Coverage control of Agricultural Fields using Heterogeneous Multi-robot Systems

by

Saba Faryadi

(Under the Direction of Javad Mohammadpour Velni)

Abstract

In the past decade, increase in world population and global demand for food has motivated scientists to develop new sensing technologies and unmanned vehicles in farms for automating agricultural tasks and improving farm management. To achieve this goal, precision agriculture has witnessed advancements with the purpose of protecting resources and increasing productivity while decreasing the cost. Furthermore, a significant number of studies have focused on area coverage methods aiming to optimally distribute multi-robot systems for different tasks including area exploration and data collection. In this study, distributed algorithms with obstacle avoidance capability are developed to deploy a group of autonomous mobile robots for farming applications. Vehicles are equipped with essential devices to navigate in the field, monitor plant rows cooperatively, and provide information on important areas.

Index words: Heterogeneous Multi-robots Systems, Coverage Control, Precision Agriculture, Reinforcement Learning

Coverage Control of Agricultural Fields using Heterogeneous Multi-robot Systems

by

Saba Faryadi

B.Sc., Iran University of Science and Technology, Tehran, Iran, 2013

A Thesis Submitted to the Graduate Faculty
of The University of Georgia in Partial Fulfillment
of the Requirements for the Degree

Master of science

Athens, Georgia

2020

Coverage Control of Agricultural Fields using Heterogeneous Multi-robot Systems

by

Saba Faryadi

Approved:

Major Professor: Javad Mohammadpour Velni

Committee: Changying Charlie Li,
Mark Trudgen

Electronic Version Approved:

Ron Walcott
Interim Dean of The Graduate School
The University of Georgia
Aug 2020

Acknowledgments

I would first like to thank my thesis advisor Dr. Javad Mohammadpour Velni, of the college of engineering at the University of Georgia. He shows support in everything I do and teaches me the right direction for my thesis writing. Also, I would like to thank the rest of my thesis committee Dr. Changying Charlie Li, and Dr. Mark Trudgen for their encouragement and insightful comments.

I would also like to thank my lab-mate, Dr. Mohammadreza Davoodi, who was involved in the validation survey for this research project.

My sincere thanks go to my beloved husband, Dr. Sorush Omidvar, for his consistent support and guidance during the running of my project.

I must express my very profound gratitude to my parents and my sister to provide me with unfailing support and continuous encouragement throughout my years of study. None of this accomplishment would have been possible without the infinite love and support from my family.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION AND MOTIVATION

In the last several years, Precision Agriculture (PA) and smart farming have received considerable attention to improving farmers' safety, monitoring field, and saving time and energy. PA combines sensing devices, agricultural machines, information systems with environmental-friendly technologies to provide sustainable food supplies and reduce farm costs [Gebbers and Adamchuk, 2010]. One of the primary purposes of PA is to increase the production efficiency while decreasing the farming inputs by monitoring the whole system [Zhang et al., 2002]. In other words, PA focuses on recognizing environmental characteristics and monitoring plants' behavior to control the required amount of seeds, pesticides, and fertilizers [Srinivasan, 2006, Zacepins et al., 2012]. Moreover, agriculture plays a critical role in climate change as it affects the carbon dioxide content and humidity ra-

tio; therefore, PA can mitigate the global warming issue through greenhouse gas reduction [Balafoutis et al., 2017].

In the late 1970s, the advent of the Global Positioning System (GPS) was a breakthrough in the development of PA [Stafford, 2000]. As discussed in [Zhang et al., 2002], with the use of GPS, remote sensing, crop monitoring, and soil sampling, a map-based approach is more applicable to recognize the field's dynamic compared to the sensor-based method. Although aerial photos provided by satellites or other remote sensing techniques are useful to measure some essential variables, a significant number of characteristics can only be monitored throughout the direct measurement system [Camilli et al., 2007]. In [Camilli et al., 2007], two main categories of field data acquisition are explained: 1) stationary devices mounted at fixed points, which are more common for variables that need to be measured continuously like temperature and humidity, and 2) portable sensors, which are mostly embedded in vehicles.

In order to employ sensors in different applications, the Internet of Things (IoT) is developed to provide communication between devices and transfer data. One of the main components of the IoT is the Wireless Sensor Networks (WSN) as an optimum way to collect real-time data from an environment using a network of sensors distributed in the area communicating with each other. [Srbinovska et al., 2015] suggested a WSN system containing sensor nodes and base stations to collect data, control environmental parameters, and make appropriate decisions. Furthermore, authors in [Karim et al., 2017] presented an application of IoT for controlling the water stress of plants using a WSN.

Despite the fact that environmental parameters (temperature, light intensity, etc.) can be recorded repeatedly using a network of stationary sensors, these devices are not very useful for some agricultural tasks like visual data collection, soil sampling, and pest detection. To address this problem, agricultural robotic systems consisting of essential sensors and devices are developed to use as a substitute for human workers. Aurora is one of the first platforms designed to move inside greenhouses and facilitate farming tasks, especially the hard and dangerous operations which are usually performed by human [Mandow et al., 1996]. Aurora is a mobile robot carrying appropriate devices and capable of navigating autonomously or through human control [Mandow et al., 1996]. As wheeled machines have continuous contact with soil and limited mobility, they may damage the environment and plant rows. This problem motivated authors in [Ion et al., 2002] to build Mero, which is a four-legged walking robot that interacts with the environment and performs farm tasks.

As mentioned above, during the past 20 years, several robotic platforms have been designed and tested to be employed in greenhouses or farms for operating various agricultural tasks. All of these platforms require an appropriate control system to avoid obstacles and navigate autonomously. A great number of studies are focused on the development and implementation of control algorithms for numerous tasks, including navigation, path-planing, area coverage, etc. For example, [Hagras et al., 2002] introduced a Fuzzy-Genetic system as an online learning strategy that allows a ground robot (outfitted with sonar sensors) to learn an unknown dynamic agricultural environment and then it helps to calibrate its con-

troller in real time.

Some control methods are vision-based in such a way that vehicles navigate and find free path or obstacles using visual data. In [Ayala et al., 2008], a visual control method is presented, which allows a mobile robot to compute its current position based on the information resulting from the process and segmentation of photos captured by the robot while moving between plant rows. One of the disadvantages of visual sensors is their performance in extreme weather conditions with limited visibility like fog or rain. Therefore, ordinary cameras have been replaced by 3D sensors, allowing more accuracy in vehicles' controlling and navigation. 3D LIDAR sensor can be implanted in a mobile robot to detect plants, self localize and navigate in the field [Weiss and Biber, 2011].

Unmanned ground vehicle (UGV) movements are limited by the presence of obstacles or any unpredicted objects; on the other hand, the aerial vehicles are agile and can fly freely. Therefore, they are heavily exploited in PA in recent years. As an example, [Mogili and Deepak, 2018] described a new technique using an unmanned aerial vehicle (UAV) equipped with a multispectral camera to monitor crops and find the areas which require spraying with pesticide. Moreover, authors in [Tripicchio et al., 2015] presented a vision-based method enforcing a drone to fly over the land and collect data using the Kinet v2.0 sensor for further investigations and analyzing soil characteristics.

Nowadays, one of the most interesting challenges in PA is the development of multi-robot systems, including both UAVs and UGVs. Based on the applications and limitations of UGVs and UAVs, the cooperation between a group of

autonomous vehicles is studied in different scenarios. [Chen et al., 2015] reviewed and categorized the interaction of UAVs and UGVs into centralized and decentralized approaches for decision making. A centralized and distributed controller for a team of unmanned vehicles is presented in [Li and Cassandras, 2006] as a Receding Horizon (RH) controller, which solves a sequence of optimization problems to find vehicles' routs. In some cases, robots have to decide independently (decentralized) for a specific task like target tracking. Authors in [Nascimento et al., 2013] explained a decentralized nonlinear model predictive approach for the formation of mobile robots and controlling them towards the desired goal.

As the application of multi-robot systems is growing rapidly, new studies are mostly focusing on coverage control and collaboration of a team of heterogeneous unmanned vehicles. [Michael et al., 2007] explained controlling multiple UGVs with a UAV in such a way that the drone commands the linear and angular velocities to the ground vehicles and control them as a team. Furthermore, to improve the robotic system's performance, the application of a multi UAV-UGV system is proposed in [Roldán et al., 2016] to record environmental parameters in greenhouses. Compared to the multi UAV-UGV systems, in some cases, vehicles are heterogeneous in terms of their dynamics, speed, sensing capabilities, and battery capacities. For instance, authors in [Conesa-Muñoz et al., 2015] developed a path-planning algorithm to deploy a team of heterogeneous ground robots aiming to minimize total traveling time, distance, and cost.

Automated agricultural systems empower farmers to control their products and contribute to saving time/energy, economic growth, and food quality im-

provement. Furthermore, the motivation for this research is to apply and test heterogeneous multi-robot systems to navigate in a graph-based field and collect data for further agricultural analysis. This thesis consists of six chapters where the first and last one provide the introduction and concluding remarks, and the other four are as follows. Chapter 2 presents and validates a distributed algorithm to deploy multiple UGVs in the field. Chapter 3 examines the application of the discrete coverage control method in monitoring multiple regions of interest in a greenhouse using a single ground robot. In chapter 4, we address the optimization problem of finding the minimum traversal time for a team of heterogeneous UAVs flying over the field and monitoring plant rows. Chapter 5 describes the application of a Q-learning based method that enforces multiple UGVs to explore an unknown changing farm, learn its model in real time, and then optimally distribute the UGVs in the field to cover essential areas.

Chapter 2

AGRICULTURAL FIELD COVERAGE USING COOPERATING UNMANNED GROUND VEHICLES[1]

[1]Saba Faryadi, Mohammadreza Davoodi, and Javad Mohammadpour Velni. Accepted by *American Society of Mechanical Engineers (ASME) 2019, Dynamic Systems and Control Conference*. Reprinted here with permission of publisher.

Abstract

In this paper, a distributed algorithm with obstacle avoidance capability is presented to deploy a group of ground robots for field-based agriculture applications. To this end, the field (consisting of many plots) is first modeled as a directed graph, and the robots are deployed to collect data from some important areas of the field (e.g., areas with high water stress or biotic stress). The key idea is to formulate the underlying problem as a locational optimization problem and then find the optimal solution based on the Voronoi partitioning of the associated graph. The proposed partitioning method is validated through simulation studies, as well as experiments using a group of mobile robots.

2.1 Introduction

With the world's growing population, farming and food production are becoming more important than ever. An instrumental component of "smart farming" and "precision agriculture" is the process of automating tasks through the use of ground robots, drones, and other advanced sensing technologies – this has indeed led to a better decision making and more efficient farm management. On the other hand, in order to improve safety and save time, remote sensing has become an alternative to human observance and has played a key role in site specific management. Moreover, remote sensing allows a faster estimation/detection of the most interesting areas of the field and provides a better visual understanding [Casady and Palm, 2002]. In the automation process, intelligent devices such as robotic

networks have found their way into smart farming aiming at assisting humans for an improved quality and quantity of the crops [Lalwani et al., 2016]. Having more productive and sustainable agricultural production, based on more precise and resource-efficient approaches, is the main purpose of deploying Unmanned Ground Vehicles (UGVs) and Unmanned Aerial Vehicles (UAVs) in field-based agriculture. The use of a single robot, especially in smart farming, brings several limitations; it is not only time consuming but also requires a robot with a large battery storage. Therefore, collaboration of a team of agents (robots) working together is a new research direction in field-based agriculture. Tokekar *et al.* in [Tokekar et al., 2016] presented a method for coordinating ground robots and drones to take plant images and collect soil samples with the purpose of covering the maximum points, finding the best trajectory and minimizing the time of traveling over the field. This approach needs agents to communicate with each other to arrange and share duties based on their locations and capabilities [Barrientos et al., 2011]. Yu *et al.* in [Yu and LaValle, 2016] studied optimizing multi-agents path planning (MPP) over a graph. The latter work made use of integer linear programming (ILP) and heuristic approaches to increase the efficiency of the algorithmic solutions. In [Gonzalez-de Santos et al., 2017], authors focused on remote monitoring by deploying a group of UGVs and UAVs in a farm for effective control over weed and pest and to diminish the effects of chemicals. Bhandari *et al.* in [Bhandari et al., 2017] investigated collaboration between robots equipped with multispectral/hyperspectral cameras and RGB cameras to determine water stress and nitrogen deficiency; one of the robots was also equipped with an arm

to remove the unhealthy plants. Since the collection of sensor data around the regions of interest needs higher density, authors in [Nolan et al., 2017] designed a coverage control algorithm using Voronoi diagrams and presented a path planning algorithm to achieve precise sensor measurements around the important areas in the farm. The authors of [Liu and Michael, 2014] designed an energy-aware control strategy for robots with limited amount of energy reserves to navigate them to docking stations for recharging.

Coverage control with guaranteed collision avoidance has been examined in several works. Teraoka *et al.* in [Teraoka et al., 2011] proposed a distributed control algorithm for sensors deployment in such a way that both of the following objectives are simultaneously satisfied: obstacle avoidance and communication cost reduction. In our recent work (see [Davoodi et al., 2018]), an energy-aware coverage control strategy for a team of UGVs was presented. In this paper, similar to [Davoodi et al., 2018], we develop and solve the coverage problem by modeling the field as a directed graph and transforming the original problem into a locational optimization problem over a graph. However, unlike [Davoodi et al., 2018], an obstacle avoidance strategy is embedded here into the proposed coverage control algorithm. Furthermore, the implementation of the partitioning method on a practical case study is also provided. Experimental results include the design and programming of iRobots that can coordinate their tasks and cooperate with the purpose of monitoring the field in real time and collecting data. Implementation of controllers in the robots involves communication, localization, navigation, obstacle avoidance and task allocation modules.The remainder of this paper is organized

as follows. In Section 2, the problem statement and formulation is described. In Section 3, the main result of this work, distributed coverage control with collision avoidance, is given. The performance of the proposed algorithm using simulations, as well as hardware experiments are demonstrated in Section 4, followed by the description of concluding remarks and future work in Section 5.

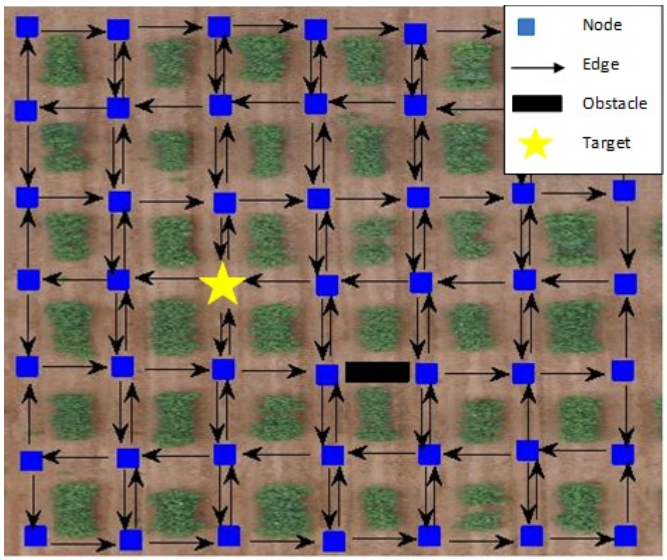


Figure 2.1: An illustration of the field graph.

2.2 Problem Statement

The main goal of this work is to present a solution for optimal coverage of an agricultural field using a fleet of ground vehicles equipped with relevant sensors (such as multi-spectral cameras) so that a number of predefined regions of interest can be precisely monitored. A region of interest is a part of the field, where there is health abnormality of soil or crops, e.g., identifying plant pathogens such as

fungal or areas where water stress is suspected.

For the deployment of robots, a graph-based topological map is used to model the field. First, we define a directed adjacent graph $G(V, A)$ as shown in Fig. 2.1 to represent the farm. It is a $|V| \times |V|$ matrix, where $V \in \{1, 2, ..., n\}$ and n is the number of nodes. Each vertex is assigned to an intersection of two corridors. Furthermore, A is a set of directed arcs that indicate whether or not there is an edge between two nodes. Each arc $a \in A$ is specified by an ordered pair of nodes $v_i, v_j \in V$. Also, $N_G(v_i)$ is considered as a set including neighbors of v_i, i.e., $N(v_i) = \{v_j | a_{ij} \in \text{A}\}$, $a_{ij} \neq 0$ where a_{ij} indicates the arrow from v_i to v_j. The modeled graph is weighted since all nodes are assumed to contain the same value except for those assigned to regions of interest – vertices indicating such regions are given a higher cost (weight). To monitor predefined important regions in the field, a team of m Unmanned Ground Vehicles (UGVs) capable of communicating with each other are deployed in the farm. As they are assumed to have limited battery capacity, we consider a charge station for each UGV. All the robots are equipped with GPS to update their current locations. Each of the robots is responsible for making the best decision to allow sensing crops and land based on locational optimization method inspired by recent work of [Alitappeh et al., 2017, Yun and Rus, 2014].

We use $P(r_i)$ as a set whose elements represent the position of each robot and $r_i \in V$. With the purpose of recharging, charge stations are located besides the field. Furthermore, $C(r_i)$ denotes the level of battery charge for i^{th} robot. Whenever it becomes critical, a robot aborts its assigned mission in order to dock

to the nearest charge station autonomously.

2.3 Methodology

The main purpose here is to find the optimal partitioning of the field for each robot to minimize corresponding cost function. In order to achieve this goal and motivated by the work of Yun *et al.* [Yun and Rus, 2014], we design and implement a distributed coverage control algorithm in such a way that the ground robots in the farm equally share tasks. In the proposed method, we employ Dijkstra's algorithm to find the shortest path between each pair of nodes. After that, field is divided into i Voronoi sub-graphs $S(r_i)$ using our recent work [Davoodi et al., 2018]. Voronoi diagram $S(r_i)$ for i^{th} robot is a partition of the field defined as

$$S(r_i) = \{u \in V | d(u, r_i) \leq d(u, r_j), i \neq j\}, \tag{2.1}$$

where $d(v_i, v_j)$ denotes the cost of the shortest path between nodes v_i and v_j. It is noted that $S(r_i)$ is a partition whose corresponding robot i is responsible for all tasks in that partition. If a robot is out of charge and has to move towards the charging station, all steps are retaken for finding the best partition among the remaining robots. This method can be encoded within a new formulation to provide the minimum total cost. Following cost function is used by [Alitappeh et al., 2017, Davoodi et al., 2018]

$$H(u, s) = \sum_{i=1}^{n} H_i(u_i, s_i), \tag{2.2}$$

where

$$H_i(u_i, s_i) = \sum_{u \in S_i} d(u_i, v)\Phi(v), \tag{2.3}$$

where $\Phi(v)$ is the so-called priority value corresponding to node v. As interpreted from this formulation, since the region of interest has higher priority value, the cost will converge to the minimum value only when the distance between the current location of robot and target $d(u_i, v)$ converges to zero. Minimizing the cost (2.2) is the main step in finding the next best position for each robot. The main element of this approach is the implementation of a distributed algorithm based on the aforedescribed assumptions. As shown in Table 2.1, a single robot that has a reliable battery charge updates its Voronoi sub-graph, finds the next best node to move to and computes the total cost function sequentially until it converges.

To avoid collisions between the robots and with any undefined object, we formulated the problem for two different conditions including the presence of an object in a corridor or in an intersection. If a corridor is blocked, the corresponding edge a will be considered zero, and if an obstacle is located in the intersection, all those edges that end to that node will be considered zero.

2.4 Validation of the Proposed Partitioning Method

In this section, we present the results of validating the proposed partitioning method in both simulation environment and a lab-scale test bed.

Table 2.1: Algorithm 1: Determining the next best point for a robot to go to.

Initialize program
Input: Q, P_i, C_i

1. **Add** information of graph G.
2. **Get** current location of the robot $P(r_i)$, its battery charge $C(r_i)$ and the center of regions of interest.
3. **Get** obstacles' location.

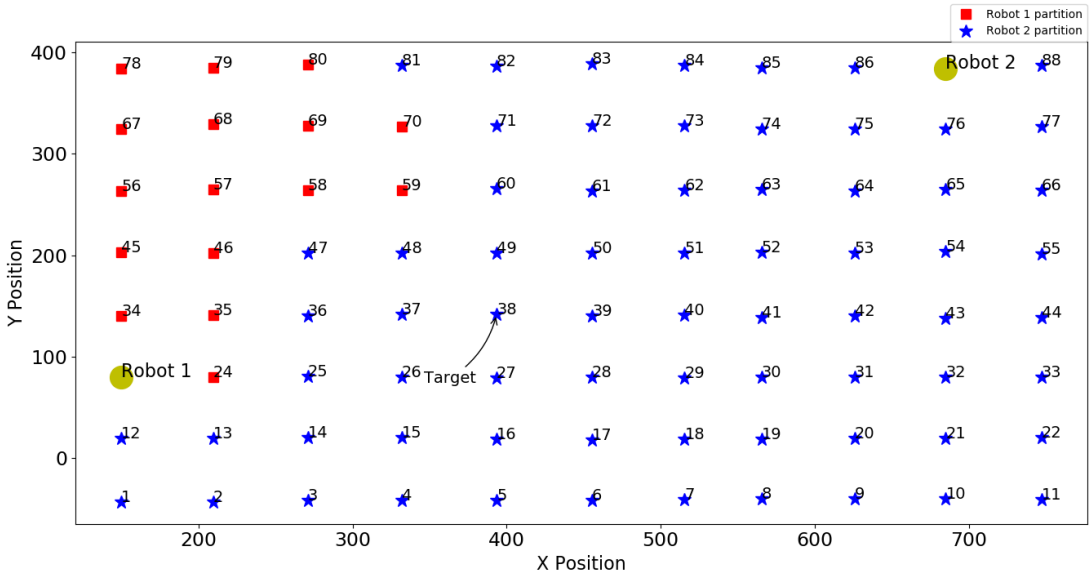
Main code

1. **Set** the edges blocked with obstacle or ones that end to an obstacle to zero.
2. **Find** shortest path between every two vertices.
3. **Compute** initial Voronoi regions for each robot.
4. **While** cost function has not converged:
   Update coverage neighboring set $N(r_i)$:
   if there is a neighboring node $u \in N(r_i)$ so that $H(u, r_i)$ is minimum in $S(r_i)$, **then:**
   Set u as the next best point.
   end if
   Update charge status of battery C_i:
   if C_i is less than the threshold, **then:**
   Move to recharging station.
   else:
   Move towards node u.
   end if

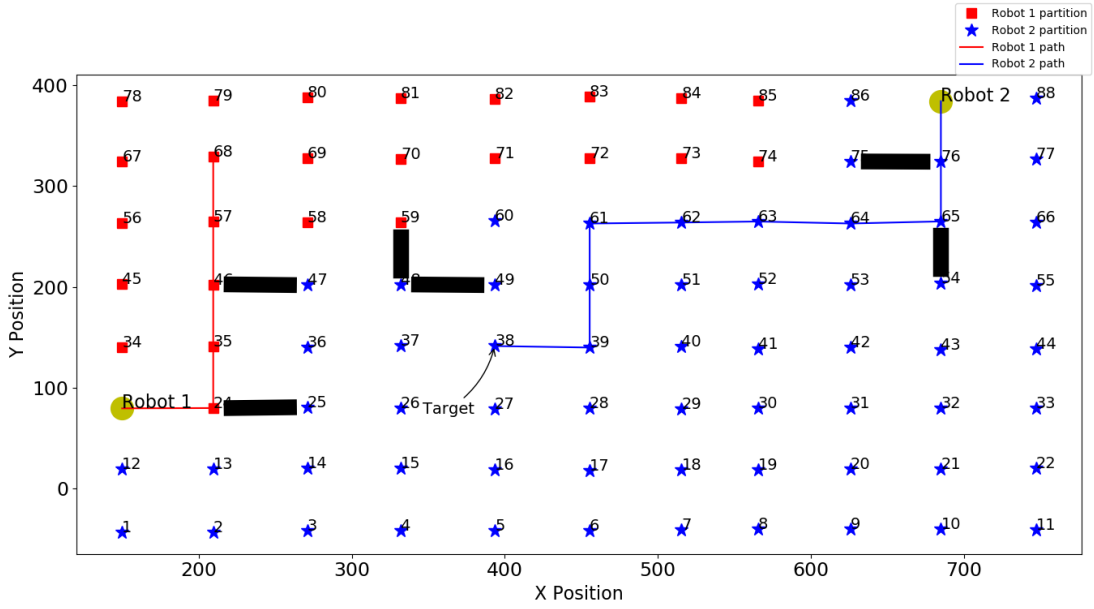
end while

2.4.1 Numerical Example

The proposed partitioning method was implemented in the programming language Python with the aim of monitoring a field that contains several obstacles. For initialization, we considered a directed graph G with 88 nodes, as depicted in Fig. 2.2a. As it is clear from Fig. 2.2b, six black stripes indicate obstacles located in corridors; as described before, for modeling purposes, the corresponding edges of these obstacles are set to zero. It is assumed that there is only one region of interest in the whole field (node 37) with the priority value Φ.



(a)



(b)

Figure 2.2: (a): The plot shows the initial location of each robot and their Voronoi partition; (b): the plot shows the final partitioning resulting in a minimum cost and one of the two robots reaching the target. Black stripes indicate obstacles.

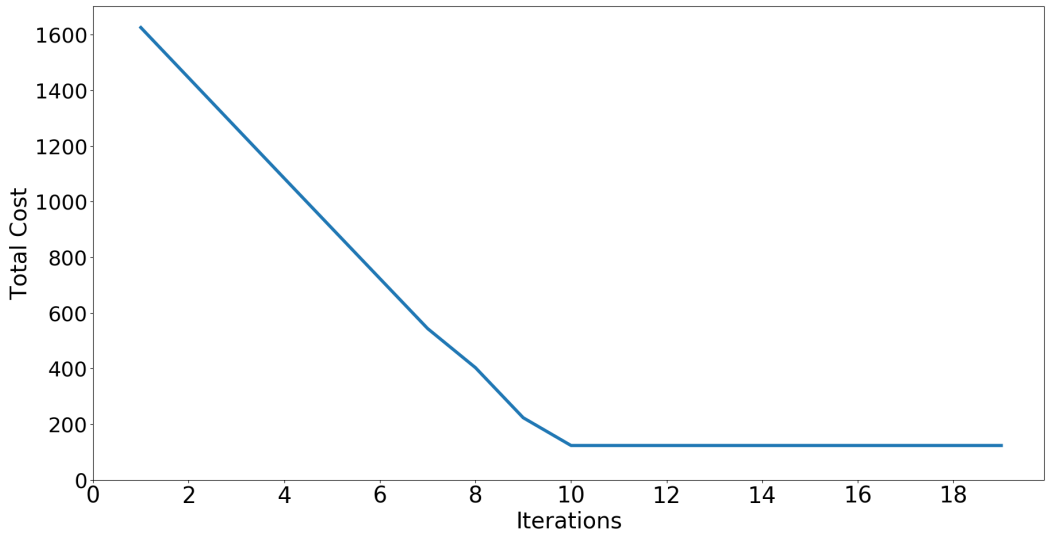
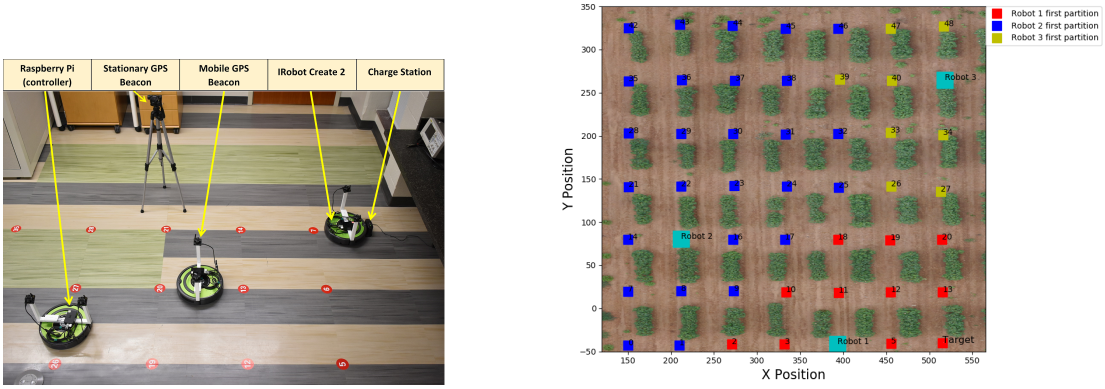


Figure 2.3: Reduction in the value of cost for the numerical example showing convergence after 10 iterations.

Simulations are done to deploy two robots (1 and 2 as shown in Fig. 2.2a) in the field which are desired to reach the target (node 37) while avoiding collision with obstacles. Fig. 2.2a shows the initial Voronoi partitions before the robots begin to move. In the next step, by solving the underlying optimization problem, each robot finds the next best node considering those paths that are free of obstacles. As observed from Fig. 2.2b, after ten iterations, robot 2 (despite being further away from the region of interest than robot 1) reaches the target since the other robot's (robot 1) path to reach the target is blocked by several obstacles. Nonetheless, robot 1 moves towards the center of its Voronoi partition to minimize the cost function shown in Fig. 2.3.

2.4.2 Experimental Results

Next, we implemented the coverage control algorithm in a lab-scale test bed to mimic the cooperation of ground vehicles in actual agricultural fields. For this purpose, we used three *iRobot Create2*. For the localization, *Marvelmind indoor GPS* module was employed providing centimeter accuracy. The GPS set includes 1 router and 4 stationary beacons mounted on the tripods located in the corners of the test bed. It also includes 4 paired mobile beacons placed on iRobots horizontally as shown in Fig. 2.4a that receive location update from the router. We used a pair of mobile beacons for each robot to find the heading of the robot and its direction. iRobot is an open source platform and easy to program. Proposed coverage control algorithm and associated modules including trajectory optimization, obstacle detection, communication and navigation are implemented in *Python* using *Raspberry Pi* with Fig. 2.4a illustrating the test bed and its various important components. All information including location, battery charge status and detected obstacles are transmitted to other ground robots through a local (WiFi) network. We consider one robot as server and others as clients, so every two agents can transmit or receive data through the server-client communication. Before deploying robots in the field, the field layout and the plots therein are constructed using images taken by a drone. We used the map in Python to set up our test bed similar to the numerical example (given earlier) by finding a matrix to model it as a graph. The field is divided into 36 regions and 49 nodes. Unlike the numerical example, in this experiment, there are three regions of interest shown as "target" in Fig. 2.4b.



(a) Picture of the test bed and its different elements

(b) Simulated test bed in Python showing initial Voronoi regions.

Figure 2.4: Our lab-scale test bed and how it is modeled as a graph.

In this experiment, robots iteratively solve the optimization problem corresponding to the coverage control while guaranteeing that no collision occurred. Initially, the three agents are located in nodes 4, 15 and 41 as shown in Fig. 2.4b. It is observed from the figure that field is divided into three Voronoi partitions corresponding to the three robots. In the next step, if there is a neighboring node with smaller cost in that partition, it will be set as the next best point to move to. Then, all the necessary steps such as computing Voronoi sub-graph, total cost function and next position are repeated until the total cost converges. In this experiment, cost converges after six iterations. Fig. 2.5 illustrates that all robots reach the targets after six steps by moving along the shortest path with an optimal cost. The robots are moved such that the whole field is optimally covered. Final Voronoi partitions are shown with different colors in Fig. 2.5.

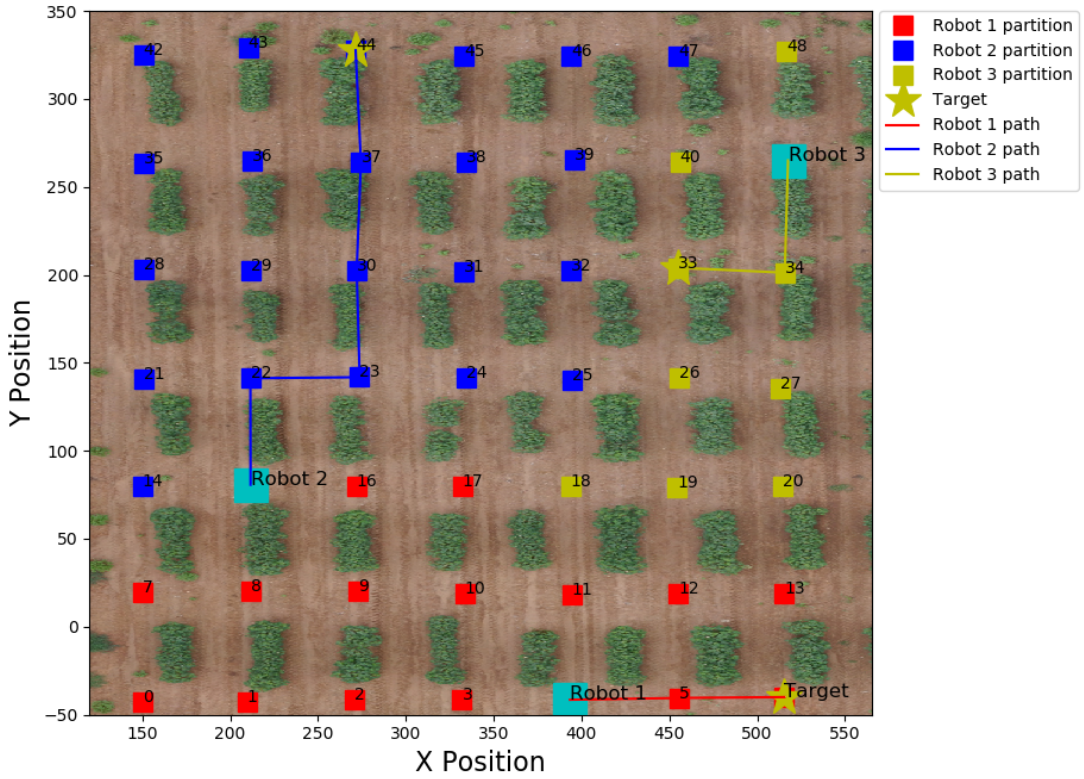


Figure 2.5: Final Voronoi partitions and robot trajectories for the experiment.

2.5 Conclusion Remarks

In this paper, a discrete coverage control method was proposed for deploying a team of UGVs in an agricultural field modeled as a directed graph. A distributed algorithm was proposed to navigate the robots in such a way that the collision with existing obstacles in the field was avoided and simultaneously multiple areas of interest were monitored and covered. The success of the proposed approach was illustrated using a numerical example, as well as experimental results from a

laboratory-scale test bed consisting of multiple robots. Our ongoing work focuses on the cooperation of UGVs and drones for deployment in the field. Drones that are faster and free to move above obstacles are flown over the field to capture images and identify areas with phenotype problems or water stress (regions of interest). Then, UGVs move towards the areas of interest to collect higher resolution data. Another aspect we are investigating is the detection of undesired objects in the path to improve the safety of the mission.

Chapter 3

AUTONOMOUS REAL-TIME MONITORING OF CROPS IN CONTROLLED ENVIRONMENT AGRICULTURE[1]

[1]Saba Faryadi, Mohammadreza Davoodi, and Javad Mohammadpour Velni. Accepted by *American Society of Mechanical Engineers (ASME) 2019, Dynamic Systems and Control Conference*. Reprinted here with permission of publisher.

Abstract

In this work, we develop a system that can be used for real-time monitoring of multiple important areas in controlled environment agriculture (and in particular greenhouses) using an autonomous ground vehicle (AGV). To model the greenhouse layout, as well as the tasks that should be accomplished by the AGV, we generate two weighted directed graphs. Based on those graphs, an algorithm is then proposed for finding the optimal (in the sense of traveled distance) trajectory of the vehicle with the goal of precisely monitoring important areas in the greenhouse. Furthermore, a data collection system and image processing algorithm is proposed and implemented so that the vehicle: (i) can capture images and detect changes that have occurred on the crops in real time, and (ii) construct (if needed) a map of the plant rows, when arriving at each one of the important areas. Based on this work, the images can either be stitched onboard the vehicle and then sent to a server or be sent directly to the server and then processed (stitched) there. Both simulation and experimental results are provided to demonstrate the effectiveness and performance of the proposed system.

3.1 Introduction

Throughout the years, controlled environment agriculture's (and in particular, greenhouse) role in achieving a sustainable food resource has been highlighted and its development has received a lot of attention [Benke and Tomkins, 2017, Shamshiri et al., 2018]. The use of modern technologies such as remote sensing and

automation has also improved greenhouse efficiency and crop yield [R Shamshiri et al., 2018]. Remote sensing can provide information of important variables such as climate parameters in different parts of a greenhouse. For instance, in [Ahonen et al., 2008], Ahonen *et al.* proposed a method to integrate three commercial sensors to measure environmental factors. Since environmental and soil sensors are unable to fully monitor physical characteristics of crops, visual sensors are becoming more popular to collect image data for various applications including crop growth analysis, pest detection, etc. Maharlooei *et al.* in [Maharlooei et al., 2017] presented an image processing method for identifying and counting different sizes of soybean aphids. An image interpreting technique to find calcium deficiency in lettuce crops in the greenhouse was proposed in [Story et al., 2010]. In [Pourdarbani and Rezaei, 2011], authors presented a method for automatic image capturing and focused on off-line image analysis to control pests and diseases of greenhouse plants. Similar to [Pourdarbani and Rezaei, 2011] and [Maharlooei et al., 2017], most studies that use visual data for greenhouse applications examine off-line data processing. Only some of the recent work that use static sensor networks focus on real-time data processing. For instance, a micro climate real-time monitoring system using a ZigBee wireless sensing network was designed and implemented in [Postolache et al., 2012].

On the other hand, developing autonomous systems to monitor greenhouse production is an emerging area that may replace human operators in the very near future. Ruiz-Larrea *et al.* in [Ruiz-Larrea et al., 2016] described the application of an autonomous ground robot to measure ground properties of the greenhouse. In

[Kitamura and Oka, 2006], a system for finding sweet peppers in the greenhouses was developed, where then an autonomous picking robot was used. In [Roldán et al., 2016], the application of a heterogeneous multi-robot system for monitoring various variables of greenhouses in the presence of environmental obstacles was addressed.

In this work, we aim to develop and test an automated system for real-time monitoring of important areas in greenhouses using an autonomous ground vehicle (AGV). It is assumed that the entire greenhouse is first scanned, *e.g.*, by an unmanned aerial vehicle. Then, the aerial images are used by the ground robot for further investigation of detected areas. To achieve this, our main objective is to solve the problem of finding the optimal trajectory for the AGV so that multiple regions of interest in the greenhouse can be effectively monitored. This is an example of multi-goal task planning problem that has been explored in the robotics community [Englot and Hover, 2011, Sagar et al., 2017].

By modeling the greenhouse as a weighted graph and using the knowledge of important areas (*e.g.*, identified by the aerial vehicle), a task graph [Bhattacharya et al., 2010] is generated to integrate different possible execution order of tasks for the robot. The optimal order of execution of the tasks is then obtained by solving a *shortest path* problem for the task graph. Finally, an image acquisition and processing algorithm is implemented to accomplish the objectives of plants map reconstruction, as well as early detection of changes in important areas in both real time (on the robot) and offline (on the server).

The remainder of this paper is organized as follows. The problem under study

is described in Section 2. Section 3 presents our proposed methodology for construction of the greenhouse and task graphs, computation of the optimal trajectory for the AGV, as well as the data collection and image processing modules. In Section 4, a detailed description of our testbed is provided. Section 5 is devoted to simulation and experimental results, and finally, Section 6 gives conclusion and directions for future work.

3.2 Problem Statement

Image reconstruction of plants and roots play a key role in monitoring all visible details from crops and soil. The ideal image data as the resource of the remote sensing for greenhouse can be a panoramic high resolution image of the plant that provides a wider field of view. Furthermore, from the viewpoint of growers and plant biologists, early detection and identification of any disorder or changes in plants is critical for greenhouse management. Color changes, bloom detection or other changes in physical characteristics can be detected using image processing methods.

This paper mainly focuses on greenhouse plants aiming to develop a system that can simultaneously detect changes in plants (such as disorders), as well as reconstruct and transfer a map of the greenhouse rows in real-time to a server for further investigation. To this end, an autonomous plant monitoring system for capturing and transferring the image data is developed. This type of data can be widely used to continuously monitor the plants growth and improve their

productivity.

In the first step, important areas of greenhouse are determined before even deploying the ground robot. To achieve this goal, a drone that is faster and free to move above obstacles is flown over the rows to collect low resolution data. Moreover, an initial offline data analysis module is used to identify regions of interest (e.g., areas with growth problems). Then, an AGV is deployed to move in the greenhouse and monitor the target plant rows for providing a better visual understanding. To achieve the aforedescribed goals, our objectives in this paper are threefold: First, to generate the optimal path for the AGV to move in the greenhouse and collect data (images) once the regions of interest are detected; Second, to obtain the reconstructed images of the important areas both online and offline. In the real-time case, the images are processed and stitched onboard the robot and then transferred to the server. However, in the offline case, at first the images are transferred to the server and then combined (stitched) there; *Third*, to detect different types of changes and disorders in the plants in real time.

3.3 Methodology

In this section, we first describe our path planning approach for monitoring multiple important areas. Then, the data collection and image processing modules are introduced.

3.3.1 Path Planing with Multiple Tasks

To achieve an optimal path planning, two graphs are considered for modeling the greenhouse environment and the order of completion of the tasks the AGV is assigned to. Then, these graphs are utilized to find the optimal path for the AGV.

Greenhouse Graph

The Greenhouse is converted into a weighted directed graph $\mathcal{G}_1(\mathcal{V}_1, \mathcal{E}_1, \mathcal{C}_1)$ with the node set $\mathcal{V}_1 = \{1, 2, ..., m\}$ (m denotes the total number of known nodes in the greenhouse environment), the directed edge set $\mathcal{E}_1 \subseteq \mathcal{V}_1 \times \mathcal{V}_1$, and the specific costs (weights) $\mathcal{C}_1$. To obtain an insight into the underlying process, a map of greenhouse with its corresponding graph is depicted in Fig. 3.1. In this work, it is assumed that the robot is equipped with a right-sided camera. Therefore, the nodes around each plant row are connected by using directed edges in such a way that the robot can capture appropriate images from the rows. Moreover, the nodes around each plant row are connected through undirected edges to appropriate nodes around other plant rows. Costs/weights of the edges of the graph are the metric lengths of the edges. This graph is used by the AGV for navigation purposes. Very recently, in [Davoodi et al., 2018], a graph-based approach was proposed to model an agricultural field as a topological map. This graph was then used in a distributed deployment strategy to navigate a group of robots towards the nodes with higher priority. In [Davoodi et al., 2018], each node represented a portion of the field, which is the region next to the plant rows. However, in

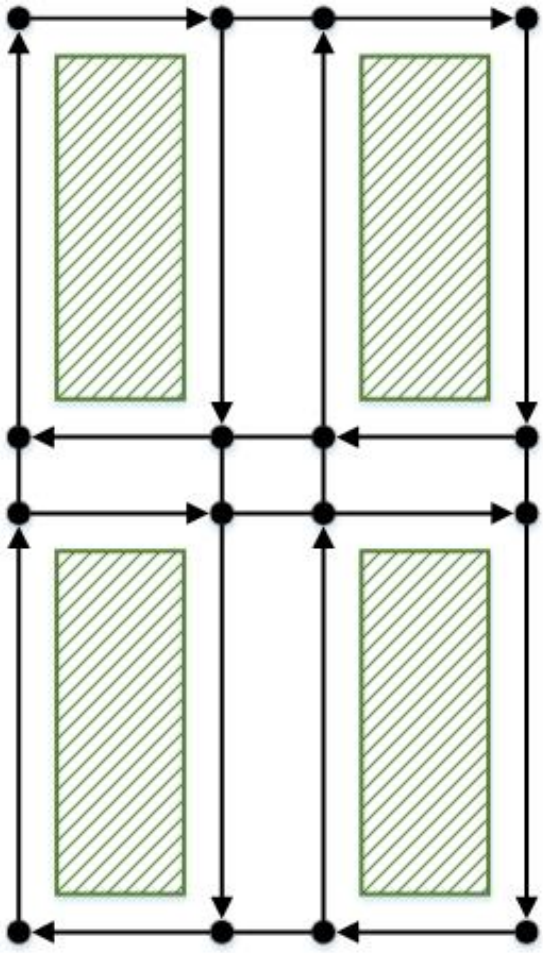


Figure 3.1: An illustrative example showing the map of the greenhouse with required information for modeling the underlying graph. The figure shows four plots of plants.

contrast with [Davoodi et al., 2018], in this work each node of the graph is defined as a point in the greenhouse around the plant rows.

Task Graph

It is assumed that monitoring of M important areas has been assigned to the AGV. In order to precisely monitor each of these areas, four subtasks are assigned to the robot. These subtasks are indeed the corresponding nodes of graph (nodes around each plant row as depicted in Fig. 3.1) that the robot should pass through for capturing the images. Consequently, the planned trajectory of robot should be computed in such a way that the robot passes in total through $4M$ nodes in the graph. However, the order of execution of the tasks is obtained as a solution to the planning problem. Similar to the results of [Bhattacharya et al., 2010], another weighted directed graph $\mathcal{G}_2(\mathcal{V}_2, \mathcal{E}_2, \mathcal{C}_2)$ with the node set $\mathcal{V}_2 = \{1, 2, ..., 4M\}$, the directed edge set $\mathcal{E}_2 \subseteq \mathcal{V}_2 \times \mathcal{V}_2$, and the specific costs (weights) $\mathcal{C}_2$, is considered to define the task (search) graph. The nodes of this graph are binary numbers. Each binary variable consists of $4M$ bits, each of which is an indicator of whether or not the corresponding task has been completed. Consequently, the value 1 is considered at each position where its corresponding task has been completed, and the value 0 otherwise. Moreover, there exists a connection between two nodes of the graph if and only if their corresponding binary values differ by exactly one bit. In this case, the direction of edge is from the node with lower binary value to the one with higher value. Weights of the edges of the graph are the shortest path between the corresponding nodes. It is noted that the graph $\mathcal{G}_1$ and also the Dijkstra algorithm [Dijkstra, 1959] are used to find the shortest path between two arbitrary tasks in $\mathcal{G}_2$. An example of such a graph for accomplishing three tasks is shown in Fig. 3.2.

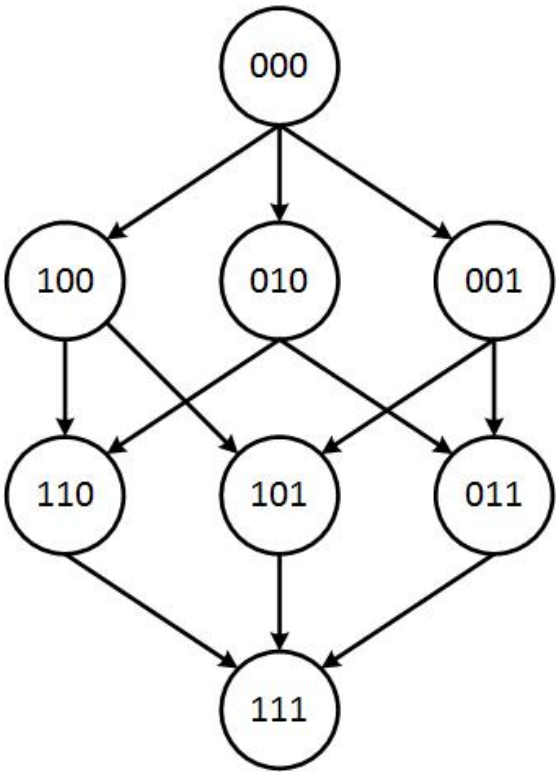


Figure 3.2: An illustrative example showing the task graph when there are three tasks to accomplish.

Computation of Optimal Trajectory for Robot

Given the problem formulation, the goal of computing the optimal trajectory for the robot would be to find the minimal path, from the traveled distance viewpoint, through the corresponding task graph. This problem can be equivalently formulated as finding the shortest path from the initial to the end points of the task graph. The Dijkstra algorithm is again used for finding the shortest path. Consequently, the result of applying the Dijkstra algorithm to the task graph provides the optimal (in the sense of traveled distance) trajectory of the robot so that

the important areas in the greenhouse can be effectively monitored.

3.3.2 Data Collection and Image Processing

In this subsection, our data (image) collection approach is discussed and then two image processing approaches are proposed for interpretation and analysis of captured images.

Data Collection

In the proposed approach, it is assumed that the first image is captured once the robot arrives at a node corresponding to a region of interest. Moreover, while the robot moves through an edge corresponding to an important area, in certain footsteps, it stops and captures more images. The stop points of the robot are obtained based on the footprint of the robot's camera on the ground (field of view of camera). In this work, we assume that the camera is in parallel to the ground plane. Consequently, as shown in Fig. 3.3, the width L of the camera's footprint is computed as [Avellar et al., 2015]:

$$L = H\frac{l}{f}, \tag{3.1}$$

where l denotes the width of the camera, f denotes the focal length of the camera's lens, and H denotes the distance between the camera and the ground. Since the field of view of camera is limited, therefore, the robot is not capable of monitoring the whole plant row by capturing just one image. Therefore, to obtain an image

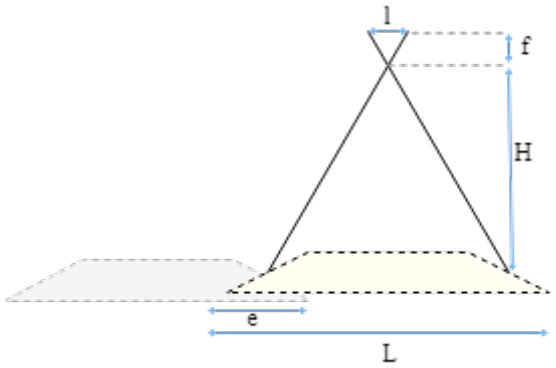


Figure 3.3: Illustrative example showing the field of view of Pi camera.

from the whole plant row, we are interested in taking overlapped images from specified parts of the plant row and then combine them together. To this end, it is assumed that the robot is stopped at some stop points for capturing images. The next stop point of robot is calculated based on the traveled distance $L-e$ from the previous stop point of the robot, where $0 < e < L/2$. Based on the above discussion, there are two sets of *stop points* for the robot to stop and capture images: first, the four nodes corresponding to each important area; and second, the position of points corresponding to each $L - e$ segment on the directed edges around that important area.

Map Reconstruction

A Python library, *OpenCV*, is used to combine all the pictures. The first step in panoramic image stitching is to extract and match common features among all

overlapping images. In this step, images with the greatest number of matches between them will be identified. There are two different sets of feature matches as follows: (1) geometrically consistent, and (2) inside the area of overlap but not consistent. Finally, a probabilistic model will be applied to verify the matches [Brown and Lowe, 2007]. The overall block diagram illustrating the process of image stitching is shown in Fig. 3.4. In this work, image processing is done both

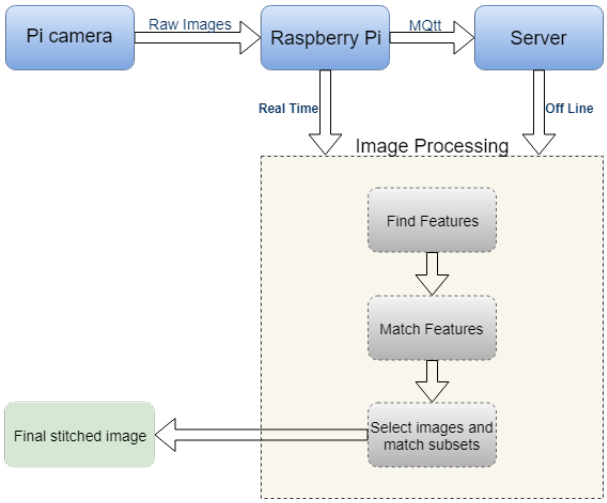


Figure 3.4: Block diagram of the image stitching process for both real-time and offline processing.

real-time and offline for stitching the images and reconstruction of the plant rows. In the first case (offline image stitching), as long as the ground robot is moving in the greenhouse and collecting data, the server waits to receive raw captured images and then starts preprocessing them. The images are then combined on a PC (defined as a server). In the second case (real-time image stitching), all raw captured images associated with each region of interest are stitched on a Raspberry Pi 3 (the processing unit we use) and then a single panoramic image of row plant will

be transferred to the server. In the presence of a good wireless coverage, PC can receive the images and process them fast. Although offline map reconstruction on a PC with a powerful CPU is faster than processing on Raspberry Pi, poor to non-existent wireless coverage in some places of the greenhouse leads to considerable delay in data transmission and affects the offline data processing. Consequently, to enhance the accuracy of the results, we propose real-time image processing on the Raspberry Pi instead of the server in real applications. This would also facilitate the use of emerging Fog (edge) Computing devices for distributed data processing and computing in large networks. It is, however, important to note that using Raspberry Pi (as the processing unit) although increases the chance of a successful real-time data analytics may result in a lower precision.

Early Detection of Changes

To quickly detect changes in the plants, the *OpenCV* library is used again for interpreting the captured images. Motivated by various agricultural needs (*e.g.*, bloom detection, leaf color change), we extract a specific color percentage in the original image. This percentage is then compared to a predefined threshold to detect a change in important areas of the greenhouse. This provides a complete interface to visualize and analyze spatio-temporal images of plants for the growers or operators.

3.3.3 Deployment of the Developed System

To implement the proposed monitoring system, the operator first deploys a drone to scan the whole greenhouse and identify important areas that require further attention. Then, the weighted directed graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are constructed for the greenhouse environment and task orders, as described earlier, respectively. The graphs are then shared with the robot that autonomously (and optimally) moves to the important areas, captures the required images, combines them and transfers the final results to the server for further investigation by the operator. Various steps involved in the process are summarized in Algorithm 1.

Table 3.1: Algorithm 1: Implementation of the proposed system.

Initialization

1. Receive the greenhouse map, initial position of the robot and important areas.
2. Construct the greenhouse graph.

Computing the optimal trajectories

3. Construct the task graph.
4. Find the shortest path between the initial and end nodes of the task graph.

Robot navigation and image acquisition

5. Use the generated shortest path for navigation.
6. **While** (robot has not reached the end node of the path)
7. Robot state ← Move
8. **if** (robot reaches one of the stop points) **then**
9. Robot state ← Stop and capture an image.
10. **end if**
11. **end While**

3.4 Description of the Experimental Testbed

The ground robot that we have used for deployment in the greenhouse is the *iRobot Create2* [iro] that is an open source, programmable platform and can be utilized in specific applications. It is also capable of communicating with PC. For the sake of autonomous navigation and localization, *Marvelmind indoor GPS* module [mar] is employed providing robot's position with centimeter accuracy. The GPS set includes 1 router and 4 stationary beacons mounted on the tripods located in the corners of the greenhouse. A pair of mobile beacons are attached to the robot to provide precise locational data and direction of the robot.

The multi-tasks path planning algorithm and associated modules, namely, communication, navigation, image acquisition and data processing, are implemented in *Python* language and compiled in *Raspberry Pi 3* (RPi). In fact, RPi is used as an onboard controller of the ground robot. Captured images are also stored in the memory of the RPi. All information including the locations of important areas and processed image data can be transferred between ground robot and PC through a local (WiFi) network. In this structure, we consider the PC as the *server* and AGV as a *client*, so that data can be transmitted or received through the server-client communication using MQTT protocol. For capturing the images, a 5M-pixel *picamera* is mounted on the iRobot and is kept parallel to the ground plane to allow taking pictures of the plants from above. Our experimental setup is shown in Fig. 3.5.

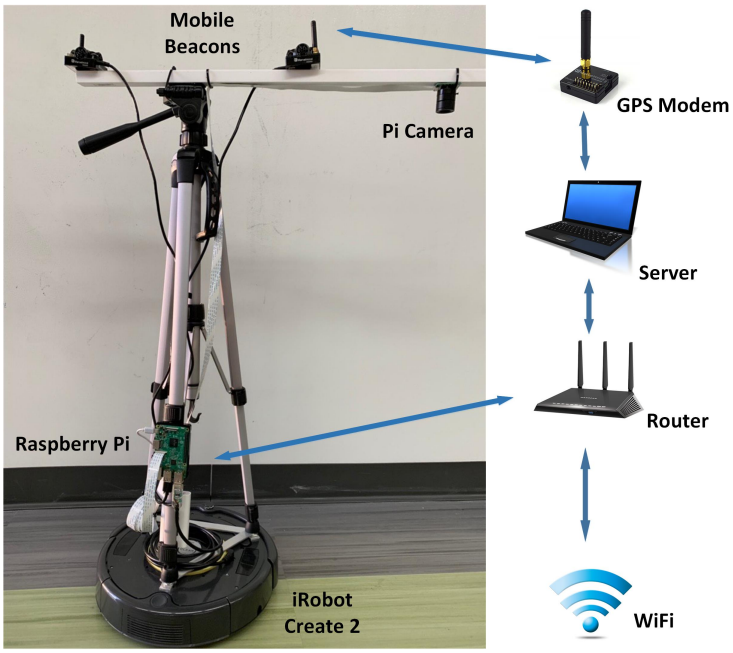


Figure 3.5: Experimental setup showing various components.

3.5 Experimental Results

To demonstrate the efficacy of the developed system, we consider a greenhouse lab setup shown in Fig.3.6. Each plant row consists of sixteen pots of the green pepper.

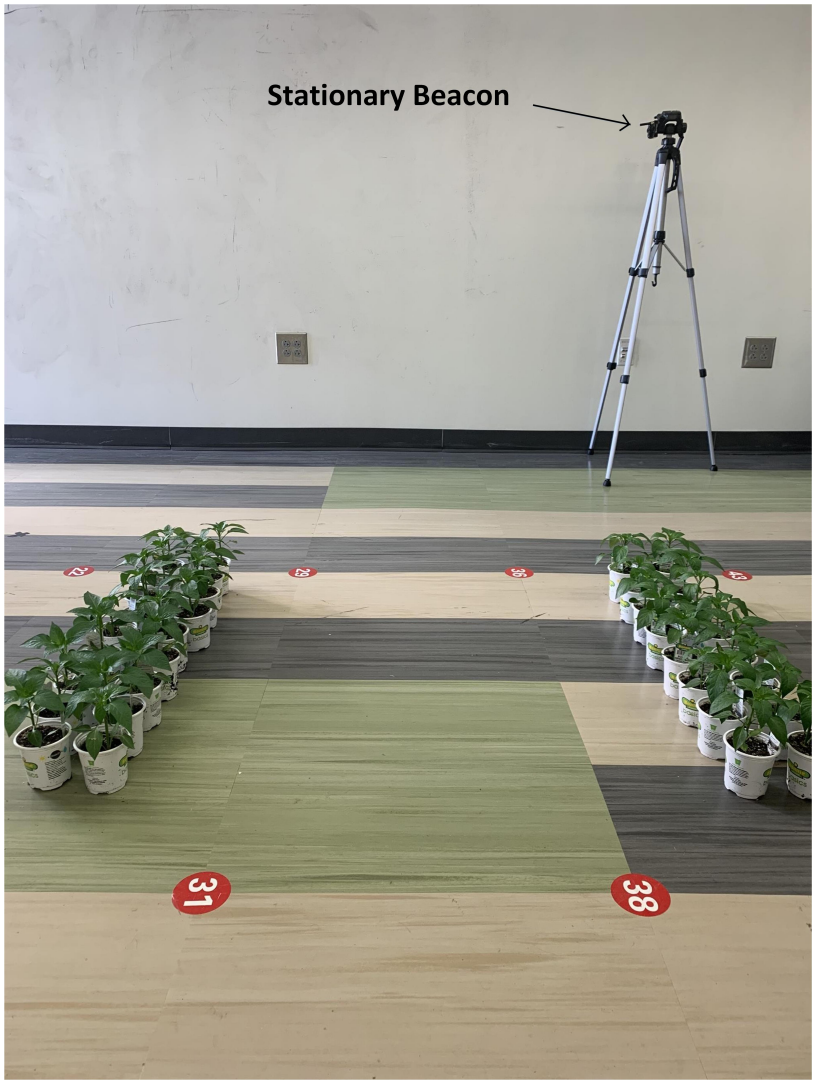


Figure 3.6: Partial view of Greenhouse lab setup used in our study.

The considered environment is modeled as a directed graph $\mathcal{G}_1$ with 44 nodes. The robot is assumed to be initially located at node 6.

3.5.1 Map Reconstruction

It is assumed that two plant rows corresponding to node sets $\{7, 8, 13, 14\}$ and $\{33, 34, 39, 40\}$ are selected as the regions of interest that require attention. The task graph for 8 important nodes ($M = 2$) is generated, the shortest path problem in Algorithm 1 is solved, from which the optimal path for the robot is generated. The simulation and experimental results of path planning are shown in Fig. 3.7. The stitching result for one of the plant rows is shown in Fig. 3.8. This image

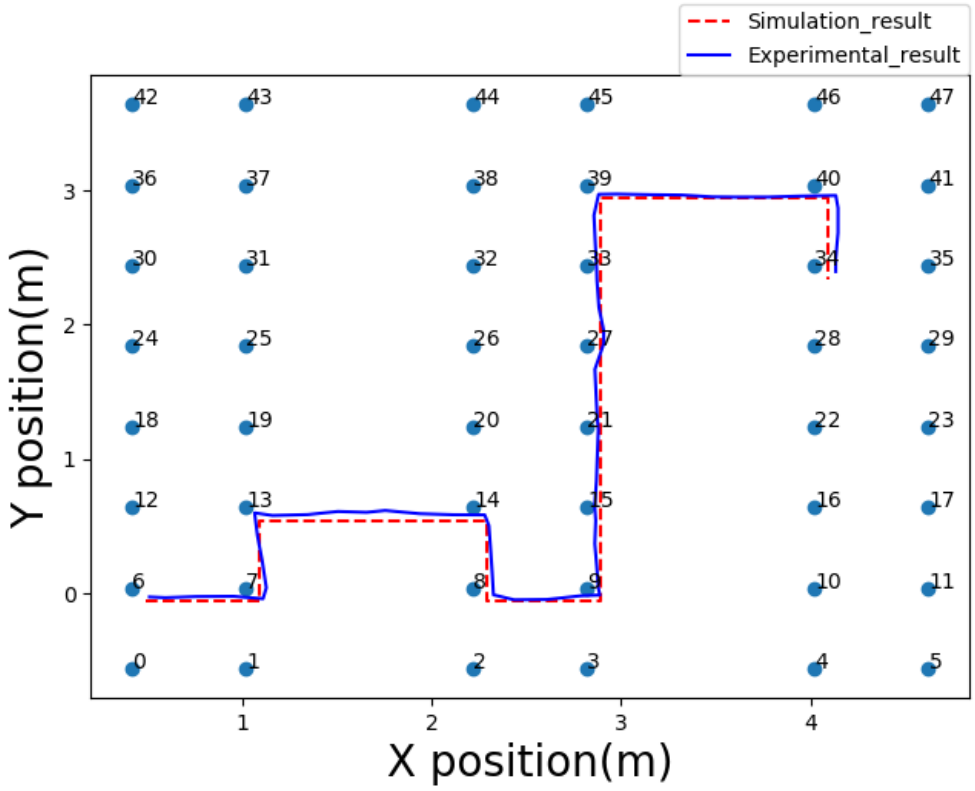


Figure 3.7: 2-D graph showing the results of path planning.

is obtained from combining five different images captured from different parts

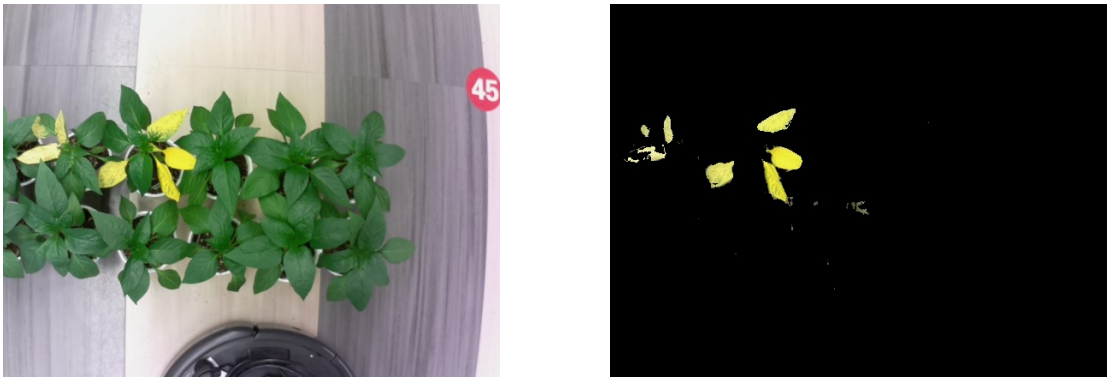
of the plant row. Stitching images on Raspberry Pi (real-time processing) took under 2 minutes and led to a successful panoramic image reconstruction. It can

Figure 3.8: Stitched image for the first plant row (combination of 5 images). The image was constructed in real time onboard the AGV.

be observed from figures 3.7 and 3.8 that the real-time monitoring of multiple important areas in the greenhouse using an AGV was achieved using the developed system.

3.5.2 Real-time Detection of Changes in Greenhouse Plants

To demonstrate the effectiveness of the proposed system for the purpose of detecting (in real-time) disorders in the plants, three other plant rows are considered as important areas requiring more investigation. Furthermore, it is assumed that one of the plant rows contains a number of plants with yellow leaves. So, it is expected that the robot monitors these plant rows, identifies the position of disordered plants and provides a real-time visualization of the detected areas to the operator. The results are shown in figures 3.9 and 3.10. As shown in the figures, the disordered plants were effectively identified and extracted from other healthy plants in the row. Moreover, the operator can monitor the results and have ac-



(a) Original plant image.

(b) Detecting and extracting the leaves undergone a color change.

Figure 3.9: Real-time detection of changes in the leaves.

cess to the plant positions in a real-time manner. Indeed, Fig. 3.10 provides the real-time information transmitted by the RPi to alert the operator on the status of each important area that the robot had covered.

3.6 Conclusion

In this work, we examined the problem of computing optimal paths for monitoring multiple areas in a greenhouse via an autonomous ground vehicle (AGV). A graph-based method for modeling a partially known greenhouse as a topological map was proposed. By utilizing this graph and knowing the important areas, another weighted directed graph, named task graph, was constructed to capture different possible task orders for the robot. By using the Dijkstra algorithm, a shortest path problem for the task graph was solved to find the optimal order of task executions.

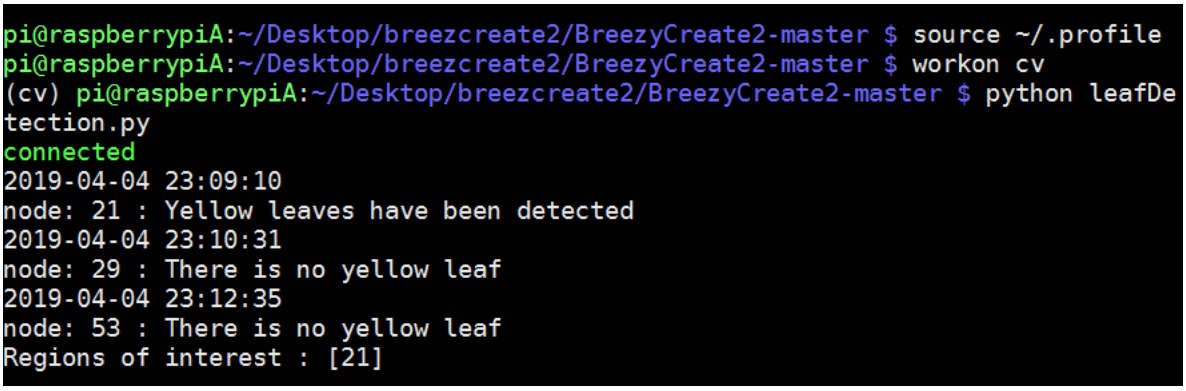


Figure 3.10: Snapshot of RPi interface showing real-time analysis results.

An image capturing and processing approach was also presented which was capable of stitching images captured from different parts of plant rows, reconstruction of the whole plant row and then transferring it to the server. Furthermore, another image processing method for online detection of changes in the greenhouse crops was proposed. The proposed system was experimentally validated using a lab-scale greenhouse testbed to demonstrate its capabilities. The future work will concentrate on the deployment of the proposed system in a large greenhouse. Moreover, deploying a single vehicle/robot (this work) is useful for a small number of plants in small farms and greenhouses. Another area for future work is to extend the results of this work to the case with a team of collaborating AGVs for monitoring of thousands of plants in large farms/greenhouses and under more practical conditions.

Chapter 4

OPTIMAL PATH PLANNING FOR A TEAM OF HETEROGENEOUS DRONES TO MONITOR AGRICULTURAL FIELDS[1]

[1]Saba Faryadi, Mohammadreza Davoodi, and Javad Mohammadpour Velni. Submitted to *American Society of Mechanical Engineers (ASME) 2020, Dynamic Systems and Control Conference*, 04/20/2020.

Abstract

In this work, we investigate the problem of finding the minimum coverage time of an agricultural field using a team of heterogeneous unmanned aerial vehicles (UAVs). The aerial robotic system is assumed to be heterogeneous in terms of the equipped cameras' field of view, flight speed and battery capacity. The coverage problem is formulated as a vehicle routing problem (VRP) [Toth and Vigo, 2002] with two significant extensions. First, the field is converted into a graph including nodes and edges generated based on sweep direction and the minimum length of UAVs' footprints. Second, the underlying optimization problem accounts for aerial vehicles having different sensor footprints. A series of simulation experiments are carried out to demonstrate that the proposed strategy can yield a satisfactory monitoring performance and offer promise to be used in practice.

4.1 Introduction

With the advancement in technology, there has been much ongoing research work on the use of robots for precision agriculture applications to improve the quality and quantity of the crop production [Xing et al., 2017]. Candiago *et al.* in [Candiago et al., 2015] deployed a multi-rotor aerial vehicle to monitor a vineyard and a tomato farm, where the collected imaging data were then analyzed and evaluated. Berni *et al.* developed an autonomous aerial system to remotely monitor a vegetation field with thermal and imaging sensors in [Berni et al., 2009]. Bakker *et al.* in [Bakker et al., 2011] suggested an autonomous navigation ground robot

to map crop rows and monitor sugar beets in a field.

Due to the limitations of a single unmanned vehicle, multiple robots have been used to reduce the time taken to accomplish tasks and improve overall system performance [Walter et al., 2018]. In [Gonzalez-de Santos et al., 2017], a system consisting of aerial and ground vehicles was used to monitor and control pests and weeds in three different agricultural fields. Authors in [Vasudevan et al., 2016] deployed a multi-robot system consisting of an unmanned ground vehicle (UGV) and an unmanned aerial vehicle (UAV) to monitor crops in a farmland and collect crop data. Besides, the use use of remote sensing with team of aerial robots has recently received considerable attention for various purposes like collecting visual data of crops and fields and the map reconstruction of the fields.

For instance, Chao *et al.* [Chao et al., 2008] proposed multiple automated aerial vehicles to remotely detect water level of crops based on image analysis. Moreover, in [De Rango et al., 2017], a team of UAVs was proposed to monitor and control the presence of pests on a farmland. In practical scenarios, teams of heterogeneous UAV systems might be deployed that have different capabilities (e.g., dynamics, fields of view, speeds or battery charges among others). Fields of view are important factors that should be considered when solving coverage optimization problems for UAVs. Ahmadzadeh *et al.* in [Ahmadzadeh et al., 2008] proposed a coverage method using four heterogeneous UAVs with different footprints on the ground. Each camera's field of view was defined to depend on the heading, back angle, and position of the vehicle and was considered in solving the coverage problem. In [Pimenta et al., 2008], the authors also deployed an

algorithm for a team of heterogeneous UAVs with different circular fields of view.

To aid optimal deployment of a team of heterogeneous UAVs, the coverage areas can be modeled as a graph so that each of the vehicles can visit the edges of the graph at least once and avoid overlapping [Toth and Vigo, 2002]. In [Faryadi et al., 2019b], Faryadi *et al.* proposed an area coverage method by modeling the field of study into a corresponding weighted graph with different nodes and edges to simplify path planning. Davoodi *et al.* in [Davoodi et al., 2018] modeled a target agricultural field into a weighted undirected graph with nodes defined around plant rows. Although drones have no limitation to fly in any direction, to avoid sharp turning and to move constantly during their missions, they are enforced to monitor the area executing back and forth motions in most studies. Huang *et al.* in [Huang, 2001] modeled a coverage area as a directed graph to determine the optimal coverage direction by visiting each region of the area in a back-and-forth motion. Similarly, in deploying multiple robots to spray insecticides on a farmland in a back-and-forth movement, Luo *et al.* [Luo et al., 2017] modeled the field by a rectangular directed graph so that the flight trajectories of the robots are parallel to the edges of the field. After converting the area into a graph consisting of nodes and edges, the coverage problem is formulated as a vehicle routing problem (VRP) and each node is considered as a customer that must be visited with a single vehicle (drone). In [Ergezer and Leblebicioğlu, 2014], a solution to the VRP problem for multi-UAVs was proposed by defining the objective function based on the information collected by the vehicles' cameras. In [Guerriero et al., 2014], Guerriero *et al.* addressed a dynamic VRP in which the nodes of the graph

represent the temporal and spatial coverage points.

In this work, our main objective is to solve the problem of minimizing the coverage time of monitoring an agricultural field using a team of heterogeneous UAVs. More specifically, we assume that the UAVs can have different fields of view, battery charges and speeds. Our work is motivated by the methodology used in [Avellar et al., 2015], in which a fleet of UAVs were deployed to complete a mission of field coverage. In [Avellar et al., 2015], the coverage area was modeled by a directed graph and the optimization problem was defined as a min-max problem such that the time spent traveling through the longest route (that is maximum flight time) is minimized. The setup time for prepping and launching the vehicles was also considered. However, in this paper, we specify the number of UAVs and consider the field of view of the vehicles' cameras in the formulated optimization problem.

The remaining sections of this paper are organized as follows: Section 2 provides a detailed description of the problem we study here, as well as our proposed methodology to address that. Section 3 presents the simulation results, and Section 4 concludes the paper and provides perspectives for future work.

4.2 Problem Formulation

The main goal of this work is to find the solution to the optimal deployment problem of M heterogeneous UAVs (drones) in an agricultural field represented as a weighed graph in such a way that the maximum traveling time by them be

minimized. We assumed that a team of autonomous drones equipped with RGB cameras with different fields of view need to be deployed to monitor the field and to collect image data from plant rows. Assuming that cameras are parallel to the ground, L_m is used to denote the camera's footprint on the plane for m-th drone. Moreover, deployed drones are assumed to be different in terms of the maximum flight time t_m and flight speed V^m, where $m \in \{1, 2, ..., M\}$ and each drone flies at a fixed height H_m which is chosen in such a way that the camera can take sufficiently high resolution images.

The deployment problem will be solved following the steps below: (i) the given field is modeled as a weighted graph by considering the parallel coverage rows with optimum distance [Huang, 2001]; (ii) the coverage rows are partitioned between different drones by formulating and solving a mixed integer linear programming (MILP) optimization problem; (iii) based on the results of solving MILP, optimal trajectories for drones are generated so that they can effectively monitor the whole field.

4.2.1 Sweep direction and coverage rows

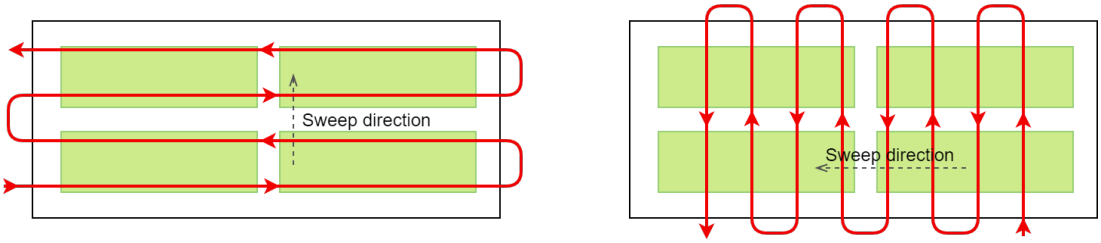


Figure 4.1: Left: sweep direction perpendicular to the plant rows; Right: sweep direction along the plant rows.

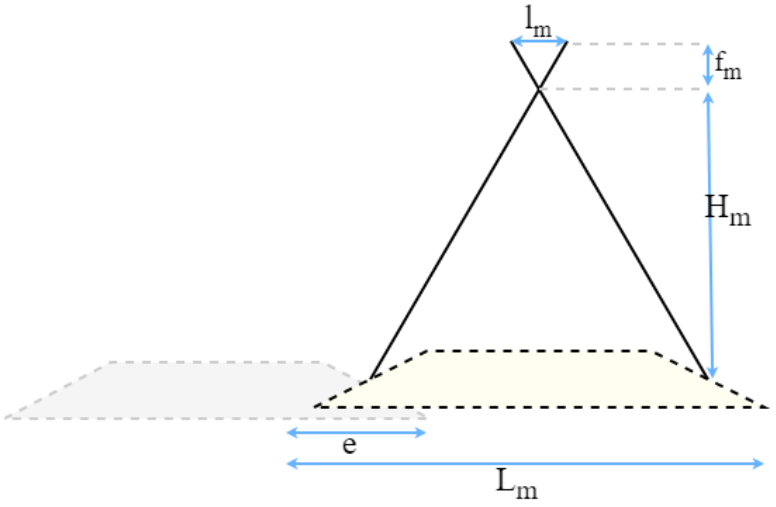


Figure 4.2: Illustrative example showing the field of view.

In most of the coverage applications, the coverage time and energy are two main factors contributing to the cost efficiency. Consequently, the coverage path must be generated in such a way that a combination of total time and energy be minimized. To this end, similar to the method proposed in [Huang, 2001], area is decomposed into sub-regions based on a specific sweep direction and all drones must fly over these regions using back and forth motion along the coverage rows, which are perpendicular to the sweep direction. For agricultural application, the main purpose to deploy drones over the field is to monitor plant rows and to capture appropriate images. To achieve this goal, the best sweep direction for drones is defined in such a way that they can fly along the plant rows, as shown in Fig. 4.1. After defining the sweep direction, coverage rows will be arranged with a fixed distance, which is calculated as a function of the smallest footprint of the on-board cameras on the ground. First, as shown in Fig. 4.2, the width L_m

representing the camera's footprint for the m-th drone is calculated as [Avellar et al., 2015]

$$L_m = H_m \frac{l_m}{f_m}, \tag{4.1}$$

where l_m is the width of the camera and f_m denotes the focal length of the camera's lens for the m-th drone. For M heterogeneous drones with different footprints, L is chosen as

$$L = \min\{L_1, L_2, ..., L_M\}. \tag{4.2}$$

Then, the number of coverage rows is

$$N_l = \Big\lceil \frac{w}{L(1-e)} \Big\rceil, \tag{4.3}$$

where w represents the field's width and $e \in (0, 1)$ is the fraction showing two images' overlap. Hence, the distance between two rows is

$$l = \frac{w}{N_l}. \tag{4.4}$$

Similar to our previous work [Faryadi et al., 2019a,b], the field is converted into a weighted graph represented by $G = (V, E, C)$ with the node set $V \in \{1, 2, ..., N\}$ shown in Fig. 4.3. Also, $E \subseteq |V| \times |V|$ is a set of arcs (edges), and C is the specific costs (metric lengths) of the edges. In this work, c_{ij} represents the Euclidean distance between nodes i and j. The novelty of this method is in arranging

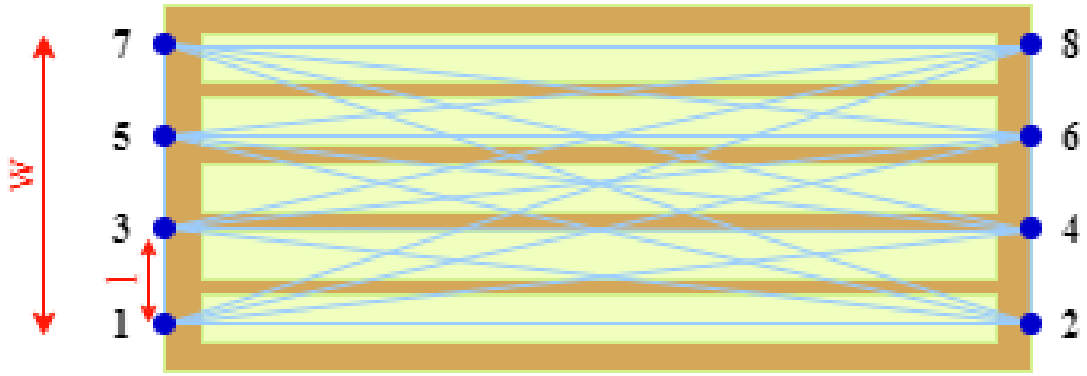


Figure 4.3: Field modeling as a graph.

coverage rows with the specific distance calculated based on the minimum length of camera's footprints (4.2) between the drones' on-board cameras.

4.2.2 Coverage rows partitioning

To divide the coverage rows between the vehicles considering their sensing differences, we define a new parameter ρ_m representing the ratio of m-th drone's footprint to the minimum length L as

$$\rho_m = \left[\frac{L_m}{L}\right]. \tag{4.5}$$

As shown in Fig. 4.4, ρ_m is defined to calculate the number of rows that can be covered (monitored) by m-th drone while it is flying over one coverage row. After generating the field graph G, inspired from the method proposed in [Avellar et al., 2015], the coverage problem is transformed to a vehicle routing problem (VRP)

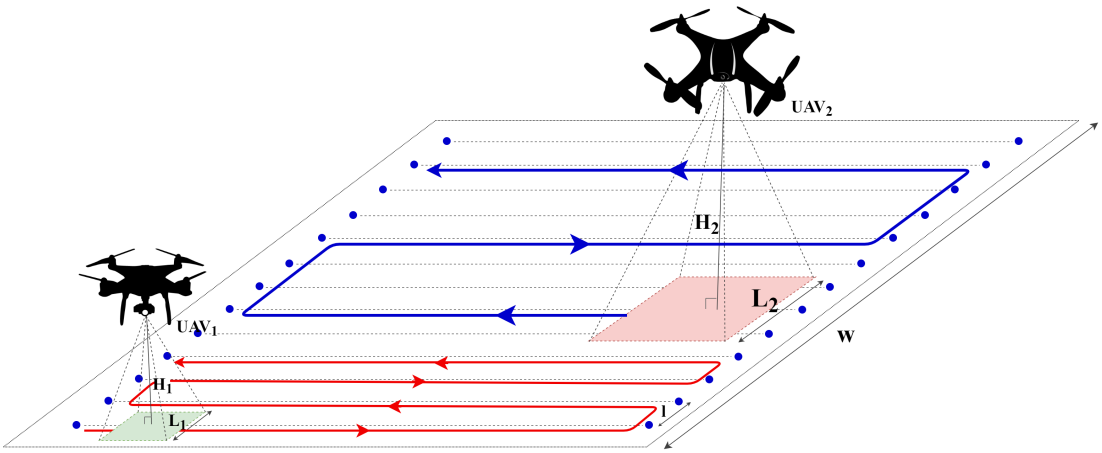


Figure 4.4: Illustrative example showing different footprints of two drones.

with specific constraints. It should be noted that our work extends the results of [Avellar et al., 2015] with two specific contributions that make our proposed problem unique. First, unlike [Avellar et al., 2015], our optimization problem (VRP) is formulated as a function of the drones' footprint and then solved for vehicles with different sensing capabilities; second, unlike most routing problems, in our work, solving the VRP problem results in partitioning coverage rows between vehicles and then optimum routes are generated based on cameras' footprints.

To formulate this optimization problem as a VRP, each node of the field graph is considered as a customer that needs to be visited by one drone. Moreover, decision variable $E_{ij}^{m} \in \{0, 1\}$ is defined indicating if $arc(i, j)$ is used on the route or not by m-th vehicle. Like most vehicle routing problems, in this application, the main objective is to partition nodes between all drones in order to minimize the time of mission. By knowing the m-th drone's flight speed V^{m} and cost matrix

C, the flight time of m-th drone is calculated as follows

$$t_m = \sum_{i=0}^{N} \sum_{j=0}^{N} \frac{c_{ij}}{\rho_m V^m} E_{ij}^m, \tag{4.6}$$

where c_{ij} denotes the entry (i, j) of the cost matrix C. Then, the longest traveling time between all drones is

$$T_{max} = \max(t_m). \tag{4.7}$$

With the purpose of minimizing the longest traveling time taken to cover the entire area, our optimization problem is

$$\min(T_{max}), \tag{4.8}$$

subject to several constraints explained below and given in equations ((4.10)) - ((4.18)). The following constraint is imposed to ensure that the m-th drone's total coverage time does not exceed the maximum time

$$\sum_{i=0}^{N} \sum_{j=0}^{N} \frac{c_{ij}}{\rho_m V_{ij}^m} E_{ij}^m \leq T_{max}. \tag{4.9}$$

For limiting the longest flight route of each drone with its battery capacity, another constraint is imposed as follows

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \frac{c_{ij}}{\rho_m V_{ij}^m} E_{ij}^m \leq \tau_m, \tag{4.10}$$

where τ_m denotes the maximum battery time duration of m-th drone.

Constraint (4.11) below enforces each vehicle to leave the depot (node 0) and arrive at a determined customer

$$\sum_{i=1}^{N} E_{0j}^{m} = 1. \tag{4.11}$$

Moreover, (4.12) guarantees that all drones would return back to the depot:

$$\sum_{j=1}^{N} E_{j0}^{m} = 1. \tag{4.12}$$

Constraint (4.13) ensures that each $arc(i, j)$ is monitored only once

$$\sum_{m=1}^{M} \sum_{i=1}^{N} E_{ij}^{m} = 1, \quad i, j \in \{1, 2, ..., N\},\ i \neq j. \tag{4.13}$$

The following constraint limits drones' entrance and exit flows to guarantee that the same vehicle would visit and leave each node of the graph

$$\sum_{i=0}^{N} E_{ip}^{m} - \sum_{j=0}^{N} E_{pj}^{m} = 0, \quad p = 0, 1, 2, ..., N. \tag{4.14}$$

In addition, equation (4.15) enforces drones to fly over coverage rows perpendicular to the given sweep direction and along the plant rows

$$\sum_{m=1}^{M} E_{i,i+1}^{m} + \sum_{m=1}^{M} E_{i+1,i}^{m} = 1, \quad i = 1, 2, 3, ..., N. \tag{4.15}$$

Two important constraints (4.16) and (4.17) are defined to impose drones executing back and forth motions to cover the area [Avellar et al., 2015]:

$$\sum_{m=1}^{M} E_{i,i+1}^{m} - \sum_{m=1}^{M} \sum_{(j=0,2,4,...\backslash i+1)}^{N} E_{(i+1,j)}^{m} = 0, \quad i = 1, 3, ... \tag{4.16}$$

$$\sum_{m=1}^{M} E_{i,i-1}^{m} - \sum_{m=1}^{M} \sum_{(j=1,3,5,...\backslash i-1)}^{N} E_{(i-1,j)}^{m} = 0, \quad i = 2, 4, 6, ... \tag{4.17}$$

And finally, constraint (4.18) given below guarantees that the VRP solution would make drones avoiding sub-tours:

$$y_i - y_j + N \sum_{m=1}^{M} E_{ij}^{m} <= N - 1, \quad i, j \in \{1, 2, 3, ..., N\}, \quad i \neq j, \tag{4.18}$$

where $y_i, y_j \in \{0, 1\}$ are additional binary variables.

After solving the optimization problem, represented as a VRP, all nodes of the field graph and the corresponding coverage rows are divided among a team of heterogeneous drones considering their footprint ratios, flight speeds and battery limitations (see Fig. 4.4). As an example, Fig. 4.5 shows coverage regions resulted from solving the VRP for two drones with different footprints. In this figure, the edges plotted with solid red lines from node 1 to node 8 are assigned to the drone with $\rho = 1$ and dashed blue edges from node 9 to 26 are assigned to the drone with $\rho = 2$.

In the ensuing subsection, a path planning method will be applied to generate the optimal flight route for each drone with the purpose of monitoring its partition

resulted from the VRP solution.

4.2.3 Drones' path planning

By solving the aforediscussed MILP optimization problem, the field graph G is partitioned into m sub-graphs, each of which is assigned to one of the drones. Then, the optimal trajectories for each drone should be generated to make it capable of flying over these rows and collect image data of corresponding sub-region.

As explained before, the camera's footprint ratio denoted by ρ_m represents the number of coverage rows monitored by the m-th drone while flying over only one row. For instance, in Fig. 4.5, the green squares show camera's footprint of the first drone with $\rho_1 = 1$ while the red ones (larger ones) represent second drone's footprint with $\rho_2 = 2$. From Fig. 4.5, it is clear that when a drone is flying over an arbitrary edge (i, j), in addition to this edge, it will simultaneously cover R_m neighboring rows ($R_m/2$ rows from the right and left sides of the (i, j)-th one), where

$$R_m = 2(\rho_m - 1). \tag{4.19}$$

For instance, in this figure, dashed blue trajectory shows the VRP solution for the drone with $\rho = 2$ and it is in the following order:

$depot \rightarrow 9 \rightarrow 10 \rightarrow 12 \rightarrow 11 \rightarrow 13 \rightarrow 14 \rightarrow 16 \rightarrow 15 \rightarrow$

$17 \rightarrow 18 \rightarrow 20 \rightarrow 19 \rightarrow 21 \rightarrow 22 \rightarrow 24 \rightarrow 23 \rightarrow 25 \rightarrow 26 \rightarrow depot.$

However, this trajectory is not optimal and using R_m as in (4.19), after completing each row, $2R_m$ nodes should be skipped to find the next best node and finally the

following optimal route is obtained

$depot \rightarrow 12 \rightarrow 11 \rightarrow 17 \rightarrow 18 \rightarrow 24 \rightarrow 23 \rightarrow depot$.

Based on the above explanation, when the second drone (with $\rho_2 = 2$) flies over an arbitrary row, in addition to that row, it also monitors $2(2-1) = 2$ more (left and right) neighboring rows. Therefore, after solving the proposed MILP optimization problem that partitions the field graph between different drones, the optimal trajectories for all the drones will be generated following the approach described above.

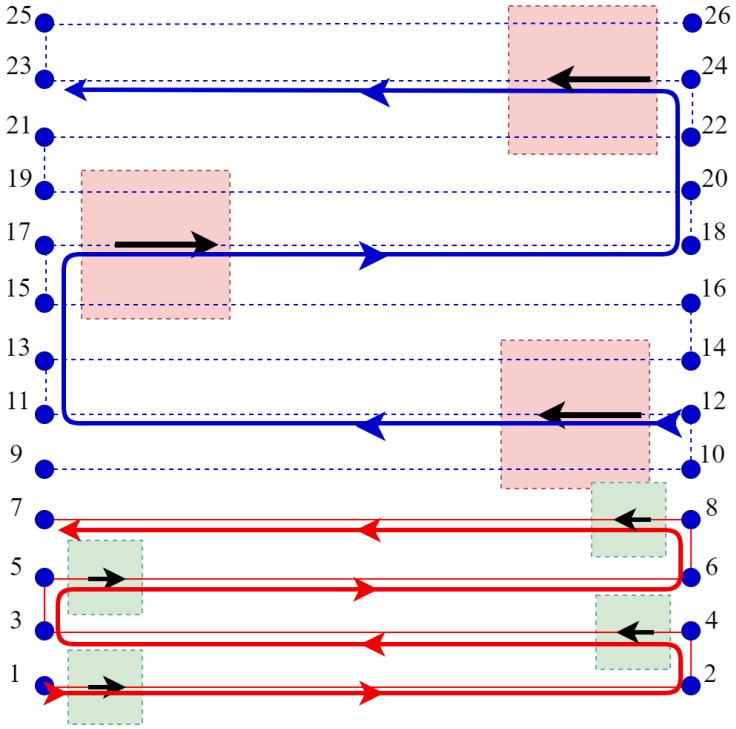


Figure 4.5: Illustrative example showing path planning for two drones with different fields of view.

4.3 Simulation Results

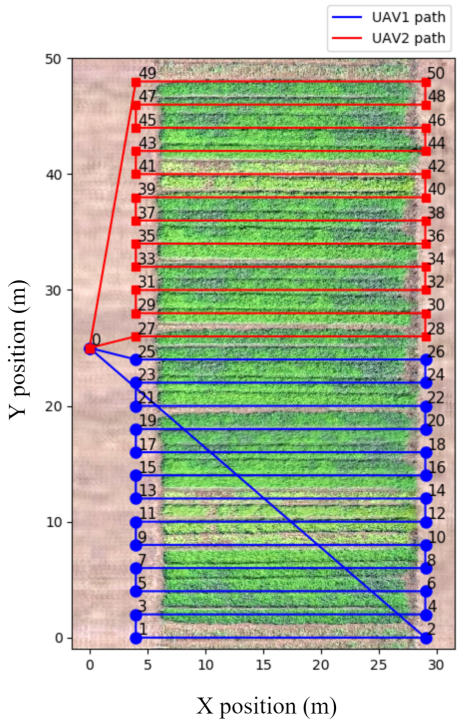


Figure 4.6: Simulation result showing the path planning outcome for two homogeneous drones.

In this section, a series of simulation results is provided to demonstrate the proposed methods and validate their effectiveness. Our proposed optimization problem was implemented in Python using MIP package to solve the underlying mixed integer linear program. In this setup, we assumed that there is a $20 \times 50\ m^2$ agricultural field including 32 plant rows. It is worth mentioning that the number of coverage rows is independent of the number of plant rows and is computed based on the width of field and the minimum footprint of the drones' cameras. Two drones are deployed from node 0 (defined as depot) and their mission is to fly

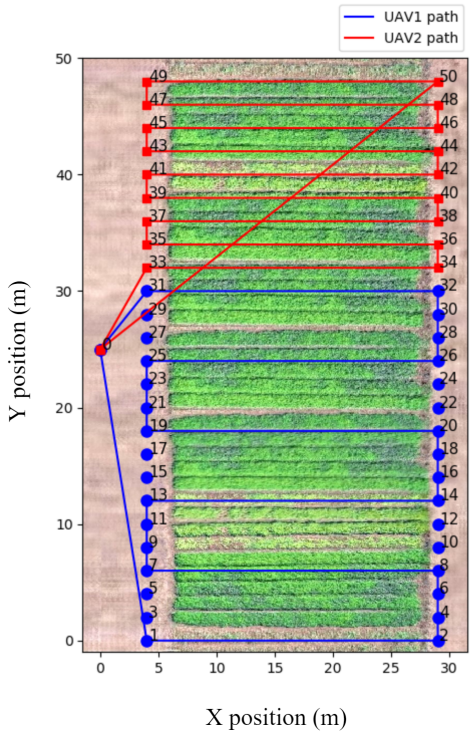


Figure 4.7: Simulation result demonstrating the path planning for two drones with different fields of view. First drone's footprint is assumed to be twice of that of the second drone, but its battery charge is half of the full state.

cooperatively over the field and collect image data. First, to convert the field into a weighted graph, we defined $L = 2\ m$ as the minimum of the cameras' footprint and then based on equations (4.3) and (4.4), the field with width of 50 m is divided into 25 rows arranged from node 1 to node 50, as presented in Fig. 4.6. Our proposed results are applied to the following three different scenarios.

In the *first scenario*, we consider two homogeneous drones. Indeed, it is assumed that they are equipped with cameras with the same properties, their bat-

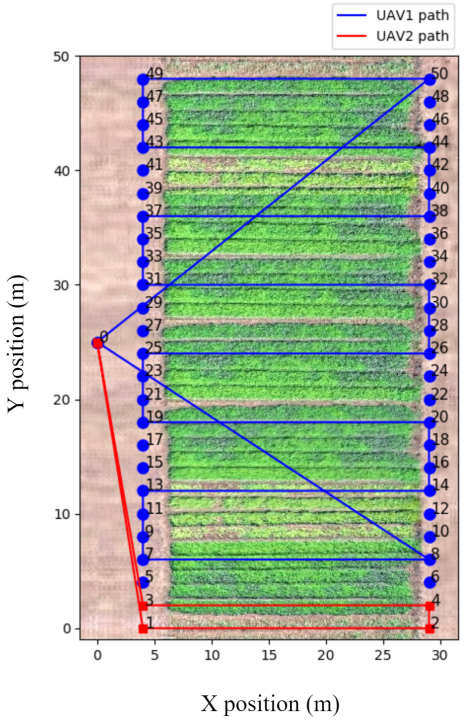


Figure 4.8: Simulation result showing the path planning for two drones with different fields of view, where the second drone's footprint is twice the first drone's but its battery charge is full.

teries are fully charged (at the beginning of the mission) and they both fly at the same speed. In this case, $\rho_1 = \rho_2 = 2\ m$ and as shown in Fig. 4.6, by solving the VRP for this setup, flight trajectories are generated for two drones with same characteristics leading to the filed being equally divided between the two drones to cover.

In the *second scenario*, it is assumed that the second drone has smaller camera's footprint than the first one ($L_1 = 2L_2 = 4\ m$), but its battery time duration is

two times larger than the first drone ($\tau_1 = \frac{1}{2}\tau_2 = 1500\ s$). When the first drone flies from node 8 to 7, considering its footprint ratio ($\rho_1 = 2$), it can monitor arcs $(5, 6)$ and $(9, 10)$ simultaneously. Fig. 4.7 shows flight trajectories for each drone based on the number of rows that they can monitor flying over each edge. As observed, the first drone is allocated and covers almost twice the field rows as the second drone.

In the *third scenario*, we assumed that the first drone that has a larger field of view is fully charged and its battery time duration is equal to the second drone. Fig. 4.8 indicates the flight routes for this scenario. As it is clear from this figure, most of the rows are assigned to the first drone (because of its large footprint ratio).

4.4 Conclusion and Future Work

In this paper, we formulated (and solved) the problem of optimal deployment of a team of heterogeneous drones in an agricultural field in such a way to minimize the coverage time. We considered a team of drones with different cameras' field of view, flight speeds and batteries' remaining charge. The field was first modeled as a graph with a set of nodes arranged considering the minimum of footprints. The underlying optimization problem was converted to a vehicle routing problem subject to a set of specific constraints enforcing vehicles to travel along the plant rows in a back and forth motion. The optimization problem was formulated as a mixed integer linear program (MILP) and the performance of the proposed VRP

solution was validated, using various simulation studies to find the, optimal flight path in a given agricultural field.

The authors are currently investigating a new approach to address the case with different flight heights for a team of drones, with the goal of finding the optimum cameras' footprint as a function of the height at which drones fly. The experimental validation (field study) of the proposed method is also left as a topic of our future work.

Chapter 5

A Reinforcement Learning-based Approach for Modeling and Coverage of an Unknown Field Using a Team of Ground Vehicles[1]

[1]Saba Faryadi and Javad Mohammadpour Velni. Submitted to *International Journal of Intelligent Systems*, 05/27/2020.

Abstract

In this work, a reinforcement learning-based method is presented for a team of unmanned ground vehicles (UGVs) to cooperatively learn an unknown dynamic field (and in particular, an agricultural field). The research problem here is to deploy autonomous vehicles to map plant rows, find obstacles, whose locations are not known a priory, and define regions of interest in the field (e.g., areas with high water stress). Once an environment model is built, the UGVs are then distributed in the field to provide full coverage of plants and update the reconstructed map simultaneously. Simulation results are finally presented to demonstrate that the proposed strategy can yield successful learning and monitoring of the coverage area and hence may be applied in practice to reduce human intervention in the field.

5.1 Introduction

During the last few years, the deployment of multi-robot systems in agricultural environments (both indoor and field) has received significant attention and is rapidly becoming reality even for large farms because of modern technologies and advancements in artificial intelligence (see [Duckett et al., 2018, Emmi et al., 2014, R Shamshiri et al., 2018] and references therein). In the very near future, a large number of farm workers will receive significant assistance from automated ground vehicles and drones [Vasconez et al., 2019]. There are also a number of studies focusing on multi-robot based distributed coverage, path planning and automated

navigation for agricultural applications [Davoodi et al., 2018, Faryadi et al., 2019a, Yun and Rus, 2014]. Although such collaboration can improve farm management and crop monitoring, the underlying systems are not fully autonomous due to the lack of a detailed map of the farm containing information of the plant rows and important areas. Hence, it is necessary to develop a system of crops-land mapping for employing mobile robots in smart farming. Weiss *et al.* in [Weiss and Biber, 2011] employed MEMS-based 3D LiDAR sensors to detect and segment plants and ground with the purpose of automating localization, mapping, and navigation for farm robots. As described in [Hiremath et al., 2014], the major limitation of vision-based methods is the sensitivity to ambient lighting conditions. To address this problem, Hiremath *et al.* developed a particle filter-based (PF) algorithm for autonomous navigation in a maize field [Hiremath et al., 2014].

Besides visual-based approaches, recently, the use of Q-learning, which is a powerful reinforcement learning technique, has been proposed in autonomous robotic systems, due to its self-training ability with no knowledge about the environment [Low et al., 2019]. The application of reinforcement learning and deep learning methods to train agents navigating autonomously is fundamental to their intelligent behavior. Konar *et al.* in [Konar et al., 2013] developed a Q-learning based approach for multi-robot path planning application. In comparison with the classical Q-learning, the proposed method in [Konar et al., 2013] is more efficient in terms of the total traveling time, number of visited states, and 90° turns required. In [Tai et al., 2017], a mapless motion planning method has been developed based on an asynchronous deep reinforcement learning method without any

predefined features.

There are several studies in the literature on applying reinforcement learning to solve the coverage problem for multiple unmanned aerial vehicles (UAVs) in an unknown environment. In [Xiao et al., 2020], authors proposed a solution to the non-optimal coverage problem of free dynamic area coverage using reinforcement learning. Pham *et al.* in [Pham et al., 2018] developed a multi-agent reinforcement learning (MARL) algorithm to enforce UAVs to learn cooperatively for fully covering an unknown field with the purpose of minimizing the overlaps among their fields of view.

Most of the algorithms developed for aerial vehicles are not useful for ground robots due to their limitations and differences. Authors in [Adepegba et al., 2016] established an area coverage control law using reinforcement learning methods by training multiple autonomous agents and showed that the control law would asymptotically converge to an optimal configuration. In [Bae et al., 2019], the authors proposed a method to combine deep Q-learning with convolutional neural networks (CNNs) to analyze the exact situation using image data on its environment and guided the robots to navigate based on the results extracted from the deep Q-learning based analysis.

One of the interesting challenges in mapless navigation in the field is how to cope with a dynamic environment that changes frequently. Authors in [Mirowski et al., 2016] presented a navigation approach in a complex environment using reinforcement learning in such a way that data efficiency and task performance are considerably improved. Yue *et al.* in [Yue et al., 2019] employed reinforcement

learning for multi-UAV sea area search map considering models of the environment, UAV dynamics, dynamics of goals, and sensor detection.

The main contribution of this work is in applying a learning-based method to enforce ground robots to navigate in a mapless dynamic agricultural field modeled as a complex maze (weighted graph). In this method, by defining appropriate reward and punishment for a given action at every single cell of the maze, at the end of the training process, agents build the field map and the corresponding graph. Then, a distributed coverage control algorithm is proposed to optimally deploy robots in the farm. The remainder of this paper is organized as follows. Section 2 describes the problem statement and methodology. Validation of the proposed method in a simulated agricultural environment will be discussed in Section 3, followed by concluding remarks in Section 4.

5.2 Problem Statement and Proposed Solution Method

In this article, we first employ a model-driven reinforcement learning (RL)-based method for deploying multiple UGVs in an unknown agricultural field to cooperatively build a model (graph) of the field including plants, obstacles, and important areas (goals) and update the model in real time while it changes. After the model is created (or updated in real time), the field graph is then generated for solving the problem of optimally distributing the team of UGVs in the coverage area.

5.2.1 Environmental model learning based on Dyna-Q+ algorithm

Most of the RL algorithms are model-free methods and applicable to problems where the environment's model is not available. Dyna-Q is a model-based RL algorithm that combines Q-learning and Q-planning, in which the planning is done online while the agents interact with (and hence explore) the environment to learn its model [Sutton and Barto, 2018]. In many applications, the main challenge in using Q-learning is its poor performance under unknown and dynamic environments, i.e., its lack of capability to update the environment model while it changes during the learning process. To address this problem, Dyna-Q+ method was developed as an improved Q-learning algorithm to accelerate the training process under unpredictable conditions [Sutton and Barto, 2018]. To apply RL for building a map of the field, let us consider a group of M ground vehicles moving at different speeds. The vehicles that are collectively responsible for exploring the environment are equipped with agricultural sensors and cameras for monitoring the plants growth and phenotypic traits. Considering sensors' footprint and the radius of their coverage as shown in Fig. 5.1, the field is divided into several states and modeled as an unknown *grid world*. In this work, $S = \{s_1, s_2, ..., s_n\}$ represents a set of n states of the grid world. During the learning process, each agent starts from one state s, interacts with neighboring cells by taking an action from a set of actions $A = \{a_{(up)}, a_{(down)}, a_{(left)}, a_{(right)}\}$ and then moves to the new state s' and receives a reward r.

In this work, the main goal is to enforce agents to find free states, the states

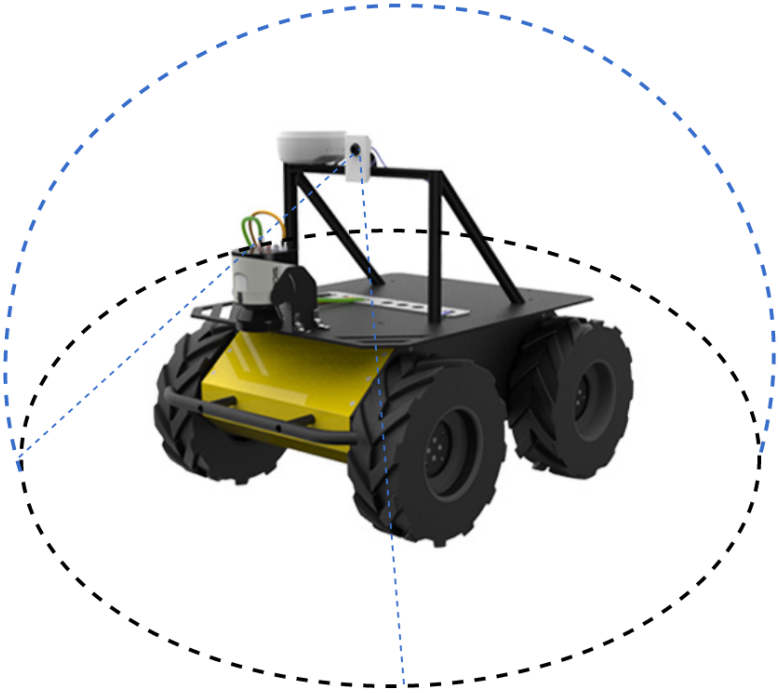


Figure 5.1: An illustration of visible area around a ground robot.

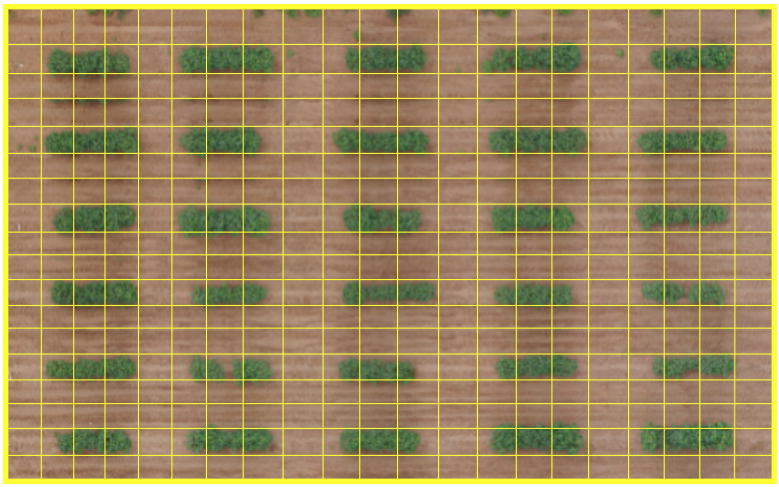


Figure 5.2: An example of a farm converted into the grid world.

blocked with obstacles (unexpected objects or plant rows) and goal states (regions of interest) aiming at creating the field's model and its graph. To this end, we assume one state as a terminal state $s_{terminal}$ which is the furthest from starting state to make sure that agents explore the whole field. Based on this definition, by taking action $a \in A$ in current state s, there are three different cases for mapping the pair of $(state, action)$ to $(next\ state, reward)$ pair as:

$$(s,a) \rightarrow \begin{cases} (s',1) & \text{if } s' = free, \\ (s,-1) & \text{if } s' = obstacle, \\ (s',10) & \text{if } s' = terminal. \end{cases} \tag{5.1}$$

As interpreted from this function, if taking action a in current state s maps agent to a free state, it moves towards the next state s' and gets reward $r = 1$, but if (s, a) results in moving towards a blocked state (obstacle or plant), the agent stays in the current state s and would be punished by receiving reward $r = -1$. In the last scenario, if the agent reaches the terminal state $s_{terminal}$ by choosing action a in state s, it moves towards the terminal state and would be rewarded with $r = 10$.

To learn a model of the maze (field) using a team of agents, inspired by the method presented in [Kivelevitch and Cohen, 2010], we define the problem as follows: *Deploy a group of M agents to first cooperatively explore the environment and then, by experiencing different episodes, determine their path towards the terminal state.* It is expected that after playing a few episodes ending at terminal state, agents can build a model of the field.

To share tasks, avoid collisions between agents and make the training process as fast as possible, first, it is assumed that there is a plain maze with no goal or obstacle. Then, considering the current state of each agent (starting state) and its speed, all states are divided among the agents similar to a simple Voronoi diagram. In the next step, they start exploring the states of their Voronoi sub-states, and in each step, agents update their sub-states and share the sub-model ($Model(s, a) \leftarrow (s', r)$) with other agents.

Next, we describe the model-based Q-learning method, Dyna-Q+, which is developed by combining simple Q-learning and Q-planning algorithms (see Fig. 5.3) and which is an improved Dyna-Q method that is applicable for a complex maze [Sutton and Barto, 2018]. In this method, first, each agent chooses an action in state s based on $\epsilon - greedy$ method to achieve a trade-off between exploration and exploitation in such a way that for a small ϵ ($0 \leq \epsilon \leq 1$), agent picks an action at random for a portion ϵ (exploration) or selects the best action for a portion $1 - \epsilon$ (exploitation).

To cope with stochastic environments whose models may be inaccurate, agents need to experience new behaviors (actions) in previously observed states when the maze is changing. In Dyna-Q+ algorithm, the agent records the last time that each ($state$, $action$) was experienced, and the longer time after experiencing that pair of state-action means the greater possibility of changing the dynamics of this pair and the corresponding model. Dyna-Q+ methods encourage agents to try actions that were not experienced for a long time by adding a special bonus on reward as a function of how many time steps T the state-action pair was last tried.

Then, the total reward for a transition is calculated as follows:

$$R = r + k\sqrt{T}, \tag{5.2}$$

where r is the transition reward based on the function defined in (5.1) and k is a small number defined as a time weight parameter. After executing chosen action a at state s, based on the agent's observation, a model associated with this pair of state-action is recorded as:

$$Model(s, a) \leftarrow (s', R). \tag{5.3}$$

The main step in Q-learning is to find the best action (optimal policy) that can maximize the total reward (see Algorithm 1). In other words, at each step, the values for state-action pairs are updated in a Q-table until it converges to the maximum value. A simple update rule for Q value is:

$$Q(s, a) \leftarrow Q(s, a) + \beta \left[R + \gamma \max_a \Big(Q(s', a) - Q(s, a) \Big) \right]. \tag{5.4}$$

where β is the learning rate and $\gamma \in [0, 1]$ is a discount factor adjusted for striking a balance between future reward and immediate reward received by acting at each state.

As shown in Fig. 5.3, in Dyna-Q algorithm, agents interact with the environment, update Q table and learn its model through the real experience. Then, this model is used to generate simulated experiences for planning step that is repeated

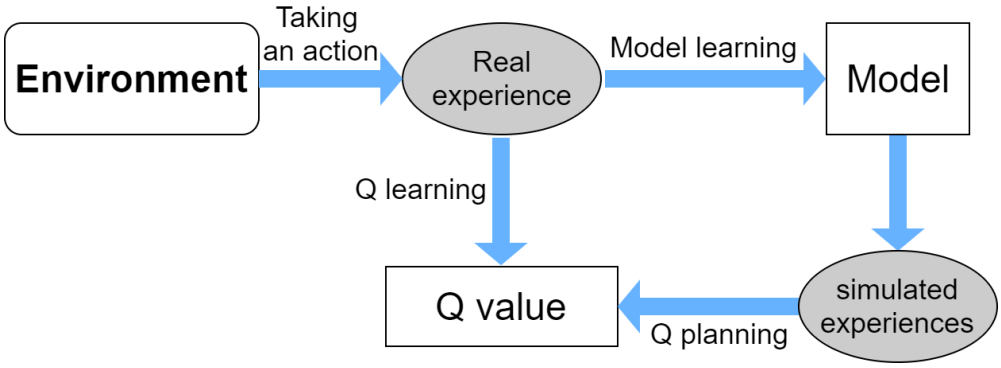


Figure 5.3: Relationship among learning, planning and acting in Dyna-Q+ algorithm.

and Q values are updated throughout both direct RL and planning updates. At the end of each learning step, the recorded model is shared among all agents and also updated when solving the coverage control problem discussed in the next section.

5.2.2 Coverage control algorithm

After searching the grid world and experiencing a number of episodes, agents build a model of maze (field) with N states. Next, the field model is converted into a weighted graph $G(S, E, C)$, where $S \in \{1, 2, ..., N\}$ is the set of states, $E \subseteq |S| \times |S|$ denotes the set of edges whose elements indicate whether or not there is a path between the center points of each pair of states. Furthermore, C is the weight matrix representing metric lengths of the edges [Alitappeh et al., 2017]. In this work, the main purpose of finding the field graph and important states is to

optimally distribute UGVs over the coverage area by minimizing the corresponding cost function. To achieve this goal, since UGVs may move at different speeds, we formulate the cost as a function of traveling time defined as follows:

$$t_{(s_{p_i},s_q)} = \frac{d_{(s_{p_i},s_q)}}{v_i}, \tag{5.5}$$

where v_i denotes the speed of the ith vehicle, and $d_{(s_{p_i},s_q)} \in C$ is the Euclidean distance between the ith vehicle's current state p_i and state q. Moreover, by knowing the optimal path starting from state p_i and ending at state s, total optimal traveling time for each vehicle is the sum of the travel times (costs) from its current state p_i to state s. In this work, the shortest path between each pair of states is computed using Dijkstra's algorithm [Dijkstra, 1959] and then by knowing $path = \{p, q, ..., r, s\}$, the total time τ is:

$$\tau_{(s_p,s_s)} = t_{s_p,s_q} + ... + t_{s_r,s_s}. \tag{5.6}$$

To minimize the corresponding cost function, after finding the minimum time between each pair of states, the field is partitioned into M Voronoi sub-graphs g_{r_i} for $i \in \{1, 2, .., M\}$ to share tasks among M vehicles proportionally. To this end, based on the Lloyd's algorithm [Alitappeh et al., 2017], the optimal Voronoi diagram g_{r_i} for ith vehicle is a partition of the area calculated as:

$$g_{r_i} = \{s_q \in S \mid \tau_{(s_{p_i},s_q)} \leq \tau_{(s_{p_j},s_q)} \ , \forall i \neq j\}. \tag{5.7}$$

Note that, in this work, it is assumed that all vehicles are initially fully charged and remain sufficiently charged for completing their assigned task.

Using the result of the Voronoi partitioning, the ith vehicle is responsible for covering the states (associated graph vertices) in its sub-graph g_{r_i}. Then, as explained in [Alitappeh et al., 2017, Davoodi et al., 2018], the total cost is formulated as follows:

$$H(p, g_r) = \sum_{i=1}^{M} \sum_{q \in g_{r_i}} \tau_{(s_{p_i}, s_q)} \phi(q), \tag{5.8}$$

where $\phi(q)$ is the priority value corresponding to state q [Faryadi et al., 2019a]. As the field is converted into a graph, the goal states are given higher priority values and lowest values are assigned to states that are far from the important states. Then, the total traveling time (cost) will be minimized, and an optimal solution is reached only when the distance between the current state of the vehicle and the goal state $d_{(s_{p_i}, s_q)}$ converges to zero.

5.3 Simulation Results and Discussion

In this section, we show the results of validating our proposed method for learning and monitoring important areas in an agricultural field through simulation studies. A small farm sized 18×21 m^2 is considered as an unknown environment and simulated as a maze (see Fig. 5.2). Based on the field size, in all scenarios, we assumed that two vehicles (agents) with different speeds are deployed to explore the environment and extract the field model and then use it to solve the coverage

Table 5.1: Algorithm 1: Implementation of the proposed method.

Input: Field dimension, sensors' footprint, agents' speed, learning parameters
Output: Environmental model of the field, field graph, Voronoi sub-states, optimal paths
Learning process:

1. Receive locational information of vehicles
2. Convert the field into a grid world (cells)
3. Compute the Voronoi sub-states for each vehicle
4. Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in S$, $a \in A$
5. **While** steps $\leq$ maximum steps:
   for $episode = 1, 2, ...$ **do**:
   $s \leftarrow$ current non-terminal state
   $a \leftarrow \epsilon - greedy\ (s, Q)$
   $(s', r) \leftarrow$ execute action a
   $T_{(step)} \leftarrow T_{(step-1)} + 1$ for all non-visited states
   $R \leftarrow r + k\sqrt{T}$
   $Q(s,a) \leftarrow Q(s,a) + \beta\Big[R + \gamma \max_a \Big(Q(s',a) - Q(s,a)\Big)\Big]$
   $Model(s,a) \leftarrow (s', R)$
   Update Voronoi diagram based on learned model and current state
   for $n =$ sampling steps repeat:
   $s \leftarrow$ previously observed state
   $a \leftarrow$ random action taken previously in state s
   $(s', R) \leftarrow Model(s,a)$.
   $Q(s,a) \leftarrow Q(s,a) + \beta\Big[R + \gamma \max_a \Big(Q(s',a) - Q(s,a)\Big)\Big]$
   end while

Coverage control

6. **Generate** weighted graph of field G
7. **Set** the edges that end to an obstacle to zero
8. **Find** shortest path between center points of each pair of cells.
9. **Compute** initial Voronoi regions for each vehicle
10. **While** cost function has not converged:
    if there is a neighboring cell u **so that**
    $H(u, g_i)$ is minimum in $g(r_i)$, **then:**
    Set u as the next best point
    Move towards state u
    if state u is blocked:
    Update field model and Voronoi diagram.
    Repeat step 10.
    end if
    end if
    end while

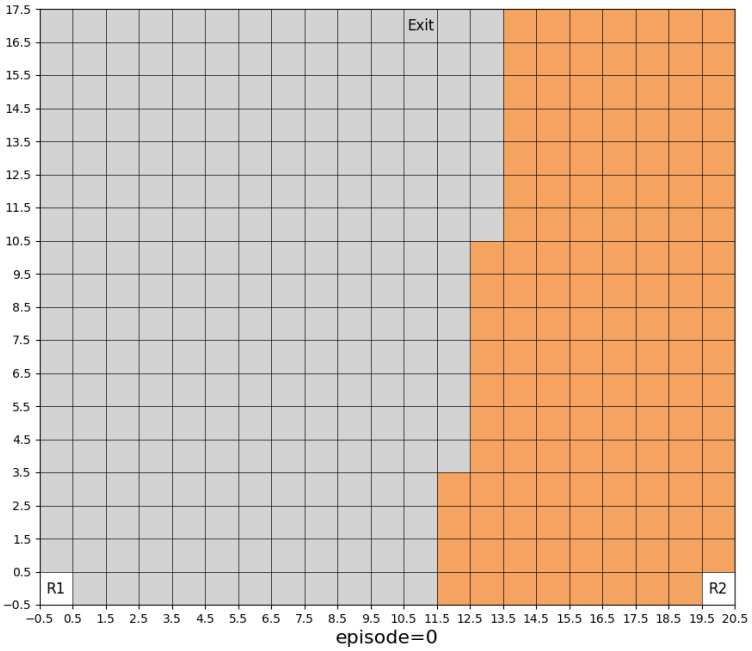
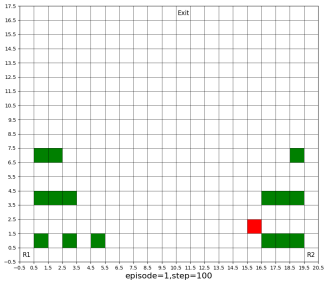
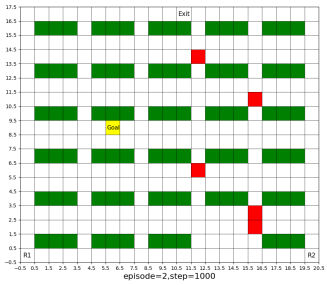


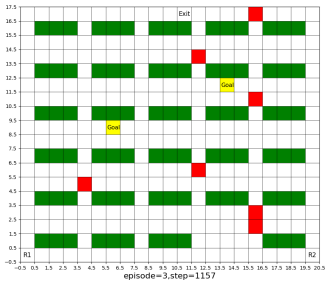
Figure 5.4: Voronoi diagram before the model learning begins in both scenarios.



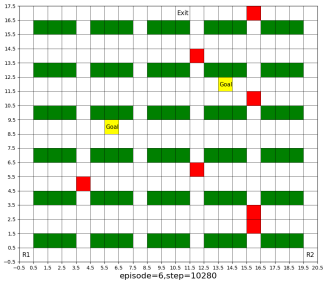
(a) Learned model 100 steps into the first episode.



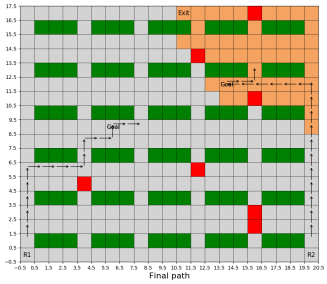
(b) Learned model 1,000 steps into the second episode.



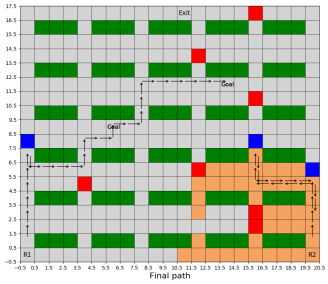
(c) Learned model at the end of the third episode (step=1,157).



(d) Learned model at the end of the last episode (step=10,280).



(e) Final model of maze after 20,000 steps. Black arrows indicate trajectories of agents towards goal states if offline learned model of the environment is used for coverage.



(f) Final model of maze after 20,000 steps assuming that some states (blue cells) have changed while coverage problem is being solved.

Figure 5.5: Plots show simulation results for the first scenario. The starting states for the two agents are shown by R1 and R2, and the terminal state is shown by Exit. Gray cells are Voronoi sub-states of the first agent, and the orange cells are those of the second agent. There are three types of obstacles found during learning and coverage process: green cells which represent plant rows, red cells representing unknown objects blocking cells/paths detected during learning process and blue cells representing new obstacles found while coverage problem is solved in real time. Subplot (e) illustrates the final Voronoi diagrams for the case that no obstacle is added while coverage problem is being solved. In comparison, subplot (f) shows final Voronoi partitions and trajectories for the case that maze changes and new obstacles (blue cells) are added (and hence learned online). This leads the vehicles to alter their directions to find free paths towards the goals and the centers of their partitions.

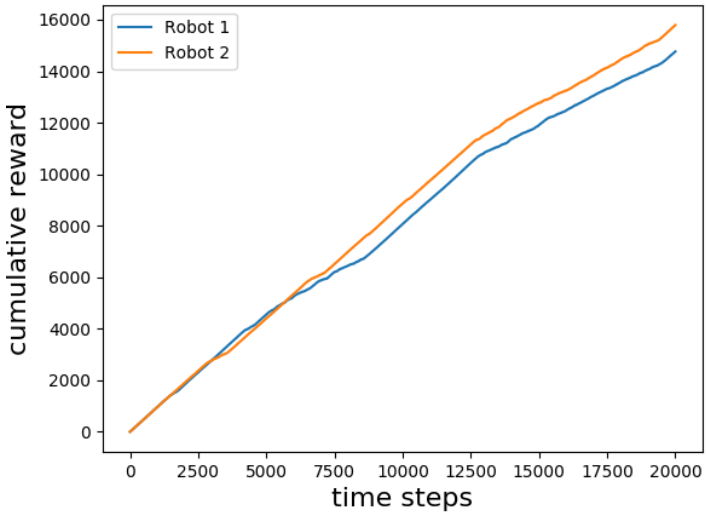
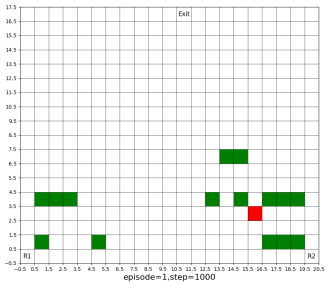


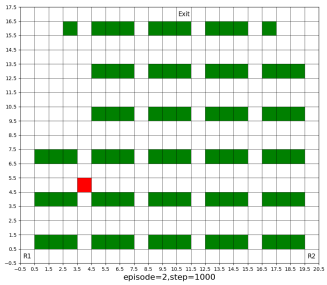
Figure 5.6: Cumulative rewards for first scenario after 20,000 steps. The agents are enforced to use past experience to explore which actions lead to higher cumulative rewards.

problem. Both agents are equipped with the same sensors to cover the area and observe their neighboring cells. All steps including the simulation of the field as a grid world (maze), learning process and coverage problem are implemented in *Python* programming language. In order to find the size of states, we assumed that the optimum sensing radius (Fig. 5.1) is 0.5 m (an arbitrary value) so each state covers $1 \times 1 = 1\ m^2$ and hence the field is converted into a maze with $18 \times 21 = 378$ cells.

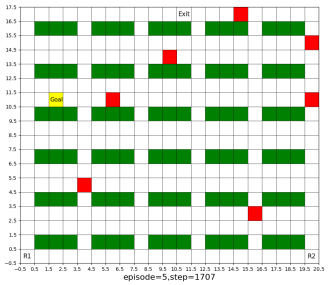
Here, two scenarios are considered for the validation of the proposed method. In both scenarios, first agent gets into the simulated field from the bottom left corner of the maze where the corresponding cell is $[0, 0]$, and second agent starts



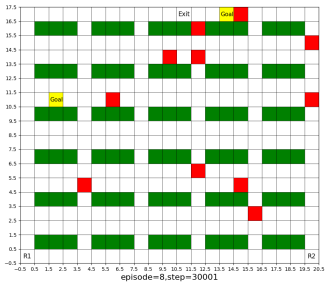
(a) Learned model 1,000 steps into the first episode.



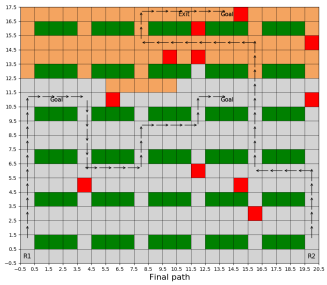
(b) Learned model 1,000 steps into the second episode.



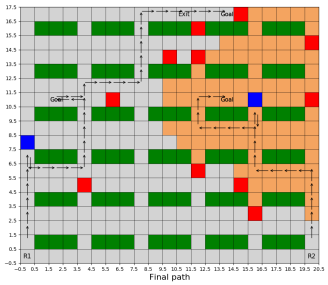
(c) Learned model at the end of the fifth episode (step=1,707).



(d) Learned model at the end of the last episode (step=30,001).



(e) Final model of the maze after 30,000 steps. Black arrows indicate final trajectories of agents towards goal states.



(f) Final model of the maze after 30,000 steps. As observed, some states (blue cells) have changed while coverage problem is being solved.

Figure 5.7: Plots show simulation results for the second scenario. In this experiment, three regions of interest (goals) are found during the learning process. Plot (e) illustrates the final Voronoi diagrams and vehicles' trajectories to reach the goals or the center of their Voronoi partitions while the maze is static after learning the field's model. In contrast, plot (f) shows final partitioning and trajectories for the case that field changes while the coverage problem is being solved. In this case, second vehicle that is slower moves towards (and covers) one of the goals while the first vehicle finds an optimal path to monitor two other goals. As observed from the plots (e) and (f), with the addition of two obstacles, both assigned tasks and trajectories have changed. It is noted that in this scenario, we enforce the agents to stop moving once all the goal states are visited.

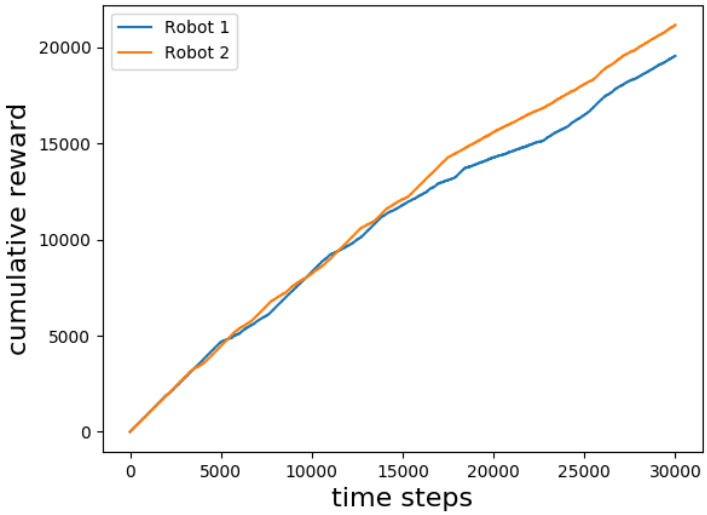


Figure 5.8: Cumulative rewards for the second scenario after 30,000 steps.

from the bottom right corner which is state $[0, 20]$. To solve the reinforcement learning problem as an episodic problem and for enforcing agents to explore the whole maze, state $[17, 11]$ is considered as terminal (exit) state (see Fig. 5.4). In each episode, agents begin experiencing from starting cell and find their trajectories towards the exit cell. For limiting total experience steps, considering the maze's size and its complexity (the number of unpredictable obstacles and goals), a maximum threshold for steps is defined, which means that when the cumulative steps of all episodes exceeds the maximum step, the learning process terminates.

One of the key factors to achieve reliable results from Q-learning and Q-planning steps is the selection of the best set of learning parameters. To this end, all of the parameters are chosen by trial and error to calibrate for the best

results. For learning rate and discount factor described earlier (that appear in (5.4)), we chose $\beta = 0.7$ and $\gamma = 0.95$, respectively. Furthermore, $\epsilon = 0.1$ was considered as a probability for exploration and the time weight in (5.2) was set to $k = 10^{-2}$. As learning the model of a dynamic field is one of the main objectives of this work, we assumed that maze is changing at some steps meaning that some blocked cells are added or removed and some cells are defined as important states during the learning process. Next, we describe each scenario and demonstrate and discuss associated results.

Scenario 1: In this scenario, first agent is assumed to be faster than the second one and moves at the speed of $v_1 = 20\ cm/s$ while second one moves at the speed of $v_2 = 15\ cm/s$. It is assumed that agents know only the number of cells and have no knowledge about the states blocked with plant rows or obstacles, and goal states. As shown in Fig. 5.4, before starting the first episode, agents solve Voronoi partitioning problem described earlier and the field cells are divided between two ground vehicles considering their speeds. Then, first episode starts and vehicles try to find their path to the exit state. As agents are exploring the maze for a free path, they also update the model of the field in real time. It is noted that the number of steps at each episode is not fixed and an episode ends when agents visit the exit state.

Fig. 5.5 illustrates the results of simulating the first scenario. The maximum running step is set to 20,000 steps, and as observed from the results, agents find the environment model after experiencing six episodes. In this setup, during the learning process, the maze changes, and some obstacles or goals are added

at random running steps. In this simulated environment, only two important regions are considered and the corresponding cells are assigned to the goal states for further investigation. Then, the field graph is generated using its model, and agents are deployed in the field to make the best coverage over the regions of interest (see Fig. 5.5e and Fig. 5.5f). As observed from Fig. 5.5e, first, it is assumed that the maze does not change after learning its model and no obstacle is added while agents are finding the next best points to move to. Therefore, each agent moves towards the closest important area to have better coverage over the field. In the second case illustrated in Fig. 5.5f, the maze alters while coverage problem is being solved, and agents find new obstacles (blue cells) along their route, update field's graph and Voronoi diagrams, and hence change their paths. In this case, although one of the goals is close to the first vehicle and the other one is further away from it, both goals are eventually covered by the first vehicle because the second vehicle's path to reach the target is blocked by several obstacles. Nevertheless, second vehicle moves towards the state at the center of its Voronoi diagram (i.e., the centroid) to minimize the corresponding cost function. In other words, 5.5e shows, each agent was initially tasked to cover one goal, but because of the new obstacles and hence re-learning of the field in real time, first agent ended up covering both goals. Another important point here is that once the goal states are covered, the agents can either stop moving or move towards the center of their Voronoi partitions – in this scenario, we considered the latter.

Scenario 2: Similar to the first scenario, we assume that $v_1 = 20\ cm/s$ and $v_2 = 15\ cm/s$, and hence the initial Voronoi diagrams for both agents are the same

as those shown in Fig. 5.4. The maze is assumed to change more often during the learning process and hence more complex compared to the simulated maze in the first scenario. For this reason, the maximum step is set to 30,000 iterations which is long enough to build the field model. Subplots in Fig. 5.7 illustrate the results of the second scenario simulations. It is shown that after running 8 episodes, the environment model is learned and agents solve the coverage problem while the maze is changing; they also update the corresponding graph in real time (Fig. 5.7f). Moreover, as discussed in section II, our main purpose in using Q learning is to maximize the total reward which enforces agents to choose the best action. To this end, as shown in Fig. 5.6 and Fig. 5.8, the cumulative rewards increase in both scenarios.

5.4 Conclusion

In this paper, a cooperative learning approach was presented to model a dynamic environment/field (in which multiple agents are present) and then optimally distribute agents in the field for coverage and data collection. Using Dyna-Q+ algorithm (a model-based reinforcement learning method), a reward and punishment function was designed to enforce vehicles to find exit state and learn the environment at the same time. The extended (cooperative) learning method introduced and formulated here for multi-agent systems used the Voronoi partitioning. The proposed approach eventually led to dividing the search space among the agents while they could share their observations through a communication net-

work. Extensive simulation results were presented to demonstrate the efficacy of the proposed model learning and planning approach.

Chapter 6

CONCLUSIONS & FUTURE WORK

This thesis focused on developing discrete coverage control algorithms and reinforcement learning-based methods in precision agriculture to deploy multiple heterogeneous unmanned ground and aerial vehicles in a farm to collect data. Throughout this study, it was assumed that both ground and aerial vehicles are equipped with essential sensors and devices to monitor the field and execute agricultural tasks.

The second chapter's main contribution is in formalizing the cooperation between multiple ground vehicles for field coverage and precisely monitoring the critical areas of a farm. The field is converted into a directed graph consisting of a set of nodes and edges. For obstacle avoidance, if an object is blocking the path between two nodes, the corresponding edge is set to zero. After solving the cover-

age problem, the ground robots are distributed over the field graph based on the Voronoi partitioning and move towards the goals by minimizing the corresponding cost function.

In the third chapter, a new strategy is integrated with the coverage control method for path planning and monitoring multiple targets in a controlled environment. Assuming that the ground robot has a right-sided camera, the field graph is directed, which forces robots to move in a specific direction and take appropriate images. We also define a task graph to find the order of missions and the optimal trajectory. Moreover, a real-time data processing method is applied to reconstruct the image of whole plant rows and detect early changes in their physical characteristics. For both chapters 2 and 3, simulations and experimental results are provided to prove the reliability of the proposed methods and demonstrate that they can be applied in the real world.

In the fourth chapter, unlike the second and third ones, the collaboration among multiple drones is studied in such a way that they can cover the whole field with a minimum traveling time. It is assumed that drones fly at a fixed height from the ground, and they differ in terms of flight speed, fields of view, and battery capacity. In this section, a sweep direction is chosen perpendicular to the plant rows, and the distance between coverage rows is computed based on the minimum length of cameras' footprints for generating the field's graph. Then, the optimization problem is formulated as a vehicle routing problem with specific constraints to minimize the total flight time.

It is noted that in chapters 2, 3 and 4, the agricultural field is predefined,

and vehicles know the locational information of plant rows, obstacles, and goals. However, in the fifth chapter, it is assumed that the field is unknown, and its environmental model changes frequently. The field is converted into a grid world, and a reinforcement learning-based method (namely, Dyna-Q+) is applied to enforce agents experiencing some episodes and find the field's model. In this chapter, a team of ground vehicles is distributed to learn the environmental model of the area in real time and make the best coverage while updating the learned model. The problem of detecting undefined objects in the robots' trajectories (chapter 2) is addressed in the fifth chapter.

Future work will focus on the cooperation between ground and aerial vehicles and task allocation between them to address their different capabilities and agricultural operations. Drones are faster than ground vehicles (operating in the farm), and they can fly everywhere regardless of obstacles. They can provide images from the top side of the field, while the ground units can capture images from different sides of plants. Therefore, the collaboration of drones and ground robots may benefit farmers for various applications. Another line of work relates to extending the experimental results of this work to monitor large farms/greenhouses consisting of a significant number of plant rows. Furthermore, the deployment of high-performance robots with powerful onboard processing systems which allow faster real-time model learning and data processing (and eventually decision making) will be a promising subject area.

Bibliography

https://www.irobot.com/.

https://marvelmind.com/.

A. A. Adepegba, S. Miah, and D. Spinello. Multi-agent area coverage control using reinforcement learning. In *The Twenty-Ninth International Flairs Conference*, 2016.

A. Ahmadzadeh, J. Keller, G. Pappas, A. Jadbabaie, and V. Kumar. An optimization-based approach to time-critical cooperative surveillance and coverage with UAVs. In *Experimental Robotics*, pages 491–500. Springer, 2008.

T. Ahonen, R. Virrankoski, and M. Elmusrati. Greenhouse monitoring with wireless sensor network. In *IEEE/ASME International Conference on Mechtronic and Embedded Systems and Applications*, pages 403–408, Oct 2008.

R. J. Alitappeh, G. A. Pereira, A. R. Araújo, and L. C. Pimenta. Multi-robot deployment using topological maps. *Journal of Intelligent & Robotic Systems*, 86(3-4):641–661, 2017.

G. S. Avellar, G. A. Pereira, L. C. Pimenta, and P. Iscold. Multi-UAV routing for area coverage and remote sensing with minimum time. *Sensors*, 15(11): 27783–27803, 2015.

M. Ayala, C. Soria, and R. Carelli. Visual servo control of a mobile robot in agriculture environments. *Mechanics based design of structures and machines*, 36(4):392–410, 2008.

H. Bae, G. Kim, J. Kim, D. Qian, and S. Lee. Multi-robot path planning method using reinforcement learning. *Applied Sciences*, 9(15):3057, 2019.

T. Bakker, K. van Asselt, J. Bontsema, J. Müller, and G. van Straten. Autonomous navigation using a robot platform in a sugar beet field. *Biosystems Engineering*, 109(4):357–368, 2011.

A. Balafoutis, B. Beck, S. Fountas, J. Vangeyte, T. V. d. Wal, I. Soto, M. Gómez-Barbero, A. Barnes, and V. Eory. Precision agriculture technologies positively contributing to ghg emissions mitigation, farm productivity and economics. *Sustainability*, 9(8):1339, 2017.

A. Barrientos, J. Colorado, J. d. Cerro, A. Martinez, C. Rossi, D. Sanz, and J. Valente. Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots. *Journal of Field Robotics*, 28(5):667–689, 2011.

K. Benke and B. Tomkins. Future food-production systems: vertical farming

and controlled-environment agriculture. *Sustainability: Science, Practice and Policy*, 13(1):13–26, 2017.

J. A. Berni, P. J. Zarco-Tejada, L. Suárez, and E. Fereres. Thermal and narrowband multispectral remote sensing for vegetation monitoring from an unmanned aerial vehicle. *IEEE Transactions on Geoscience and Remote Sensing*, 47(3): 722–738, 2009.

S. Bhandari, A. Raheja, R. L. Green, and D. Do. Towards collaboration between unmanned aerial and ground vehicles for precision agriculture. In *Autonomous Air and Ground Sensing Systems for Agricultural Optimization and Phenotyping II*, volume 10218, page 1021806. International Society for Optics and Photonics, 2017.

S. Bhattacharya, M. Likhachev, and V. Kumar. Multi-agent path planning with multiple tasks and distance constraints. In *IEEE International Conference on Robotics and Automation*, pages 953–959, May 2010.

M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, Aug 2007.

A. Camilli, C. E. Cugnasca, A. M. Saraiva, A. R. Hirakawa, and P. L. Corrêa. From wireless sensors to field mapping: Anatomy of an application for precision agriculture. *Computers and Electronics in Agriculture*, 58(1):25–36, 2007.

S. Candiago, F. Remondino, M. De Giglio, M. Dubbini, and M. Gattelli. Evalu-

ating multispectral images and vegetation indices for precision farming applications from UAV images. *Remote Sensing*, 7(4):4026–4047, 2015.

W. W. Casady and H. L. Palm. Precision agriculture: remote sensing and ground truthing. 2002.

H. Chao, M. Baumann, A. Jensen, Y. Chen, Y. Cao, W. Ren, and M. McKee. Band-reconfigurable multi-UAV-based cooperative remote sensing for real-time water management and distributed irrigation control. *IFAC Proceedings Volumes*, 41(2):11744–11749, 2008.

J. Chen, X. Zhang, B. Xin, and H. Fang. Coordination between unmanned aerial and ground vehicles: A taxonomy and optimization perspective. *IEEE transactions on cybernetics*, 46(4):959–972, 2015.

J. Conesa-Muñoz, J. M. Bengochea-Guevara, D. Andujar, and A. Ribeiro. Efficient distribution of a fleet of heterogeneous vehicles in agriculture: a practical approach to multi-path planning. In *2015 IEEE International Conference on Autonomous Robot Systems and Competitions*, pages 56–61. IEEE, 2015.

M. Davoodi, J. M. Velni, and C. Li. Coverage control with multiple ground robots for precision agriculture. *Mechanical Engineering Magazine Select Articles*, 140 (06):S4–S8, 2018.

F. De Rango, N. Palmieri, M. Tropea, and G. Potrino. Uavs team and its application in agriculture: A simulation environment. *SIMULTECH*, 2017:374–379, 2017.

E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

T. Duckett, S. Pearson, S. Blackmore, B. Grieve, W.-H. Chen, G. Cielniak, J. Cleaversmith, J. Dai, S. Davis, C. Fox, et al. Agricultural robotics: the future of robotic agriculture. *arXiv preprint arXiv:1806.06762*, 2018.

L. Emmi, M. Gonzalez-de Soto, G. Pajares, and P. Gonzalez-de Santos. New trends in robotics for agriculture: integration and assessment of a real fleet of robots. *The Scientific World Journal*, 2014, 2014.

B. Englot and F. Hover. Multi-goal feasible path planning using ant colony optimization. In *IEEE International Conference on Robotics and Automation*, pages 2255–2260, May 2011.

H. Ergezer and K. Leblebicioğlu. 3D path planning for multiple UAVs for maximum information collection. *Journal of Intelligent & Robotic Systems*, 73(1-4): 737–762, 2014.

S. Faryadi, M. Davoodi, and J. Mohammadpour Velni. Agricultural field coverage using cooperating unmanned ground vehicles. In *Dynamic Systems and Control Conference*, volume 59155, page V002T25A003. American Society of Mechanical Engineers, 2019a.

S. Faryadi, M. Davoodi, and J. Mohammadpour Velni. Autonomous real-time monitoring of crops in controlled environment agriculture. In *ASME 2019 Dy-*

namic Systems and Control Conference. American Society of Mechanical Engineers Digital Collection, 2019b.

R. Gebbers and V. I. Adamchuk. Precision agriculture and food security. *Science*, 327(5967):828–831, 2010.

P. Gonzalez-de Santos, A. Ribeiro, C. Fernandez-Quintanilla, F. Lopez-Granados, M. Brandstoetter, S. Tomic, S. Pedrazzi, A. Peruzzi, G. Pajares, G. Kaplanis, et al. Fleets of robots for environmentally-safe pest control in agriculture. *Precision Agriculture*, 18(4):574–614, 2017.

F. Guerriero, R. Surace, V. Loscri, and E. Natalizio. A multi-objective approach for unmanned aerial vehicle routing problem with soft time windows constraints. *Applied Mathematical Modelling*, 38(3):839–852, 2014.

H. Hagras, M. Colley, V. Callaghan, and M. Carr-West. Online learning and adaptation of autonomous mobile robots for sustainable agriculture. *Autonomous Robots*, 13(1):37–52, 2002.

S. A. Hiremath, G. W. Van Der Heijden, F. K. Van Evert, A. Stein, and C. J. Ter Braak. Laser range finder model for autonomous navigation of a robot in a maize field using a particle filter. *Computers and Electronics in Agriculture*, 100:41–50, 2014.

W. H. Huang. Optimal line-sweep-based decompositions for coverage algorithms. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 1, pages 27–32. IEEE, 2001.

I. Ion, I. Simionescu, and A. Curaj. Mobil mechatronic system with applications in agriculture and sylviculture. *IFAC Proceedings Volumes*, 35(2):865–870, 2002.

F. Karim, F. Karim, et al. Monitoring system using web of things in precision agriculture. *Procedia Computer Science*, 110:402–409, 2017.

S. Kitamura and K. Oka. Recognition and cutting system of sweet pepper for picking robot in greenhouse horticulture. In *IEEE International Conference Mechatronics and Automation*, volume 4, pages 1807–1812, 2006.

E. H. Kivelevitch and K. Cohen. Multi-agent maze exploration. *Journal of Aerospace Computing, Information, and Communication*, 7(12):391–405, 2010.

A. Konar, I. G. Chakraborty, S. J. Singh, L. C. Jain, and A. K. Nagar. A deterministic improved q-learning for path planning of a mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(5):1141–1153, 2013.

A. Lalwani, M. Bhide, and S. K. Shah. A review: Autonomous AGRIBOT for smart farming. *International Journal of Industrial Electronics and Electrical Engineering*, 4(2):50–53, 2016.

W. Li and C. G. Cassandras. Centralized and distributed cooperative receding horizon control of autonomous vehicle missions. *Mathematical and computer modelling*, 43(9-10):1208–1228, 2006.

L. Liu and N. Michael. Energy-aware aerial vehicle deployment via bipartite graph

matching. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 189–194. IEEE, 2014.

E. S. Low, P. Ong, and K. C. Cheah. Solving the optimal path planning of a mobile robot using improved q-learning. *Robotics and Autonomous Systems*, 115:143–161, 2019.

H. Luo, Y. Niu, M. Zhu, X. Hu, and H. Ma. Optimization of pesticide spraying tasks via multi-uavs using genetic algorithm. *Mathematical Problems in Engineering*, 2017, 2017.

M. Maharlooei, S. Sivarajan, S. G. Bajwa, J. P. Harmon, and J. Nowatzki. Detection of soybean aphids in a greenhouse using an image processing technique. *Computers and Electronics in Agriculture*, 132:63–70, 2017.

A. Mandow, J. Gomez-de Gabriel, J. L. Martinez, V. F. Munoz, A. Ollero, and A. Garcia-Cerezo. The autonomous mobile robot aurora for greenhouse operation. *IEEE Robotics & Automation Magazine*, 3(4):18–28, 1996.

N. Michael, J. Fink, and V. Kumar. Controlling a team of ground robots via an aerial robot. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 965–970. IEEE, 2007.

P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

U. R. Mogili and B. Deepak. Review on application of drone systems in precision agriculture. *Procedia computer science*, 133:502–509, 2018.

T. P. Nascimento, A. P. Moreira, and A. G. S. Conceição. Multi-robot nonlinear model predictive formation control: Moving target and target absence. *Robotics and Autonomous Systems*, 61(12):1502–1515, 2013.

P. Nolan, D. A. Paley, and K. Kroeger. Multi-UAS path planning for non-uniform data collection in precision agriculture. In *Aerospace Conference, 2017 IEEE*, pages 1–12. IEEE, 2017.

H. X. Pham, H. M. La, D. Feil-Seifer, and A. Nefian. Cooperative and distributed reinforcement learning of drones for field coverage. *arXiv preprint arXiv:1803.07250*, 2018.

L. C. Pimenta, V. Kumar, R. C. Mesquita, and G. A. Pereira. Sensing and coverage for a network of heterogeneous robots. In *2008 47th IEEE conference on decision and control*, pages 3947–3952. IEEE, 2008.

O. A. Postolache, P. d. S. Girão, J. M. C. D. Pereira, C. Grueau, H. Teixeira, and M. Leal. Greenhouses microclimate real-time monitoring based on a wireless sensor network and GIS. In *XX IMEKO World Congress Metrology for Green Growth*, pages 1–5, 2012.

R. Pourdarbani and B. Rezaei. Automatic Detection of Greenhouse Plants Pests by Image Analysis. *Journal of Agricultural Machinery Science*, 7(2):171–174, 2011.

R. R Shamshiri, C. Weltzien, I. A. Hameed, I. J Yule, T. E Grift, S. K. Balasundram, L. Pitonakova, D. Ahmad, and G. Chowdhary. Research and development in agricultural robotics: A perspective of digital farming. *Chinese Society of Agricultural Engineering*, 2018.

J. Roldán, P. Garcia-Aunon, M. Garzon, J. de Leon, J. del Cerro, and A. Barrientos. Heterogeneous multi-robot system for mapping environmental variables of greenhouses. *Sensors*, 16(7):1018, 2016.

A. Ruiz-Larrea, J. J. Roldán, M. Garzón, J. del Cerro, and A. Barrientos. A UGV approach to measure the ground properties of greenhouses. In *Robot 2015: Second Iberian Robotics Conference*, pages 3–13, 2016.

K. Sagar, D. Zlatanov, M. Zoppi, C. Nattero, and S. Muthuswamy. Multi-goal path planning for robotic agents with discrete-step locomotion. In *Proceedings of the ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 1–10, 2017.

R. Shamshiri, F. Kalantari, K. C. Ting, K. R. Thorp, I. A. Hameed, C. Weltzien, D. Ahmad, and Z. M. Shad. Advances in greenhouse automation and controlled environment agriculture: A transition to plant factories and urban agriculture. *Int J Agric & Biol Eng*, 11(1):1–22, 2018.

M. Srbinovska, C. Gavrovski, V. Dimcev, A. Krkoleva, and V. Borozan. Environmental parameters monitoring in precision agriculture using wireless sensor networks. *Journal of cleaner production*, 88:297–307, 2015.

A. Srinivasan. *Handbook of precision agriculture: principles and applications*. CRC press, 2006.

J. V. Stafford. Implementing precision agriculture in the 21st century. *Journal of Agricultural Engineering Research*, 76(3):267–275, 2000.

D. Story, M. Kacira, C. Kubota, A. Akoglu, and L. An. Lettuce calcium deficiency detection with machine vision computed plant features in controlled environments. *Computers and Electronics in Agriculture*, 74(2):238–243, 2010.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

L. Tai, G. Paolo, and M. Liu. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 31–36. IEEE, 2017.

S. Teraoka, T. Ushio, and T. Kanazawa. Voronoi coverage control with time-driven communication for mobile sensing networks with obstacles. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 1980–1985. IEEE, 2011.

P. Tokekar, J. Vander Hook, D. Mulla, and V. Isler. Sensor planning for a symbiotic UAV and UGV system for precision agriculture. *IEEE Transactions on Robotics*, 32(6):1498–1511, 2016.

P. Toth and D. Vigo. *The vehicle routing problem*. SIAM, 2002.

P. Tripicchio, M. Satler, G. Dabisias, E. Ruffaldi, and C. A. Avizzano. Towards smart farming and sustainable agriculture with drones. In *2015 International Conference on Intelligent Environments*, pages 140–143. IEEE, 2015.

J. P. Vasconez, G. A. Kantor, and F. A. A. Cheein. Human–robot interaction in agriculture: A survey and current challenges. *Biosystems engineering*, 179: 35–48, 2019.

A. Vasudevan, D. A. Kumar, and N. Bhuvaneswari. Precision farming using unmanned aerial and ground vehicles. In *2016 IEEE Technological Innovations in ICT for Agriculture and Rural Development (TIAR)*, pages 146–150. IEEE, 2016.

A. Walter, R. Khanna, P. Lottes, C. Stachniss, R. Siegwart, J. Nieto, and F. Liebisch. Flourish-a robotic approach for automation in crop management. In *Proceedings of the international conference on precision agriculture (ICPA)*, 2018.

U. Weiss and P. Biber. Plant detection and mapping for agricultural robots using a 3d lidar sensor. *Robotics and autonomous systems*, 59(5):265–273, 2011.

J. Xiao, G. Wang, Y. Zhang, and L. Cheng. A distributed multi-agent dynamic area coverage algorithm based on reinforcement learning. *IEEE Access*, 8:33511–33521, 2020.

X. Xing, J. Song, L. Lin, M. Tian, and Z. Lei. Development of intelligent information monitoring system in greenhouse based on wireless sensor network. In *2017*

4th International Conference on Information Science and Control Engineering (ICISCE), pages 970–974. IEEE, 2017.

J. Yu and S. M. LaValle. Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics. *IEEE Transactions on Robotics*, 32(5):1163–1177, 2016.

W. Yue, X. Guan, and L. Wang. A novel searching method using reinforcement learning scheme for multi-uavs in unknown environments. *Applied Sciences*, 9 (22):4964, 2019.

S.-k. Yun and D. Rus. Distributed coverage with mobile robots on a graph: locational optimization and equal-mass partitioning. *Robotica*, 32(2):257–277, 2014.

A. Zacepins, E. Stalidzans, and J. Meitalovs. Application of information technologies in precision apiculture. In *Proceedings of the 13th International Conference on Precision Agriculture (ICPA 2012)*, 2012.

N. Zhang, M. Wang, and N. Wang. Precision agriculture—a worldwide overview. *Computers and electronics in agriculture*, 36(2-3):113–132, 2002.