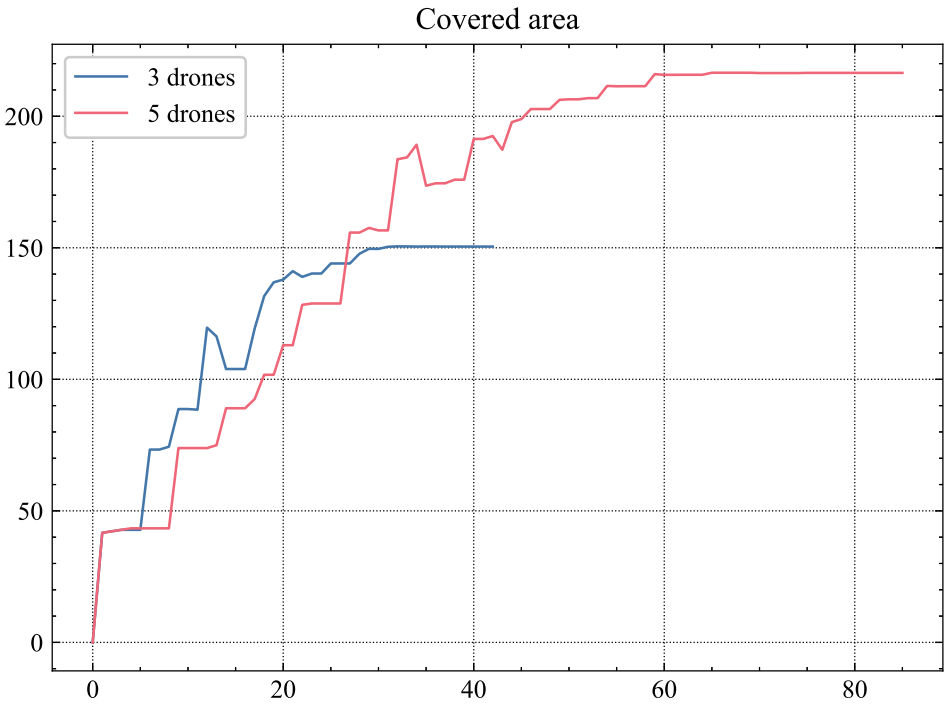


Figure 17: Comparison of covered area using 3 and 5 drones, reduction factor of 0.5.

4.2 Reduction factor of 0.8

Changing how the step size is reduced changes the behavior of the network. Using the same initial step size as for the networks shown in figs. 15 and 16 but multiplying the step size by 0.8 instead of 0.5 after each iteration, yielded the results shown in figs. 18 and 19.

It is clear from figs. 18, 19 and 20 that the selection of step size has a big impact of the final area the network covers. The network of three drones in fig. 18 covers an area of 124.4. This is almost 20% less than what was achieved in fig. 15.

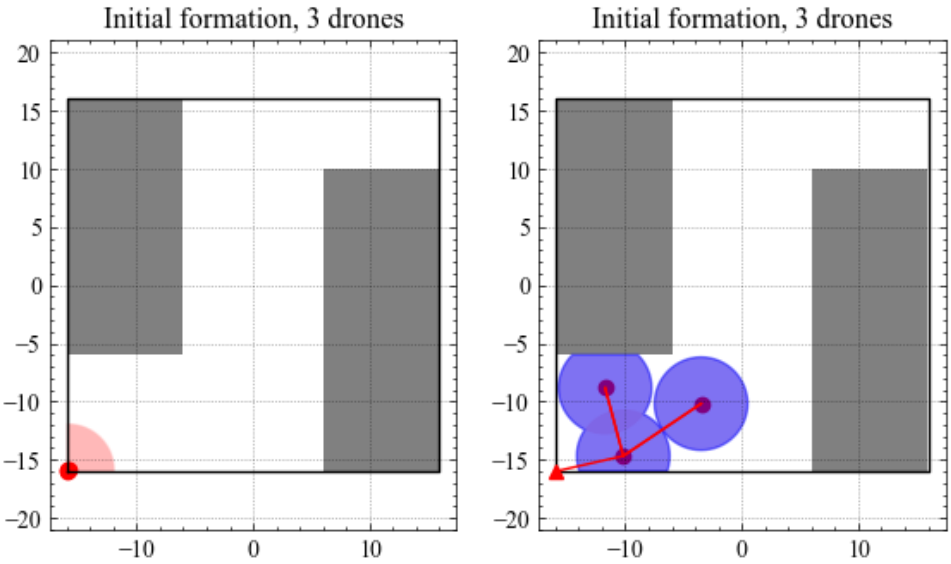


Figure 18: Initial and final positions using a step size reduction factor of 0.8.

When deploying five drones, an area of 194.7 ended up being covered, which also is considerably less than what was covered in fig. 16. Figure 19 also displays how the strong connections might form paths containing loops and can therefore not be used directly as communication paths to the base station.

Comparing the performance when 0.8 is used as step size reduction factor with what was used in section 4.2, it is worth noting that the network consisting of three drones takes longer to reach its final configuration. For the five-drone-network on the other hand, the amount of time steps is the same.

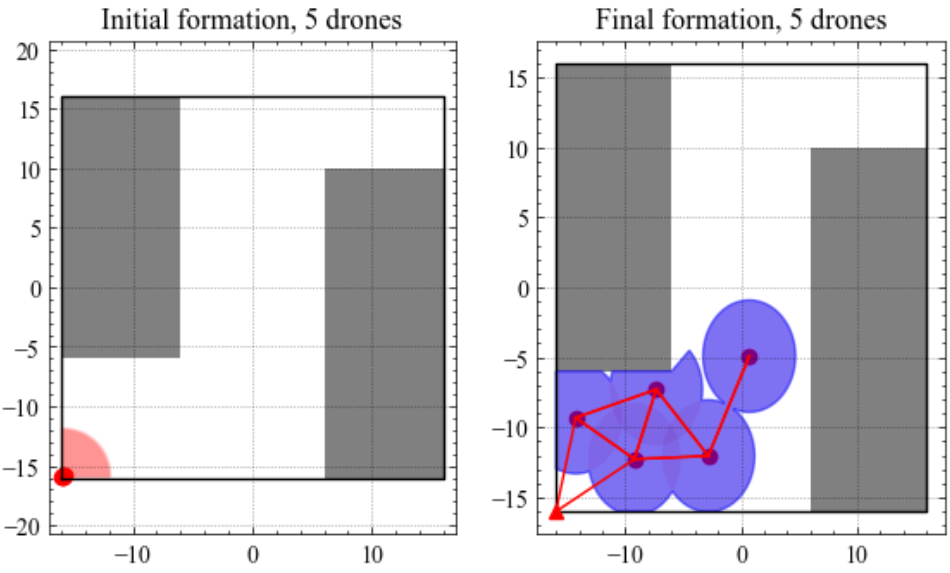


Figure 19: Initial and final positions using a step size reduction factor of 0.8.

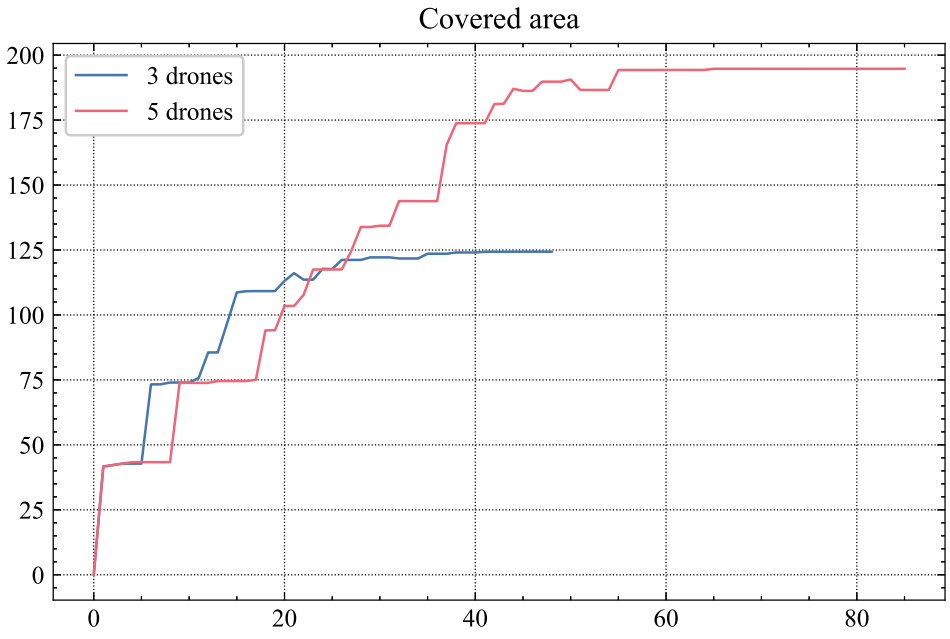


Figure 20: Comparison of covered area using 3 and 5 drones, reduction factor of 0.8.

5 Conclusion

This report has provided a thorough description of the area coverage problem using a multi-agent network and presented how four research papers have solved it. A similar approach to what was presented in one of these articles was then implemented as a simulation using Python. The article that inspired the simulated algorithm formulated the coverage problem as a maximization problem and solved it by using the gradient ascent method, as well as utilized a routing algorithm to ensure connectivity. How the implemented algorithm performed using different parameter values was then analyzed, and one could definitely see that the choice of parameters impacted how well the network covered the given area.

5.1 Future work

There are multiple paths that can be further explored from this work. It would be very interesting to try to implement a routing algorithm in the same way as done in [18] to see how the connectivity can be ensured in a better way, leading to a more efficient coverage. Investigating how one could develop an algorithm to get the three other approaches to guarantee connectivity at all times would also be an interesting path to follow in the future.

It would of course be of interest to study how one could apply a coverage algorithm in real life, using the Crazyflie 2.1. If one is able to achieve this, a natural next step is to get one or mulitple robots to map an area and letting the network of drones use the provided data in a known-map algorithm.

5.2 Personal reflection

Working with this project has been exciting from start to end. At times, it has been demanding to get my head around all the different topics that one needs to have a decent understanding of when working with the multi-agent area coverage problem. With that being said, it has been of great interest to be able to get a detailed look at how researchers present their findings and how they present the associated theory.

It has also been very interesting to see how the discussed problem can be solved in many different ways. All the articles that I have read as part of this work have of course not been as relevant for the problem defined in section 1.3 as the four presented in section 3. Even if there are many articles that I have read and studied that have not provided some visible results, they have given me a broader understanding of how the general problem of area coverage can be solved. This understanding is something that I will take with me in future work and has definitely given me ideas of what I want to further study.

References

[1] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[2] C. G. Cassandras and Wei Li. "Sensor Networks and Cooperative Control." In: *Proceedings of the 44th IEEE Conference on Decision and Control*. 2005, pp. 4237–4238.

[3] C.G. Cassandras. "Distributed Coverage Control in Sensor Network Environments with Polygonal Obstacles." In: vol. 41. July 2008, pp. 4162–4167.

[4] Z. Drezner and H. Hamacher. "Facility location - applications and theory." In: 2001.

[5] Qiang Du, Vance Faber, and Max Gunzburger. "Centroidal Voronoi Tessellations: Applications and Algorithms." In: *SIAM Rev.* 41.4 (Dec. 1999), 637–676. URL: https://doi.org/10.1137/S0036144599352836.

[6] M. Elbanhawi et al. "Enabling technologies for autonomous MAV operations." In: *Progress in Aerospace Sciences* 91 (2017), pp. 27 –52. URL: http://www.sciencedirect.com/science/article/pii/S0376042116300367.

[7] Andrew Howard, Maja J. Matarić, and Gaurav S. Sukhatme. "Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem." In: *Distributed Autonomous Robotic Systems 5*. Ed. by Hajime Asama et al. Tokyo: Springer Japan, 2002, pp. 299–308.

[8] O. Khatib. "Real-time obstacle avoidance for manipulators and mobile robots." In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. Vol. 2. 1985, pp. 500–505.

[9] Guilherme Pereira et al. "Decentralized motion planning for multiple robots subject to sensing and communication constraints." In: *Departmental Papers (MEAM)* (Apr. 2003).

[10] M. Schwager, J. Slotine, and D. Rus. "Decentralized, Adaptive Control for Coverage with Networked Robots." In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. 2007, pp. 3289–3294.

[11] Yiannis Stergiopoulos and Anthony Tzes. "Voronoi-based Coverage Optimization for Mobile Networks with Limited Sensing Range - A Directional Search Approach." In: July 2009, pp. 2642 –2647.

[12] Xinmiao Sun, C.G. Cassandras, and Kagan Gokbayrak. "Escaping Local Optima in a Class of Multi-Agent Distributed Optimization Problems: A Boosting Function Approach." In: *Proceedings of the IEEE Conference on Decision and Control* 2015 (Sept. 2014).

[13] H. Tnunay et al. “Distributed collision-free coverage control of mobile robots with consensus-based approach.” In: *2017 13th IEEE International Conference on Control Automation (ICCA)*. 2017, pp. 678–683.

[14] Wei Li and C. G. Cassandras. “Distributed Cooperative coverage Control of Sensor Networks.” In: *Proceedings of the 44th IEEE Conference on Decision and Control*. 2005, pp. 2542–2547.

[15] You-Chiun Wang, Chun-Chi Hu, and Yu-Chee Tseng. “Efficient deployment algorithms for ensuring coverage and connectivity of wireless sensor networks.” In: *First International Conference on Wireless Internet (WICON'05)*. 2005, pp. 114–121.

[16] M. Zhong and C. G. Cassandras. “Asynchronous distributed optimization with minimal communication.” In: *2008 47th IEEE Conference on Decision and Control*. 2008, pp. 363–368.

[17] M. Zhong and C. G. Cassandras. “Asynchronous distributed optimization with minimal communication and connectivity preservation.” In: *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. 2009, pp. 5396–5401.

[18] M. Zhong and C. G. Cassandras. “Distributed Coverage Control and Data Collection With Mobile Sensor Networks.” In: *IEEE Transactions on Automatic Control* 56.10 (2011), pp. 2445–2455.

[19] Y. Zou and Krishnendu Chakrabarty. “Sensor deployment and target localization based on virtual forces.” In: *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*. Vol. 2. 2003, 1293–1303 vol.2.