SWARM ROBOTICS: HARVESTING OF MOVING SWARMS REPRESENTED BY A MARKOV PROCESS

A Thesis

Presented to

the Faculty of the Department of Electrical and Computer Engineering

University of Houston

in Partial Fulfillment

of the Requirements for the Degree

Master of Science

in Electrical Engineering

by

Shriya Bhatnagar

May 2019

SWARM ROBOTICS: HARVESTING OF MOVING SWARMS REPRESENTED BY A MARKOV PROCESS

Shriya Bhatnagar

Approved:

Chair of the Committee
Aaron T. Becker, Assistant Professor
Department of Electrical and Computer Engineering

Committee Members:

Nicolaus Radford, CTO
Houston Mechatronics Inc.

David R. Jackson, Professor
Department of Electrical and Computer Engineering

Suresh K. Khator, Associate Dean,
Cullen College of Engineering

Badrinath Roysam, Professor and Chair,
Electrical and Computer Engineering

Acknowledgements

I would like to thank all the amazing people who have been there for me through this wonderful opportunity. I would like to thank Dr. Aaron Becker for giving me an opportunity to pursue a masters and to take me under your wing. You are the sole reason I was able to open up this exciting chapter in my life. You have been an amazing professor and mentor and you have shown me great patience and understanding. You definitely have set an unreachable bar for all professors.

I would also like to thank Nicolaus Radford for taking a chance on me, believeing in me and helping me grow my confidence in the field of robotics. I am truly grateful for the doors you have opened for me and allowing me to pursue a wonderful career path.

I want to thank my family and friends, especially my mother, Chhaya Bhatnagar, for helping me get through this degree without having to worry about things back at home. I would like to thank my siblings, Krish and Khushi, for being my motivation to strive to become the best. I would like to thank all my friends who studied with me and helped me through my classes.

Lastly, I would like to thank my significant other, Alexander Bui. Thank you for always believing in me when no one else did. I appreciate all the support and encouragement you gave me through all the difficult times that allowed me to keep moving forward. I would not be the person I am today without you. I love you very much, thank you so much for always being there for me.

SWARM ROBOTICS: HARVESTING OF MOVING SWARMS REPRESENTED BY A MARKOV PROCESS

An Abstract

of a

Thesis

Presented to

the Faculty of the Department of Electrical and Computer Engineering

University of Houston

in Partial Fulfillment

of the Requirements for the Degree

Master of Science

in Electrical Engineering

by
Shriya Bhatnagar

May 2019

Abstract

This project presents methods of harvesting moving swarm agents, with two different experimental applications. We investigated motion planning for one or more robot(s). These methods differ from traditional motion planning problems because the agents move. This allows areas that were previously cleared to become recontaminated. The movement of agents is represented by a Markov process that encodes the agents' preferred regions and their speed of motion. There are two different categories of controllers presented in this project, one for single robot applications and a second for multi-robot applications. We conducted experiments using the single robot controllers. For the first application we studied a destructive survey of mosquitoes. We researched behavior and other characteristic of mosquitoes to design an electrified screen that can be carried on a UAV. For the second application we studied the harvesting of moving swarms using an autonomous robotic boat with an acoustic larvicide unit in a body of water.

Table of Contents

Acknowledgements **v**

Abstract **vii**

Table of Contents **viii**

List of Figures **xi**

List of Tables **xvi**

1 Introduction **1**

1.1 Harvesting of Moving Swarms 1

1.2 Destructive Surveying of Mosquito Populations using a UAV 3

1.3 Harvesting Larvae Using Autonomous Robotic Boat 3

2 Robotic Harvesting of a Moving Swarm **7**

2.1 Overview and Related Work 7

2.2 Analytical Coverage of a Gaussian Distribution 9

2.3 Modeling 11

2.3.1 Uncontrolled system 13

2.3.2 Controlled system 14

2.4 Controllers 14

2.4.1 Controllers for single robot applications 14

2.4.2 Controllers for multi-robot applications 16
2.5 Simulation 19
2.6 Conclusions 22

3 Destructive Survey of Mosquitoes **24**
3.1 Overview and Related Work 24
3.2 Hardware Design 25
3.2.1 UAV 25
3.2.2 Screen design 26
3.2.3 Screen location 26
3.2.4 Wind tunnel verification of net angle 29
3.2.5 Data logger 30
3.2.6 Energy budget 31
3.3 Path Planning 31
3.3.1 Modelling 31
3.3.2 Computational complexity 34
3.3.3 Mathematical optimization 35
3.3.4 Computational results 38
3.4 Experiments 40
3.5 Conclusion and Future Work 44

4 Harvesting Larvae with an Autonomous Robotic Boat **45**
4.1 Overview and Related Work 45
4.2 Physical Prototyping 46

4.3 Conclusions 47

5 Conclusion **49**

References **50**

List of Figures

1.1 Three successive snapshots of harvesting a moving swarm. Robots (represented as green points) pass through an agent distribution (represented by a density plot). Although many agents are harvested, the swarm reforms and contaminates the cleared regions. 2

1.2 A hexacopter UAV carrying a 48 cm × 61 cm rectangular bug-zapping screen. An onboard micro controller monitors the voltage across the screen and records the time, GPS location, humidity, and altitude for each mosquito strike. At right are three frames recorded by the onboard camera showing mosquito hits, during the day (top) and at twilight. See attachment for videos of flight experiments [1]. 4

1.3 Hardware system. Top Left: The autonomous boat carries an underslung Larvasonic© transducer for killing mosquito larvae, twin motors, and an array of sensors. Bottom Left: Sensors include a LIDAR for detecting the boundaries of the water feature and precision GPS for building maps and following trajectories. Top Right: 2D slice of the Larvasonic© transducer power in polar coordinates. The lines illustrates how the acoustic power is focused in an arc in front of the transducer. Bottom Right: Image of the boat on water. 6

2.1 Plots of analytical results for covering a Gaussian distribution. Plots are for $\sigma = 1, a = 1$, and $f_{\text{kill}} = 1/2$ with the robot starting at the mean position. (left) If the mosquitos reform the Gaussian distribution every second, the optimal strategy is for the robot to stay at the mean location. The stair-step plot shows the result for the first 50 time steps. If the mosquitos are non-moving, then covering successively larger annuli is the optimal strategy at first, however at some point the robot should start covering smaller annuli. The red vertical line shows the point that there are the same number of mosquitos inside the region already covered as there are in the uncovered region. This time is illustrated in the 3D plot at right. 10

2.2 Sequence of simulations showing the agent distribution as a density plot and the path of the robot in red. The agents' position evolves according to a Markov process, but the robot harvests $f_{\text{kill}} = 1$ fraction of the agents in the cell it is covering and the agents diffuse with parameter $k = 0.005$. (a) Greedy Controller (b) h-step horizen heuristic controller See simulation video at `https://youtu.be/u1OTBK5kq7O` [2]. 13

2.3 Simulations using different multi-robot coverage algorithms with $n_r = 9$ robots and $N = 10,000$ agents over 1,000 simulation steps, with a $d = 3$ depth greedy search, $k = 0.05$, and $n_r = 10$. (A) Forest coverage boustrophedon path. (B) Formation boustrophedon path. (C) Greedy-Depth implementation. (D) Greedy-Depth search with initial goal destinations assigned based on population density. 16

2.4 (a) Simulations comparing the controllers presented in Section 2.4 with $n_r = 1$ robot, $N = 10,000$ agents, 1,000 simulation steps, and diffusion parameter $k = 0.05$. (b) Simulations varying agent dispersion parameter k, as described in Section 3.3, with $n_r = 1$ robot, $N = 10,000$ agents over 1,000 simulation steps, with a $d = 3$ depth greedy search. (c) Simulations varying the controllers as described in Section 2.4, with $N = 10,000$ agents, 1,000 simulation steps, a $d = 3$ depth greedy search, $k = 0.05$, and $n_r = 10$. See simulation video [2]. 17

2.5 Simulation parameters: $N = 10,000$ agents, 1,000 simulation steps, running Heuristic with Goals DFS, $k = 0.05$, and $n_r = 10$ (a) Simulation comparing the number of agents harvested as the kill rate parameter varies, $f_{\text{kill}} \in [0.1, 1]$, run in World 1. (b) Simulation comparing the number of agents harvested as the kill rate parameter varies, $f_{\text{kill}} \in [0.1, 1]$, run in World 2. 18

2.6 Simulations varying the number of robots with $N = 10,000$ agents, $1,000$ steps, $d = 3$ depth greedy search, and diffusion parameter $k = 0.05$. Subplots show the heat map of the agents, and the robot paths (in multicolors) after 50, 100, and 250 steps. 22

3.1 The UAV suspends a rectangular bug-zapping screen beneath it. Propwash pushes incoming mosquitoes downwards, and the UAV clears a volume $h_m \times d_s \times v_f$ each second. Circles show two mosquitoes at equal time intervals relative to the UAV. 27

3.2 The volume cleared by a UAV is a function of screen angle θ and forward velocity v_f. Dotted line shows the optimal angle given in (3.4). 29

3.3 Frames from wind tunnel test with free-flying UAV at 3 m/s windspeed with smoke for streaklines [1]. As shown in the frames at right, the proposed screen position (in red) captures free flowing air and air entrained by the UAV propellers. Each black square is 25.4 mm in width. 30

3.4 Current, voltage, and power traces for five *Culex quinquefasciatus* mosquitoes as each contacts the bug-zapping screen at $t = 0$. Contact causes a brief short that recovers in 160 ms. 32

3.5 Mosquito hunting drone 33

3.6 Optimal cycle covers with different density scaling. The middle has twice and the right instance has four times the density as the left. In these instances, the cost of a 90° is five times that of a straight pixel transition. 37

3.7 Runtime of solving benchmark instances to optimality. Shown are the times of ten instances for each size, with a timeout at 900 s, as well as the percentage of solved instances. Only the number of turns is minimized in these instances. 39

3.8 Left is an optimal penalty cycle cover. Cycles (blue) cover all areas with high density. After three applications of the tour constraints, a single cycle remains (right). In the intermediate solutions, the subcycles first try to evade the new constraints by reshaping. The final tour omits two of the small hotspots because the cost of integrating them into the single tour is prohibitively expensive. 40

3.9 Fountain 41

3.10 The UAV's path for flight 3 is in red. Strikes collected along this path are represented by yellow dots. 42

3.11 Density map showing mosquito distribution on the field, overlaid by flight path 4 in white. 43

3.12 The UAV and screen during a flight trial near the ocean. 43

4.1 Schematic showing the kill zone for the Larvasonic© boat (kill zone at 0 s is orange and at 1 s is blue). In 1 s, the boat kills larvae within 3 meters of the device in a 25° arc in front of the transducer. Limiting the linear speed of the vehicle to 1 m/s and rotational speed to 25°/s ensures that approximately a 1×1 m^2 area in front of the boat is cleared of larvae. . . 47

4.2 The boat was programmed to follow a boustrophedon path set by way points, which is indicated by the orange line. The red line with the arrow is the autonomous boat following the path as closely as possible. 48

List of Tables

1.1 Spectrum of robotic coverage with time-varying characteristics. This thesis focuses on the middle column. 1

Chapter 1

Introduction

1.1 Harvesting of Moving Swarms

Canonical robotic coverage seeks to navigate a robot such that the robot's *coverage footprint* passes over every point in the workspace. Coverage tasks have received significant attention from the robotics community with applications from search and rescue to painting [3]. Coverage problems with dynamic workspaces, such as having moving targets, changing or unknown environments, or coverage uncertainty are more challenging than variants with static workspaces and deterministic actions. This chapter focuses on a form of time-varying coverage where a large population of moving agents are distributed in the workspace, and the robot harvests a fraction of all the agents within its coverage footprint. Furthermore, we assume that the agents' movements can be represented by a spatially discretized Markov model that is specified by a stationary distribution and a scalar diffusion coefficient. The model chosen moves the agents probabilistically, positioning this problem between periodic coverage problems and pursuit-evasion problems, as illustrated in Table 1.1, between periodic coverage problems and pursuit-evasion problems.

Time-Varying Coverage Problems		
Item count changes but no cell-to-cell movement	**Probabilistic motion models for many agents**	**Pathological cases-intelligent agents**
Lawn mowing/vacuuming Persistent sensing Data ferry/sensor recharge	Killing mosquitoes/larvae Commercial fishing/hunting Pesticides applications	Art gallery problem Pursuit/evasion problems Coverage and tracking

Table 1.1: Spectrum of robotic coverage with time-varying characteristics. This thesis focuses on the middle column.

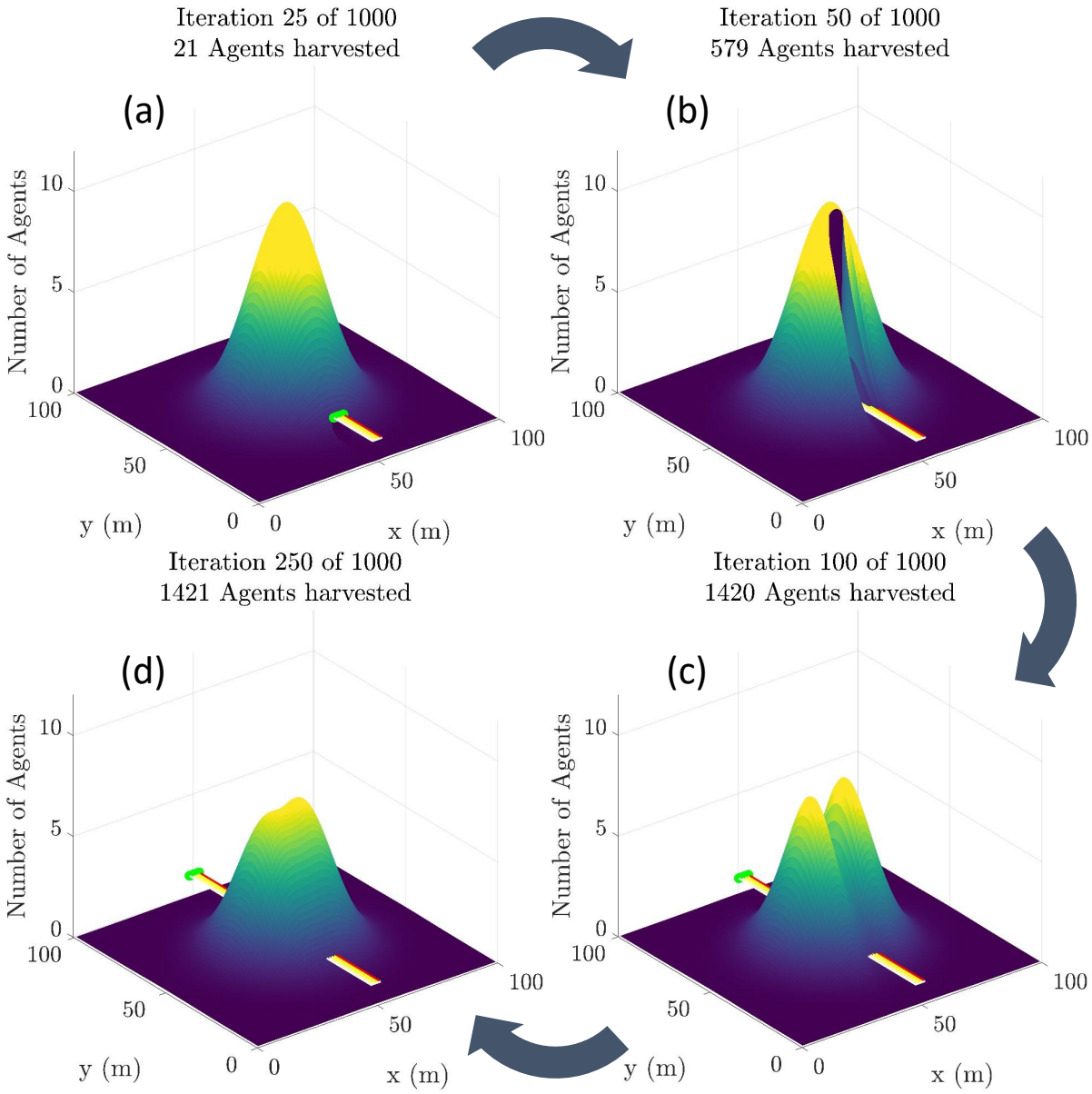

Figure 1.1: Three successive snapshots of harvesting a moving swarm. Robots (represented as green points) pass through an agent distribution (represented by a density plot). Although many agents are harvested, the swarm reforms and contaminates the cleared regions.

This model enables encoding preferences swarm agents might have for certain locations. The coverage goal is to maximize the number of agents harvested in a given time budget. Areas previously covered by the robot may be recontaminated, as illustrated in Fig. 1.1.

1.2 Destructive Surveying of Mosquito Populations using a UAV

Mosquito-borne diseases kill millions of humans each year [4]. Because of this threat, governments worldwide track mosquito populations. Tracking individual mosquitoes is difficult because of their small size, wide-ranging flight, and preference for low-light. Tracking studies of individual mosquitos have chosen to use small ($1.2\,\mathrm{m} \times 2.4\,\mathrm{m}$) indoor regions [5], or mating swarms backlit against a solid background [6].

The dominant tools for tracking mosquito populations are stationary traps that are checked at weekly intervals (*e.g.* Encephalitis Vector Surveillance traps and/or gravid traps [7]). Recent research has focused on making these traps smaller, cheaper, and capable of providing real-time data [8,9]; however, they still rely on attracting mosquitoes to the trap. This section presents an alternate solution using an electrified bug-zapping screen mounted on an unmanned aerial vehicle (UAV) as shown in Fig. 1.2 to seek out the mosquitoes in their habitat. As the UAV follows a path, it sweeps out a volume of air, temporarily removing all the mosquitoes in this volume. By monitoring the voltage across this screen, we can track individual mosquito contacts. UAVs have strict energy budgets, so optimized flight patterns are of crucial importance. As a consequence, putting the UAV to good use requires methods for computing trajectories that minimize energy consumption along the way, but maximize the total volume of mosquitoes at visited locations.

1.3 Harvesting Larvae Using Autonomous Robotic Boat

Mosquito-borne diseases kill millions of humans every year, making them the world's deadliest animal [4]. As a result, there is a worldwide effort in the development of effective control methods. This effort is primarily focused on developing countries where the risks of contracting a disease are severe [10]. While chemical-based insecticides and biological controls have been successful in the past, some mosquito species

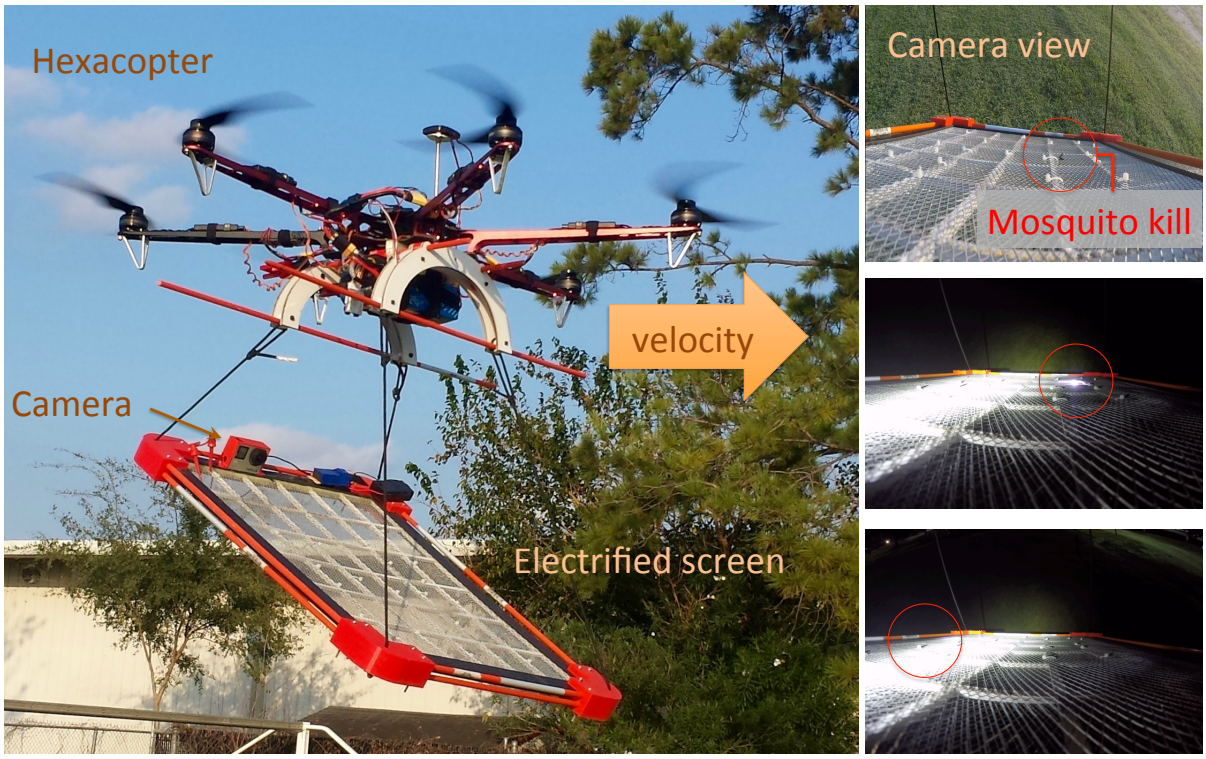

Figure 1.2: A hexacopter UAV carrying a 48 cm × 61 cm rectangular bug-zapping screen. An onboard micro controller monitors the voltage across the screen and records the time, GPS location, humidity, and altitude for each mosquito strike. At right are three frames recorded by the onboard camera showing mosquito hits, during the day (top) and at twilight. See attachment for videos of flight experiments [1].

and their larvae have developed resistance to chemicals or avoid locations with natural predators [11,12]. In this work, we examine control methods that target mosquito larvae. Because mosquito larvae are part of the *hyponeuston*, organisms that live underneath the surface of water, they are susceptible to focused acoustic energy [13].

This chapter presents an autonomous robotic boat carrying an acoustic larvicide© unit, as shown in Fig. 1.3. The larvicide unit is a piezoelectric transducer that emits acoustic pulses that kill all mosquito larvae in a cone of length 3 m with a 25° angular arc. The robotic boat uses GPS to follow trajectories, such as a lawnmower-type coverage path. However, mosquito larvae are not stationary [14–17], and their movement [18] can recontaminate zones that were previously cleared by the larvicide unit. Moreover,

mosquito larvae distribution is not uniform [19]. Mosquito larvae tend to prefer areas that are near lake edges and shallow waters [20]. The robotic boat carries a camera that is used to identify larvae concentration. This data is used to generate a stationary distribution for the mosquito larvae.

This chapter is organized as follows: research in coverage and tracking tasks are discussed in Section 4.1. Then, analytical elimination of a population of mosquito larvae using a Gaussian distribution is presented in Section 2.2. Markov processes for modeling are covered in Section 2.3, and several controllers for the robotic boat are presented in Section 2.4. Simulation results are reported in Section 3.3. The Larvasonic© robotic boat and the operation of the acoustic larvicide are described in Section 4.2.

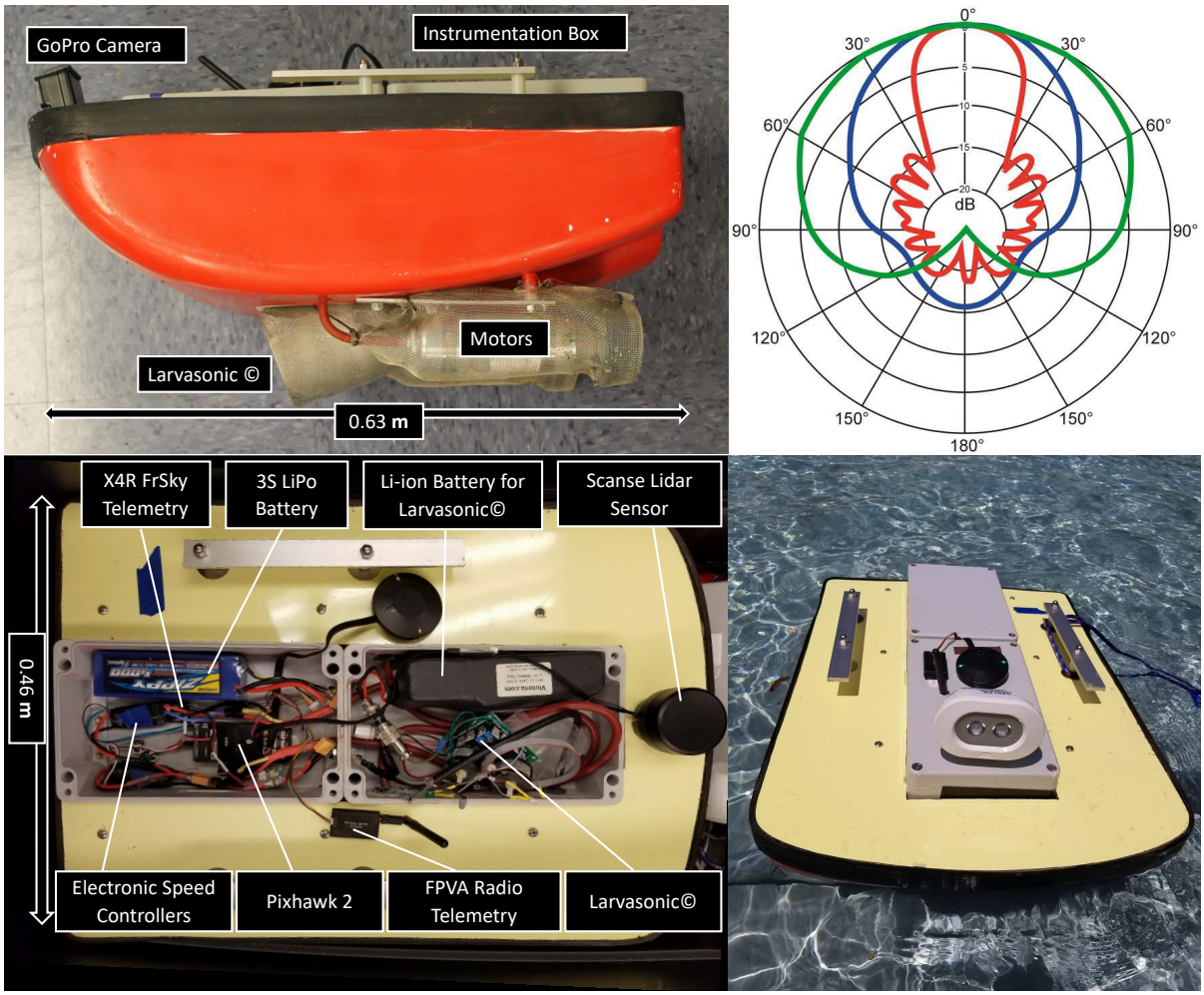

Figure 1.3: Hardware system. Top Left: The autonomous boat carries an underslung Larvasonic© transducer for killing mosquito larvae, twin motors, and an array of sensors. Bottom Left: Sensors include a LIDAR for detecting the boundaries of the water feature and precision GPS for building maps and following trajectories. Top Right: 2D slice of the Larvasonic© transducer power in polar coordinates. The lines illustrates how the acoustic power is focused in an arc in front of the transducer. Bottom Right: Image of the boat on water.

Chapter 2

Robotic Harvesting of a Moving Swarm

2.1 Overview and Related Work

This research has much in common with visibility-based coverage tasks. In these tasks, robots clear adversaries within their coverage footprint, and attempt to construct a series of movements that prevent adversaries from being able to enter previously cleared regions while enlarging the cleared area [21–24]. Such problems assume that adversaries are infinitely clever, so planners attempt to design a solution that guarantees coverage, or returns failure. In contrast, this project assumes adversary motion is predictable in the aggregate, and represents adversaries by a Markov motion model. This project is also related to the work of Pimenta et al., who devised a controller that minimizes the time required for coverage of an environment while also tracking moving targets [25]. Because the targets are moving, their coverage solution requires constant adjustment. Similarly, our coverage robots must continue to move due to the changing environment.

Previous work in path planning for persistent sensing resulted in algorithms that allow a robot to modify a path by shifting a set of waypoints to focus on dynamic areas of interest [26]. The controllers are custom designed to find a locally optimal path given an unknown environment. While the current population of agents in each workspace cell corresponds to the areas of interest in [26], this research compares a variety of controllers that either follow a consistent path or plan using greedy or heuristic policies.

Other previous research on coverage under uncertainty provided a "probably approximately correct" measure of coverage. This enabled a robot to generate policies that guarantee (to an arbitrary level of certainty) the coverage of a fraction of the free

space [27]. Similarly, in this work, agent populations obey dynamics with large variance and we wish to design policies that reliably harvest the agents.

Optimization problems related to this research include the traveling salesman problem (TSP) and the art gallery problem [28]. Both problems are proven to be nondeterministic polynomial-time hard (NP-hard) and typically rely on heuristics for a complete, but not optimal solution. This project explores using heuristic-based algorithms to cover a swarm of moving agents.

This work is inspired by motion planning challenges involved in mosquito control. Coverage of mosquitoes is difficult because movement of both larva and adults causes recontamination of previously cleared regions. However, mosquito motion is at least partially predictable in the aggregate. Mosquitos exhibit preferences for certain regions. Recent research on mosquito control methods that incorporate robotics include the use of Unmanned Aerial Vehicles (UAV) with chemical larvicides [29], UAVs with controlled release of sterile adult male mosquitoes [30], and experiments in trajectory-planning for a UAV equipped with a mosquito-zapping electric screen [31]. Past work in mapping of mosquito larvae exploit satellite imagery and vision processing techniques to locate sections of a body of water where mosquito larvae are most likely to be found [32].

Another application for this research is environmental tracking in marine environments. Tokekar et al., designed a robotic boat capable of covering a region to look for tagged invasive fish [33]. They introduced a coverage problem that requires a robot to cover only specific regions of a lake by assuming that fish have preferred regions over others. They assumed that these preferred regions were distinct and solved a TSP to minimize the time to move between the preferred regions. In contrast, we assume that agents can move probabilistically throughout the workspace, but have higher probabilities of moving toward preferred regions.

2.2 Analytical Coverage of a Gaussian Distribution

A Gaussian distribution is a natural representation for a population of mosquitos. However, if the mosquito removal process makes the distribution non-Gaussian and the mosquitos move, finding an analytical expression for the mosquito distribution is challenging. Updating the mosquito distribution requires integration over the probability distribution, which becomes intractable unless the distribution is discretized. There are two notable exceptions that occur: (1) when the mosquitos do not move, and (2) when the mosquitos re-form a Gaussian distribution every time step. A representative plot for each limit case is shown in Fig. 2.1. Both exceptions will be modeled analytically in this section, while the following sections will model mosquito distributions using a discretized approach.

If the mosquitos are distributed in a symmetric Gaussian distribution with standard deviation σ and mean $[0, 0]$, the expected distribution is

$$P(x, y) = \frac{1}{2\pi\sigma^2} \mathrm{e}^{\frac{-(x^2+y^2)}{2\sigma^2}}. \tag{2.1}$$

If an agent covers area a per second, the agent can cover $a\,t$ area in t seconds. If the area cleared is a disk, that disk has radius r and area πr^2. To preserve symmetry, we will assume the robot is initialized at the mean position of the distribution, $(0, 0)$.

Covering a Gaussian Distribution with non-moving mosquitos:

If we make the gross simplification that the agent covers successively larger (and correspondingly thinner) annuli, the area covered from time 0 to t has radius

$$r(t) = \sqrt{\frac{at}{\pi}}. \tag{2.2}$$

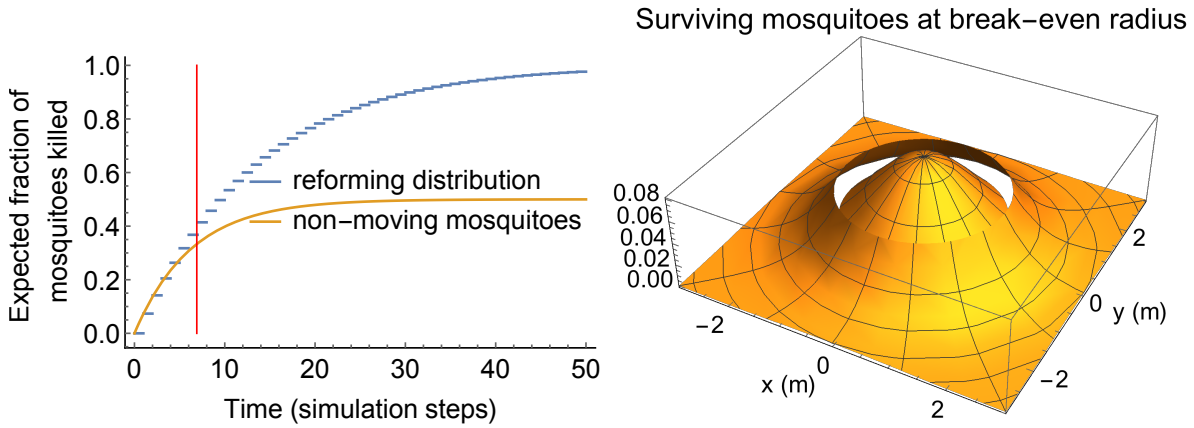

Figure 2.1: Plots of analytical results for covering a Gaussian distribution. Plots are for $\sigma = 1, a = 1$, and $f_{\text{kill}} = 1/2$ with the robot starting at the mean position. (left) If the mosquitos reform the Gaussian distribution every second, the optimal strategy is for the robot to stay at the mean location. The stair-step plot shows the result for the first 50 time steps. If the mosquitos are non-moving, then covering successively larger annuli is the optimal strategy at first, however at some point the robot should start covering smaller annuli. The red vertical line shows the point that there are the same number of mosquitos inside the region already covered as there are in the uncovered region. This time is illustrated in the 3D plot at right.

The probability mass within a radius r of the mean position is

$$P(\sqrt{x^2+y^2} < r) = 2\pi \int_0^r x \left(\frac{1}{2\pi\sigma^2} \mathrm{e}^{\frac{-(x^2)}{2\sigma^2}} \right) \mathrm{d}x,$$

$$= 1 - \mathrm{e}^{\frac{-(r^2)}{2\sigma^2}}. \tag{2.3}$$

Substituting (2.2) into (2.3) gives the probability mass covered as a function of time:

$$sP_{\text{mass covered}}(t) = 1 - \mathrm{e}^{\frac{-at}{2\pi\sigma^2}}. \tag{2.4}$$

If we assume our robot kills a fraction f_{kill} of the mosquitos it encounters, where $f_{\text{kill}} \in [0, 1]$, and that there are N mosquitos in the distribution, the expected number of kills is

$$\mathbb{E}[\text{kills}_{\text{stationary}}(t)] = N f_{\text{kill}} \left(1 - \mathrm{e}^{\frac{-at}{2\pi\sigma^2}} \right). \tag{2.5}$$

In this analysis, the robot is continuously spiraling outwards to pursue more mosquitos. However, if f_{kill} is less than 1, at some point there are more surviving mosquitos in the region already covered than in the uncovered region. This occurs at $t = \frac{2\pi\sigma^2}{a} \ln[\frac{f_{\text{kill}}-2}{f_{\text{kill}}-1}]$.

Mosquitos that reform a scaled Gaussian distribution:

In this case, assume the robot removes all the mosquitos within a disc of area a every second. If the robot is at the mean position, the probability mass eliminated is given by (2.3) with $r = a/\pi$. Because the highest mosquito density is located at the center, the robot should stay at the mean position. We will assume that each second the remaining mosquitos reform into a Gaussian distribution at the original mean and the original standard deviation. The system then kills $f_{\text{kill}}\left(1 - \mathrm{e}^{\frac{-a}{2\pi\sigma^2}}\right)$ fraction of the remaining mosquitos each second. The number of surviving mosquitos is then $N\left(1 - f_{\text{kill}}\left(1 - \mathrm{e}^{\frac{-a}{2\pi\sigma^2}}\right)\right)^{\lfloor t \rfloor}$. The expected number of kills is then

$$\mathbb{E}[\text{kills}_{\text{reformed}}(t)] = N - N\left(1 - f_{\text{kill}}\left(1 - \mathrm{e}^{\frac{-a}{2\pi\sigma^2}}\right)\right)^{\lfloor t \rfloor}. \tag{2.6}$$

Generating analytical results for intermediate situations where the mosquitos are moving, but do not immediately reform their stationary distribution is challenging. It is also challenging to model robot motions that are asymmetric, or non-Gaussian distributions. For these reasons, the next section describes a technique to model mosquito distributions as discretized 2D histograms whose time evolution is governed by a Markov process.

2.3 Modeling

We assume that some agents prefer certain regions over others. In previous work, we represented these preferences by a 2D histogram where each grid cell represented the relative prevalence of agents in that cell and designed paths that maximized the number of agents harvested while respecting an energy budget [31,34]. That model assumed agents were immobile. Representing agent mobility requires increasing the model complexity. One method to represent mobility is to directly model individual agents, as in [35],

however this approach becomes unwieldy with large numbers of agents and required running a statistically significant number of trials to obtain representative data.

We model aggregate behavior of the agent population as a whole. This eliminates the need for running many trials for each situation. We can then generalize the behavior of the agents' movement by representing it as a Markov process.

Algorithm 1 Markov Transition Matrix $P(w, k)$

w is the stationary distribution ($w_{i,j} \geq 0$, $\sum w = 1$) and $k \in [0, 1/4]$ is the dispersion parameter (fraction of swarm that leaves a cell in any direction).

1: $(\ell_w, \ell_h) \leftarrow \text{size}(w)$
2: $P \leftarrow$ zero matrix that is $(\ell_w \ell_h) \times (\ell_w \ell_h)$
3: **for** $r = 1$ to ℓ_h **do**
4: **for** $c = 1$ to ℓ_w **do**
5: $x \leftarrow \ell_w(c-1) + r$ ▷ current cell
6: $e \leftarrow \ell_w c + r$ ▷ cell to east
7: $s \leftarrow \ell_w(c-1) + r + 1$ ▷ cell to south
8: **if** $c < \ell_w$ **then**
9: $\begin{bmatrix} P_{e,x} \\ P_{x,e} \end{bmatrix} \leftarrow \begin{cases} \begin{bmatrix} k \\ k\frac{w(r,c)}{w(r,c+1)} \end{bmatrix} & w(r,c) < w(r,c+1) \\ \begin{bmatrix} k\frac{w(r,c+1)}{w(r,c)} \\ k \end{bmatrix} & \text{else} \end{cases}$
10: **end if**
11: **if** $r < \ell_h$ **then**
12: $\begin{bmatrix} P_{s,x} \\ P_{x,s} \end{bmatrix} \leftarrow \begin{cases} \begin{bmatrix} k \\ k\frac{w(r,c)}{w(r+1,c)} \end{bmatrix} & w(r,c) < w(r+1,c) \\ \begin{bmatrix} k\frac{w(r+1,c)}{w(r,c)} \\ k \end{bmatrix} & \text{else} \end{cases}$
13: **end if**
14: **end for**
15: **end for**
16: **for** $i = 1$ to $\ell_w \ell_h$ **do**
17: $P_{i,i} = 1 - \sum_{j=1}^{\ell_w \ell_h} P_{i,j}$ ▷ columns sum to 1
18: **end for**
19: **return** P

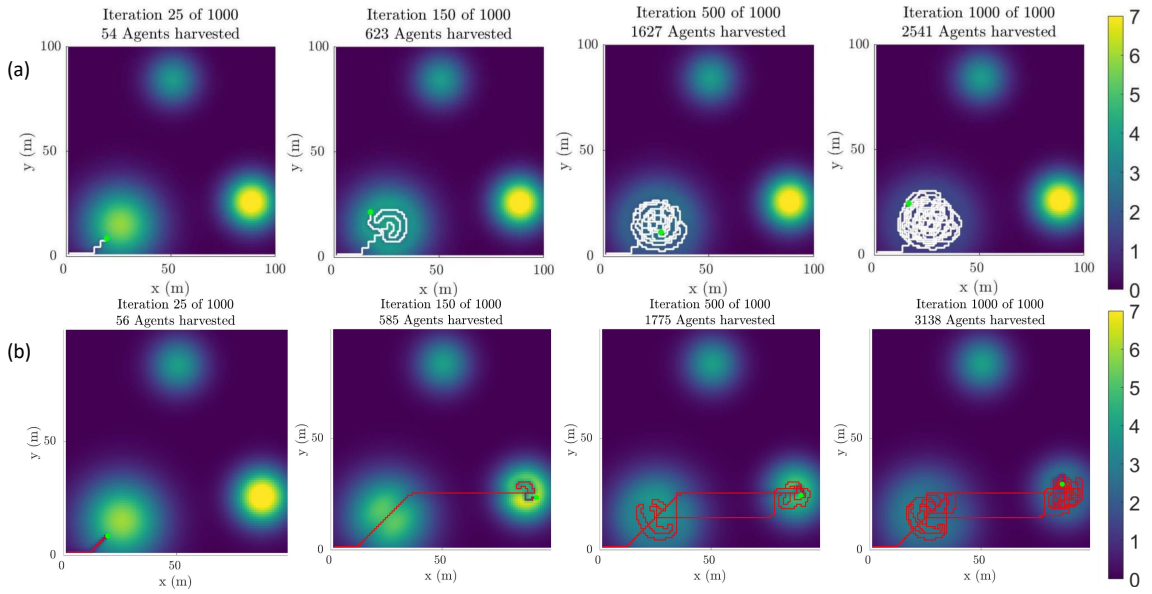

Figure 2.2: Sequence of simulations showing the agent distribution as a density plot and the path of the robot in red. The agents' position evolves according to a Markov process, but the robot harvests $f_{\text{kill}} = 1$ fraction of the agents in the cell it is covering and the agents diffuse with parameter $k = 0.005$. (a) Greedy Controller (b) h-step horizen heuristic controller See simulation video at `https://youtu.be/u1OTBK5kq7O` [2].

2.3.1 Uncontrolled system

We assumes agents are distributed in a 2D grid and that the system state is represented at time t by $\mathbf{x}(t)$. $\mathbf{x}(t)$ is a 1D vector of length $\ell_w \times \ell_h$ that represents the agent population in each grid cell at time t. To propagate the moving swarm one time step, we use

$$\mathbf{x}(t+1) = P\mathbf{x}(t). \tag{2.7}$$

The system evolution matrix P assumes interaction only between grid cells that share edges, and is parameterized by the diffusion rate k. The transition matrix P is determined entirely by the stationary distribution w and the diffusion rate k, and is computed once in $\mathrm{O}(w^2)$ time by Alg. 1. If $\ell_w = \ell_h = \ell$, this produces a sparse $\ell^2 \times \ell^2$ matrix $P_{x,y}$, with only $\ell(5\ell - 4)$ non-zero entries. If all the w values are greater than zero and $k > 0$, the resulting Markov chain is aperiodic and irreducible. Distributions

that are disturbed, heal and return to their stationary distribution after some time. The evolution of such a system is shown in Fig. 1.1.

2.3.2 Controlled system

We assume that the robot's coverage footprint is the size of one grid cell, that all n_r robots are in different cells, and that each robot harvests $f_{\text{kill}} \in [0, 1]$ of the agents in its grid cell.

1: $\mathbf{x}' \leftarrow \mathbf{x}(t)$
2: **for** $j = 1$ to n_r **do**
3: $\quad i =$ position of robot j
4: $\quad \mathbf{x}'_i = (1 - f_{\text{kill}})\mathbf{x}'_i$
5: **end for**
6: $\mathbf{x}(t + 1) = P\mathbf{x}'$.

2.4 Controllers

This section compares several controllers used for single and multi-robot applications.

2.4.1 Controllers for single robot applications

The *boustrophedon* controller steers the robot back and forth from west to east, moving one row upwards each time it reaches a boundary. If the agents are evenly distributed and cannot move, the boustrophedon is the optimal solution. When these assumptions are violated, the boustrophedon performs poorly.

The *random coverage* controller randomly commands the robot to move one grid cell east, south, west, or north. If this move is blocked by the boundary, a different direction is chosen. Since the random controller does not use any sensor measurements besides

boundary detection, it performs poorly, but it does not get stuck in local minimums.

In contrast, a *greedy* controller with 1-step lookahead compares the utility of moving one grid cell east, south, west, north, or for staying still. The movement that results in the largest number of agents harvested is selected. This type of controller is an exploitation strategy that does that does little exploration. Similarly, a greedy controller with d-step lookahead computes the number of expected agents harvested by all possible 5^d movement sequences, and implements the first step of the sequence that harvests the largest number of agents.

All these controllers have a limited ability to predict into the future. In the simulation of Fig. 2.2.A, the Gaussian peaks are at least 70 units apart. Due to the curse of dimensionality, the brute-force greedy controller cannot predict far enough into the future to make crossing between the Gaussian hills competitive with staying in the initial distribution ($5^{70} \approx 8 \times 10^{48}$).

To address this, our final controller is the *h-step horizon heuristic*, which switches between exploration and exploitation strategies by simulating a small number of deep searches, as shown in Fig. 2.2.B. This controller is similar to model predictive control because each time step it simulates the expected number of agents harvested h steps into the future by following different strategies. Each strategy is simulated, and the strategy that harvest the most agents is selected. All strategies start by setting a goal destination. If the simulated robot reaches the destination within h steps, the robot uses its remaining time by obeying a d-step greedy controller. The null policy sets the current position as the goal destination, so the robot obeys a d-step greedy policy for all h iterations. The implementation compares the null policy and sending the robot to each of the Gaussian peaks. If there are i peaks, $i + 1$ policies are compared. While moving to a destination the robot makes locally optimal choices if the movement requires both horizontal and vertical movements, choosing the option that harvests more agents.

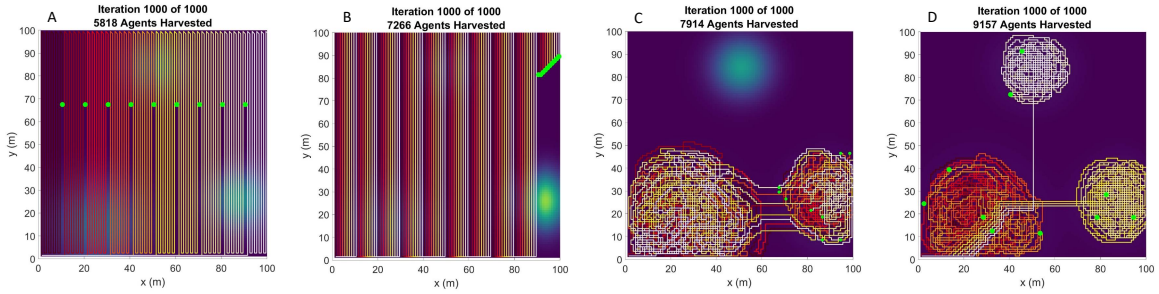

Figure 2.3: Simulations using different multi-robot coverage algorithms with $n_r = 9$ robots and $N = 10,000$ agents over 1,000 simulation steps, with a $d = 3$ depth greedy search, $k = 0.05$, and $n_r = 10$. (A) Forest coverage boustrophedon path. (B) Formation boustrophedon path. (C) Greedy-Depth implementation. (D) Greedy-Depth search with initial goal destinations assigned based on population density.

There is a tradeoff when choosing h because at each time step, the heuristic only simulates one switch of goal destination. If the horizon h is too short, the robot will not harvest many agents from a distant peak (and may not even reach the destination). If h is too long, the simulated robot will overexploit the destination.

2.4.2 Controllers for multi-robot applications

The *forest boustrophedon* [36] controller moves the robots to starting positions that evenly divide the workspace, and then robots perform boustrophedon paths. This is shown in Fig. 2.3(A).

The *formation boustrophedon* controller performs coverage paths in echelon form with the robots moving in a diagonal formation. This can be seen in Fig. 2.3(B).

Both controllers (A) and (B) do not use any sensor measurements. Adding sensing can improve performance. The *greedy depth* controller in Sec. 2.4.1 can be implemented on multiple robots to achieve higher amounts of harvesting. However, Fig. 2.3(C) shows an instance where the robots fail to discover a distribution peak in the north. They could be more effective if they were spread out based on the initial agent density map.

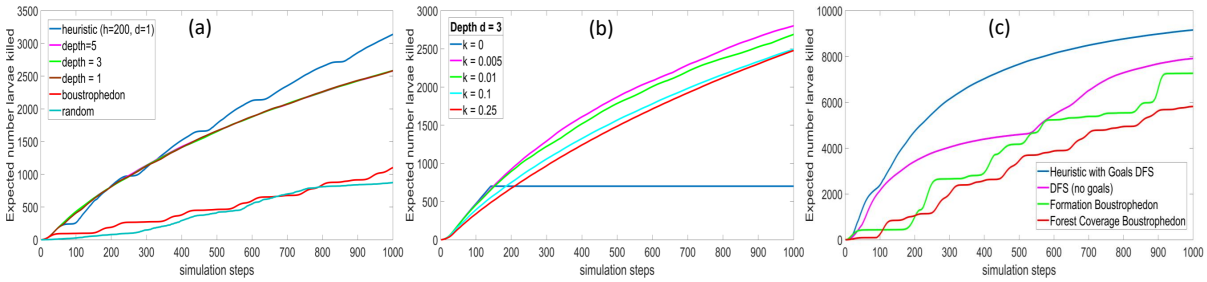

Figure 2.4: (a) Simulations comparing the controllers presented in Section 2.4 with $n_r = 1$ robot, $N = 10,000$ agents, 1,000 simulation steps, and diffusion parameter $k = 0.05$. (b) Simulations varying agent dispersion parameter k, as described in Section 3.3, with $n_r = 1$ robot, $N = 10,000$ agents over 1,000 simulation steps, with a $d = 3$ depth greedy search. (c) Simulations varying the controllers as described in Section 2.4, with $N = 10,000$ agents, 1,000 simulation steps, a $d = 3$ depth greedy search, $k = 0.05$, and $n_r = 10$. See simulation video [2].

To implement such a controller, goal destinations are introduced to the greedy depth controller. We build a mixture of Gaussians model that is fit to $x(0)$ using an expectation maximization (EM) algorithm. This expectation maximization identifies τ_i, the variables that encode the relative probabilities of each Gaussian. The robots are distributed so that τ_i robots are sent to each Gaussian peak. Once the robots are positioned, they continue to perform their regular greedy algorithm. These initial goal positions can send robots to distant peaks that the regular greedy depth controller may be unable to reach. A resulting path is shown in Fig. 2.3(D). In this simulation, two robots are sent to the Gaussian peak at the top of the map that was missed by the greedy depth controller.

To combat the exponential cost of simulating n_r robots d steps into the future, our approach instead uses a priority-based system, where the $(i+1)^{\text{th}}$ robot computes its controller after simulating the harvesting performed by robots 1 to i.

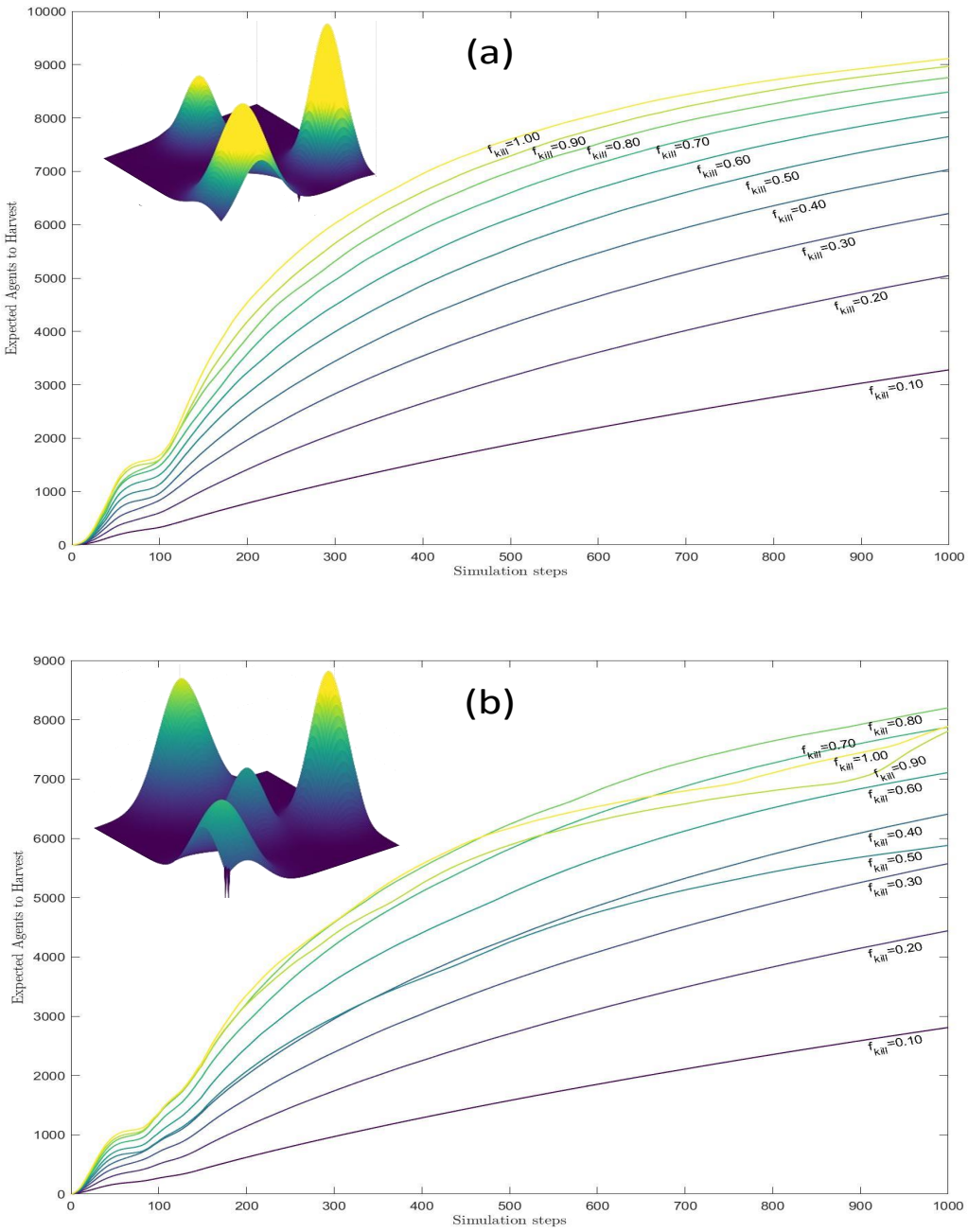

Figure 2.5: Simulation parameters: $N = 10,000$ agents, 1,000 simulation steps, running Heuristic with Goals DFS, $k = 0.05$, and $n_r = 10$ (a) Simulation comparing the number of agents harvested as the kill rate parameter varies, $f_{\text{kill}} \in [0.1, 1]$, run in World 1. (b) Simulation comparing the number of agents harvested as the kill rate parameter varies, $f_{\text{kill}} \in [0.1, 1]$, run in World 2.

2.5 Simulation

For most simulations, we used a stationary distribution of agents described as the sum of three Gaussian distributions on an $\ell \times \ell$ grid with $\ell = 100$ and used World 1, defined with

- 1/2 of the agents distributed with mean $\ell(1/7, 1/4)$ and $(\sigma_x, \sigma_y) = \ell(1/8, 1/8)$,
- 1/3 of the agents distributed with mean $\ell(1/4, 7/8)$ and $(\sigma_x, \sigma_y) = \ell(1/12, 1/12)$, and
- 1/6 of the agents distributed with mean $\ell(1/6, 1/2)$ and $(\sigma_x, \sigma_y) = \ell(1/12, 1/12)$.

The robot was initialized in the southwest corner of the workspace, and the percentage of population harvested when a robot visits a cell was set to 100% ($f_{\text{kill}} = 1$). All simulations were performed in MATLAB. Code is available at [37].

Comparing controllers Our first set of simulations compare six controllers with the same agent diffusion parameter, $k = 0.05$. The amount of harvested agents as a function of simulation steps are shown in Fig. 2.4(a). As expected, the random strategy performed the worst with 874 agents harvested. The boustrophedon pattern performed only slightly better, with 1103 agents harvested. The greedy controller exploits local information and kills approximately twice as many as boustrophedon. However, the greedy controller cannot not explore further than d steps ahead. Due to local maximas, having a deeper lookahead function did not always translate into a higher number of harvested agents. For the d-step greedy strategy, $d = \{1, 3, 5\}$ harvested $\{2583, 2583, 2580\}$ agents.

In contrast, the heuristic controller balances exploration and exploitation. This enables the heuristic controller to eventually outperform the greedy controller. However, due to the time spent on moving toward goal destinations, the greedy controller often

outperforms the heuristic during the early stages of the simulation. Each time the heuristic controller travels to a new maxima, the harvesting rate flattens, which shows up as the five plateaus in Fig. 2.4(a). The heuristic with a 200-step horizon harvested 3138 agents, even though its greedy search had only a $d = 1$ step lookahead.

Varying k, the agent diffusion rate The second set of simulation experiments compared the effect of the dispersion parameter, which is the maximum fraction of agents in a cell moving in a cardinal direction ($k \in [0, 1/4]$). The results are shown in Fig. 2.4(b). If $k = 0$ the agents are stationary and the problem is equivalent to traditional coverage. As k increases, the distribution returns to the stationary distribution more quickly. With one notable exception, performance was inversely proportional to k. For the largest, $k = 0.25$, only 2478 agents were harvested. Diffusions of $k = \{0.1, 0.01, 0.005\}$ harvested $\{2495, 2689, 2801\}$ agents. However, for $k = 0$ only 702 agents were harvested, even though for the first 142 steps this strategy performed better than all others. The system reached a configuration where all the agents in every direction had been harvested and the limited 1-step lookahead was unable to plan a trajectory.

Comparing controllers for multiple robots The third experiment compared four different controllers with multiple robots. For each controller the number of robots is $n_r = 10$, with $k = 0.05$. The amount of harvested agents as a function of simulation steps are shown in Fig. 2.4(c). The forest boustrophedon coverage performed the worst, with 5815 agents harvested. It had a spike in agents harvested at 125 steps, and briefly outperformed formation boustrophedon coverage. The formation boustrophedon coverage performed better, harvesting 7266 agents. The greedy depth search without goal destinations for n_r robots captured 7914 agents. The best performance, harvesting 9157, was earned by the $d = 3$ greedy depth search with robots assigned to goal destinations based on the agent population density.

Varying f_{kill} in two different worlds The fourth experiment compared two different worlds while also varying the f_{kill} parameter from 0.1 to 1, with an increment of 0.1 between each run. World 1 is the same world used in all the other simulations. World 2 is a stationary distribution of agents described as the sum of four Gaussians with the following parameters: $\ell = 100$, with

- 1/6 of the agents distributed with mean $\ell(1, 1/2)$ and $(\sigma_x, \sigma_y) = \ell(1/8, 1/8)$,
- 1/3 of the agents distributed with mean $\ell(2/5, 8/9)$ and $(\sigma_x, \sigma_y) = \ell(1/12, 1/12)$,
- 1/6 of the agents distributed with mean $\ell(1/2, 1/2)$ and $(\sigma_x, \sigma_y) = \ell(1/12, 1/12)$,
- 1/3 of the agents distributed with mean $\ell(1/5, 1/10)$ and $(\sigma_x, \sigma_y) = \ell(1/8, 1/8)$.

In this experiment, the efficacy of the robots is put the test. When f_{kill} is set to 0 the robots completely fail at harvesting the agents. When f_{kill} is set to 1, the robots harvest all agents present in their cell. In Fig. 2.5.A, increasing f_{kill} increases the amount of agents harvested. After increasing f_{kill} above 0.7, the marginal utility decreases. However, in Fig. 2.5.B, in World 2, increasing f_{kill} did not always increase the amount of agents harvested. This could be the effect of resource starvation, where after a certain amount of simulation steps robots are not sent back to areas with recontaminated areas. This could be avoided by implementing the h-step horizon heuristic controller for multi-robot applications.

Marginal utility of additional robots The fifth experiment compared the marginal effect of additional robots. All the robots were programmed to follow a $d = 3$ depth greedy search once they had reached their assigned initial goal destination, with $k = 0.05$ and $n_r \in [1, 30]$. After $n_r = 10$, additional robots have low impact on the number of agents harvested. Figure 2.6 fits this data with an exponential decay.

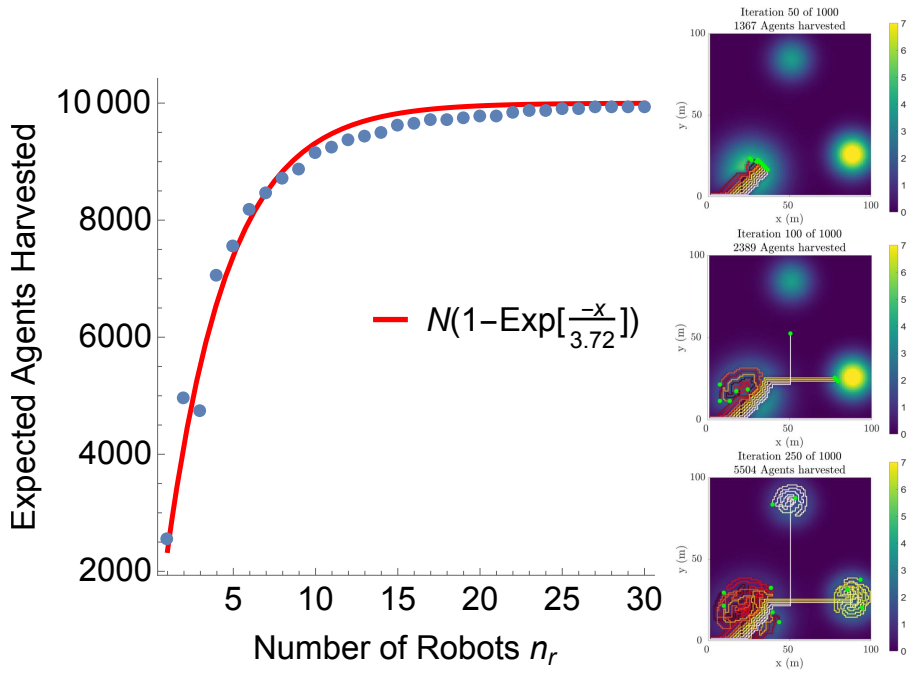

Figure 2.6: Simulations varying the number of robots with $N = 10,000$ agents, $1,000$ steps, $d = 3$ depth greedy search, and diffusion parameter $k = 0.05$. Subplots show the heat map of the agents, and the robot paths (in multi-colors) after 50, 100, and 250 steps.

2.6 Conclusions

From this research we presented an alternate coverage problem where the objects to be covered are moving agents that obey a Markov motion model and have a stationary distribution. We presented analytical results for a version of this problem with Gaussian distributions, and presented heuristic and greedy controllers that outperform standard coverage techniques. This research was motivated by current challenges in mosquito larvae control, but may have applications to commercial fishing, pesticide treatments, or steering a predator to maximize the number of prey harvested when the predator has a limited coverage footprint and the prey obey a Markov motion model. For future work, we plan on expanding the h-step horizon heuristic controller to multi-robot applications, where every n number of simulations steps, it re-evaluates the expected return for re-distributing the robots across the world. We also plan to enable the controller to avoid overexploitation by choosing the amount of time it spends at a goal. Finally, we plan on

verifying the results seen in the simulations through hardware experiments.

Chapter 3

Destructive Survey of Mosquitoes

This chapter focuses on the research performed for the 2018 ICRA paper "Using A UAV for Destructive Surveys of Mosquito Population", by A. Nguyen, D. Krupke, M. Burbage, S. Bhatnagar, S.P. Fekete, and A.T. Becker [31].

My role in this research was to construct a electrified screen that safely zapped mosquitoes we were interested in, and not harm other insects in that particular area. I also designed the frame, the body of the net and 3D printed attachments to connect to the UAV. I also participated in conducting the wind tunnel experiments to test the most effective angle for the screen, and presented the research in Brisbane Australia.

3.1 Overview and Related Work

Robotic Coverage: Robotic coverage has a long history. The basic problem is one of designing a path for a robot that ensures the robot visits within r distance of every point on the workspace. For an overview see [3]. This work has been extended to use multiple coverage robots in a variety of ways, including using simple behaviors for the robots [38, 39].

Mosquito Control Solutions: Mosquito control also has a long history of efforts associated both with monitoring mosquito populations [40] and with eliminating mosquitoes. Mosquito control involves both draining potential breeding grounds and destroying living mosquitoes [41]. An array of insecticidal compounds has been used with different application methods, concentrations, and quantities, including both larvicides and compounds directed at adult mosquitoes [42].

Various traps have been designed to capture and/or kill mosquitoes with increasing sophistication in imitating human bait, as designers strive to achieve a trap that can rival the attraction of a live human [43]. In recent history, methods have also included genetically modifying mosquitoes so that they either cannot reproduce effectively or cannot transmit diseases successfully [44], and with the recent genomic mapping of mosquito species, new ideas for more targeted work have been formulated [45].

Popular methods to control mosquitoes such as insecticides are effective, but they have the potential to introduce long-term environmental damage and mosquitoes have demonstrated the ability to become resistant to pesticides [46]. Traditional electrified screens (bug zappers) use UV light to attract pests but have a large bycatch of non-pest insects [47].

Robotic Pest Management: As GPS technology has flourished and data processing has become cheaper and more readily available, researchers have explored options for implementing the new technologies in breeding ground removal [48] and more effective insecticide dispersion [49]. Low-cost UAVs for residential spraying are under development [50]. Even optical solutions have been considered, including laser containment [51] or, by extension, exclusion and laser tracking and extermination [52].

3.2 Hardware Design

This section examines the components of the mosquito UAV system, shown in Fig. 1.2. This includes the UAV, electrified screen, surveying electronics, and a discussion of the energy budget.

3.2.1 UAV

The UAV is a custom-built, 177 cm wingspan hexacopter, controlled by a Pixhawk flight controller running ArduPilot Mega flight software. The UAV has a 3DR GPS

module using the UBlox NEO-7 chipset.

3.2.2 Screen design

The mosquito screen is designed to eliminate high density mosquito populations. This screen was constructed from two expanded aluminum mesh panels, spaced apart by 3 mm thick ABS grid. These mesh panels have 12 mm diamond-shaped openings, and is held taught by nylon bolts around the perimeter. The bottom mesh panel is offset by half a diamond (6 mm) to the right to ensure all insects greater than 6 mm cannot pass through the net. The top mesh is held at the reference voltage and the bottom mesh is energized to 1.8 kV above the reference voltage.

The perimeter is reinforced by two sets of 7 mm diameter fiberglass rods that are inset into 3D printed corner fixtures. These rods protect the frame from getting damaged from any side, and allows the UAV to land without damaging the net.

Once assembled, the net weighs 0.948 kg and has an overall area of 0.194 m^2, with the spacer occupying 0.0325 m^2. This makes the effective net area 0.161 m^2.

3.2.3 Screen location

The UAV carries the bug-zapping screen, which is suspended by paracord rope at each corner. The location of this screen determines the efficacy of the mosquito UAV, measured in mosquitoes detected per second of flight time. The following describes a simplified analysis to optimize the screen location.

For manufacturing ease, the electrified screen is a rectangle with a width of d_s. The screen is suspended a distance h_s beneath the UAV flying at height h_d. We chose to suspend the screen beneath the UAV to avoid the weight of the rigid frame that would be required if the screen were above the UAV and because most mosquito species prefer low flight [53]. This screen can be suspended at any desired angle θ in comparison to

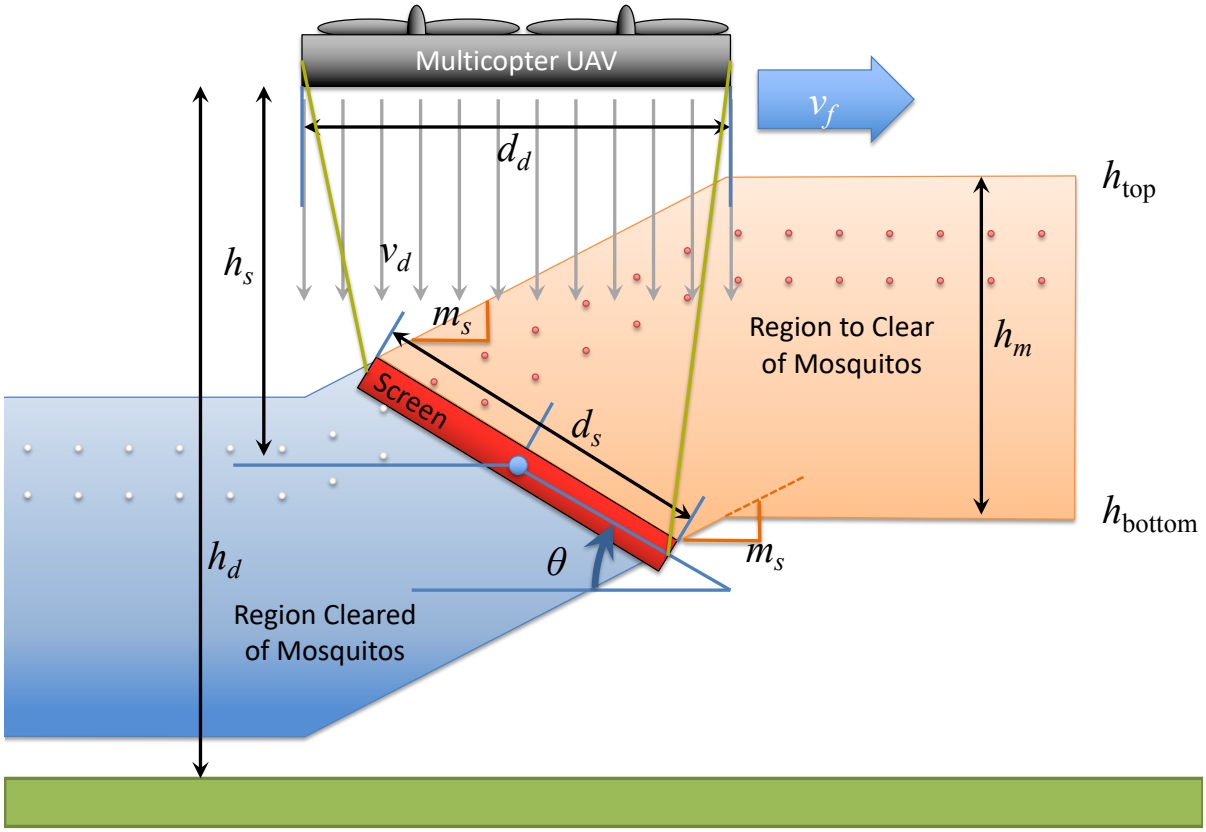

Figure 3.1: The UAV suspends a rectangular bug-zapping screen beneath it. Propwash pushes incoming mosquitoes downwards, and the UAV clears a volume $h_m \times d_s \times v_f$ each second. Circles show two mosquitoes at equal time intervals relative to the UAV.

horizontal, as shown in Fig. 3.1. Two key parameters are the distance h_s and the optimal angle θ. The goal is to clear the greatest volume of mosquitoes per second, a volume defined by the UAV forward velocity v_f and the cross-sectional area $h_m \times d_s$ cleared by the screen, as shown in Fig. 3.2.

To hover, the UAV must push sufficient air down with velocity v_d to apply a force that cancels the pull of gravity. The UAV and screen combined have mass m_d and its cross section can be approximated as a square with a side length of d_d. The mass flow of air through the UAV's propellers is equal to the product of the change in velocity of the air, the density of the air ρ_a, and the cross sectional area.

We assume that air above the UAV is quiescent, so the change in velocity of the

air is v_d m/s.

$$\text{Force gravity} = (\text{mass flow}) \cdot \text{air velocity},$$

$$m_d \cdot g = (v_d \cdot \rho_a \cdot d_d^2) \cdot v_d. \tag{3.1}$$

Then the required *propwash*, the velocity of air beneath the UAV, for hovering is

$$v_d = \sqrt{\frac{m_d g}{\rho_a d_d^2}}. \tag{3.2}$$

The flight testing site in Houston, Texas is 15 m above sea level. At sea level the density of air ρ_a is $1.225\,\mathrm{kg/m^3}$. The UAV and instrumentation combined weigh 5.1 kg with a width of 0.75 m. The acceleration due to gravity is $9.871\,\mathrm{m/s^2}$. Substituting these values gives $v_d = 8.5\,\mathrm{m/s}$.

Due to propwash, an initially hovering mosquito will fall when under the UAV at a rate of v_d. Relative to the UAV, the mosquito moves horizontally at a rate of $-v_f$. As shown in Fig. 3.1, we can extend lines with slope $-v_d/v_f$ from the screen's trailing edge to h_{top} and from the leading edge to h_{bottom}.

$$h_{\text{top}} = h_d - h_s + \frac{d_s}{2}\sin(\theta) + \frac{d_d + d_s\cos(\theta)}{2}\frac{v_d}{v_f},$$

$$h_{\text{bottom}} = h_d - h_s - \frac{d_s}{2}\sin(\theta) + \frac{d_d - d_s\cos(\theta)}{2}\frac{v_d}{v_f},$$

$$h_m = h_{\text{top}} - h_{\text{bottom}} = d_s\left(\frac{v_d}{v_f}\cos(\theta) + \sin(\theta)\right). \tag{3.3}$$

The optimal angle is therefore a function of forward and propwash velocity:

$$\theta = \arctan\left(\frac{v_f}{v_d}\right). \tag{3.4}$$

To ensure the maximum number of mosquitoes are collected, the screen must be sufficiently far below the UAV $h_s > \frac{d_s}{2}\sin(\theta) + \frac{d_d + d_s\cos(\theta)}{2}\frac{v_d}{v_f}$ and the bottom of the screen must not touch the ground, $h_d > h_s + \frac{d_s}{2}\sin(\theta)$.

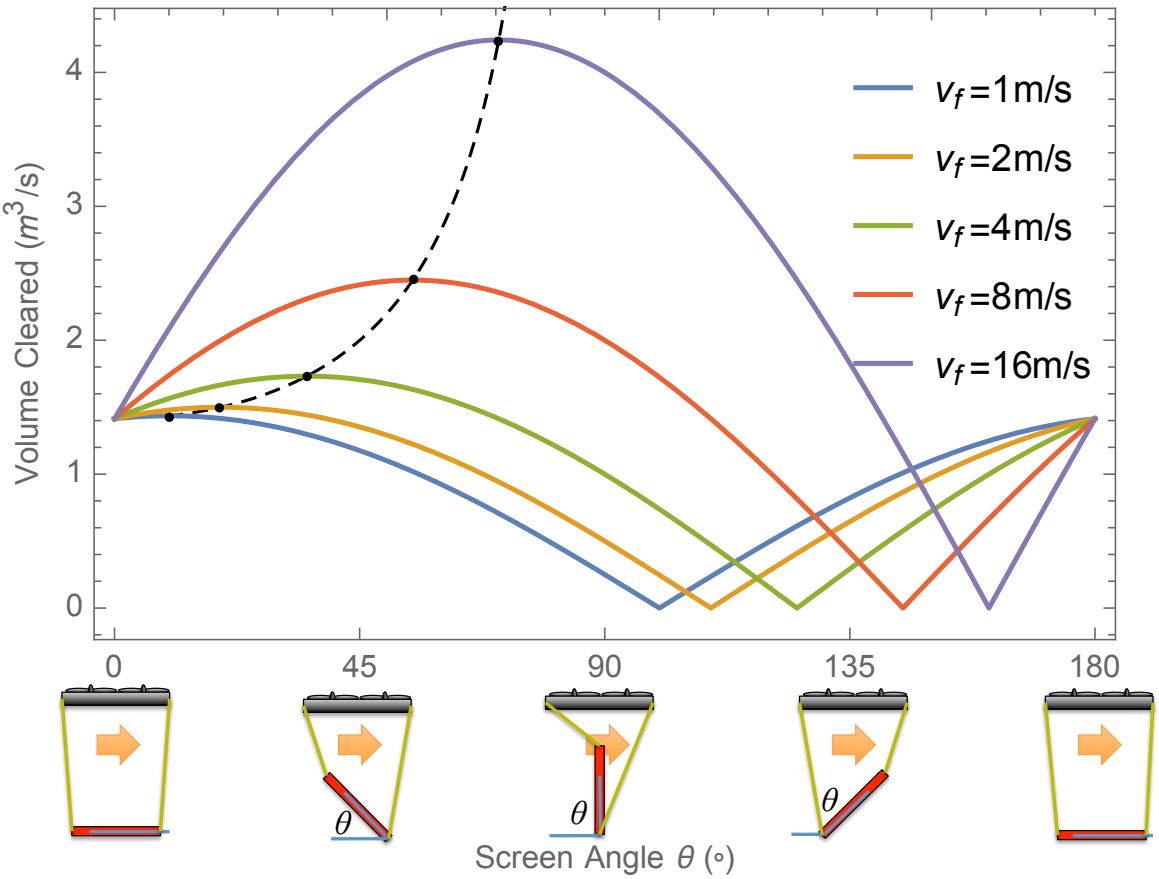

Figure 3.2: The volume cleared by a UAV is a function of screen angle θ and forward velocity v_f. Dotted line shows the optimal angle given in (3.4).

There are practical limits to h_s as well. Tests with $h_s > 2\,\text{m}$ were abandoned because the long length caused the screen to act as a pendulum, introducing dynamics that made the system difficult to fly.

Changing the flying height h_d of the UAV will target different mosquito populations because mosquitoes are not distributed uniformly vertically. Gillies and Wilkes demonstrated that different species of mosquitoes prefer to fly at different heights [53].

3.2.4 Wind tunnel verification of net angle

This section describes experiments run in a wind tunnel to verify the simplified net angle analysis in the previous section. Smoke streaklines were used to visualize the flow

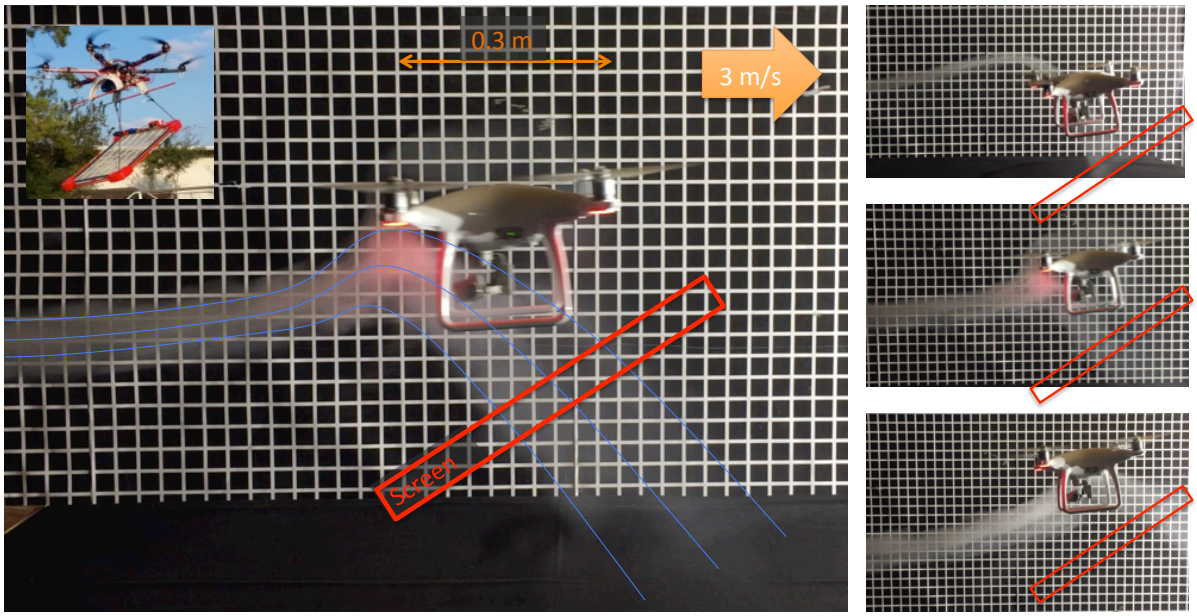

Figure 3.3: Frames from wind tunnel test with free-flying UAV at 3 m/s windspeed with smoke for streaklines [1]. As shown in the frames at right, the proposed screen position (in red) captures free flowing air and air entrained by the UAV propellers. Each black square is 25.4 mm in width.

of air as it passed by the UAV. Due to space constraints in the wind tunnel, a free-flying phantom 4 was used instead of the hexacopter used for carrying the zapper. The wind tunnel was set to a 3 m/s flow speed, and the UAV manually flown in approximately stable hovering. The solo UAV is $0.3 \times 0.3 \times 0.2$ m. The windtunnel has a $1\,\text{m} \times 1\,\text{m}$ cross section. As seen from Fig. 3.3, the proposed screen position captures free flowing air and air entrained by the UAV propellers. This test encouraged us to mount the net as close to the UAV as possible, so that air, and flying mosquitos, entrained by the propellers are pushed into the net.

3.2.5 Data logger

The electrical detection and logging system is powered by a 9 V lithium ion battery applied directly to the controller and two AA 3 V lithium ion batteries applied to the power circuit for the screen. The controller uses a GPS shield for monitoring the location

and altitude as well as a real time clock to timestamp each data point collected from the system. A Raspberry Pi 3 is used for data logging, sensors include a GPS sensor (NEO-6M Ublox), a capacitive humidity sensor, a thermistor (DHT22), and an INA219 high side, 12-bit DC current sensor for monitoring the supply-side current delivered to the net. The net current draw is logged at 100 Hz, while GPS and weather sensor data is logged at 1Hz. All data is stored on an onboard SD card.

3.2.6 Energy budget

Tests with an oscilloscope show that in the steady state, a 30.5 cm × 61 cm screen and electronics have a power consumption of 3.6 W. During a zap, the screen voltage monitoring circuit shorts briefly when the mosquito contacts the screen. Figure 3.4 shows the time sequences for battery and screen voltages, current, and power during five mosquito zaps. Multiplying voltage by current to find the instantaneous power ($p = iv$) and integrating the area under the power curve show a total energy consumption of 4.2 mJ for each zap. Recharging the screen requires more power and is represented in the latter part of the curves. The overall recovery time is about 160 ms. Most of the energy is consumed charging and maintaining the charge on the screen rather than in zapping the mosquitoes.

3.3 Path Planning

3.3.1 Modelling

The data on the distribution of mosquitoes is given for a two-dimensional grid environment; the grid size is induced by the size of the screen, the available data and the desired resolution of the extracted map. For each pixel $p_i \in P$, we are given a relative value $c(p_i)$ that describes the estimated a-priori density of mosquitoes, based on data obtained from boustrophedon (back and forth) scans of the area by the UAV; this

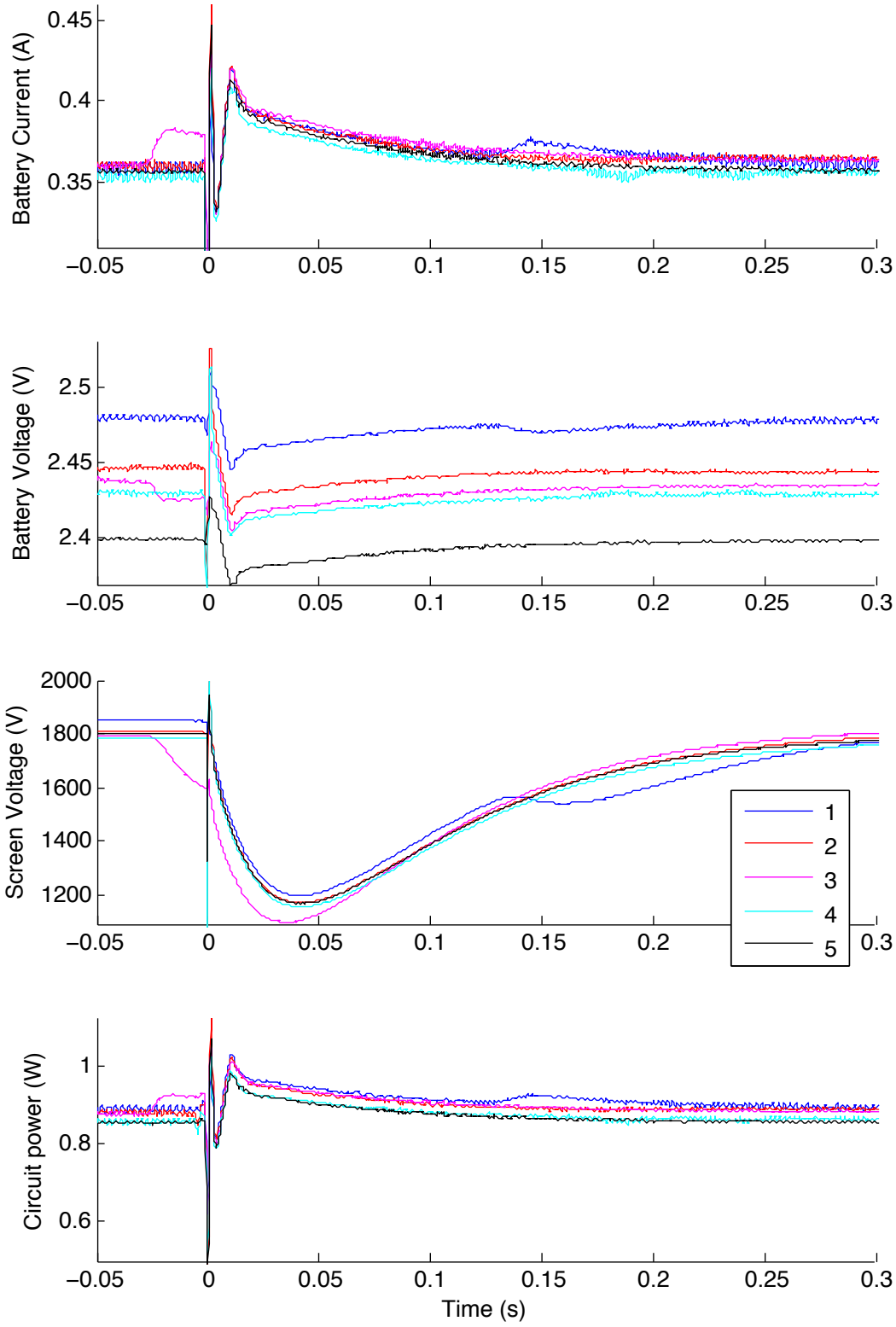

Figure 3.4: Current, voltage, and power traces for five *Culex quinquefasciatus* mosquitoes as each contacts the bug-zapping screen at $t = 0$. Contact causes a brief short that recovers in 160 ms.

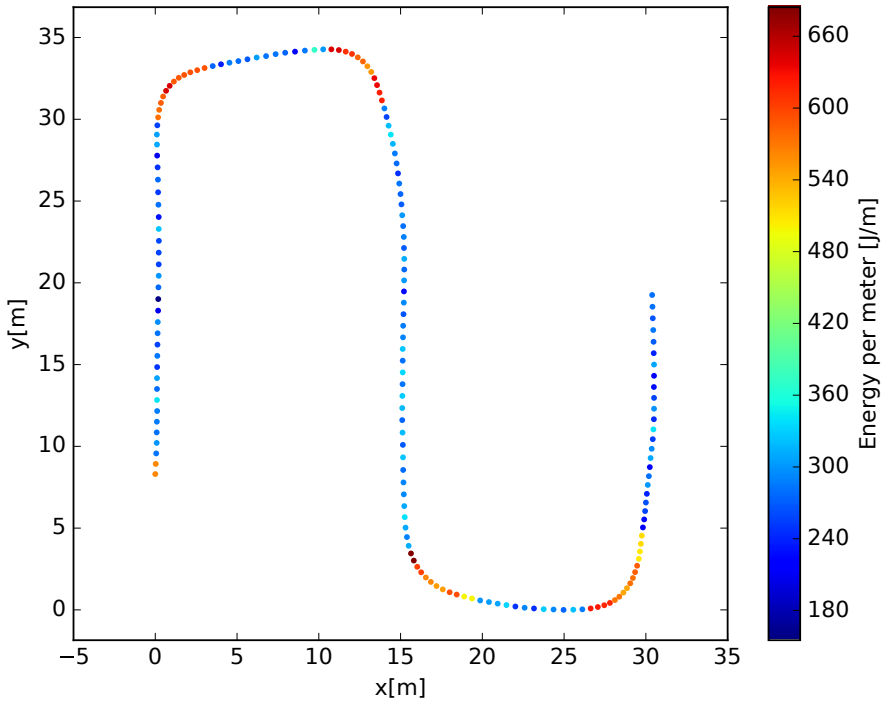

Figure 3.5: Turns are expensive. See our related video at `https://youtu.be/SFyOMDgdNao` for details, and [34] for an accompanying abstract.

implies that only a subset of pixels carry a significant value. Visiting one of the pixels corresponds to sampling and mapping the actual density distribution of mosquitoes. For a dense distribution of mosquitoes (which is the case for the instances relevant for pest control), multiple visits to the same pixel do not contribute additional knowledge. As a consequence, the objective is to maximize the sampling value of the set $S \subseteq P$ of visited pixels, i.e., $\max_{S \subseteq P} \sum_{p_i \in S} c(p_i)$ within the available battery capacity; this may be over the course of a single closed trajectory, or over a combination of multiple roundtrips.

Planning good trajectories for a UAV is not subject to the same curvature constraints of an ordinary aircraft because UAVs can turn on the spot. However, turns are a critical aspect of path planning due to their impact on energy consumption. Battery capacity is *the* limiting factor for UAV flight time. As shown in Fig. 3.5, the power output for a desired trajectory is non-uniform. Flying along a straight path is relatively inexpensive but turning is energy intensive.

As a consequence, we must consider the total turn costs associated with changing direction, as measured by the turn angle. As we are not limited by trajectory curvature, we refer to straight-line connections and a finite set of 2ω different headings for visiting vertices. For the most natural case of orthogonal grids $\omega = 2$. When surveying non-isolated mosquito hotspots (whose size greatly exceeds the size of the UAV), we are not dealing with isolated pixels and the modeling error of this restriction is small.

Now we consider different trajectory types. A *cycle* is a roundtrip of a subset $S \subseteq P$ that visits all points in S and returns to the origin, a *cycle cover* of P is a set of cycles that together visit all points in P, and a *tour* is a single cycle that visits all points in P. A *subset cycle cover* for $S \subset P$ is a cycle cover that covers at least the points in S, while a *subset tour* is a tour of at least the points in S. For any of these structures, we are interested in cycle covers or tours of *minimum total travel cost*. The travel/battery cost is a linear combination of the number of pixel transitions (distance) and the weighted number of turns, corresponding to the total turn angle. In addition, a *minimum turn-cost penalty cycle cover* or a *minimum turn-cost penalty tour* visits a subset $R \subset P$, such that the sum of total travel cost and the sum $\sum_{i \notin R} c(p_i)$ of values of unvisited pixels is minimized.

3.3.2 Computational complexity

Finding optimal covering paths that map a given region is closely related to the famous *Traveling Salesman Problem (TSP)*, which asks to minimize total length of a single tour that covers all of a given set of locations. The TSP is one of the classic NP-hard problems, so we cannot expect a general method that finds a provably optimal solution for any instance in polynomially bounded time. A generalization of the TSP is the *Lawnmower Problem* (see Arkin et al. [54], which considers coverage by a tool of nontrivial size. For the objective of minimizing the total cost (in particular, the turn cost), Arkin et al. [55] showed that finding minimum-turn tours in grid graphs is NP-hard,

even if a minimum-turn cycle cover is given. The complexity of finding a set of multiple cycles that cover a given set of locations at minimum total turn cost had remained elusive for many years; *Problem 53* in *The Open Problems Project* asks for the complexity of finding a minimum-cost (full) cycle cover in a 2-dimensional grid graph. This is not obvious: large parts of a solution can usually easily be deduced by local information and 2-factor techniques. Arkin et al. showed [55, 56] that the full coverage variant in *thin* grid graphs (which do not contain a 2×2 square, so every pixel is a boundary pixel) is solvable in polynomial time. In separate work [57], two of us were able to resolve this issue by showing that finding a cycle cover of minimum turn cost is NP-hard.

3.3.3 Mathematical optimization

A powerful approach for finding optimal solutions to instances of NP-hard problems is the use of Integer Programming (IP). While solving an IP still requires exponential time in the worst case, using carefully crafted mathematical models in combination with specific algorithm engineering and available IP solvers enables solving instances of considerable size to provable optimality. For our purposes, we can describe the problem as follows.

Penalty Cycle Covers The set P of pixels corresponds to a given grid graph $G(P, E)$ in which each pixel $p_j \in P$ is adjacent to the set $N(p_j)$ of pixels in P that share an edge with p_j. Each vertex $p_j \in P$ has a scalar reward $c(p_j)$ for visiting (or penalty for not visiting), and a function $\text{cost}_j(i, k) \in \mathbb{Z}_0^+$ that maps the cost of traveling from p_i to p_j to p_k, where $p_i, p_k \in N(p_j)$ are adjacent pixels to p_j. This cost is symmetric, i.e. $\text{cost}_j(i, k) = \text{cost}_j(k, i)$. The integer program uses two types of variables: integer variables $x_{ijk} = x_{kji}$ that state how often passage $p_i - p_j - p_k$ or $p_k - p_j - p_i$ is used and Boolean variables y_j that indicate that the pixel $p_j \in V$ is not covered, i.e., the penalty

is paid. This results in the following formulation:

$$\min \sum_{p_j \in P} \sum_{p_i, p_k \in N(p_j)} \text{cost}_j(i,k) \cdot x_{ijk} + \sum_{p_j \in P} c(p_j) \cdot y_j. \tag{3.5}$$

with constraints

$$1 \leq 4 \cdot y_j + \sum_{p_i, p_k \in N(p_j)} x_{ijk} \leq 4, \quad \forall p_j \in P, \tag{3.6}$$

$$2x_{jij} + \sum_{p_k \in N(p_i), p_k \neq p_j} x_{jik} = 2x_{iji} + \sum_{p_k \in N(p_j), p_k \neq p_i} x_{ijk} \quad , \quad \forall \{p_i, p_j\} \in E, \tag{3.7}$$

$$x_{ijk} \in \mathbb{N}_0, y_j \in \mathbb{B}, \quad \forall p_j \in P, \{p_i, p_k\} \subseteq N(p_j). \tag{3.8}$$

The objective function in Eq. 3.5 minimizes the total cost of the cycles and the uncovered pixels. Eq. 3.6 enforces a pixel to be covered or the *not covered* variable to be set to `true`. Arkin et al. [55] showed that no pixel needs to be visited more than four times, otherwise a simple local optimization can be performed. Eq. 3.7 enforces the transitions between two adjacent pixels to match. Eq. 3.8 enforces that the variables are integers or booleans.

We can solve a wide spectrum of instances with different kinds of probability distributions up to a size of 1500 pixels to provable optimality. Optimal solutions for different densities scalings of an instance with 1783 pixels are shown in Fig. 3.6. To solve larger instances the optimality constraint can be relaxed or the grid graph can be split and the subgraphs solved separately.

Tours Computing a minimum cycle cover may result in several subcycles that need to be visited separately, which is appropriate for the use of several UAVs or when several separate roundtrips by the same UAV are convenient. If we want to determine connected roundtrips by a single UAV, we need to connect the components of a cycle cover to a

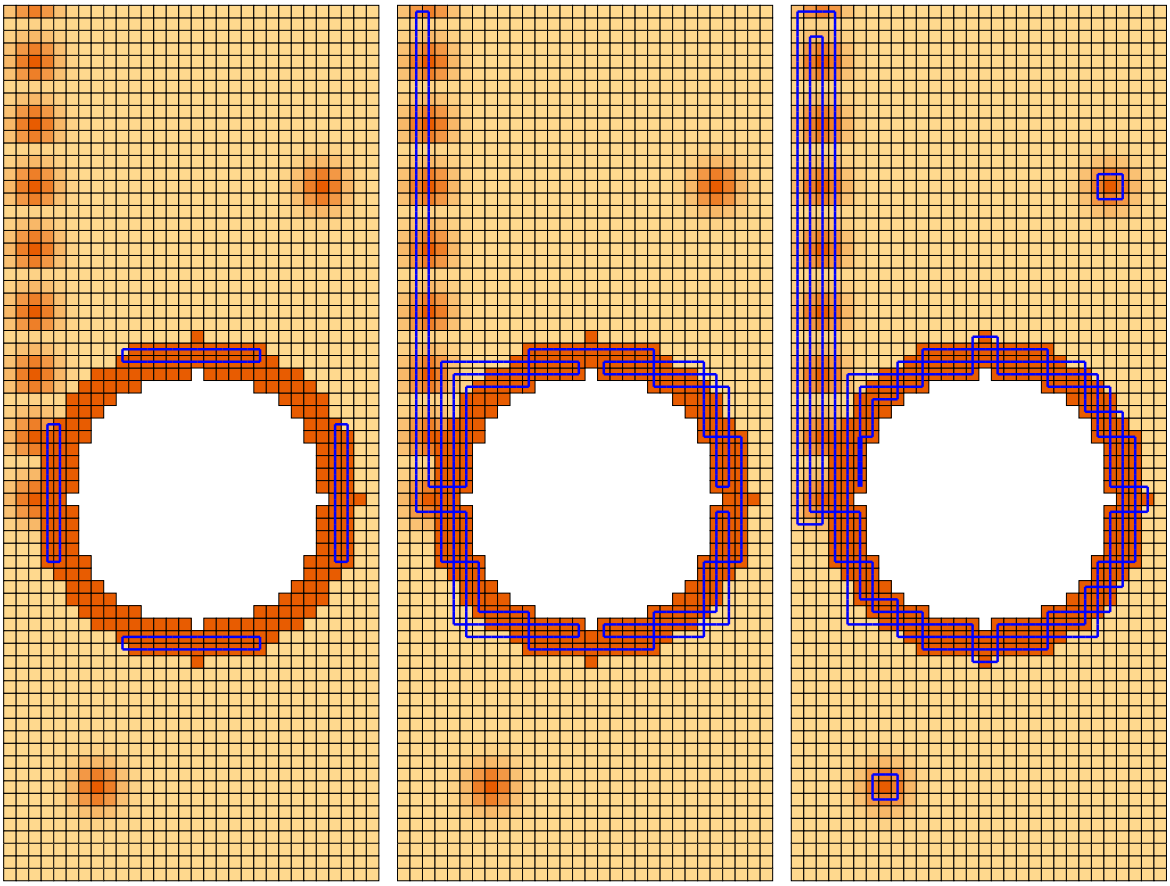

Figure 3.6: Optimal cycle covers with different density scaling. The middle has twice and the right instance has four times the density as the left. In these instances, the cost of a 90° is five times that of a straight pixel transition.

tour. This can be achieved via integer programming by adding additional constraints for separating these *subtours*.

This separation of subtours is more complicated than for the classic TSP because there may be tours that cross but are not connected. Instead of connecting two subtours, one subtour can also be discarded.

We first consider a constraint (Eq. 3.9) that is able to separate any given solution with multiple subtours. Let Q be the pixels of a selected subtour. Let $p_\ell \in Q$ be a pixel with high density and no other subtours crossing it and $p_{\ell'} \notin Q$ be another covered pixel with high density. These two pixels are used for 'defusing': if one of them is no longer covered, the constraint is automatically fulfilled. We denote by Q_s the pixels that are

covered only by straight paths in the subtour. $T(p_j)$ describes the turn variables of a pixel p_j. x' refers to the variable assignment in the current solution.

$$1 \leq y_\ell + y_{\ell'} + \sum_{p_i,p_k \in N(p_\ell), x'_{i\ell k}=0} x_{i\ell k} + \sum_{t \in T(v), v \in Q_s - p_\ell} t + \sum_{p_j \in Q \setminus (Q_s + p_\ell), p_i \neq p_k \in N(p_j), x'_{ijk}=0} x_{ijk}. \tag{3.9}$$

While this constraint suffices for capturing the mathematical conditions, its practical performance is unsatisfactory for connecting distant subtours. A better approach is described in the following; this is not always sufficient but more efficient in connecting distant cycles. We use the same definitions as for the previous constraint, but consider an additional set Q' that is a superset of Q. Q are the pixels of a subtour and p_ℓ is a valuable pixel. Q' is a superset of Q (possibly equal to Q). $p_{\ell'}$ is a valuable pixel outside of Q'. The constraint enforces that either p_ℓ or $p_{\ell'}$ is uncovered, or there is a path on the margin of Q' that connects p_ℓ and $p_{\ell'}$.

$$y_\ell + y_{\ell'} + \sum_{x \in \text{Leaving}(Q')} x \geq 1. \tag{3.10}$$

We use two ways to choose Q' for these subtour elimination constraints: $Q' = Q$ which is similar to the classical TSP constraint or Q' is the Voronoi cell of the subtour.

3.3.4 Computational results

We evaluated the effectiveness of our optimization method by testing it on a suite of benchmark instances based on random natural grid graphs with random densities; the probability of a pixel to be added during test instance generation is correlated with its neighborhood, resulting in smoother boundaries which are more natural than purely random instances. The tests were carried out for 10 instances for each size in the range up to 1.400 pixels. We used modern desktop computers equipped with an *Intel(R) Core(TM)*

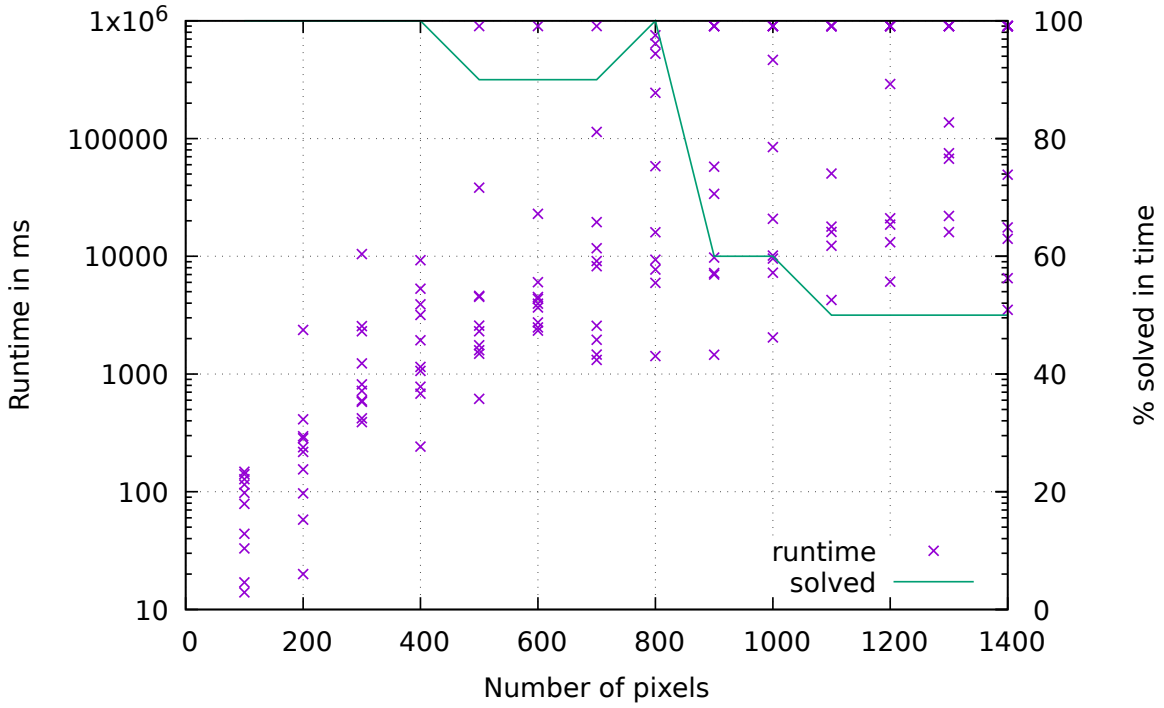

Figure 3.7: Runtime of solving benchmark instances to optimality. Shown are the times of ten instances for each size, with a timeout at 900 s, as well as the percentage of solved instances. Only the number of turns is minimized in these instances.

i7-6700K CPU @ 4.00 GHz and 64 GB of RAM. The integer programs were computed with CPLEX version 12.5.0.0 and the parameters `EpInt=0`, `EpGap=0`, `EpOpt=1e-9`, and `EpAGap=0`. Fig. 3.7 shows runtimes for solving penalty cycle cover to optimality. Instances that took longer than 15 min were aborted. As shown in the figure, even at 1400 pixels we were still able to solve half of the instances to provable optimality. Even for the aborted instances, the computed solutions were within a few percentage points of the provable lower bound, meaning that they were nearly optimal.

Fig. 3.8 shows an example of iteratively computing an optimal tour with the described integer program. This example took less than a minute of total computing time. It assumed that 90° turns cost five times as much as a straight pixel transition (distance).

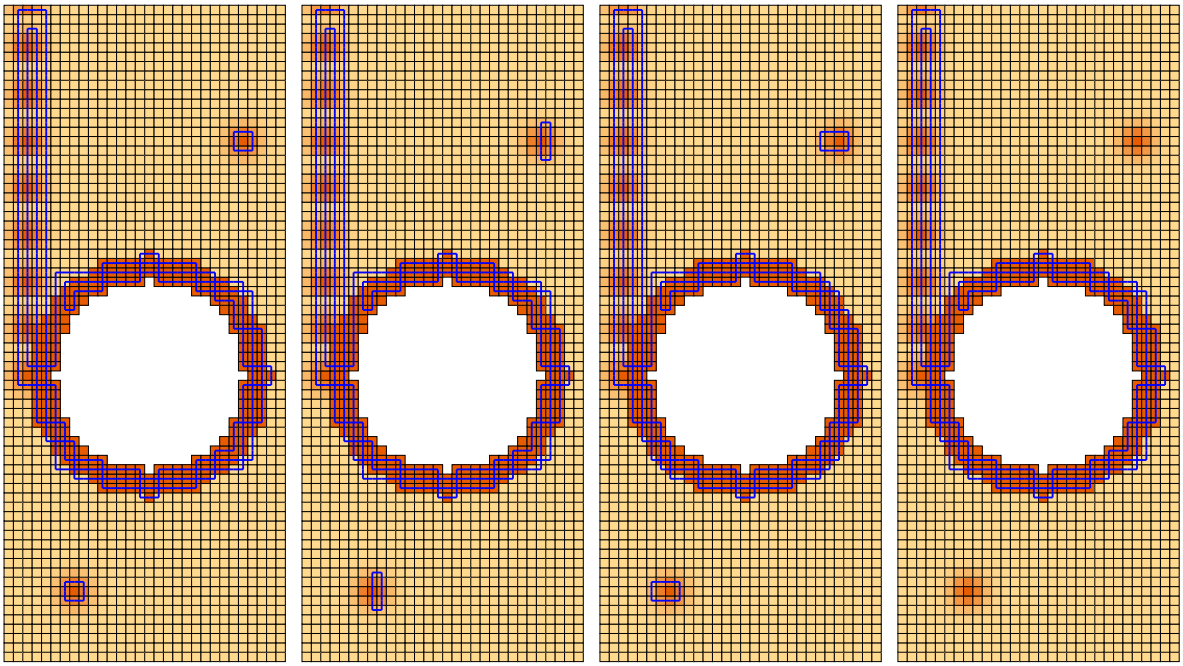

Figure 3.8: Left is an optimal penalty cycle cover. Cycles (blue) cover all areas with high density. After three applications of the tour constraints, a single cycle remains (right). In the intermediate solutions, the subcycles first try to evade the new constraints by reshaping. The final tour omits two of the small hotspots because the cost of integrating them into the single tour is prohibitively expensive.

3.4 Experiments

The results for representative flights are described below. Figure 3.9 compares the energy consumption for three coverage schemes for a region including a large obstacle in the center. A boustrophedon path requires 50 turns, 187 kJ, 160 s, and 181 m. A hand-designed path requires 45 turns, 214 kJ, 155 s, and 178 m. A path computed using the optimal penalty cycle cover requires only 33 turns, 184 kJ, 133 s, and 176 m.

A boustrophedon (back-and-forth) path with 2 m spacing was generated to cover a region 120 m × 15 m at height 1.5 m. The path was generated using Mission Planner software from ardupilot.org [58].

For each trial the UAV took off from a resting position on top of the screen. Flight began manually, with a piloted takeoff of the UAV. After establishing a stable hover at

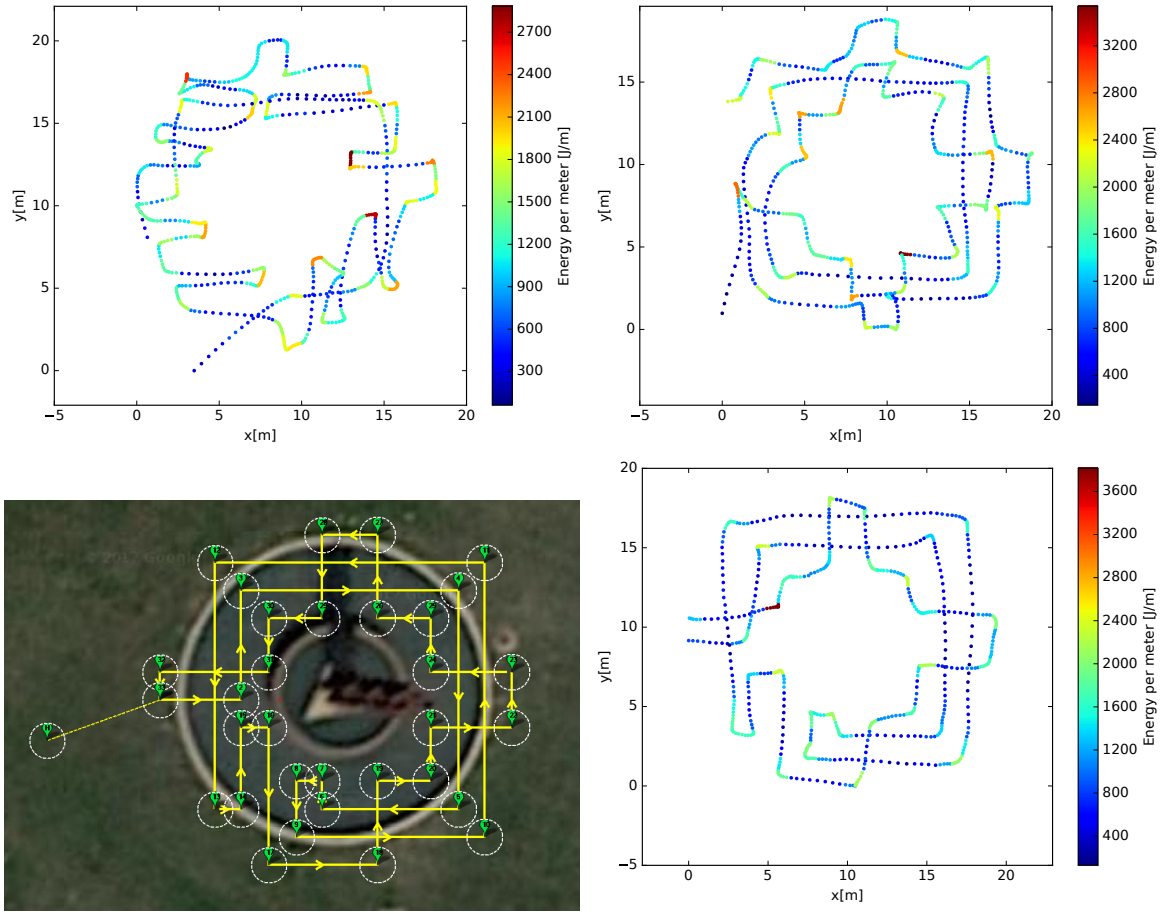

Figure 3.9: Paths for an environment surrounding a fountain, which poses an obstacle for the UAV. (Top left) The energy consumption during a real world flight for a boustrophedon path. (Top right) The energy consumption during a real world flight for a hand-designed loop path. (Bottom left) The optimal penalty cycle cover path. (Bottom right) The energy consumption during a real-world flight along the optimal path.

3 m, control was switched to the autonomous flight plan. The pilot monitored the flight with the ability to switch to manual operation in case of potential crashes due to GPS error or hazards in the flight plan. Mosquito strikes detected by the data logger were verified using a GoPro Hero 4 Silver camera attached at the top of the net, as shown in Fig. 1.2. At night and twilight, the sparks could be detected both visually and audibly from the recorded video. During the day, the sparks were loud enough to observe over the audio channel of the videos.

Figure 3.10: The UAV's path for flight 3 is in red. Strikes collected along this path are represented by yellow dots.

The UAV flew eight missions on this field, covering the same path. It was mainly flown in the early morning and late afternoon, when mosquito activities are more active. Three flights were flown at noon and early afternoon to ensure that mosquito activities during these periods were not ignored. However, only two mosquito strikes were observed during this period. The path covered is about 1 km long and typically takes 12 min.

Over the eight missions on this field, there were a total of 11 mosquito strikes. Figure 3.10 shows the mission's flight path and the map of all collected strikes. The mosquito strikes are concentrated at the north and south ends of the field, where there are more trees. A density map was generated from the collected strikes' position by representing each strike by a Gaussian distribution with the norm on the strike's location and a σ of 10 m. Figure 3.11 shows the density map generated by summing these Gaussian distributions.

These results not only tell where mosquitoes were but also show where mosquitoes were not. This is a key difference from stationary traps such as [8, 9]. Figure 3.12 shows the UAV during a dawn flight test near the ocean.

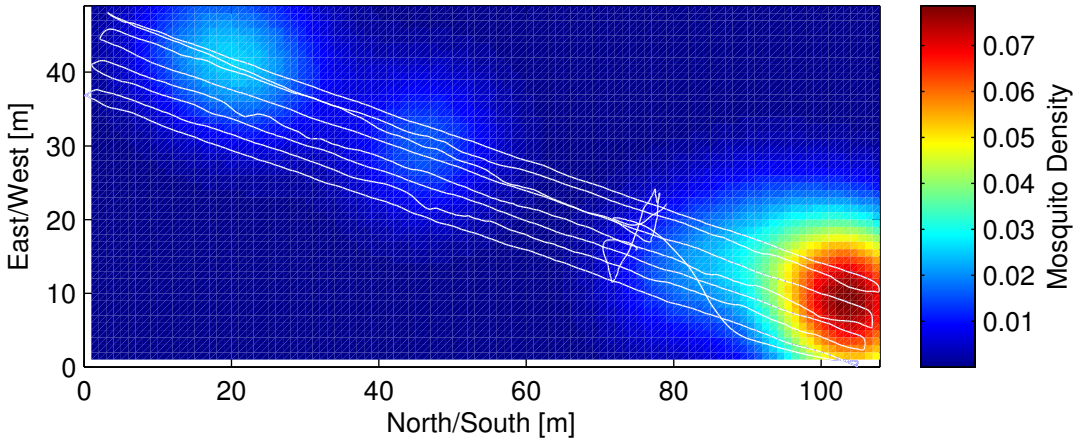

Figure 3.11: Density map showing mosquito distribution on the field, overlaid by flight path 4 in white.

Figure 3.12: The UAV and screen during a flight trial near the ocean.

3.5 Conclusion and Future Work

This chapter presented an approach for finding optimal tours given turn costs and an energy budget, inspired by a mosquito-killing UAV with limited battery life. Initial experiments with the UAV and electrified screen track the location of a mosquito-killing UAV as it patrols a field and maps mosquito kills.

Many refinements to the algorithm could be pursued in future work, including changes to both the mosquito-biasing algorithm and the robot flight simulation. The model may be expanded to continuous space, three dimensions, and to arbitrary turn angles. These and other considerations will make a more realistic model for future work.

Further testing of the multi-copter UAV is indicated and will allow more extensive testing of the robustness and accuracy of the hardware design. New sensors that can identify and detect flying insects [8] may be added to the UAV and enable it to proactively steer toward insect swarms and identify insects in realtime.

The concept may be extended to a non-destructive population survey in which the screen could be replaced with a net and, with appropriate lighting, the camera used to record capture events. Teams of UAVs could work together to map areas more quickly and, by measuring gradients of the distribution, quickly find large mosquito populations.

Chapter 4

Harvesting Mosquito Larvae with an Autonomous Robotic Boat

4.1 Overview and Related Work

Coverage tasks are well-suited to robot capabilities and have received commensurate attention from the robotic community. This chapter focuses on a variant of time-varying coverage where a large population of moving agents are collected whenever they are within the coverage footprint of the robot. The model chosen moves the particles probabilistically, positioning this problem, as illustrated in Table 1.1, between periodic coverage problems and pursuit-evasion problems. This line of research has much in common with visibility-based coverage tasks, where teams of agents ensure no adversaries are in a region by working to reduce the size of *contaminated regions* that may contain an adversary by clearing them and placing agents to ensure the area cannot become re-contaminated [22–24]. The major difference is that the adversary in our problem obeys a motion model that is predictable in the aggregate.

Previous research on coverage under uncertainty provided a "probably approximately correct" measure of coverage. This enabled a robot to generate policies that guarantee (to an arbitrary level of certainty) the coverage of a fraction of the free space [27]. Similarly, in this work, mosquito larva populations obey dynamics with large variance and we wish to design policies that reliably eliminate the larvae. This chapter is also related to work in persistent coverage of changing environments [59], and simultaneous coverage and tracking [25]. Since mosquito larvae can move towards different regions, areas previously cleared will need to be covered again over time.

Recent research on alternatives to mosquito control that incorporate robotics include the use of Unmanned Aerial Vehicles (UAV) with chemical larvicides [29], UAVs with controlled release of sterile adult male mosquitoes [30], and experiments in trajectory-planning for a UAV equipped with a mosquito-zapping electric screen [31].

Past work in mapping of mosquito larvae exploit satellite imagery and vision processing techniques to locate sections of a body of water where mosquito larvae are most likely to be found [32]. Our work also seeks to map the distribution for mosquito larvae, but directly measures this through field measurements onboard a robotic boat.

4.2 Physical Prototyping

A Larvasonic© remotely operated vehicle (ROV) is used in this work. The acoustic larvacide unit is placed underneath the ROV, which is manually operated. Modifications to the vehicle include a real time kinematic (RTK) GPS unit (HERE+ RTK), and a Pixhawk 2.1 for telemetry, control, and interfacing with the sensors. The ROV's thrusters are Blue Robotics T100 thrusters which provide a maximum forward thrust of 23.1 N and a maximum reverse thrust of 17.8 N at 12 volts. The ROV has an onboard GoPro camera that can record video from the front of the boat, to observe the effect of the Larvasonic on the mosquito larvae.

For the experiment in this work, the ROV operated in autonomous mode to follow preset GPS waypoints using QGroundControl software. The waypoints were set based on the coverage paths tested in the simulation, such as the boustrophedon and greedy policy based paths.

The Larvasonic© uses acoustic waves to kill larvae [60]. Larvae use an air-filled bladder called the dorsal tracheal trunk to aid in breathing and maintain buoyancy. The size of this bladder gives it an acoustic resonance frequency between 18-30 kHz, depending on the size and species of the mosquito larva. Because water is incompressible

and air is not, large acoustic pulses can rupture this bladder. The transducer produces pulses that cycle between 18 and 30 kHz with one frequency sweep per second.

Larvae within 3 meters of the device in a 25° arc in front of the transducer are instantly killed or severely damaged. As illustrated in Fig. 4.1, this places a limit of 1 m/s on the maximum speed of the boat and a rotational speed of 25°/s to ensure a continuous coverage region.

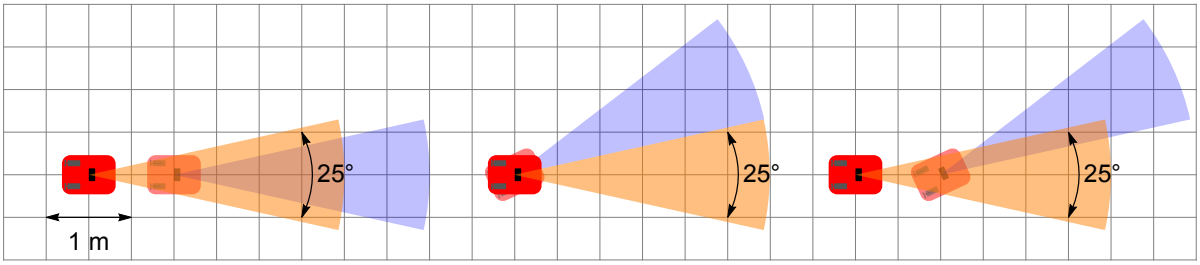

Figure 4.1: Schematic showing the kill zone for the Larvasonic© boat (kill zone at 0 s is orange and at 1 s is blue). In 1 s, the boat kills larvae within 3 meters of the device in a 25° arc in front of the transducer. Limiting the linear speed of the vehicle to 1 m/s and rotational speed to 25°/s ensures that approximately a 1×1 m^2 area in front of the boat is cleared of larvae.

The boat is intended to be placed in any body of water suspected to contain mosquito larvae, and follow a lawnmower-type survey path to map the undisturbed locations of mosquito larvae, as shown in Fig. 4.2. Then the harvesting algorithms presented in this work can be implemented to verify our simulation results.

4.3 Conclusions

This chapter presented an alternate coverage problem where the objects to be covered are moving agents that obey a Markov motion model and have a stationary distribution. We presented analytical results for a version of this problem with Gaussian distributions, and presented heuristics and greedy control policies that outperform standard coverage techniques. This chapter presented a technique to convert location data into a Markov model. This research was motivated by current challenges in mosquito

Figure 4.2: The boat was programmed to follow a boustrophedon path set by way points, which is indicated by the orange line. The red line with the arrow is the autonomous boat following the path as closely as possible.

larvae control, but may have applications to commercial fishing, pesticide treatments, or steering a predator to maximize the number of prey collected when the predator has a limited collection footprint and the prey obey a Markov motion model.

Chapter 5

Conclusion

In this thesis, many different types of controllers for area coverage problems have been described and studied. The controllers are first studied using simulations on MATLAB. Where a swarm of agents are represented by a mixture of Gaussians and the movement of these agents are represented by a Markov process. This means that as the robot makes its way through the population of the agents and clears areas based on the controller used, those clears can be recontaminated since the agents are able to move. If the environment was to be left alone after a few iterations, the agents would be more likely to reconstruct the distribution as it was before.

For future work, a better controller can be applied for multi-robot applications. In this new controller, as the distribution of agents changes there should be a check that looks to see if there any more peaks being created. Through this check, the robots can be re-assigned new goals to be more efficient. To further this research, the heuristic controller should be applied to an experiment and the performance should be studied in the real world.

References

[1] S. Bhatnagar, A. Nguyen, D. Krupke, S. P. Fekete, and A. T. Becker, "Uav for destructive surveys of mosquito population (video)," Feb. 2018. [Online]. Available: https://youtu.be/OTQSR03Bv5g

[2] A. Becker, "Robotic harvesting of a moving swarm," https://youtu.be/u1OTBK5kq70, 2018.

[3] H. Choset, "Coverage for robotics – a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1, pp. 113–126, Oct 2001. [Online]. Available: https://doi.org/10.1023/A:1016639210559

[4] C. J. Murray, L. C. Rosenfeld, S. S. Lim, K. G. Andrews, K. J. Foreman, D. Haring, N. Fullman, M. Naghavi, R. Lozano, and A. D. Lopez, "Global malaria mortality between 1980 and 2010: a systematic analysis," *The Lancet*, vol. 379, no. 9814, pp. 413–431, 2012.

[5] J. E. Parker, N. Angarita-Jaimes, M. Abe, C. E. Towers, D. Towers, and P. J. McCall, "Infrared video tracking of anopheles gambiae at insecticide-treated bed nets reveals rapid decisive impact after brief localised net contact," *Scientific Reports*, vol. 5, 2015.

[6] S. Butail, N. Manoukis, M. Diallo, A. S. Yaro, A. Dao, S. F. Traoré, J. M. Ribeiro, T. Lehmann, and D. A. Paley, "3d tracking of mating events in wild swarms of the malaria mosquito anopheles gambiae," in *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2011, pp. 720–723.

[7] G. M. Williams and J. B. Gingrich, "Comparison of light traps, gravid traps, and resting boxes for west nile virus surveillance," *Journal of Vector Ecology*, vol. 32, no. 2, pp. 285–291, 2007.

[8] Y. Chen, A. Why, G. Batista, A. Mafra-Neto, and E. Keogh, "Flying insect classification with inexpensive sensors," *Journal of Insect Behavior*, vol. 27, no. 5, pp. 657–677, 2014.

[9] A. Linn, "Building a better mosquito trap," *International Pest Control*, vol. 58, no. 4, p. 213, 2016.

[10] WHO, "A global brief on vector-borne diseases," *World Health Organization*, pp. 1–56, 2014.

[11] R. J. Peter, P. Van den Bossche, B. Penzhorn, and B. Sharp, "Tick, fly, and mosquito control - lessons from the past, solutions for the future," *Veterinary Parasitology*, vol. 132, pp. 205–215, 2005.

[12] D. Roberts, "Mosquito larvae change their feeding behavior in response to kairomones from some predators," *Journal of Medical Entomology*, vol. 51, no. 2, pp. 368–374, 2014.

[13] A. N. Clements, *The Biology of Mosquitoes*. CABI Publishing, 1999.

[14] J. R. Linley, "Swimming behavior of the larva of culicoides variipennis (diptera: Ceratopogonidae) and its relationship to temperature and viscosity," *Journal of Medical Entomology*, vol. 23, no. 5, pp. 473–483, 1986.

[15] I. Thomas, "The reactions of mosquito larvae to regular repetitions of shadows as stimuli," *Australian Journal of Biological Sciences*, vol. 3, no. 1, pp. 113–123, 1950.

[16] D. Roberts, "Mosquito larvae change their feeding behavior in response to kairomones from some predators," *Journal of Medical Entomology*, vol. 51, no. 2, pp. 368–374, 2014.

[17] A. K. Awasthi, J. C. Molinero, C.-H. Wu, K.-H. Tsai, C.-C. King, and J.-S. Hwang, "Behavioral changes in mosquito larvae induced by copepods predation," *Hydrobiologia*, vol. 749, no. 1, pp. 113–123, 2015.

[18] W. Nachtigall, "Locomotion: swimming (hydrodynamics) of aquatic insects," *The physiology of Insecta*, vol. 2, pp. 255–281, 1974.

[19] M. Burrows and M. Dorosenko, "Rapid swimming and escape movements in the aquatic larvae and pupae of the phantom midge chaoborus (diptera, chaoboridae)," *Journal of Experimental Biology*, pp. jeb–102 483, 2014.

[20] M. Laird, *The Natural History of Larval Mosquito Habitats*. Academic Press, 1988.

[21] T. H. Chung, G. A. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics," *Autonomous Robots*, vol. 31, no. 4, p. 299, Jul 2011. [Online]. Available: https://doi.org/10.1007/s10514-011-9241-4

[22] A. Kolling and A. Kleiner, "Multi-UAV motion planning for guaranteed search," in *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 79–86.

[23] A. Kleiner and A. Kolling, "Guaranteed search with large teams of unmanned aerial vehicles," in *Robotics and Automation (ICRA), 2013 IEEE International Conference*. IEEE, 2013, pp. 2977–2983.

[24] N. M. Stiffler and J. M. O'Kane, "Complete and optimal visibility-based pursuit-evasion," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 923–946, 2017.

[25] L. C. Pimenta, M. Schwager, Q. Lindsey, V. Kumar, D. Rus, R. C. Mesquita, and G. A. Pereira, "Simultaneous coverage and tracking (SCAT) of moving targets with robot networks," in *Algorithmic Foundation of Robotics VIII*. Springer, 2009, pp. 85–99.

[26] D. E. Soltero, M. Schwager, and D. Rus, "Generating informative paths for persistent sensing in unknown environments," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 2172–2179. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6385730/

[27] C. Das, A. Becker, and T. Bretl, "Probably approximately correct coverage for robots with uncertainty," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference*. IEEE, 2011, pp. 1160–1166.

[28] D. Lee and A. Lin, "Computational complexity of art gallery problems," *IEEE Transactions on Information Theory*, vol. 32, no. 2, pp. 276–282, March 1986.

[29] J.-T. Amenyo, D. Phelps, O. Oladipo, E. Sewovoe, S. Jadoonanan, S. Jadoonan, T. Tabassum, S. Gnabode, T. D. Sherpa, M. Falzone, A. Hossain, and A. Kublal, "Ultra-low cost, low-altitude, affordable and sustainable UAV multicopter drones for mosquito vector control in malaria disease management," in *IEEE Global Humanitarian Technology Conference*. IEEE, 2014, pp. 590–596.

[30] E. Ackerman, "Drones distribute swarms of sterile mosquitoes to stop zika and other diseases," *IEEE Spectrum*, 2017. [Online]. Available: https://spectrum.ieee.org/robotics/drones/drones-distribute-swarms-of-sterile-mosquitoes-to-stop-zika-and-other-diseases

[31] A. Nguyen, D. Krupke, M. Burbage, S. Bhatnagar, S. P. Fekete, and A. T. Becker, "Using a UAV for destructive surveys of mosquito population," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2018.

[32] L. Zou, S. N. Miller, and E. T. Schmidtmann, "Mosquito larval habitat mapping using remote sensing and GIS: Implications of coalbed methane development and west nile virus," *Journal of Medical Entomology*, vol. 43, no. 5, pp. 1034–1041, 2006.

[33] P. Tokekar, E. Branson, J. Vander Hook, and V. Isler, "Tracking aquatic invaders: Autonomous robots for monitoring invasive fish," *IEEE Robotics Automation Magazine*, vol. 20, no. 3, pp. 33–41, Sep. 2013.

[34] A. T. Becker, M. Debboun, S. P. Fekete, D. Krupke, and A. Nguyen, "Zapping zika with a mosquito-managing drone: Computing optimal flight patterns with minimum turn cost (multimedia contribution)," in *LIPIcs-Leibniz International Proceedings in Informatics*, vol. 77. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[35] M. Burbage, "Maximizing swarm coverage: Hunting for members of a moving population," Master's thesis, University of Houston, Houston, TX, May 2017. [Online]. Available: https://github.com/MCBurbage/Mosquito_drone/blob/master/Mary_Burbage_Thesis.pdf

[36] X. Zheng, S. Jain, S. Koenig, and D. Kempe, "Multi-robot forest coverage," in *Intelligent Robots and Systems, 2005.(IROS). 2005 IEEE/RSJ International ConferenceF*. IEEE, 2005, pp. 3852–3857.

[37] S. Bhatnagar, "Motion planning for killing mosquitoes," https://github.com/RoboticSwarmControl/2018mosquitoCoverage, 2018.

[38] D. Spears, W. Kerr, and W. Spears, "Physics-based robot swarms for coverage problems," *The International Journal of Intelligent Control and Systems*, vol. 11, no. 3, 2006.

[39] S. Koenig, B. Szymanski, and Y. Liu, "Efficient and inefficient ant coverage methods," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1, pp. 41–76, Oct. 2001.

[40] J. A. Dennett, A. Bala, T. Wuithiranyagool, Y. Randle, C. B. Sargent, H. Guzman, M. SIIRIN, H. K. Hassan, M. Reyna-Nava, T. R. Unnasch *et al.*, "Associations between two mosquito populations and west nile virus in harris county, texas, 2003–06," *Journal of the American Mosquito Control Association*, vol. 23, no. 3, p. 264, 2007.

[41] R. Peter, P. Van den Bossche, B. L. Penzhorn, and B. Sharp, "Tick, fly, and mosquito control—lessons from the past, solutions for the future," *Veterinary parasitology*, vol. 132, no. 3, pp. 205–215, 2005.

[42] W. H. Organization, "Guidelines for laboratory and field testing of mosquito larvicides," *World Health Organization communicable disease control, prevention and eradication WHO pesticide evaluation scheme*, 2005.

[43] D. V. Maliti, N. J. Govella, G. F. Killeen, N. Mirzai, P. C. Johnson, K. Kreppel, and H. M. Ferguson, "Development and evaluation of mosquito-electrocuting traps as alternatives to the human landing catch technique for sampling host-seeking malaria vectors," *Malaria Journal*, vol. 14, no. 1, p. 1, 2015.

[44] J. M. Marshall and C. E. Taylor, "Malaria control with transgenic mosquitoes," *PLoS Med*, vol. 6, no. 2, p. e1000020, 2009.

[45] C. A. Hill, F. C. Kafatos, S. K. Stansfield, and F. H. Collins, "Arthropod-borne diseases: vector control in the genomics era," *Nature Reviews Microbiology*, vol. 3, no. 3, pp. 262–268, 2005.

[46] M. O. Ndiath, S. Sougoufara, A. Gaye, C. Mazenot, L. Konate, O. Faye, C. Sokhna, and J.-F. Trape, "Resistance to ddt and pyrethroids and increased kdr mutation

frequency in an. gambiae after the implementation of permethrin-treated nets in senegal," *PLoS One*, vol. 7, no. 2, p. e31943, 2012.

[47] ScienceDaily. (1997, Jul.) Snap! Crackle! Pop! Electric bug zappers are useless for controlling mosquitoes, says UF/IFAS pest expert. [Online]. Available: http://www.sciencedaily.com/releases/1997/07/970730060806.htm

[48] P. Anupa Elizabeth, M. Saravana Mohan, P. Philip Samuel, S. Pandian, and B. Tyagi, "Identification and eradication of mosquito breeding sites using wireless networking and electromechanical technologies," in *Recent Trends in Information Technology (ICRTIT), 2014 International Conference*. IEEE, 2014, pp. 1–6.

[49] B. Hur and W. Eisenstadt, "Low-power wireless climate monitoring system with rfid security access feature for mosquito and pathogen research," in *Mobile and Secure Services (MOBISECSERV), 2015 First Conference*. IEEE, 2015, pp. 1–5.

[50] J.-T. Amenyo, D. Phelps, O. Oladipo, F. Sewovoe-Ekuoe, S. Jadoonanan, S. Jadoonanan, T. Tabassum, S. Gnabode, T. D. Sherpa, M. Falzone *et al.*, "Medizdroids project: Ultra-low cost, low-altitude, affordable and sustainable uav multicopter drones for mosquito vector control in malaria disease management," in *Global Humanitarian Technology Conference*. IEEE, 2014, pp. 590–596.

[51] C. Boonsri, S. Sumriddetchkajorn, and P. Buranasiri, "Laser-based mosquito repelling module," in *Photonics Global Conference (PGC), 2012*. IEEE, 2012, pp. 1–4.

[52] J. Kare and J. Buffum, "Build your own photonic fence to zap mosquitoes midflight [backwards star wars]," *IEEE Spectrum*, vol. 5, no. 47, pp. 28–33, 2010. [Online]. Available: http://spectrum.ieee.org/consumer-electronics/gadgets/backyard-star-wars

[53] M. Gillies and T. Wilkes, "The vertical distribution of some West African mosquitoes (Diptera, Culicidae) over open farmland in a freshwater area of The Gambia," *Bulletin of entomological research*, vol. 66, no. 01, pp. 5–15, 1976.

[54] E. M. Arkin, S. P. Fekete, and J. S. B. Mitchell, "Approximation algorithms for lawn mowing and milling," *Comput. Geom.*, vol. 17, no. 1-2, pp. 25–50, 2000.

[55] E. M. Arkin, M. A. Bender, E. D. Demaine, S. P. Fekete, J. S. B. Mitchell, and S. Sethia, "Optimal covering tours with turn costs," *SIAM Journal on Computing*, vol. 35, no. 3, pp. 531–566, 2005.

[56] ——, "Optimal covering tours with turn costs," in *Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA01)*, 2001, pp. 138–147.

[57] S. P. Fekete and D. Krupke, "Computing optimal covering tours and cycle covers with turn cost," 2017, manuscript.

[58] ardupilot.org. (2016) Motion planner overview. [Online]. Available: http://ardupilot.org/planner/docs/mission-planner-overview.html

[59] S. L. Smith, M. Schwager, and D. Rus, "Persistent robotic tasks: Monitoring and sweeping in changing environments," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 410–426, 2012.

[60] H. Nyberg, "Fundamentals of acoustic larvicide©," New Mountain Innovation, 6 Hawthorne Road, Old Lyme, CT 06371, Tech. Rep., sept 2013.