MAXIMIZING SWARM COVERAGE: HUNTING FOR MEMBERS OF A MOVING POPULATION

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
Mary Burbage

May 2017

MAXIMIZING SWARM COVERAGE: HUNTING FOR MEMBERS OF A MOVING POPULATION

Mary Burbage

Approved:

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

Committee Members:

David Mayerich, Assistant Professor
Department of Electrical and Computer Engineering

Nikolaos Tsekos, Associate Professor
Department of Computer Science

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

Badrinath Roysam, Professor and Chair,
Electrical and Computer Engineering

Acknowledgements

I would like to thank my advisor Aaron T. Becker for all his guidance and direction. It has been a privilege to work with him. Thank you also to my committee members. The members of the Swarm Robotics Control Lab under the direction of Dr. Becker have been of assistance along the way, and I would like to express my appreciation. Most of all, I wish to thank my husband and parents for their support throughout my return to the world of academia. They have done a great deal to relieve me of tasks and allow me the time to do my work, particularly when lecture schedules kept me at the university through the evening. Thank you also to my sister for providing for me when schedules were inconvenient.

MAXIMIZING SWARM COVERAGE: HUNTING FOR MEMBERS OF A MOVING POPULATION

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
Mary Burbage

May 2017

Abstract

We explore search methods for finding and removing members of a large population of mobile particles in different environments. We begin by simulating the movement of individual swarm members biased by a map image with four search patterns and find that simulation duration affects the results. We then treat particle motion as a Markov process in an environment bounded with walls that retain a portion of the swarm and in an environment that attracts the particles into a normal distribution at the center of the space. We test six search patterns with several parameter variations. We show that a two-step greedy algorithm has the best performance in all cases but second best performance varies with the parameters of the swarm and the search.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Definition

We consider the problem of planning a coverage path for a robot such that it encounters the maximum number of members of a population of mobile particles in a workspace. Once each particle is encountered, it is removed from the population. Figure 1.1 shows two examples of such applications: a mosquito-killing drone carrying an electrified net that eliminates mosquitoes as it collides with them and a fishing trawler that gathers fish in its nets. With this type of sampling without replacement, the path followed by the robot up to a given time influences the particle population for the remaining time. This is different from the classic robotic coverage problem which is concerned with full spatial exploration.

For stationary particles and unconstrained time, any path that visits every location in the workspace will cover the entire population in a single pass. When the particles

Figure 1.1: Two examples of particle-hunting robots. Left: Mosquito-hunting drone. Right: Fishing trawler [1].

are mobile, they move as the robot moves, possibly moving into space that has already been covered. For unconstrained time, a full spatial coverage path is still satisfactory because it can be repeated until the robot has encountered every particle. What we seek is the best path for covering a population of mobile particles within a limited time period. In particular we wish to know what path will have the highest probability of encountering the largest fraction of the population when the whole workspace cannot be covered within the robot's limited search time.

We model the process as sampling without replacement [3] from a point cloud of mobile particles using a mobile agent. The point cloud particles are generated from a known or unknown distribution, and the mobile agent clears all particles in a swept-out region each time step. Many robots have strict energy budgets, with UAV budgets being particularly strict, limiting the search time to a maximum time T. The goal is to design a trajectory for the robot that, in probability, samples the most particles for a given time T. We assume the agent can detect each particle collision and can use this information to modify a planned trajectory.

1.2 Robotic Coverage

Robotic coverage has a long history. The classic problem is one of designing a path for a robot that ensures the robot visits within some maximum distance of every point in the workspace. For an overview see the summaries in [4, 5]. The simplest method of converage considered is a path in which the robot traces a path similar to that used by an ox plowing a field, giving rise to the name "boustrophedon," meaning "turning like oxen in plowing" [6]. Figure 1.2 illustrates this type of pattern composed of parallel path segments spaced by the width of the robot's sensing range. This path guarantees complete coverage of a workspace in the absence of uncertainty [7].

A significant body of research then investigates ways to ensure complete coverage in

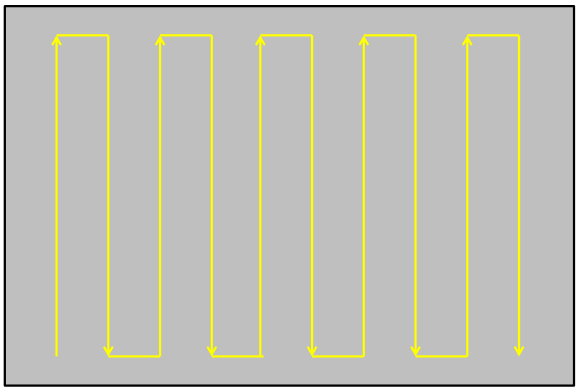

Figure 1.2: Sample boustrophedon path.

a workspace that contains obstacles. Many methods decompose a workspace into smaller spaces as in [7] or [8]. A boustrophedon coverage pattern may be used in each of these smaller spaces, and coverage is guaranteed if a path exists that visits every cell. Further work addresses actions for a robot to perform in the presence of moving obstacles [9] or when encountering unexpected or unknown obstacles [10]. Work has also been extended to use multiple coverage robots in a variety of ways, including using simple behaviors for the robots [11, 12].

Localization uncertainty is studied in [13], giving rise to a probability of coverage for any planned path. As measurement uncertainty increases, [14] finds that a random bounce coverage pattern like that shown in Figure 1.3 can be effective. This is particularly true when multiple search robots can be built inexpensively. The possibility of moving search targets is considered in passing, but the perspective is one of keeping the path unpredictable so that targets that are actively avoiding discovery are less likely to hide successfully.

1.3 Extensions

The key difference in the coverage problem presented here is that the particles can move, recontaminating an area previously cleared. This is similar to a persistent moni-

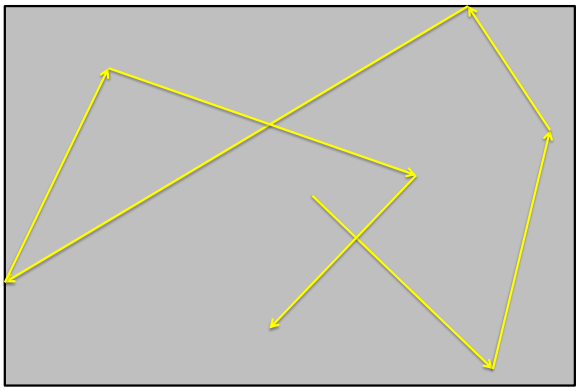

Figure 1.3: Sample random bounce path.

toring task [15] but with sampling without replacement. This gives us a probability of coverage, as in [13]. This is closely related to the art gallery problem [16] but with limited range of visibility. A second difference is in the available time. Much of the literature deals with finding a complete coverage path without regard to time or tries to find a minimum time path as in [17]. Other path optimization connects given waypoints without attempting full searches as in [18], streamlines paths to search areas of interest [19], or considers energy conservation in searches by connecting valuable locations [20]. While a minimum time path would indicate that a robot would cover the largest percentage of the area within T, it may not be the case that the most valuable parts of the workspace would be covered within the restricted time allowance.

1.4 Proposed Methods

Using a MATLAB environment, we create a simulation of moving particles in a workspace and a robot to search for them. The particles perform a random walk, either biased or unbiased, and we simulate different coverage paths for the robot in order to establish the relationships between the environment and the performance of the search algorithms. We will consider three environments:

1. a population in a mapped environment with a random walk biased toward features

of the map,

2. a population with a random walk that is biased toward a normal distribution in a bounded environment, and

3. a population with an unbiased random walk bounded by walls where particles bounce off the walls.

We use some assumptions in the work. The robot will move much faster than the particles. The robot will move at a constant velocity without regard to turns. The velocity of the particles will vary within limits. The particles will use the toroidal assumption for the space except when specifically bounded by walls as in the third environment listed above. This assumes that the particles treat the workspace as a toroid as though $x = 0$ were identified with $x = L$ and $y = 0$ were identified with $y = L$. As the original inspiration for this work was an application with a mosquito-hunting drone, we will often refer to the moving particles as mosquitoes.

Chapter 2

Map Simulation

We begin by considering a world in which particles are mosquitoes that move across a map. Their random walk movement model is biased toward certain features in the map. The goal is to design a trajectory with duration less than the time limit T that is likely to eliminate the greatest number of these particles. We assume the robot can detect each encounter and use that information to modify a planned trajectory.

2.1 Description of Methods

Ten thousand mosquitoes or particles are randomly placed within a square area 100 m on a side. A satellite image of Houston provides the map in Figure 2.1 for the simulation environment, and each particle moves according to a biased random walk at a speed up to 0.4 m/s [21] and with a direction heading biased toward the greenest of the pixels surrounding its current position. This imitates the live mosquitoes' preference for vegetative areas [22]. Because the environment consists of discrete pixels, we use a Gaussian mask to smooth the color values and improve the reliability of the bias. A toroidal mapping keeps the particles in the workspace because the world extends beyond the boundaries of the image. The robot does not leave the workspace or use the toroidal mapping. The simulation begins by running the particle movement simulation for 5000 iterations at one loop iteration per second before the robot begins to search. This allows particles that began in a uniform random distribution across the workspace as on the left side of Figure 2.2 to move into a non-uniform distribution as on the right side of Figure 2.2.

Figure 2.1: Map image [2].

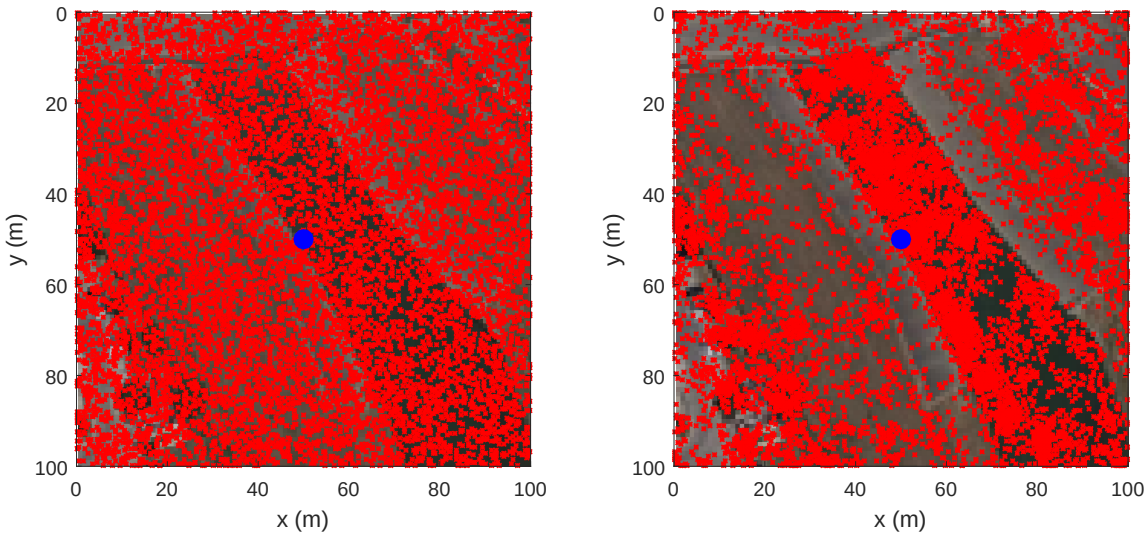

Figure 2.2: Particles (red markers) uniformly distributed (left) and after biasing (right). A robot (blue circle) is in the center.

The searching robot with a width of one unit eliminates or kills any mosquito that intersects its path. The robot is unaware of the features of the map so that it cannot predict where the most valuable regions of the workspace are. It can, however, detect the time each particle is eliminated and use this data as feedback for a motion policy. The robot is tested with four different policies. The first is a boustrophedon path, which is a very simple method that guarantees full coverage. The robot begins in one corner of the workspace and methodically progresses back and forth, advancing one unit width at each turn. If it covers the entire field in the allotted time, it begins covering the field again.

The second uses a random bounce algorithm. The robot begins in the center of the workspace and moves with a heading that varies randomly up to ± 0.2 rad from its previous heading and bounces off the perimeter of the workspace with a random heading equally biased between 0 and 2π rad, excluding headings leading outside the workspace.

The third path begins with the boustrophedon path but uses feedback from particle encounters to dwell in areas with higher particle populations. It uses a square spiral path with turns spaced one unit width apart when the rate of discovery exceeds a set threshold. Once the number of encounters falls below a second threshold, the robot returns to the boustrophedon path. The fourth and final path combines the random bounce path with the square spiral in the same way as the third path.

For the main body of the simulation, a loop runs a series of iterations in which each particle moves one step and the robot moves one step. In that step, the path traced by the robot is calculated, and any particles in that path are counted and removed. To keep the routines comparable, the robots use the same speed and same number of iterations in each test as well as the same image for biasing the particle movement. The baseline simulations used 12 m/s and fifteen minutes of search time, which is sufficient time for the robot to completely cover the 100 m workspace.

Since the thresholds for switching between the boustrophedon or random bounce base path and the square spiral affect the overall path performance, we wanted to use values that were approximately optimal. Before running the tests for comparison, we ran a parametric array of tests for the spiral thresholds in order to find the best combination for the analysis. The robot uses an average of the number of particles found in each of the last few steps to determine whether the area should be further searched with a spiral. When the average over these steps is above a set threshold, it switches into the square spiral mode. While in the spiral, it watches for the particle count to drop below another threshold to return to the previous mode. We tested the set $\{3, 6, 9, 12, 15, 18, 21\}$ for both the spiral start and end thresholds and the set $\{3, 4, 5, 6\}$ for the number of iterations to include in the running average. We ran ten tests for each combination of parameters.

Test results showed the number of steps to include in the running average must be kept quite low to keep the robot from moving too far away from the areas of higher concentration before beginning the spiral, but it must not be reduced too far to avoid switching for very small areas with a spike in concentration. This applies equally to the random bounce and boustrophedon paths. The threshold values have differing effects for the two base methods. If enough time is allowed for a boustrophedon path to cover the whole field, the spiral is of no advantage, leading to best results for a maximum threshold for both entering and leaving the spiral. As the time available decreases, we can use wider spacing between the rows and gain some value from the spirals. This led us to choose a spiral entry threshold of ten as well as a spiral exit threshold of ten. Simulation results are shown in Figure 2.3. In the random bounce method, the spiral is more advantageous and shows a peak in performance for an entry threshold of nine. It is best to stay in the spiral until the running average is quite low. We use nine for the entry threshold and three for the exit threshold. Figure 2.4 shows these results.

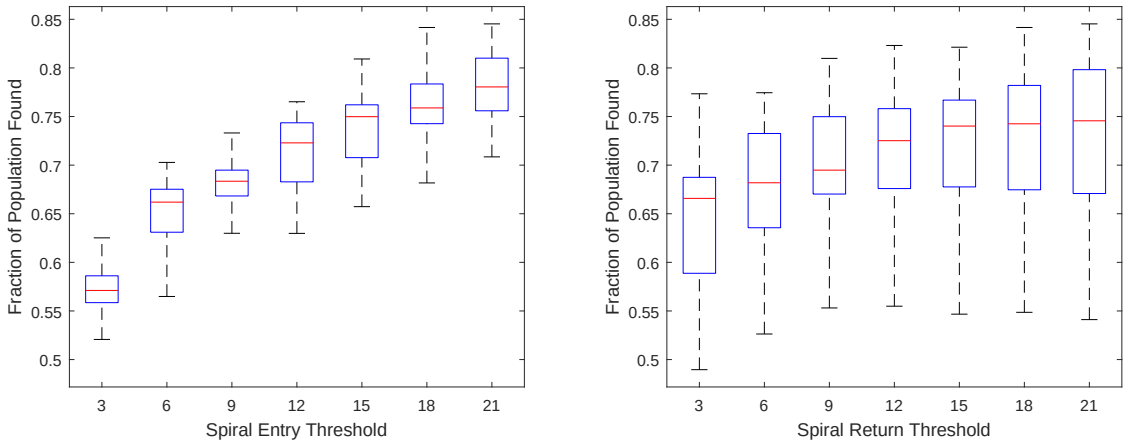

Figure 2.3: Particle collection rates for spiral entry and return thresholds with boustrophedon base path.

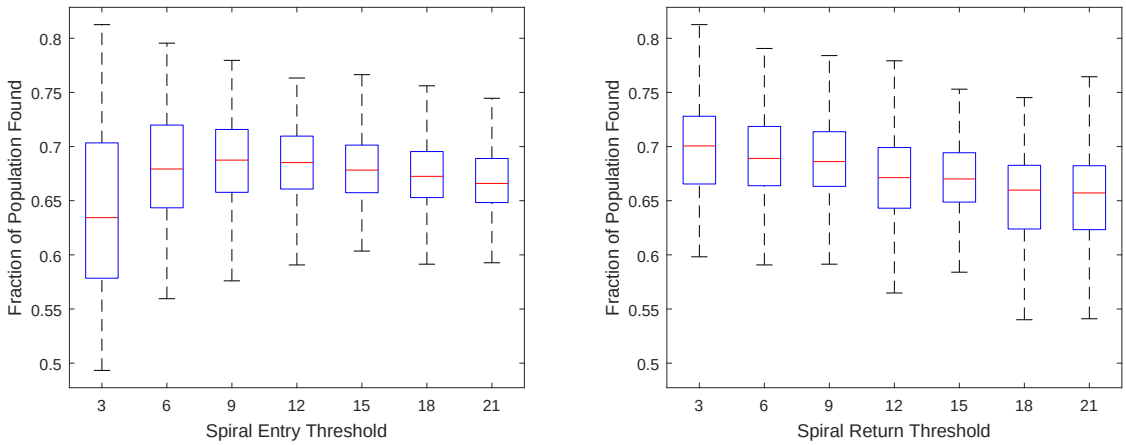

Figure 2.4: Particle collection rates for spiral entry and return thresholds with random bounce base path.

2.2 Baseline Results

One hundred trials were performed with each coverage path and the results evaluated. The boustrophedon successfully covered the entire field in every trial, while the random bounce covered only 64.7% of the field on average with a standard deviation of 1.1%. The boustrophedon with spiral averaged 91.5% coverage of the field with a standard deviation of = 0.6%, and the random bounce with spiral covered 66.0% with a standard devation of 1.3%. Due to the higher coverage rates, the boustrophedon found

significantly more particles ($\mu = 84.9\%$, $\sigma = 0.4\%$) than the random bounce ($\mu = 65.9\%$, $\sigma = 2.5\%$), though the addition of the spiral to the random bounce showed an improvement over the plain random bounce ($\mu = 71.9\%$, $\sigma = 3.8\%$). Including the spiral with the boustrophedon degraded its performance ($\mu = 79.8\%$, $\sigma = 1.7\%$). Figure 2.5 shows a set of sample paths for the baseline trials with a robot (blue circle) with the area it covered in a loop iteration (blue rectangle) and the path it has followed (yellow line). Of the $10,000$ mosquitoes used in the simulation, those that were killed by the robot are white, and the ones that were not found are red. The left column shows the random bounce paths, and the right column show the boustrophedon paths. The top row does not include the spiral option while the bottom row does. Figure 2.6 shows the aggregate results of 100 trials.

2.3 Benchmark Results

Having set a baseline in which full coverage was clearly the best method, we next considered the problem of a field which is too large for full coverage within the allotted time. To simulate this, we used the same environment but set the flight time to five minutes ($T = 300\ s$) and repeated the 100 iterations of each method. We also increased the distance between path rows for the boustrophedon path with feedback spirals so that the robot could cover a wider area looking for increased concentrations of particles.

With reduced flying time, the coverage and kill rates were much lower than in the baseline tests, but the feedback spiral improved the kill rate for both the boustrophedon and the random bounce paths. Sample paths are displayed in Figure 2.7 and use the same symbols as those discussed for Figure 2.5 in Section 2.2. Results are illustrated in Figure 2.8, which shows the best kill rate of 35.0% ($\mu = 35.0\%$, $\sigma = 3.0\%$) for the boustrophedon with spiral path, followed by the random bounce with spiral path at 32.5% ($\mu = 32.5\%$, $\sigma = 3.5\%$). The boustrophedon and random bounce paths had equal

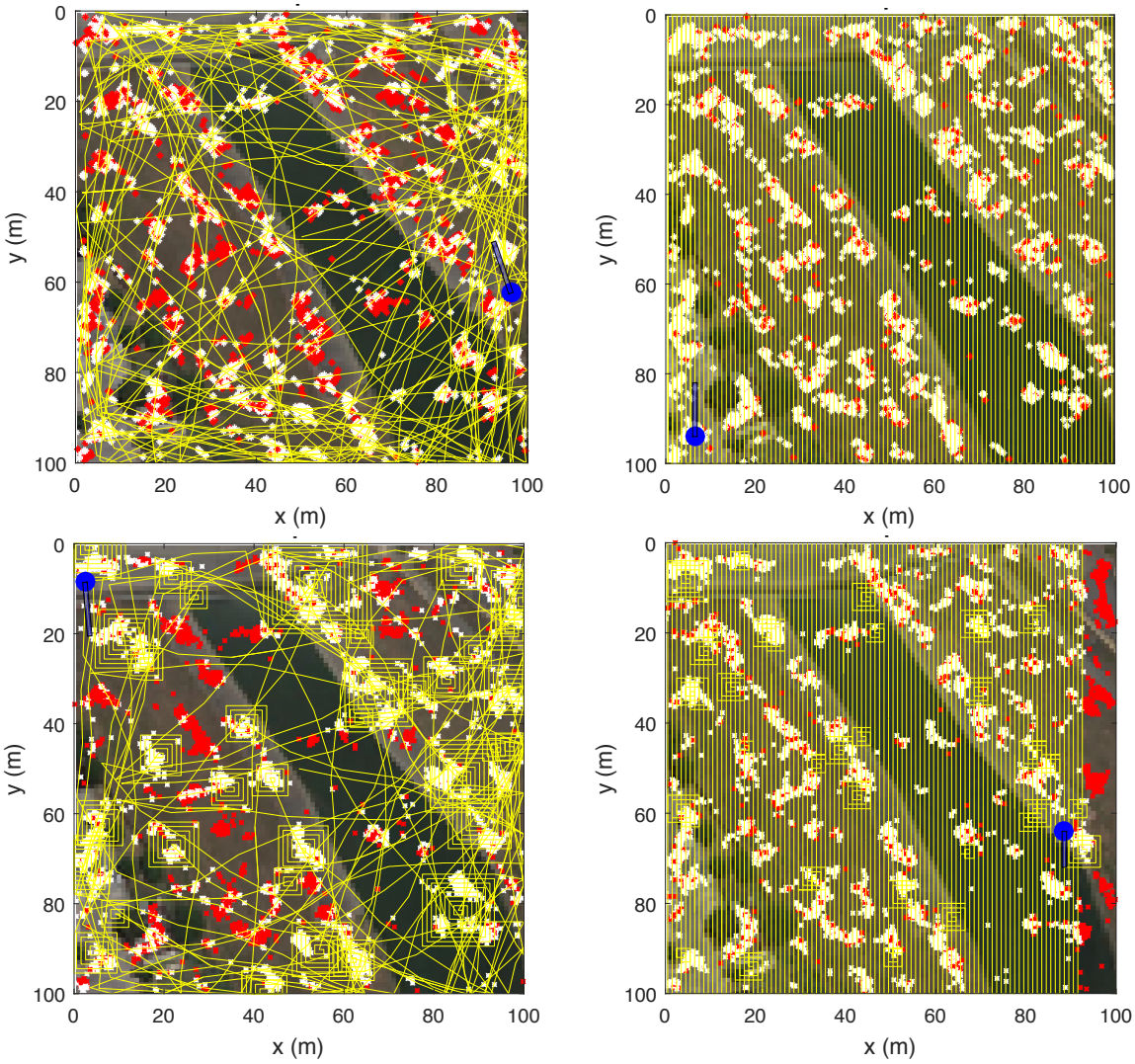

Figure 2.5: Four sample simulations for $T = 900\ s$. Left column: random bounce, right column: boustrophedon. Bottom row adds spiral movements when the particle density is high.

kill rates of 31.2%, though the boustrophedon had a narrower standard distribution ($\sigma = 0.5\%$) than the random bounce ($\sigma = 2.8\%$) and covered a greater percentage of the workspace, with 35.7% of the area covered as opposed to 30.6%. We consider these as benchmarks for testing further coverage policies and provide the code on MATLAB Central [23].

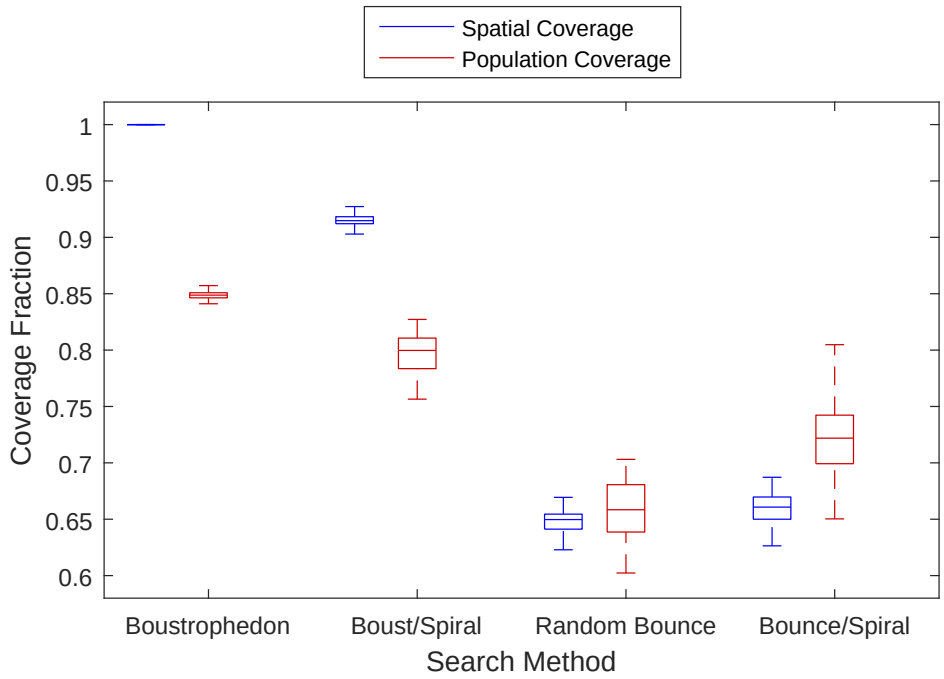

Figure 2.6: Comparison of percentage of area covered and percentage of particles found for $T = 900$ s for the four coverage patterns across 100 trials.

2.4 Difficulties with This Approach

While simulation of individual members of the population allows for motion patterns that may be closely aligned with the true behavior of the particles, it requires a great deal of computation time to generate the results of each type of path. The stochastic nature of the particle behavior means the performance of the simulation for any given trial should lie within a range of normal performance but is not necessarily representative of the general behavior and is therefore insufficient for complete analysis. A large number of trials is needed to find the mean performance, and any variation in parameters requires running multiple trials to establish a fair comparison.

Another way to view the biased population density is as a probability distribution function (PDF) like that shown in Figure 2.9. The population in Figure 2.9 is also shown with contour lines and the environment image in Figure 2.10. If we can treat every segment of the environment as having some probability of containing part of the

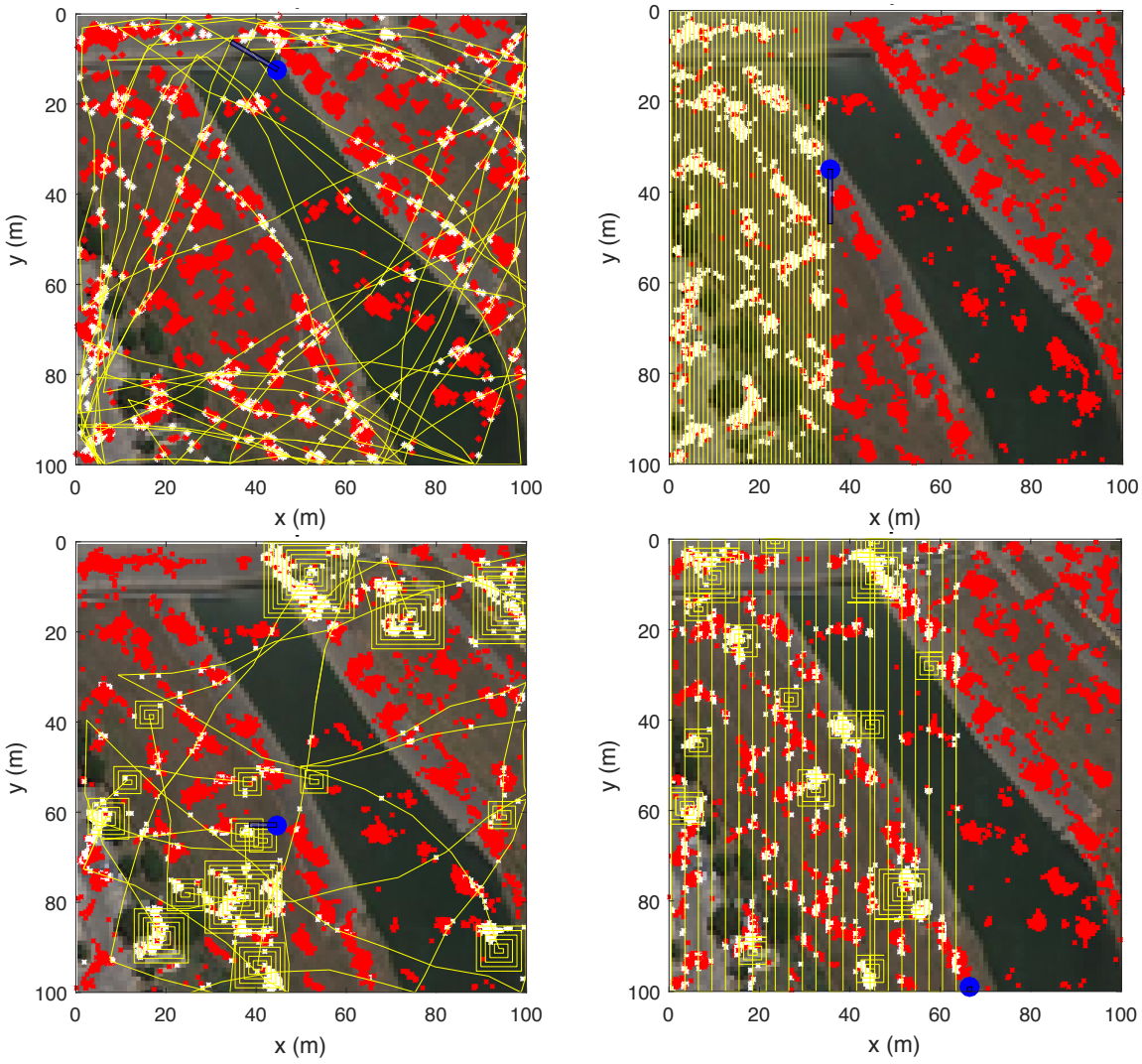

Figure 2.7: Simulations with insufficient time for full coverage. Left column: random bounce, right column: boustrophedon. Bottom row with feedback spiral movement.

swarm, we can avoid simulating each member of the swarm and simulate the behavior of the swarm as a whole.

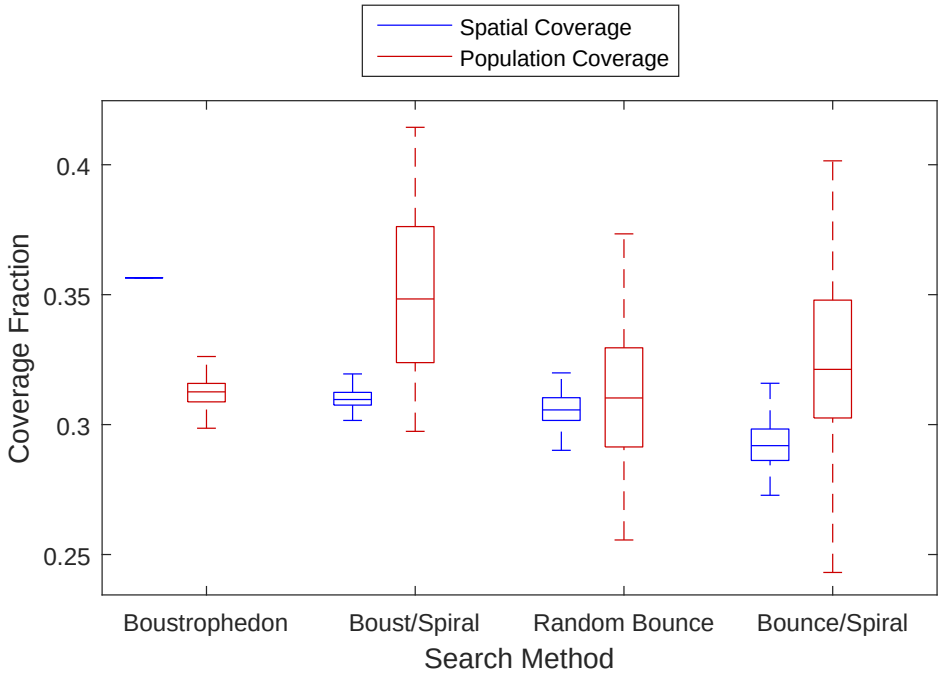

Figure 2.8: Comparison of percentage of area covered and of particles found for $T = 300\ s$ for the four coverage patterns across 100 trials.

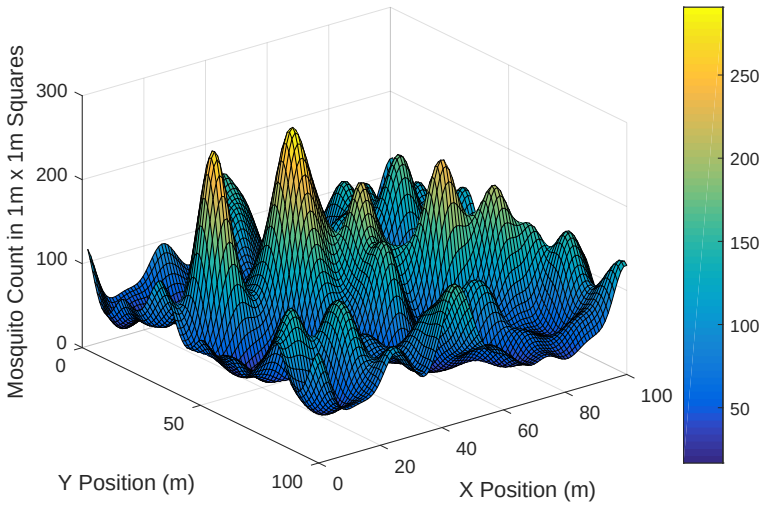

Figure 2.9: PDF of particle location with 10k particles after 5k iterations.

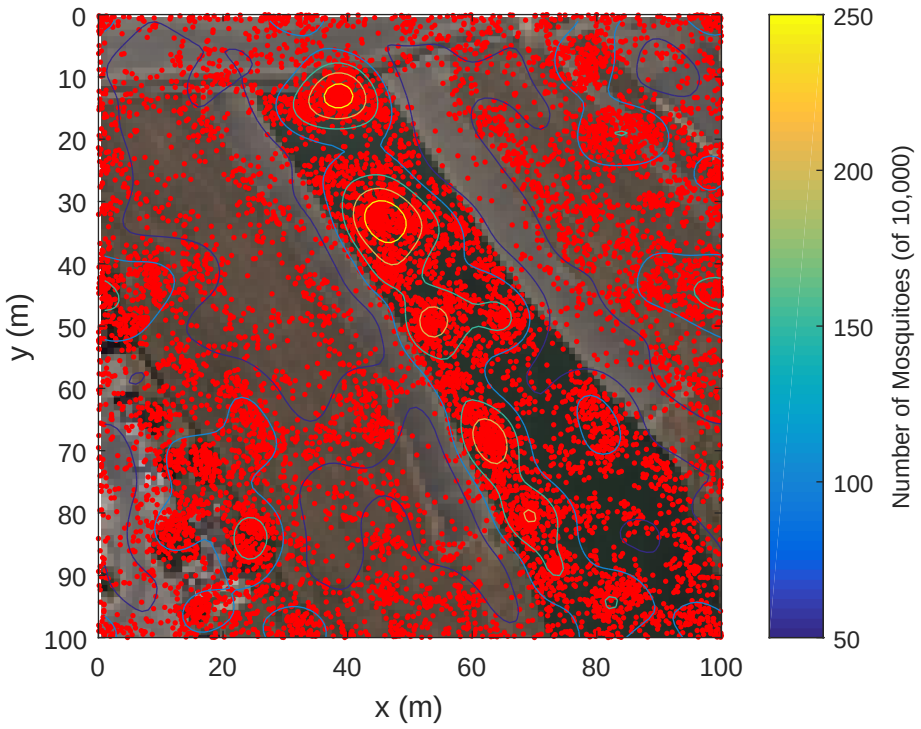

Figure 2.10: Individual particle positions (red dots) and PDF contour lines on the satellite image of a Houston location.

Chapter 3

Population Movement as a Markov Process

3.1 Introduction

To reduce the computation time required to evaluate the effectiveness of any given set of parameters, we wish to treat the problem in a way that handles the overall behavior of the population instead of working with each individual member of it. If we can find the average behavior of the population as a whole, each test with a given set of parameters will give results that are representative of the average efficacy of the method. This eliminates the need for running a large number of tests for each situation. We generalize the behavior of the moving particles by treating their motion as a Markov process.

3.2 Markov Processes

A Markov process or Markov chain considers a series of events in which the likelihood of moving from one state to another may be defined by a series of state transition probabilities [24]. For a group of states $S = \{s_1, s_2, ..., s_n\}$, the probability of moving from s_i to s_j is defined as p_{ij}, where $i, j \in [1, n]$. This assumes that the probability of moving from one state to another is not dependent upon any past history; only the current state affects the next state. Figure 3.1 shows a simple set of state transitions with their transition probabilities.

These individual probabilities are then gathered into a transition matrix of the

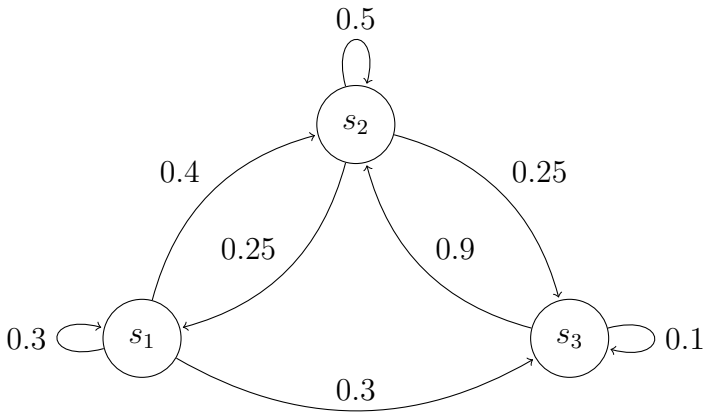

Figure 3.1: Sample state transition probabilities.

form

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{bmatrix}, \tag{3.1}$$

where the values in row i are the probabilities that a particle in s_i will move to any other state in S. Because these are probabilities and contain all possible outcomes after a transition step, each row must sum to unity with $\sum_{j=1}^{n} p_{ij} = 1$ for each row i with each p_{ij} in the range $[0, 1]$. For the diagram in Figure 3.1, this matrix is

$$\mathbf{P} = \begin{bmatrix} 0.3 & 0.4 & 0.3 \\ 0.25 & 0.5 & 0.25 \\ 0.0 & 0.9 & 0.1 \end{bmatrix}. \tag{3.2}$$

With a vector of initial states

$$\mathbf{x}(\mathbf{0}) = \begin{bmatrix} x_1 & x_2 & \ldots & x_n \end{bmatrix} \tag{3.3}$$

that contains the probabilities of having a particle in each cell for time $t = 0$, we can multiply the state vector by the transition matrix to find the probabilities of having a particle in each cell at time $t = 1$ with

$$\mathbf{x}(\mathbf{1}) = \mathbf{x}(\mathbf{0})\mathbf{P}. \tag{3.4}$$

This is repeated for a second time step with

$$\mathbf{x}(\mathbf{2}) = \mathbf{x}(\mathbf{1})\mathbf{P}, \tag{3.5}$$

but substituting for $\mathbf{x}(\mathbf{1})$ we see that

$$\mathbf{x}(\mathbf{2}) = \mathbf{x}(\mathbf{0})\mathbf{P}\mathbf{P} = \mathbf{x}(\mathbf{0})\mathbf{P}^2. \tag{3.6}$$

This generalizes to

$$\mathbf{x}(\mathbf{t}) = \mathbf{x}(\mathbf{0})\mathbf{P}^t \tag{3.7}$$

for any $t \in \mathbb{N}$.

As $t \to \infty$, the vector $\mathbf{x}$ becomes a limiting or stationary distribution

$$\mathbf{x} = \mathbf{x}(\mathbf{0})\mathbf{P}^\infty. \tag{3.8}$$

We define

$$\mathbf{W} = \lim_{t\to\infty} \mathbf{P}^t. \tag{3.9}$$

Each row of $\mathbf{W}$ is the same and is equal to $\mathbf{w}$. Each element w_i in $\mathbf{w}$ represents the fraction of the population that is in state s_i after many steps of the process and is independent of the initial distribution at time $t = 0$. It can be shown that $\mathbf{w} = \mathbf{w}\mathbf{P}$. For a proof see [24]. Mathematically this gives the same result as normalizing the left eigenvector of $\mathbf{P}$ for the eigenvalue $\lambda = 1$ [25]. Since calculating the eigenvectors of $\mathbf{P}$ is much faster than evaluating $\lim_{t\to\infty} \mathbf{P}^t$, this is the method we use to establish the stationary distribution for a transition matrix.

3.3 Environment Definitions

We next define the environments in which we apply Markov processes to the particle behavior. In considering the more general behavior of a population, we leave behind the mapped world discussed in Chapter 2 and consider two canonical workspaces, which are defined geometrically. One is a walled workspace with a uniform particle distribution, and the other is an unwalled workspace with a normal distribution of the population.

3.3.1 Uniform Walled Environment

In the walled environment, we begin with a population that is distributed uniformly in a workspace that is divided into square cells. The boundaries of the workspace are walls that the particles cannot pass through or over, and so they are confined to the workspace. Each particle follows an unbiased random walk model in which it moves in a random direction θ selected uniformly from the interval $[0, 2\pi)$ rad and a distance v selected uniformly from the interval $[0, v_{max}]$. In our simulations, we continue to use $v_{max} = 0.4$ m/s as in Chapter 2. Since the particles cannot pass through the walls, a particle whose path intersects a wall will move the portion of v that takes it to the wall, then bounce off the wall at a new angle θ uniformly selected from the portion of the interval $[0, 2\pi)$ rad that keeps the particle inside the boundaries for the remainder of v.

We recognize that this does not create a uniform distribution on a disk with radius v_{max} but has a higher concentration in the center [26] as shown in Figure 3.2. We can calculate the probability distribution function (PDF) given by this motion by realizing that each sector of the disk with a given $\Delta\theta$ has a equal probability of being selected and that each ring on the disk with a given Δv has an equal probability of being selected [24]. That makes any small sector like that shown in Figure 3.3 have an equal probability of being selected

$$p_{sector} = p(\Delta v)p(\Delta\theta) = \frac{\Delta v}{v_{max}}\frac{\Delta\theta}{theta_{max}} = \frac{\Delta v \Delta\theta}{2\pi v_{max}} \tag{3.10}$$

and an area

$$A_{sector} = \frac{\Delta\theta}{2}(v + \Delta v)^2 - \frac{\Delta\theta}{2}v^2 = \frac{\Delta\theta}{2}(2v\Delta v + \Delta v^2). \tag{3.11}$$

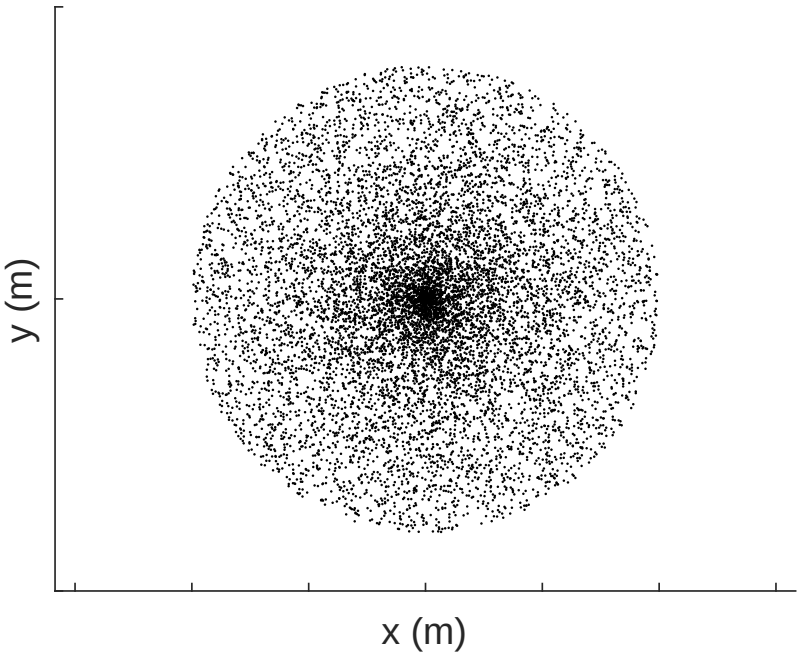

Figure 3.2: Results of $n = 10,000$ Monte Carlo samples of a single step.

Then we have

$$\begin{aligned} p(v) &= \frac{p_{sector}}{A_{sector}} \\ &= \frac{\frac{\Delta v \Delta \theta}{2\pi v_{max}}}{\frac{\Delta\theta}{2}(v+\Delta v)^2 - \frac{\Delta\theta}{2}v^2 = \frac{\Delta\theta}{2}(2v\Delta v + \Delta v^2)} \\ &= \frac{\Delta v}{\pi v_{max}(2v\Delta v + \Delta v^2)}. \end{aligned} \tag{3.12}$$

As the sector becomes infinitesimally small, we have the PDF

$$\lim_{\Delta v \to 0} p(v) = \frac{1}{2\pi v_{max} v}. \tag{3.13}$$

This curve is shown in Figure 3.4, where the center of the distribution is the previous position of the particle. When this PDF is replicated for a large population of particles, it creates a uniform distribution.

Since the uniform distribution does not have any areas that are more valuable for searching than any others, we add a feature that allows the particles to behave differently

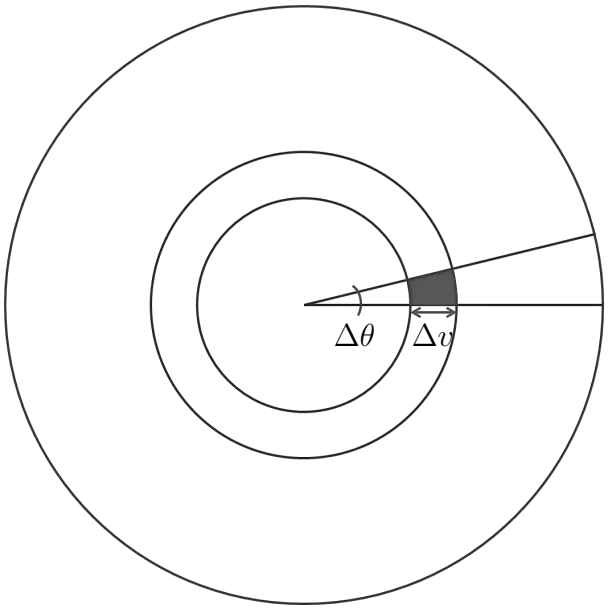

Figure 3.3: Segment of a circle for calculating the PDF.

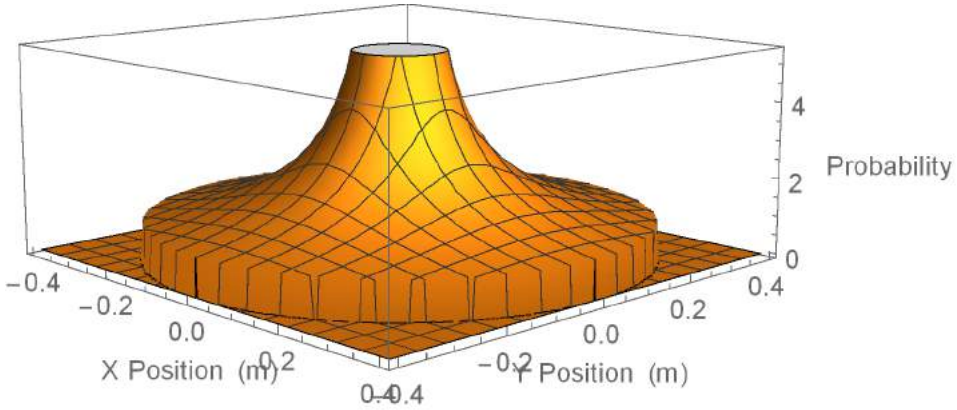

Figure 3.4: PDF of a particle's new position after a single move.

near the walls of the workspace. We introduce a sticking coefficient s that increases the likelihood of a particle remaining in a cell beside a wall. This coefficient can take any value on the interval of $[0, 1]$ and represents the fractional decrease in the probability that a particle will leave an edge cell, *i.e.* for $s = 0$, the distribution would be a standard uniform distribution, while for $s = 1$, all particles that entered an edge cell would be stuck along the walls. Addition of the sticking coefficient leads to a stationary distribution that looks like that in Figure 3.5 for a 100 m workspace with $s = 0.25$. Figure 3.6 shows a cross-section of the area at $y = 50$ m for different values of s which illustrates the

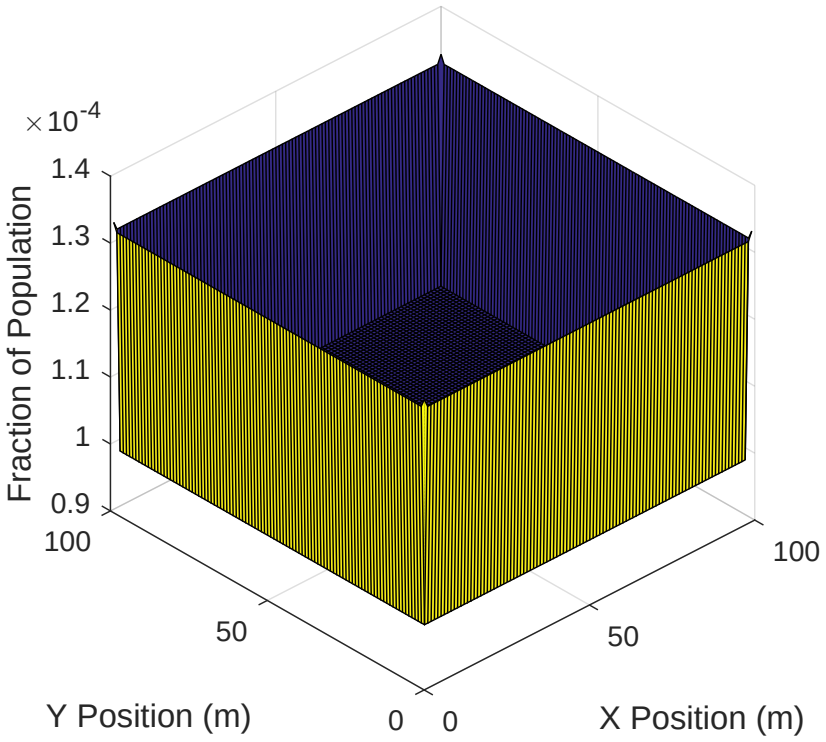

Figure 3.5: Stationary distribution of the walled environment with sticky walls.

relationship between the sticking coefficient and the distribution. This relationship is independent of the rate at which the particles leave the cell.

3.3.2 Normal Distribution Environment

The second environment that we use when the moving population's behavior follows a Markov process is unbounded but has something attractive in the center of it so that the population forms a normal distribution centered in the workspace. The fraction of the population of each cell can be calculated using the cumulative distribution function (CDF) for the normal distribution for a given standard deviation value σ. This kind of distribution is shown in Figure 3.7 for $10,000$ particles with a standard deviation of one-tenth of the workspace. Cross-sections of the distribution at $y = 50\ m$ for varying standard deviations are shown in Figure 3.8.

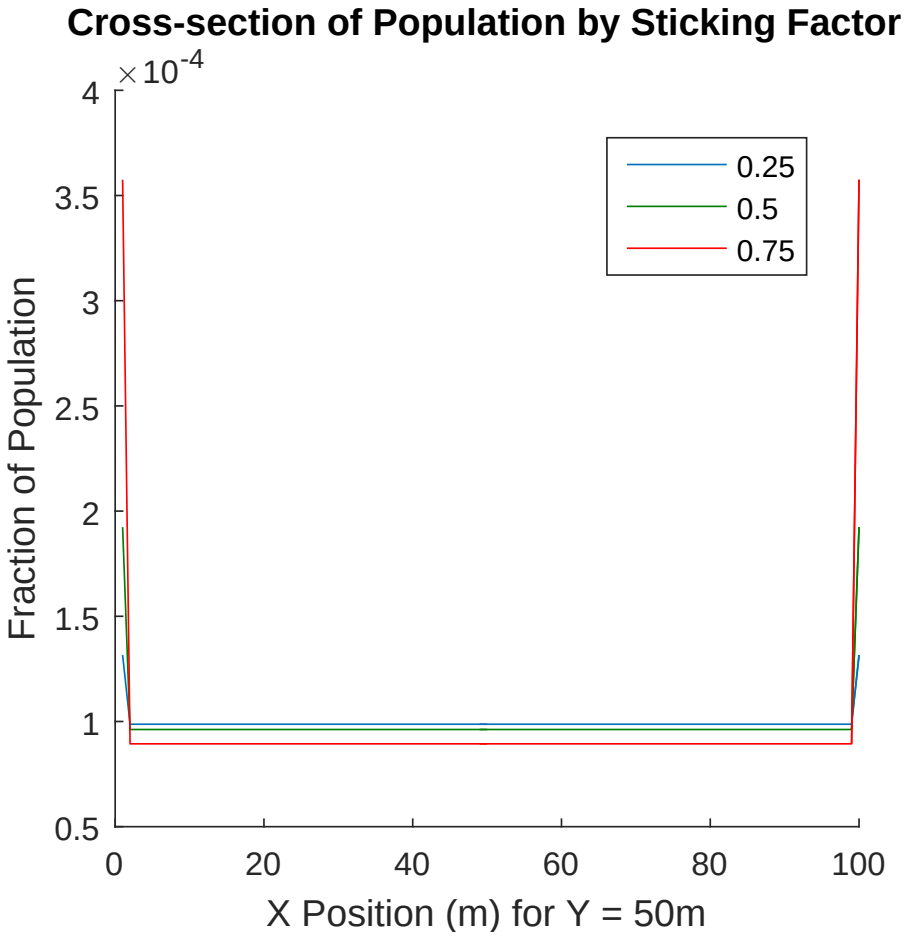

Figure 3.6: Cross-section of population distribution for $s = \{0.25, 0.5, 0.75\}$.

3.4 Creating the Transition Matrices

The basis of the Markov process is the transition matrix $\mathbf{P}$ described previously. For each cell we must determine the likelihood that a particle in that cell will move to an adjacent cell. We do that through both simulation and calculation.

3.4.1 Simulation

We initially determine these matrices by running Monte Carlo simulations on particles that follow a prescribed motion rule. We use a random number generator to select starting locations for a million particles within one cell then select a direction and step

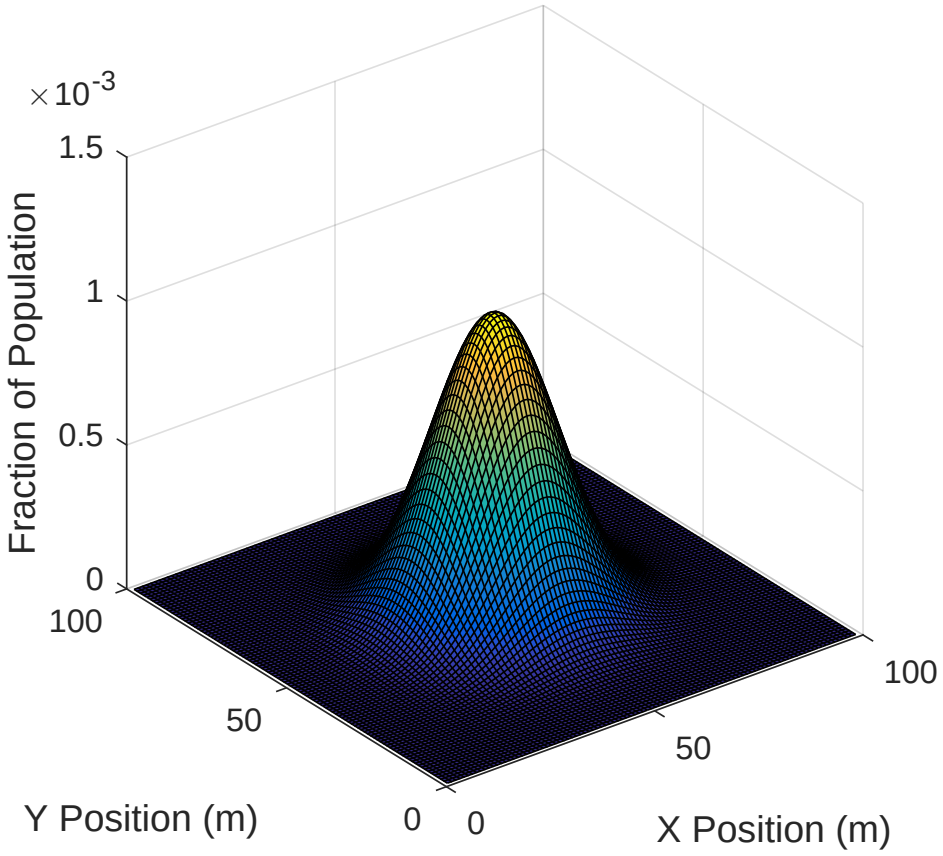

Figure 3.7: Stationary distribution of the normal distribution environment.

length for each of those particles to take a step. Since we are using a maximum step size that is less than the width of a cell ($v_{max} < 1$,) this results in a group of cells in a 3×3 cell array N with the starting cell as the center cell. The fraction of results in each of the nine cells is the probability of transitioning in that direction.

For the uniform walled environment, the Monte Carlo trial values taken from the uniform distributions as $\{v \sim U[0, 0.4], \theta \sim U[0, 2\pi)\}$ result in the following cell array:

$$N = \begin{bmatrix} 0.0043 & 0.0552 & 0.0043 \\ 0.0553 & 0.7623 & 0.0551 \\ 0.0043 & 0.0551 & 0.0043 \end{bmatrix}. \tag{3.14}$$

Since we know that the uniformity of the distribution implies symmetry in the distri-

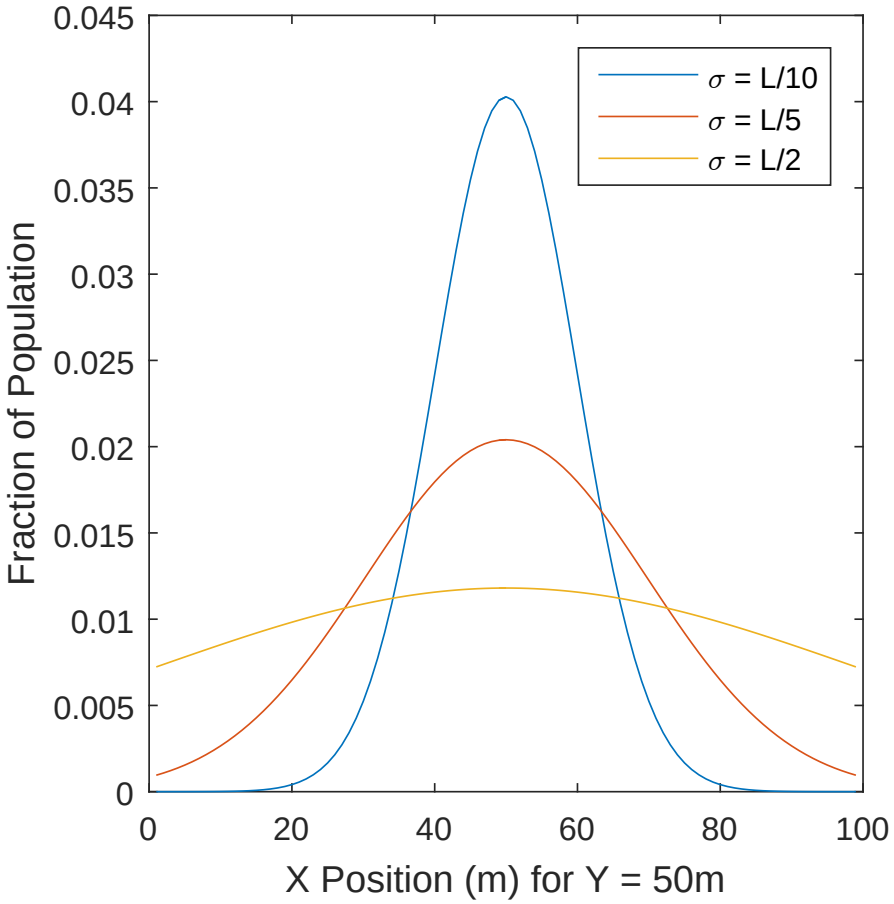

Figure 3.8: Cross-section of population distribution for $\sigma = \{L/10, L/5, L/2\}$ where $L = 99\ m$.

bution, we balance the values in the cells that share an edge with the starting cell, the cardinal directions, and in the cells that share only a corner with the starting cell, the diagonal directions. If this is not done, a slight difference will weight the stationary distribution toward the larger value in the imbalance so that it is not uniform. We also use the uniformity to shorten our calculations by calculating one set of values and applying it to every cell. In the boundary cells, we use the way the particles bounce off the walls to assume that particles that would go beyond the boundary stay in the cells next to the simulated cell. Figure 3.9 shows how the values are shifted for the left edge and bottom left corner. The fraction that would have moved into the non-existent cells are added to the cells shown by the arrows. This preserves the uniform distribution of the population

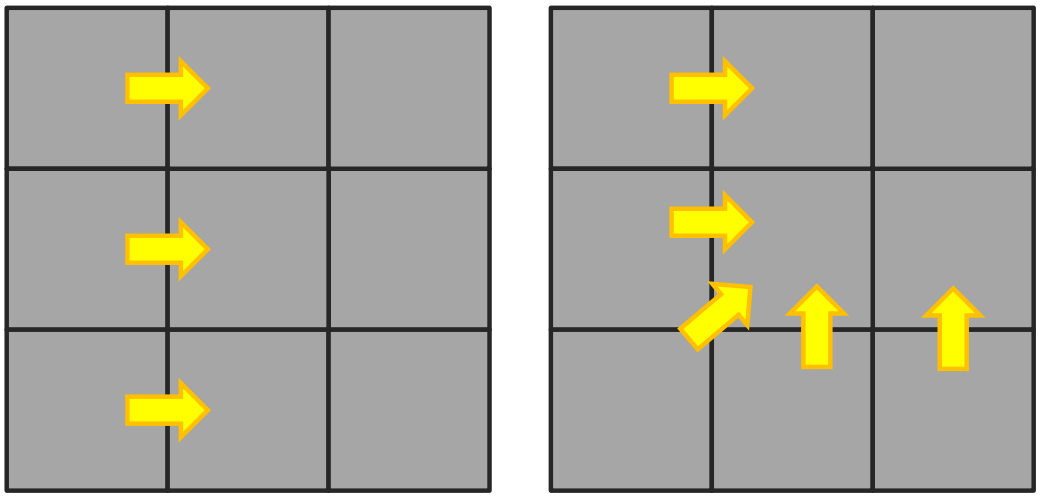

Figure 3.9: Adjusting the probabilities for edges (left) and corners (right).

since the bouncing particles take the place of particles that would have moved in from cells just beyond the walls.

The environment with a normal distribution of particles is more complex. For the particle movement we select a location drawn from a normally distributed random number generator provided by the MATLAB function `randn`. We then calculate the heading the particle would need to follow to move toward that location and add a random component to it. The distance traveled is again chosen from the uniform distribution on the range $[0, 0.4]$, and for a million particles uniformly placed within one cell, a 3×3 cell array is created just as in the uniform distribution.

This environment has a different type of symmetry from the uniform environment so we cannot replicate the same transitions for every cell, but we can use the symmetry through the center of the distribution to reduce the number of different sets of simulations we need to do to 25% of the total number of cells. After calculating an N for each cell in the top left quadrant, we can create mirrored N arrays for each of cells in the other quadrants in the following ways:

- Top Right - Reverse columns,

- Bottom Left - Reverse rows, and
- Bottom Right - Reverse rows and columns.

For example, with a workspace 100 m long, the array $N^{20,40}$ would be the N array for the cell at $\{x, y\} = \{20, 40\}$. The corresponding cell in the top right quadrant would be the cell at $\{x, y\} = \{80, 40\}$ and would have the array $N^{80,40}$. This array would be created as

$$N^{80,40} = \begin{bmatrix} n_{1,3}^{20,40} & n_{1,2}^{20,40} & n_{1,1}^{20,40} \\ n_{2,3}^{20,40} & n_{2,2}^{20,40} & n_{2,1}^{20,40} \\ n_{3,3}^{20,40} & n_{3,2}^{20,40} & n_{3,1}^{20,40} \end{bmatrix}, \tag{3.15}$$

where $n_{i,j}^{20,40}$ is the value in row i and column j of $N^{20,40}$.

Once we have the results of the Monte Carlo simulations in 3×3 arrays for each cell, we can build the $\mathbf{P}$ matrix. Each cell is numbered, and each row is populated with the values from one of the N arrays. This makes a sparse $\mathbf{P}$ matrix because each row has no more than nine values, regardless of the size of the workspace. An area 100×100 cells would have $\mathbf{P} \in \mathbb{R}^{10^4 \times 10^4}$, but very few of them are filled. We use the `sparse` type in MATLAB to take advantage of the sparsity to increase calculation times by orders of magnitude.

3.4.2 Calculation

For a uniform distribution where only one set of Monte Carlo simulations is required, the simulation method is fast and logical. When the population is spread in other ways, it can take a very long time to simulate each cell, even presuming that the rule for movement is fairly simple. In these cases it is much more convenient to calculate the transition matrices by other methods. We determine the $\mathbf{P}$ matrix differently for the two environments: in the walled environment we know the probabilities directly, while in the normal distribution environment we only know the stationary distribution associated with the process.

Walled Environment

For the walled environment, we begin with a probability k that the particle will leave its current cell and a fraction of k for particles that move in a cardinal direction, denoted as sk, and a fraction that move in a diagonal direction dk. We used the results of the N arrays from our Monte Carlo simulations to select these proportions as 23% and 2%, respectively, but these could be selected in any way that matched the movement of the particles. The new N array is of the form

$$N = \begin{bmatrix} dk & sk & dk \\ sk & 1-k & sk \\ dk & sk & dk \end{bmatrix}. \tag{3.16}$$

It is used to construct $\mathbf{P}$ in the same way discussed in Section 3.4.1.

This method has the advantage of easily extending itself to the sticky wall environment that we find of greater interest than the uniform distribution environment that is so simple to simulate. We can use the sticking coefficient s to modify the N arrays around the edges of the workspace. Then we have

$$N = \begin{bmatrix} 0 & dk + sk + s\ dk & (1-s)dk \\ 0 & sk + (1-k) + s\ sk & (1-s)sk \\ 0 & dk + sk + s\ dk & (1-s)dk \end{bmatrix} \tag{3.17}$$

for the cells along the left boundary and similar values for other boundaries through rotating and reversing the matrix. The bottom left corner would have

$$N = \begin{bmatrix} 0 & sk + dk + s\frac{dk}{2} & (1-s)dk \\ 0 & 2sk + dk + (1-k) & dk + sk + s\frac{dk}{2} \\ 0 & 0 & 0 \end{bmatrix} \tag{3.18}$$

with the additional reflection. The $s\frac{dk}{2}$ terms in $n_{1,2}$ and $n_{2,3}$ split the "stuck" $s * dk$ particles from $n_{1,3}$ evenly between the two cells that neighbor it. It is easily seen that setting $s = 0$ will create the original uniform distribution while setting $s = 1$ will trap the particles in the edge cells without necessarily making any individual cell an absorbing state.

Normal Distribution

We begin to determine $\mathbf{P}$ for the normal distribution by viewing the two-dimensional normal distribution as the product of two independent one-dimensional distributions with equal standard deviations and means in the center of the workspace. This is valid because the bivariate normal PDF is

$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{(x-\mu_x)^2}{2\sigma_x^2} - \frac{(y-\mu_y)^2}{2\sigma_y^2}} \tag{3.19}$$

when the underlying univariate distributions are independent. It can then be rewritten as

$$f(x,y) = \left(\frac{1}{\sqrt{2\pi\sigma_x^2}} e^{\frac{(x-\mu_x)^2}{2\sigma_x^2}}\right)\left(\frac{1}{\sqrt{2\pi\sigma_y^2}} e^{\frac{(y-\mu_y)^2}{2\sigma_y^2}}\right) = f(x)f(y). \tag{3.20}$$

From the mean and standard deviation parameters, we use the cumulative distribution function

$$\Phi(x) = \int_{-\infty}^{x} f(t)dt \tag{3.21}$$

to find the stationary distribution $\mathbf{w}$ in each dimension with

$$w_i = \Phi(i) - \Phi(i-1), \tag{3.22}$$

which is conveniently calculated by the `normcdf` function in MATLAB. Since our workspace is bounded, we set the values in $\mathbf{w}$ for $i \in [1, n]$. The normal distribution trails out on the range $(-\infty, \infty)$, however, so we then normalize $\mathbf{w}$ such that $\sum_{i=1}^{n} w_i = 1$.

Because the stationary distribution is not unique to a given transition matrix, we cannot simply solve $\mathbf{w} = \mathbf{wP}$ as a system of linear equations. Instead we develop a set of constraints for a linear programming problem which will find a solution for $\mathbf{P}$ subject to some minimization criteria. These constraints come from the definition of a transition matrix and its relationship to the stationary distribution. The problem is set up as $\mathbf{Ax} = \mathbf{B}$ where $\mathbf{x}$ contains the values we wish to solve for in the transition matrix.

In one dimension, a particle can move either left or right or stay where it is. This means that there are only three values for each row of the transition matrix except for the edges where there are only two values making a total of $3n-2$ variables to solve for. In order to to reduce the number of variables required we use the symmetry of a distribution centered in the workspace to eliminate almost half the variables. For $n=9$, the 9×9 matrix

$$\mathbf{P}=\begin{bmatrix} x_1 & x_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_3 & x_4 & x_5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & x_6 & x_7 & x_8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & x_9 & x_{10} & x_{11} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_{12} & x_{13} & x_{14} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_{15} & x_{16} & x_{17} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & x_{18} & x_{19} & x_{20} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x_{21} & x_{22} & x_{23} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_{24} & x_{25} \end{bmatrix} \tag{3.23}$$

can be reduced to a 5×5 matrix

$$\mathbf{P}=\begin{bmatrix} x_1 & x_2 & 0 & 0 & 0 \\ x_3 & x_4 & x_5 & 0 & 0 \\ 0 & x_6 & x_7 & x_8 & 0 \\ 0 & 0 & x_9 & x_{10} & x_{11} \\ 0 & 0 & 0 & 2x_{12} & x_{13} \end{bmatrix} \tag{3.24}$$

so that the $3n-2$ variables are reduced to $3\lceil n/2\rceil-2$ variables.

With the variables determined, we establish the constraint equations. The first set of constraints requires each row of $\mathbf{P}$ to sum to unity. This gives the first rows of $\mathbf{A}$ as $\mathbf{A}_{top}\in\mathbb{R}^{\lceil n/2\rceil\times 3\lceil n/2\rceil-2}$ as a matrix containing mostly ones and zeroes and the first rows of $\mathbf{B}$ as $\mathbf{B}_{top}\in\mathbb{R}^{\lceil n/2\rceil\times 1}$ filled with ones. Continuing the example for $n=9$, we have

$$\mathbf{A}_{top}=\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \end{bmatrix}, \mathbf{B}_{top}=\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{3.25}$$

where the two in the last row of $\mathbf{A}_{\mathbf{top}}$ comes from the symmetry of the distribution.

The second set of constraints forces the solution to satisfy $\mathbf{w} = \mathbf{wP}$. This creates a set of equations that begins

$$\begin{aligned} w_1 &= w_1 x_1 + w_2 x_3 \\ w_2 &= w_1 x_2 + w_2 x_4 + w_3 x_6 \\ w_3 &= w_2 x_5 + w_3 x_7 + w_4 x_9 \end{aligned} \tag{3.26}$$

and continues until there are $\lceil n/2 \rceil$ equations. These form the lower rows of $\mathbf{A}$ as $\mathbf{A}_{bot} \in \mathbb{R}^{\lceil n/2 \rceil \times 3\lceil n/2 \rceil - 2}$ and the lower rows of $\mathbf{B}$ as $\mathbf{B}_{bot} \in \mathbb{R}^{\lceil n/2 \rceil \times 1}$ which is equal to $\mathbf{w}^T$. For the $n = 9$ example

$$\mathbf{A}_{bot} = \begin{bmatrix} w_1 & 0 & w_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_1 & 0 & w_2 & 0 & w_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_2 & 0 & w_3 & 0 & w_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_3 & 0 & w_4 & 0 & w_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2w_4 & 0 & w_5 \end{bmatrix}, \mathbf{B}_{bot} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix}. \tag{3.27}$$

The $2w_4$ in the last row of $\mathbf{A}_{bot}$ again comes from the symmetry of the distribution.

We add one final constraint to allow some control over the rate at which the particles move. We call this the diffusion rate k, and it can take any value on the range $[0, 1]$. It is the proportion of particles that move out of the center cell of the distribution. This leads to the constraint

$$x_{3\lceil n/2 \rceil - 3} = \frac{k}{2}, \tag{3.28}$$

which in matrix form is

$$\mathbf{A}_{last} = \begin{bmatrix} 0 & \dots & 0 & 1 & 0 \end{bmatrix}, \mathbf{B}_{last} = \begin{bmatrix} k/2 \end{bmatrix}. \tag{3.29}$$

The constraints are all combined into pair of matrices with

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{top} \\ \mathbf{A}_{bot} \\ \mathbf{A}_{last} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{B}_{top} \\ \mathbf{B}_{bot} \\ \mathbf{B}_{last} \end{bmatrix}. \tag{3.30}$$

We next set the limits on the values of $\mathbf{x}$ from the rules of probability, that is $x_i \in [0,1]\ \forall\ i \in [1,n]$. The solver also requires minimization criteria which we set to be one for every variable, giving $f = [1 \ldots 1]$. These components are supplied to MATLAB's `linprog` function, which solves $\mathbf{Ax} = \mathbf{B}$. We can fill the first rows of $\mathbf{P}$ with the values of $\mathbf{x}$ from their definition, which is shown for the $n = 9$ cell example in Equation 3.23. The remaining rows are filled by recognizing that the rows for cells to the right of the center of the distribution mirror the rows for cells to the left of the center. For $n = 9$, symmetry gives $x_{14} = x_{12}$, $x_{15} = x_{11}$, *etc.*

With the linear programming problem, we are able to establish a transition matrix from the stationary distribution with a single tuning variable that controls the rate at which the population diffuses from the center into the surrounding cells. This transition matrix can be verified by calculating the stationary distribution as shown in Section 3.2. It should match the original definition of the population distribution.

Now that we know how to find the probabilities of transition in one dimension, we can expand it to two dimensions. We find $\mathbf{Px} \in \mathbb{R}^{n \times n}$ for one dimension and $\mathbf{Py} \in \mathbb{R}^{n \times n}$ for the other. We can then fill $\mathbf{P} \in \mathbb{R}^{n^2 \times n^2}$ from $\mathbf{Px}$ and $\mathbf{Py}$. For a cell in row i and column j of the workspace, we can create an N array

$$N = \begin{bmatrix} py_{j,j-1} \\ py_{j,j} \\ py_{j,j+1} \end{bmatrix} \begin{bmatrix} px_{i,i-1} & px_{i,i} & px_{i,i+1} \end{bmatrix} = \begin{bmatrix} px_{i,i-1}py_{j,j-1} & px_{i,i}py_{j,j-1} & px_{i,i+1}py_{j,j-1} \\ px_{i,i-1}py_{j,j} & px_{i,i}py_{j,j} & px_{i,i+1}py_{j,j} \\ px_{i,i-1}py_{j,j+1} & px_{i,i}py_{j,j+1} & px_{i,i+1}py_{j,j+1} \end{bmatrix} \tag{3.31}$$

which is used to fill $\mathbf{P}$ in the same way it was in Section 3.4.1.

3.5 Applying the Markov Process

With a transition matrix to describe the population movement, we can apply the Markov process to our simulations. Since the stationary distribution $\mathbf{w}$ is independent of initial conditions and represents the state of the population a "long time" after initial-

ization, we use it as the state of our population at the beginning of a coverage path trial. We then loop for the duration of the trial, applying the transition matrix to the current population distribution to simulate particle movement and the search path coordinates to the robot position to simulate its movement. We determine which cells the robot has visited in the step and remove a fraction of the population from each of them. This repeats for the length of the trial. Figure 3.10 illustrates this with a series of snapshots from the recovery of the normal distribution after removing a 2 m-wide swath of particles through the center of the distribution. From top left to bottom right, it begins at $t = 0$ s and progresses until it has recovered a normal distribution at $t = 50$ s. The robot is not searching during this example series.

Because the Markov process makes the population's behavior repeatable, we can run one trial for a pre-planned path that does not vary based on any kind of feedback, and the results will be identical to results from repeat tests of the same path. This allows us to test variations in the parameters for path planning or particle movement rapidly. While paths that include feedback to make decisions often make the same decisions in successive processes, errors in mathematical precision may periodically cause differences in runs. Since they are still likely to be similar, tests can be run by the dozen instead of by the hundred. We next apply the Markov process to our environments and consider the merits of the various paths.

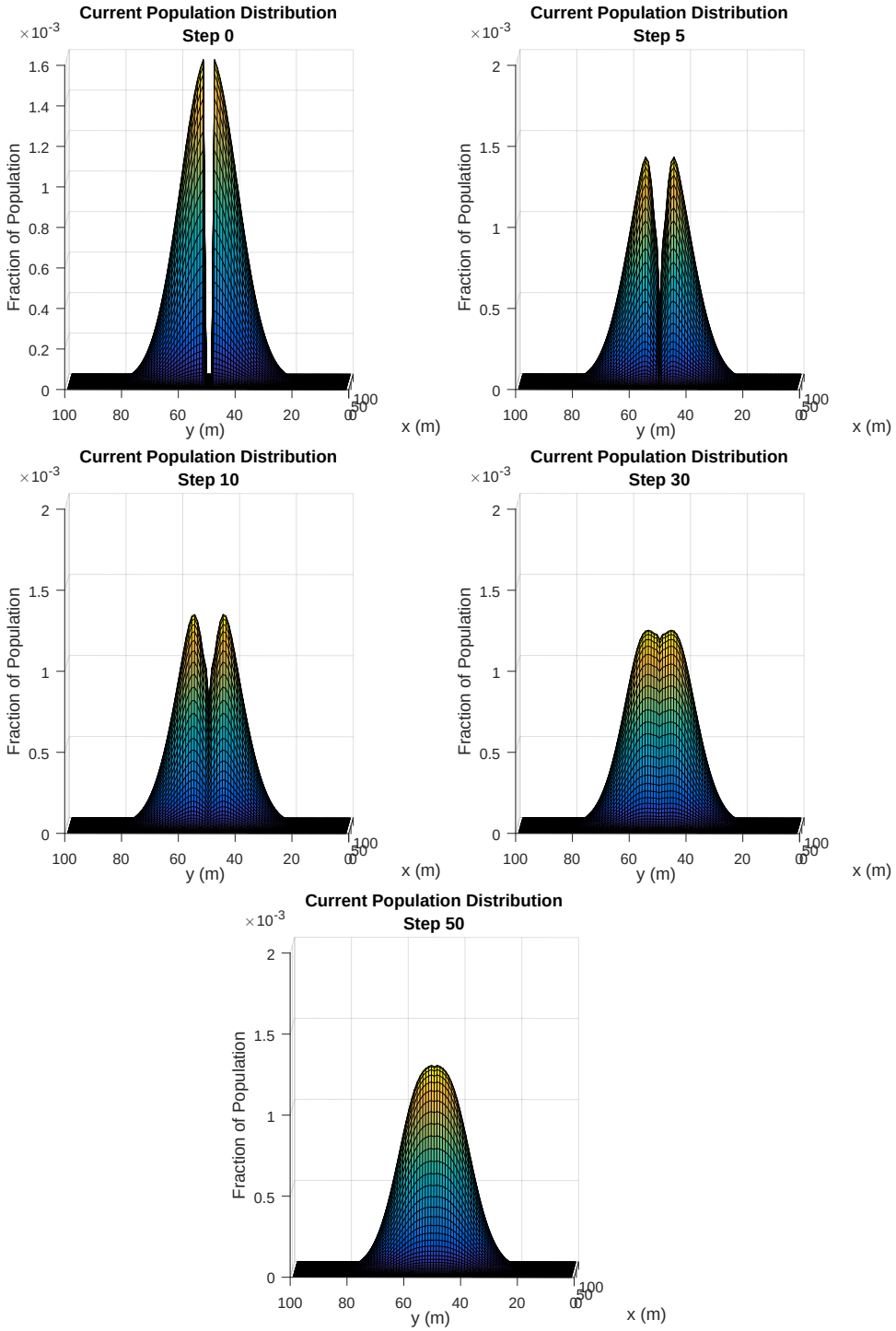

Figure 3.10: Recovery of the normally distributed population after removing all particles in a swath through the center of the distribution. Times are $t \in \{0, 5, 10, 30, 50\}$ s.

Chapter 4

Path Performance

4.1 Introduction

As we described in the introduction to this work, when given a static population or unlimited time, a boustrophedon coverage path with spacing equal to the sensing range will guarantee perfect coverage. In our case we have neither a static population nor unlimited time, and we seek a path that will sample the largest number of members of the population within a limited working time.

4.2 Offline and Online Planning

Path-planning methods may be divided into two categories: offline and online. With an offline algorithm, a path is completely specified using an existing map of the area, and the robot carries out the plan without deviation. The plan may use assumptions about the probable value of different parts of the workspace, but the robot does not make any decisions about the course once it begins. It just uses localization information to maintain its position on the route. In an online algorithm, the robot makes decisions about its path as it moves, generally using some kind of sensor data. It may react to obstacles that are not included in the map or dwell in a rewarding area as with the methods in Chapter 2 that included the square spiral.

4.3 Path Plan Descriptions

We consider a variety of paths that are composed of straight-line segments. This simplifies determining how much time the robot spends in each cell in a time step because we can calculate the intersections between the path segment and the workspace grid and find the length of the path between those intersections. For each path, the robot can begin the path again if time permits. We do not require the robot to return to a starting point or home position, presuming that the energy will be budgeted to allow the robot to move from its home position to the starting point and from the ending point back to its home position.

4.3.1 Boustrophedon

The boustrophedon path described in Chapter 1 and shown in Figure 1.2 is the first path option we consider, but we do not limit it to closely spaced rows that guarantee full coverage of the workspace. We run the path from edge to edge of the workspace in one direction but vary the spacing between rows so that they may be further apart than the sampling mechanism's range. This allows us to cover the area somewhat more evenly when the time restrictions preclude covering the whole field in a trial. This is an offline path.

4.3.2 Wall-Following

By adding the sticking coefficient to the walled environment, we create a space in which the boustrophedon path is not necessarily the best path. Since we know that the cells along the walls contain a larger share of the stationary distribution than the center cells, we use a wall-following path. It makes circuits of the workspace, moving through the edge cells in a path like that shown in Figure 4.1. This is also an offline path.

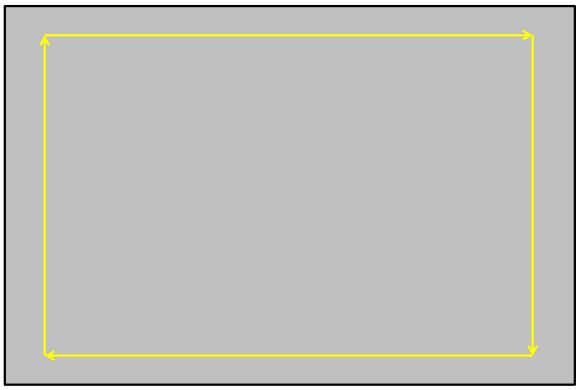

Figure 4.1: Sample wall-following path.

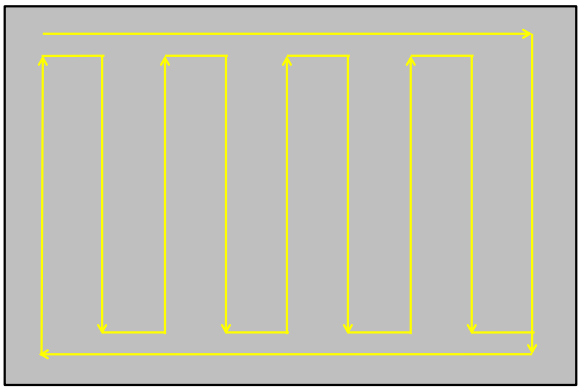

Figure 4.2: Sample hybrid wall-following and boustrophedon path.

4.3.3 Hybrid

We also use a hybrid of the wall-following and boustrophedon methods to produce a path that explores the more highly populated walls before exploring the central area as indicated in Figure 4.2. This hybrid first executes one circuit of a wall-following path then follows a boustrophedon path through the interior of the workspace. The boustrophedon path spacing may be varied to cover more or less of the center.

4.3.4 Square Spiral

Another attractive path is an Archimedean spiral with turns separated by the sensor's range as in Figure 4.3. Since it is difficult both to divide this kind of smooth spiral

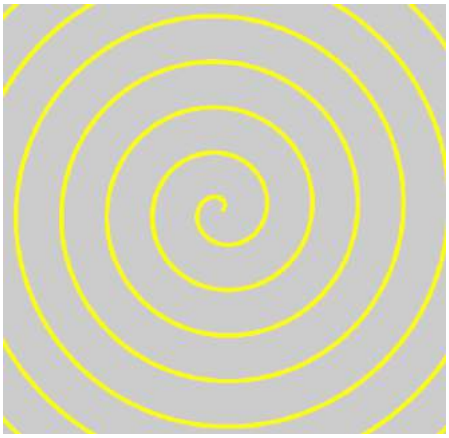

Figure 4.3: Archimedean spiral.

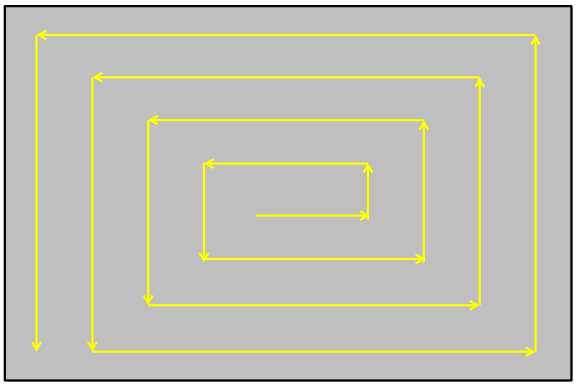

Figure 4.4: Sample square spiral path.

into uniformly sized steps and to calculate the time spent in each cell for curved paths, we use a squared version of the spiral shown in Figure 4.4. As with the boustrophedon path, the turns of the spiral may be separated by more than the sensor's range in order to have more even coverage in limited time. The spiral may begin in the center and spiral out as shown by the arrows in Figure 4.4, but it can also begin at one corner and spiral in toward the center. We also consider methods in which the direction of the spiral reverses based on the proportion of the hunted population that is inside or outside of the robot's current position on the spiral. When the robot cannot decide to change direction, this is an offline path, but it becomes an online path when the direction changes based on population information.

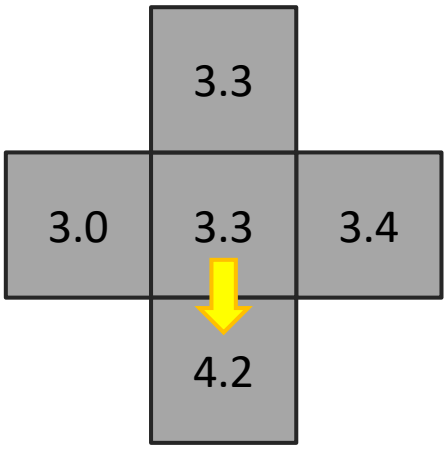

Figure 4.5: An example of a single greedy step.

4.3.5 Greedy

The final path is an online path in which the robot behaves in a greedy or opportunistic way, choosing the most advantageous of its current movement options. We assume that the robot can sense the particle count in its current cell and the cells surrounding it. At every step it has five options; it can remain where it is or move one cell up, down, left, or right. Whichever of the five cells has the highest particle count will be selected for a move. Diagonal moves are not permitted because they are longer than moves in the cardinal directions. A sample move is shown in Figure 4.5.

We can extend this to a robot with a wider sensor range that can sense the particle count up to two cells away. In this case the plan considers two steps at a time and chooses the most opportune of the options. First it looks at the surrounding cells. Then it calculates what the reward will be for a second step if it takes each of the five possible steps, giving it 25 options. Figure 4.6 shows a sample move.

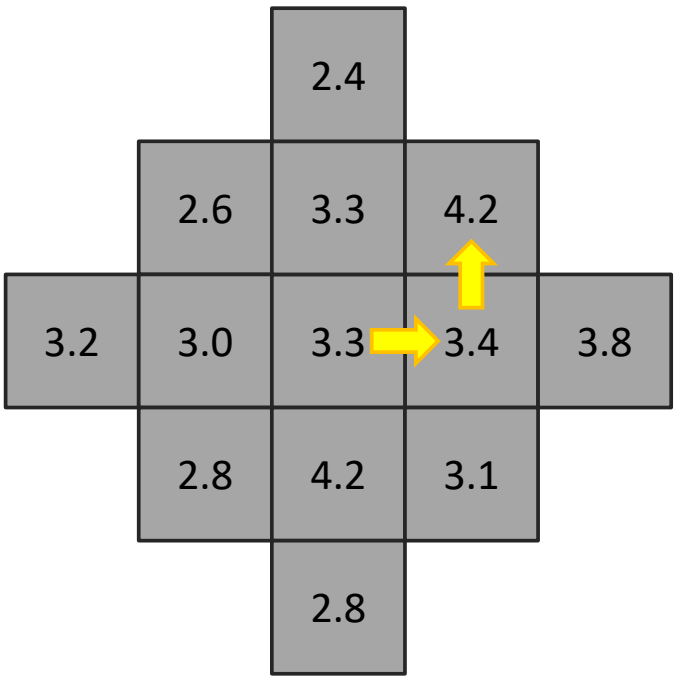

Figure 4.6: An example of a double greedy step.

4.4 Results

We consider a set of paths for each environment. The walled environment with the added sticking factor has a higher concentration of particles around the outer edges and a lower concentration in the center so we begin all tests in one corner and consider each of the following path options:

1. Boustrophedon
2. Wall-following
3. Hybrid
4. Square Spiral
5. One-step greedy
6. Two-step greedy

For the normally distributed environment, the population is concentrated in the center of the space so we begin each test in the center of the workspace. We use the following path options:

1. Square spiral
2. Square spiral with 50% threshold for turning in and continuing all the way to the center
3. Square spiral with 80% thresholds for both turning in and turning out
4. One-Step Greedy
5. Two-Step Greedy

Because this environment clearly has a more valuable area and boustrophedon coverage would spend a lot of time away from that area, we did not test that path in this world. Likewise we do not examine the wall-following path because it would search only the least useful part of the space. The square spiral can offer complete spatial coverage like the boustrophedon, but by starting in the center, it will cover the most rewarding places first.

Since the parameters of a search can affect the best search methods, we test each path with a variety of parameters. We use both 300 s and 600 s for the length of the trial T and robot velocities of 6 m/s, 9 m/s, and 12 m/s. The probability of a particle leaving any cell in the uniform distribution environment or of a particle leaving the center cell in the normal distribution environment is 0.2, 0.4, or 0.6. The robot's removal rate for particles in the cells it visits is in the set $\{0.5, 0.7, 0.9\}$, and the spacing of the boustrophedon rows or square spiral turns is set as a fraction of the workspace length L and is 2%, 5%, 10%, or 20%. In addition to these parameters, which apply to both environments, we test varying sticking coefficients with $s \in \{0.25, 0.5, 0.75\}$ in

the uniform environment with sticky walls and the standard deviation of the normal distribution with $\sigma \in \{L/20, L/10, L/5, L/2\}$.

For each of the offline, pre-planned paths, we perform one trial for each parameter set. Since the greedy algorithm has the ability to change its path based on the best of the current options, it may make slightly different decisions in consecutive runs due to numerical precision errors. To account for this, we average the results of twenty runs for each parameter set.

4.4.1 Uniform Distribution with Sticky Walls

We begin our analysis with the uniform distribution in a sticky wall environment. For every combination of parameters, the two-step greedy algorithm outperforms all others. We expect to see methods based on feedback perform better than open loop methods, but we see that this is not necessarily the case here. While the two-step greedy method is the best, the one-step greedy method is often out-performed by methods with offline planning. Figure 4.7 shows this overall ranking of

1. Two-step greedy
2. Hybrid
3. Square spiral
4. Boustrophdeon
5. Wall-following
6. One-step greedy.

This ordering is not universal, and we find that varying the parameters of the path and transition matrix can affect the relative merit of the paths.

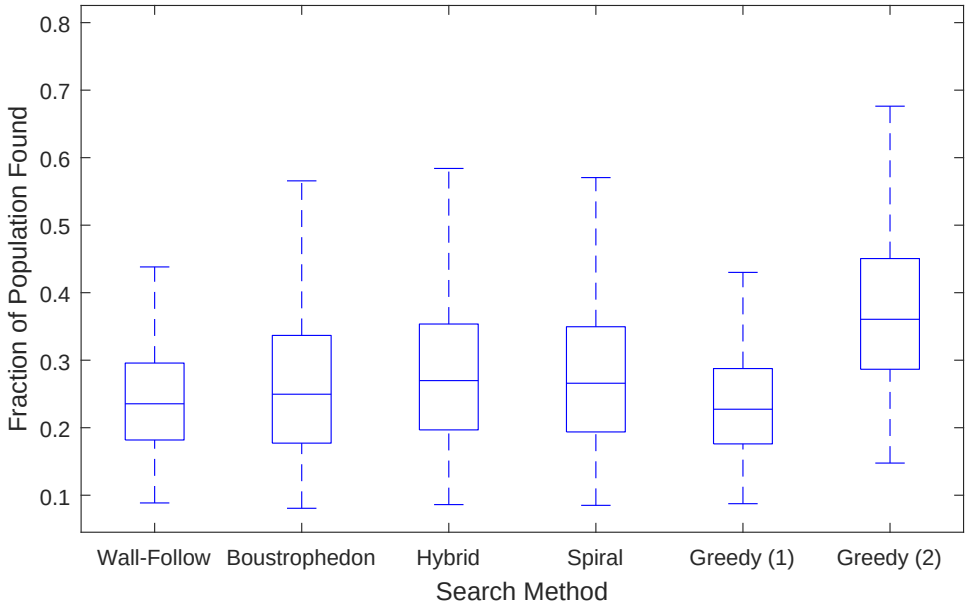

Figure 4.7: Comparison of success rates for paths across all parameter combinations.

We consider first the effects of trial time and robot speed on the results. While a longer trial obviously has better success in collecting swarm members, the relative merit of the planning methods changes with duration. For brief runs, no path has a marked advantage over any other with the exception of the two-step greedy path, as Figure 4.8 illustrates. Higher robot speeds have similar effects. Faster speeds again have higher success rates, but the magnitude of the improvement varies by path with the boustrophedon, hybrid, and spiral paths being much more strongly influenced by speed than the wall-following path and the two-step greedy path more than the one-step. (See Figure 4.9.) When we consider only the combination of lowest speed and time in comparison with the combination of highest speed and time as in Figure 4.10, we see that the wall-following path is second only to the two-step greedy path when little of the workspace can be covered but is overtaken by the other pre-planned paths when the robot is able to explore a greater area.

Varying the robot's removal rate shows similar behavior to varying its velocity as seen in Figure 4.11. It is more significant for the same methods and shows the same

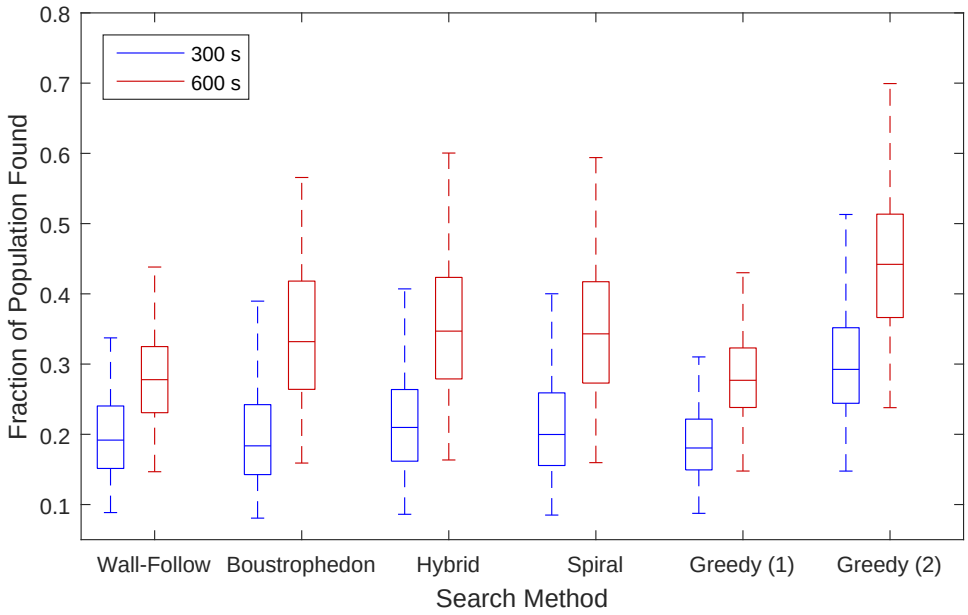

Figure 4.8: Comparison of success rates for paths for long and short trials.

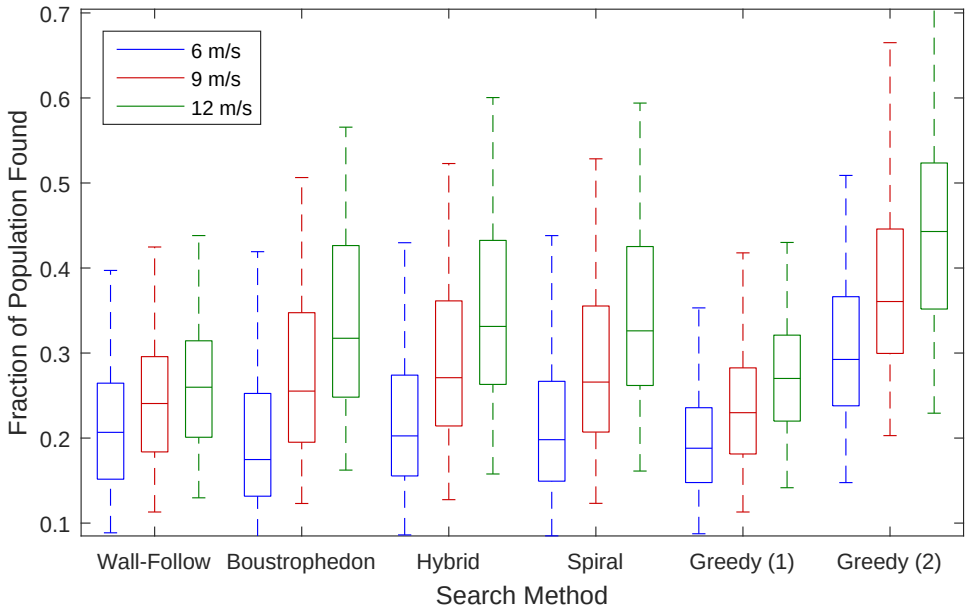

Figure 4.9: Comparison of success rates for paths with varying robot velocities.

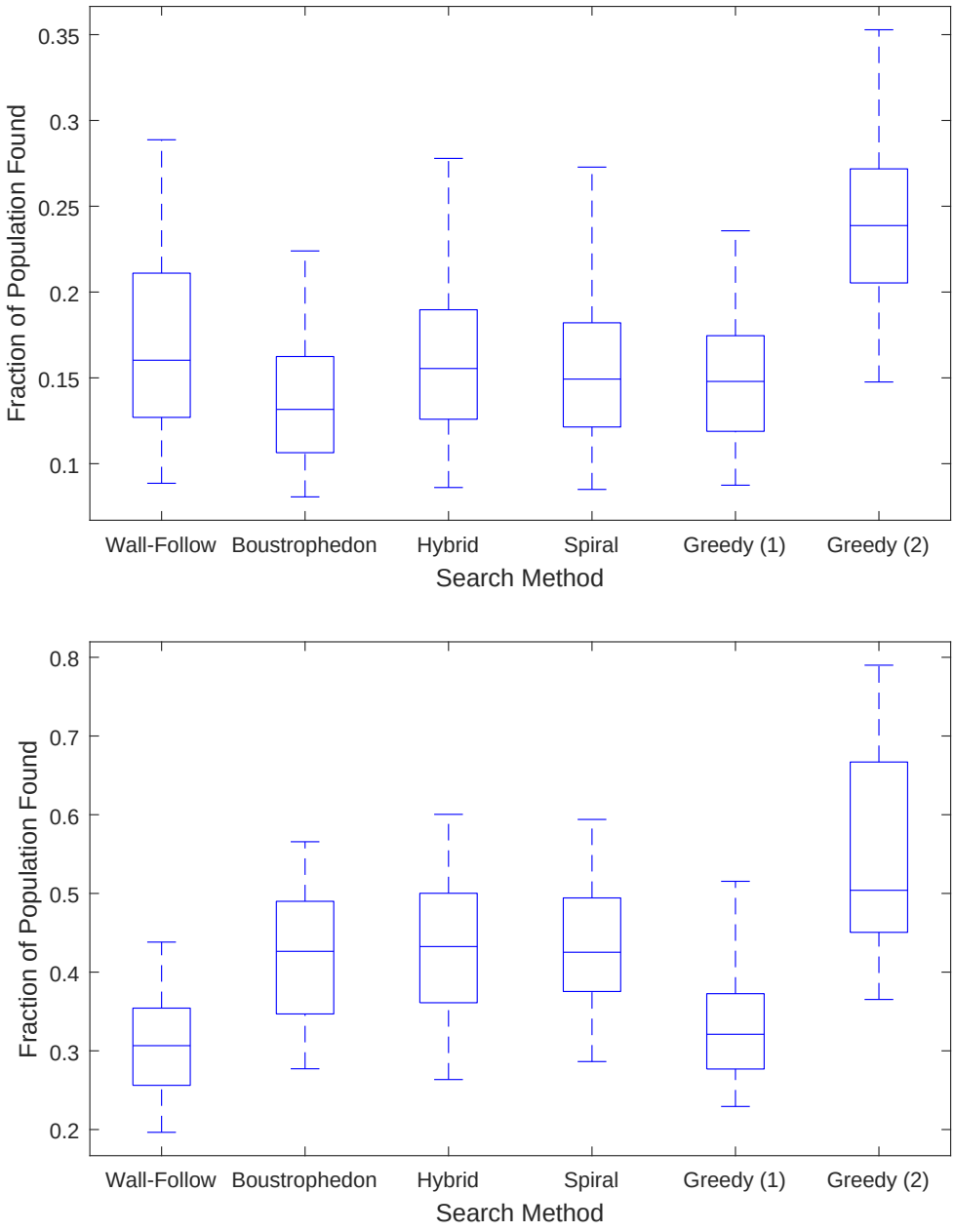

Figure 4.10: Comparison of success rates for paths with low speed and time (top) vs. high speed and time (bottom).

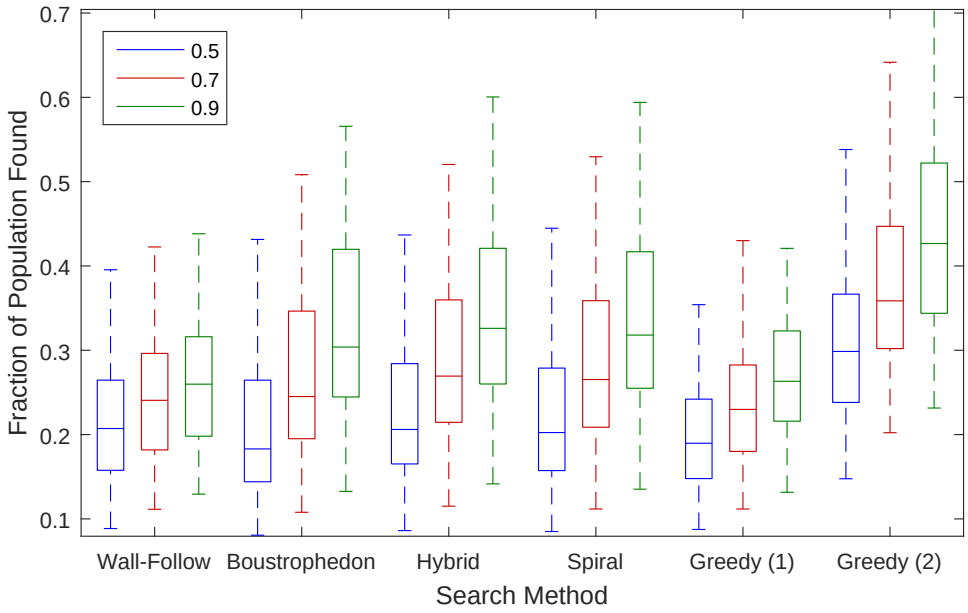

Figure 4.11: Comparison of success rates for paths with varying removal rates.

impact on relative performance. With lower removal rates, the methods that tend to remain near the walls are more compare more favorably to other methods than they do with higher removal rates.

The final robot parameter is the row spacing for path-planning. Since there is no spacing associated with either the wall-following or greedy paths, they are not considered here. The paths for which row spacing does apply are shown in Figure 4.12, and we see that it does not have a strong influence on the performance, but that $L/10$ is slightly better for the hybrid and spiral methods with $L/20$ giving the best results for the pure boustrophedon method. This is because the widest spacing allows the boustrophedon to spend more time running along the edges of the workspace as it repeats the path, giving it something more of the character of the hybrid path.

We now consider the effects of two parameters of the swarm transition matrix. The performance dependence upon the sticking coefficient is seen in Figure 4.13. This has its strongest effect on the wall-following path, which might be predicted from Figure 3.6, but even at 75%, it does not have a strong enough influence to make the wall-following

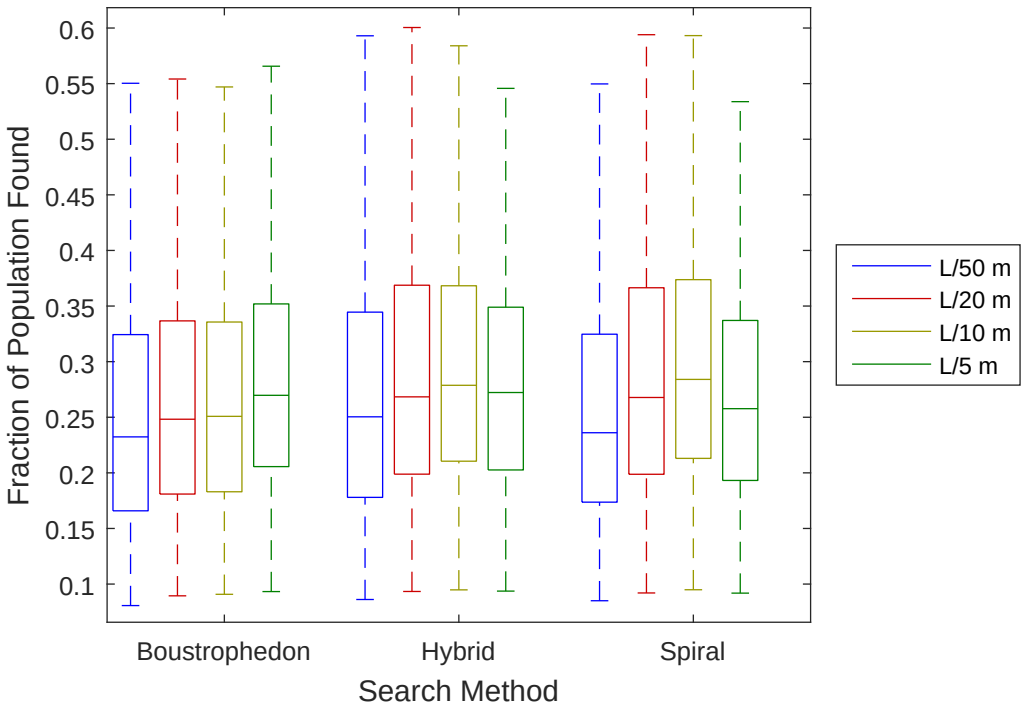

Figure 4.12: Comparison of success rates for paths with varying row spacing.

method better than the hybrid. Sticking coefficient has less effect upon the feedback-driven greedy methods. The second swarm parameter is the probability of transition, and Figure 4.14 demonstrates that this is not a very important parameter. Higher transition rates are beneficial to the wall-following and one-step greedy methods but have very little effect on any of the others.

When we look at all these results in combination, we see that the effects on the robot's success rate is driven by how much time the robot spends near the walls combined with how likely the swarm members are to be at the edges of the workspace. This drives most of the dependencies discussed above. We know the walls should be explored first, and with less time, this means that focusing on the walls is advantageous. High sticking coefficients and lower removal rates both keep higher populations near the walls longer into the trial, improving the reward accrued when traveling along the edges of the workspace. The differences between the low and high time and speed performance for the

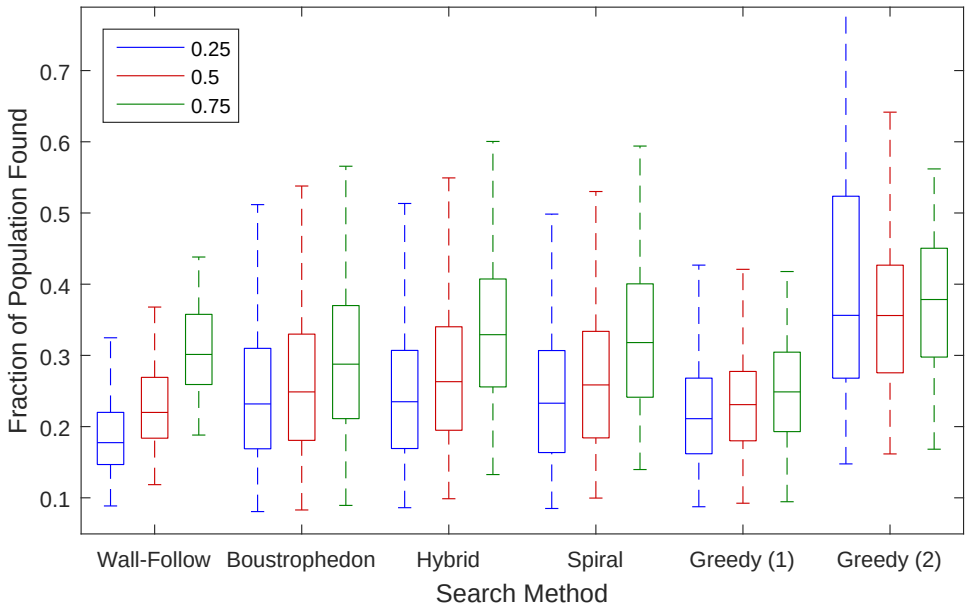

Figure 4.13: Comparison of success rates for paths with varying sticking coefficients.

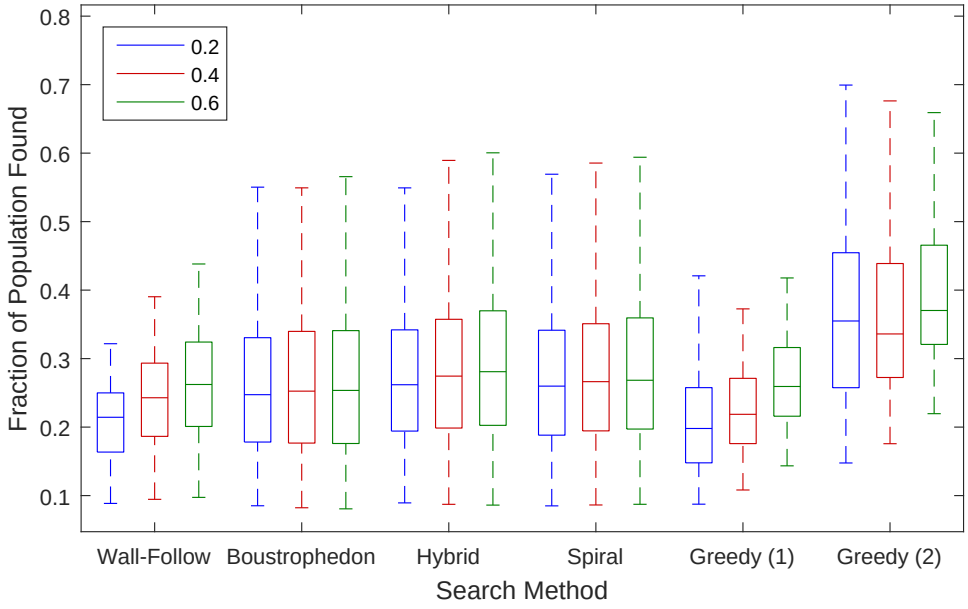

Figure 4.14: Comparison of success rates for paths with varying particle transition probabilities.

	95	96	97	98	99
1	0.07	0.06	0.31	0.47	0.39
2	0.44	0.42	0.38	0.32	0.25
3	0.58	0.55	0.49	0.41	0.31
4	0.70	0.66	0.59	0.48	0.37

Figure 4.15: Local distribution around robot at $\{x, y\} = \{1, 97\}$ (blue square) at $t = 74\ s$. One step greedy chooses $\{x, y\} = \{1, 98\}$ for the next move, but two-step greedy would choose $\{x, y\} = \{2, 97\}$.

wall-following and one-step greedy paths from Figure 4.10 are driven by the reduction in value of the walls as time extends and the population has been greatly reduced around the walls.

This also drives the often poor results from the one-step greedy search. The robot initially follows the walls because the highest value cells are along the walls. Eventually, the local distribution looks like Figure 4.15, where the robot's reference location is the blue square, and the one-step greedy method is simply too short-sighted to move away from the walls because the cells one step away from the walls tend to be depleted even further than the wall cells by the stickiness of the walls. Eventually, the edge cells will be so little populated that it will move into the center, but only by looking forward two steps can the greedy algorithm go beyond the less rewarding neighboring cell to begin exploring the interior of the area at a reasonable point in time. The path in Figure 4.16 shows a typical path for the two-step greedy method with initial steps following the walls and an eventual foray into and exploration of the interior of the workspace.

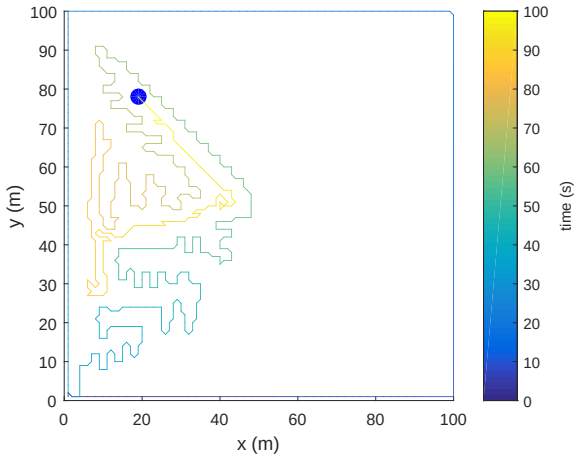

Figure 4.16: Sample two-step greedy path after a 100 s simulation at 6 m/s.

4.4.2 Normal Distribution

We now turn to the test results for the environment with the normally distributed swarm. If the normal distribution reforms instantly, it is clear that the robot's best course of action is to stay in the center cell of the distribution, and this is corroborated by simulations with the two-step greedy path that show that the robot never moves as in Figure 4.17. In this case, with a removal rate of r and a static collection area, we can calculate the total remaining population at each time interval. Since the normal distribution is well-defined, a constant fraction of the population will be in the collection area at each time step. We call this fraction a. In each iteration, the robot then removes ra particles from the population. As time passes, this follows the trajectory

$$n(t) = n(0)(1 - ra)^t, \tag{4.1}$$

where $n(t)$ is the number of members of the population that have not been collected as of time t. By subtracting this from the initial total population, we have

$$d(t) = n(0) - n(t) = n(0)(1 - (1 - ra)^t), \tag{4.2}$$

where $d(t)$ is the number of particles that have been removed at time t.

If the normal distribution takes an infinite time to reform, *i.e.* the particles are

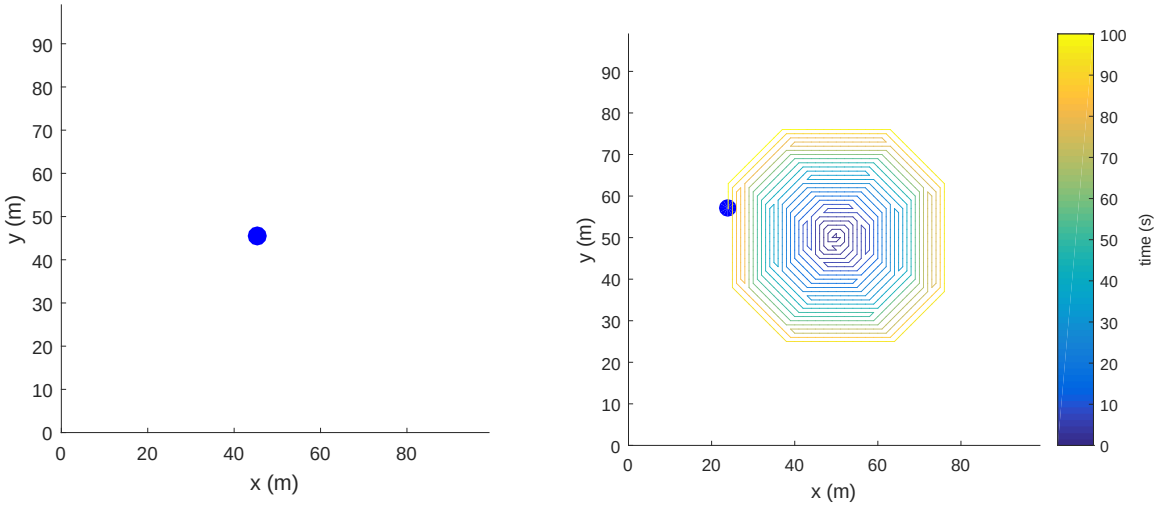

Figure 4.17: Robot paths for extreme cases of normal distribution: instantly reforming (left) and stationary (right).

stationary, the robot should attempt complete coverage moving out from the center until it runs out of time or reaches the edges of the workspace, providing it can collect every particle it encounters. This is accomplished with a spiraling method with no gaps in coverage between turns of the spiral. Simulations of the two-step greedy method support this by creating a path like that shown on the right side of Figure 4.17, which closely resembles a spiral. In this case, the area covered by the robot in each time step is constant, but the fraction a is not constant. This makes the robot's success difficult to calculate analytically, but it can be simulated.

The question we try to answer regards what happens between these two extreme cases. The instant reestablishment of a normal distribution is simple to calculate a solution to, and the stationary population is representative of the classic coverage problem. We use a population that reforms a normal distribution in finite time to investigate the problem.

As with the paths that include spirals in the map-based simulations discussed in Chapter 2, the square spiral paths with turns that we test here are dependent upon the thresholds used for turning. Before we compare these paths to the other paths, we first

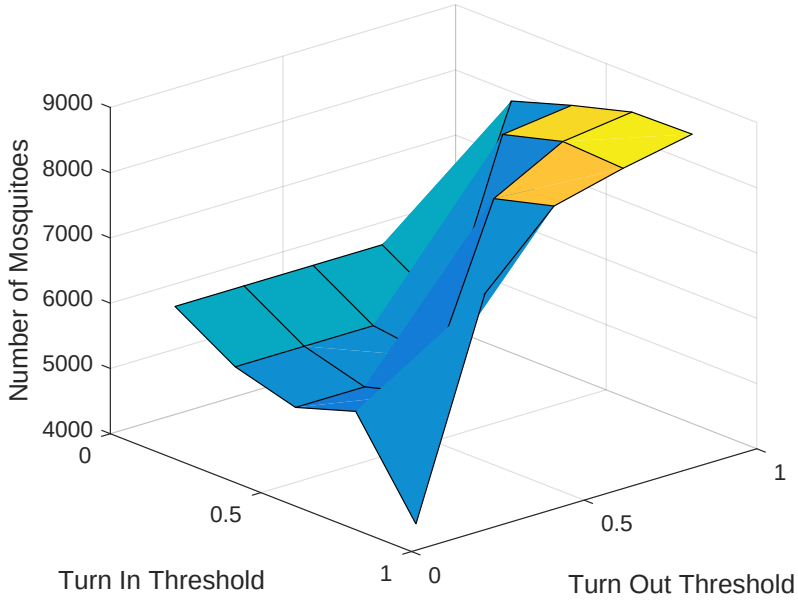

Figure 4.18: Comparison of success rates for varied turning thresholds.

establish a near-optimal level for the thresholds. The turn in threshold is used to switch the robot from spiralling out to spiralling in and is the fraction of the particles that are closer to the center of the workspace than the robot is. The turn out threshold is used to switch the robot from spiralling in to spiralling out and comes from the fraction of the particles that are farther from the center of the workspace than the robot is. Figure 4.18 shows the number of particles out of $10,000$ encountered using turn in and turn out thresholds from the set including $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. This demonstrates that it is best to make turns only when large percentages of the population are not in the direction of travel. We selected 0.8 as the setpoint for both turning thresholds.

As with the sticky wall environment, the two-step greedy algorithm has the best performance with every combination of parameters as shown in Figure 4.19. The order of performance is

1. Two-step greedy

2. One-step greedy

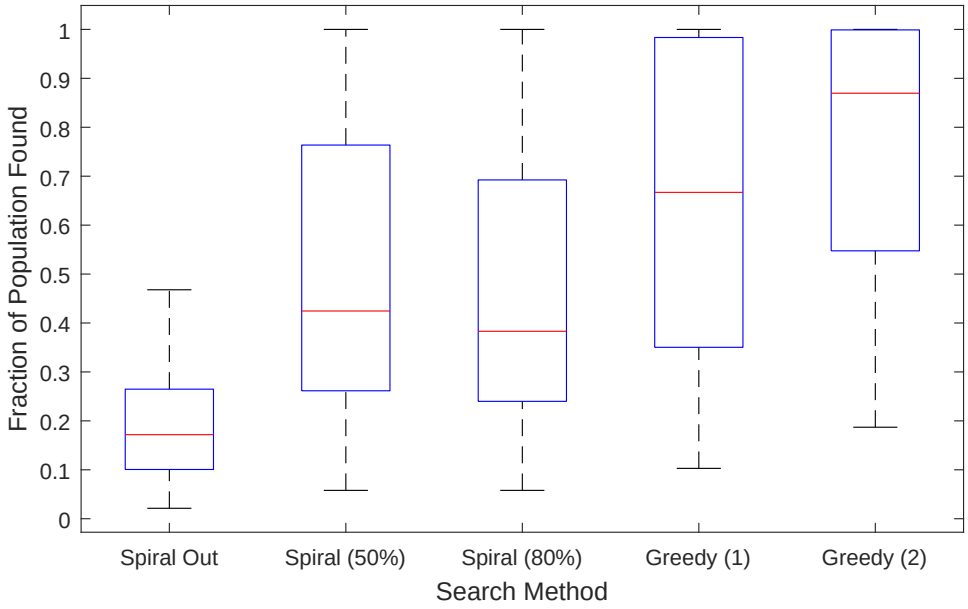

Figure 4.19: Comparison of success rates for paths across all parameter combinations.

3. Square spiral with turn in threshold at 50% and returning to the center

4. Square spiral with turning thresholds at 80%

5. Square spiral with no turning thresholds.

The third path, which returns to the center of the workspace any time it turns back in is more effective than the fourth path because the fourth path does not necessarily return all the way to the center, which tends to be the most populated part of the workspace.

Unlike the walled environment, the relative performance of the path-planning methods did not change as the path and distribution parameters varied. Figures 4.20 and 4.21 show that with longer duration or greater speed the success rates increase, but these affect each path in the same way and to a similar degree. Even when comparing the shorter time and slowest speed to the longer time and highest speed we see the same pattern. Figure 4.22 illustrates this behavior. Figure 4.23 indicates that the removal rate has a similar impact.

The only path-planning parameter that seems to have an uneven effect on the paths

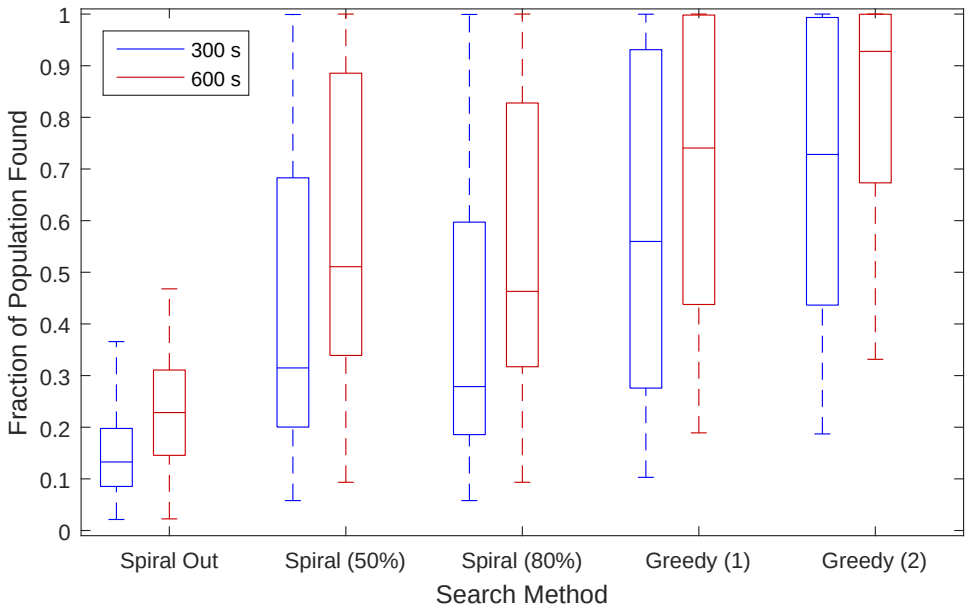

Figure 4.20: Comparison of success rates for paths for long and short trials.

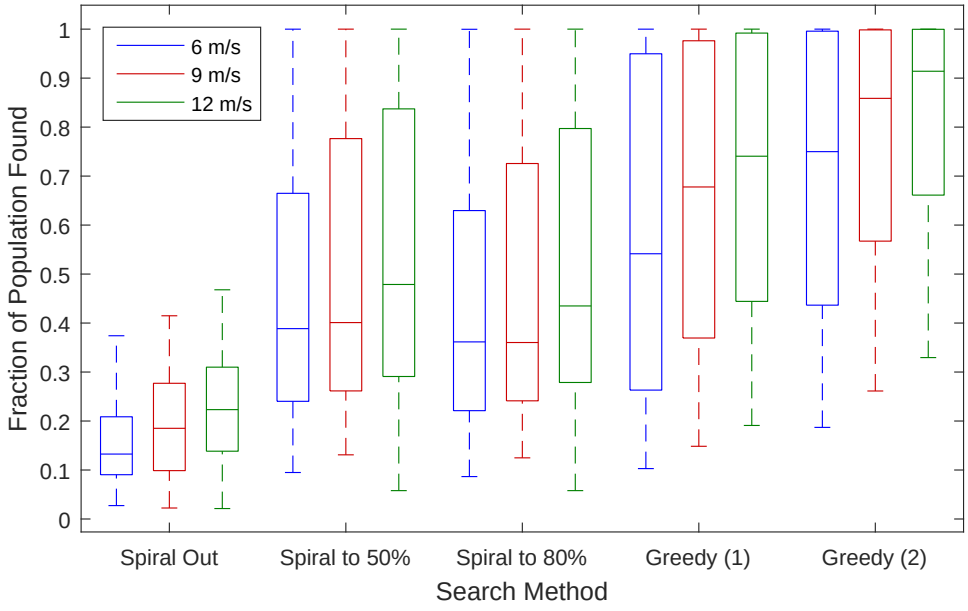

Figure 4.21: Comparison of success rates for paths with varying robot velocities.

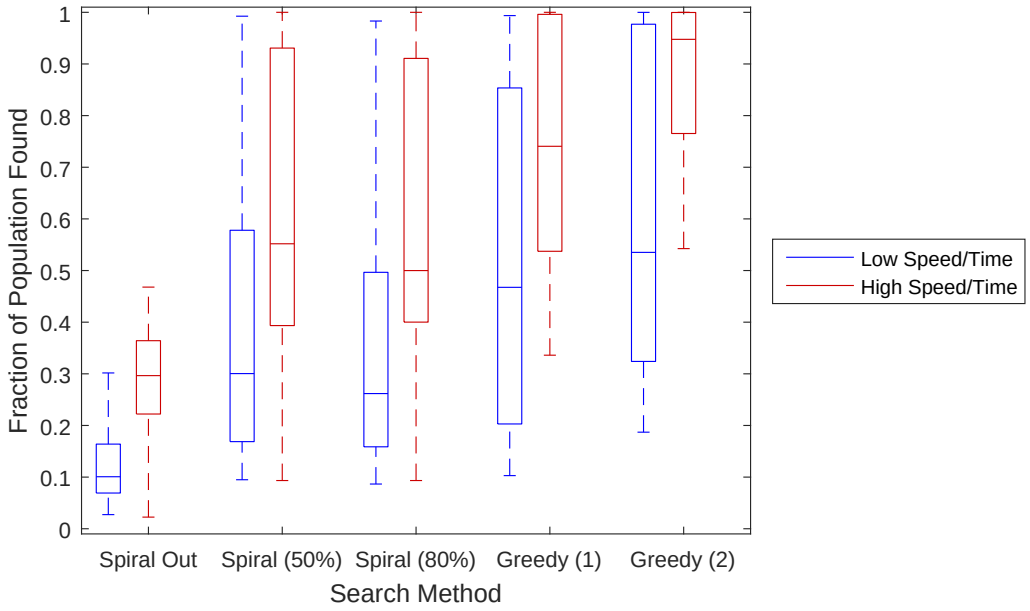

Figure 4.22: Comparison of success rates for paths with low speed and time vs. high speed and time.

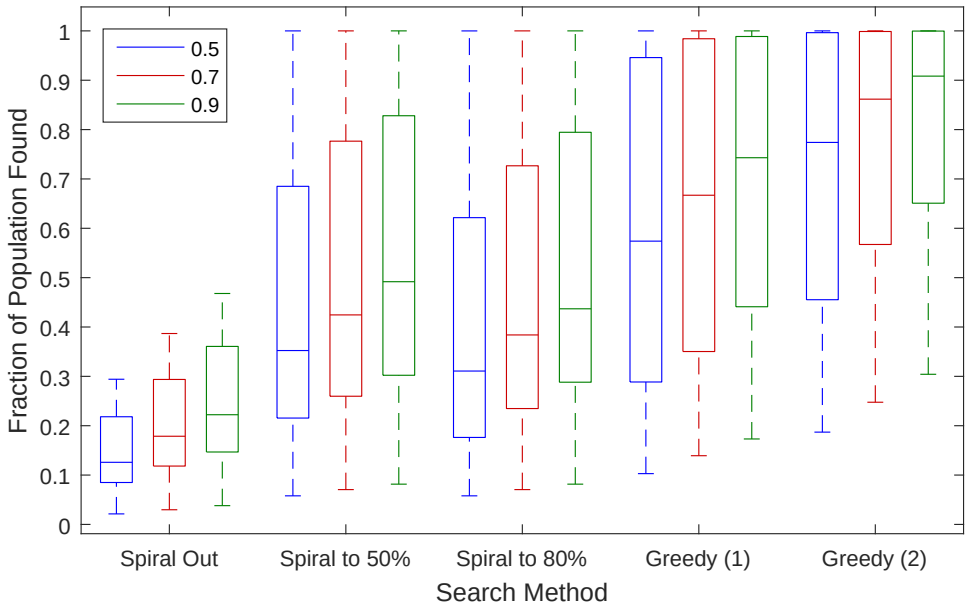

Figure 4.23: Comparison of success rates for paths with varying removal rates.

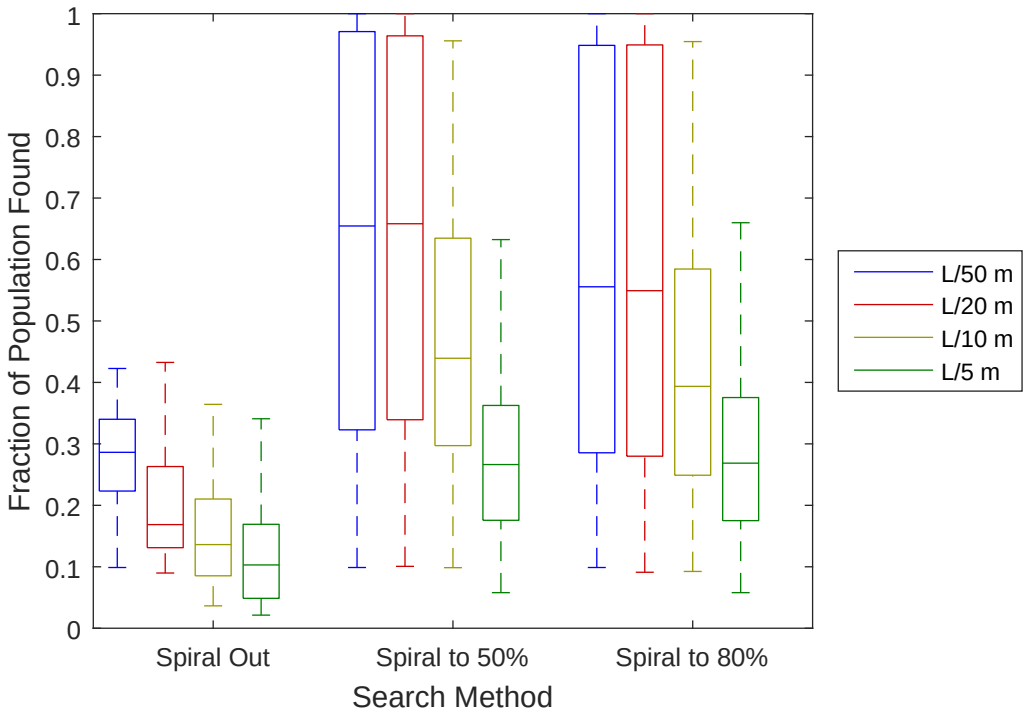

Figure 4.24: Comparison of success rates for paths with varying row spacing.

is the row spacing, which only applies to the square spiral paths both with and without turns. Larger path spacing decreases performance for all three of the paths, but the decrease is much less dramatic for the spiral that only turns when it reaches the edge of the workspace. This is because the wider space between paths makes the total length of the spiral shorter, and the robot turns back to the center sooner and so spends more time revisiting the most rewarding part of the workspace. Since the other paths already do this, their success decreases to a greater degree as they "wait" for the population to recover along the path as it does in Figure 3.10.

As in the sticky wall environment, we see that the probability of transition for the individual particles has little effect on the robot's success in finding the particles. Figure 4.25 shows just how negligible this is. The standard deviation of the population has a much stronger influence on the path performance. It does not modify the order of preference for the paths, but it does significantly impact the efficacy of the search.

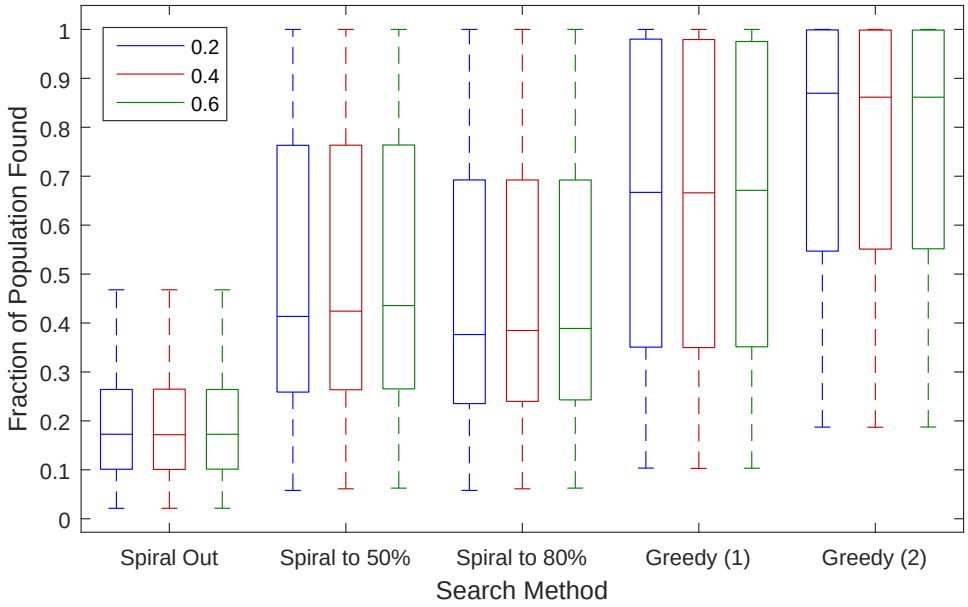

Figure 4.25: Comparison of success rates for paths with varying particle transition probabilities.

Figure 4.26 shows that populations with very tight distributions like $\sigma/20$ can be devastated by the greedy methods for any combination of other parameters. The spirals with turns can also achieve near-perfect removal of the swarm under the right combinations of parameters. As the distribution widens, the performance decreases for all paths except the spiral without threshold turns because the cells become more evenly rewarding and the robot can no longer have so much success for very little motion. The spiral without threshold turns sees improvement in its results as the distribution spreads because the more even value of the cells makes exploration farther from the center of the workspace more rewarding.

4.4.3 Comparisons

From these results we see some general behavior. The greedy algorithms have mixed results. In the sticky wall environment, a one-step greedy algorithm tends to do more poorly than the pre-planned paths because it tends to get stuck in a wall-following

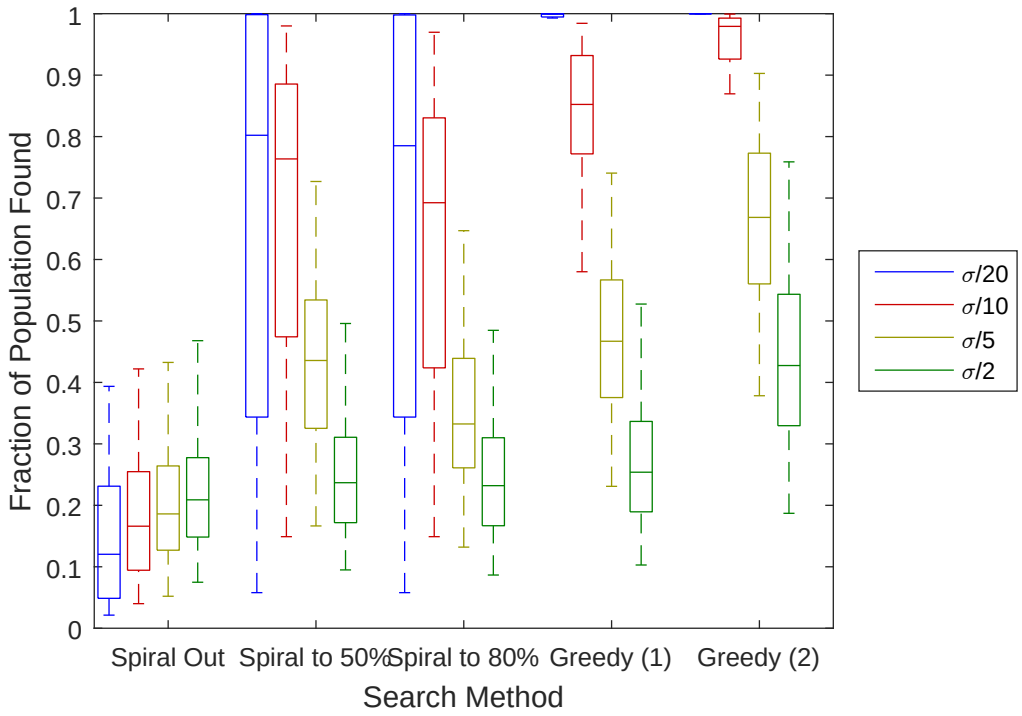

Figure 4.26: Comparison of success rates for paths with varying standard deviations of the distribution.

pattern. One step cannot pass the space one cell away from the wall that tends to have a lower population after the robot has passed between it and the wall. A two-step algorithm can pass that lower value area when the cells beyond it offer more value. Thus a two-step greedy algorithm tends to outperform the pre-planned paths. By extension, a three-step greedy algorithm ought to improve upon the two-step and a four-step improve upon the three-step. In the limit, looking forward for N steps for a simulation of N steps would obviously find an optimal path since it is equivalent to testing all possible combinations, but this comes at a cost in computation time. In Figure 4.27 the computation run time for simulations of different lengths is compared for the one- and two-step greedy methods. The two-step method takes longer than the one-step method because each iteration requires six best move searches instead of one. Using more steps would take still longer with the number of calculations being $\mathcal{O}(5^n)$ in the number of steps projected. As the number of time steps expands, it becomes intractable.

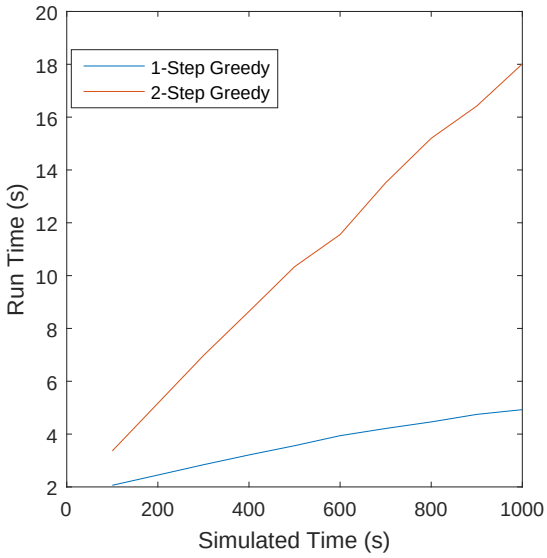

Figure 4.27: Simulation run time for one- and two-step greedy algorithms with different simulation times.

We have shown that using sensors to make path-planning decisions leads to the best paths for the normal distribution and, when it can sense at least two cells away, the best paths for the uniform distribution with a sticking coefficient. We have also shown that selecting a best path for the normally distributed population is independent of the variation in the path and swarm parameters that we tested but that the sticky wall environment shows some dependence on the parameterization. Though a two-step greedy algorithm is always best, in the case that the sensor is not available or is prohibitively expensive, less expensive sensors may provide satisfactory performance with a wall-following pattern in the right circumstances.

4.5 Immobile Collection

For a baseline comparison, we also tested an immobile collection agent. Since a stationary collector is likely to have a constant power source, it would not be subject to the energy budget constraints that a mobile robot has. This makes comparisons somewhat difficult because the time for the simulation should not be limited to the

battery life of the mobile robot and should instead be related to the number of times the mobile robot can perform a search in a day. For our simulations, we assumed a search frequency of four times per day for the mobile robot. This gives a comparable collection period of six hours for $T = 21600\ s$. In this time, our simulations showed that a collector placed at the center of the normal distribution with a removal rate of 90% could remove 99.9% of the particles. We tested several locations for the uniform distribution with sticky walls and found the best location for the collector is in the center of the workspace. Even at its best, though, the collector could only remove 9.7% of the particles. These results indicate that an immobile collection agent with a steady power source may be quite effective when the particles are attracted toward it or its location but does not perform as well as a mobile agent with a more widely distributed population.

Chapter 5

Conclusion

5.1 Conclusions

We investigated a new problem in robotic coverage that attempts to maximize the number of moving particles detected when they are sampled without replacement. We modeled mosquitoes as moving particles against a map background image and used a robot to hunt for them with four different path simulations. This provided a benchmark simulation and showed that a simple boustrophedon coverage path provided the best results when the robot has a long working time. When the allotted time is insufficient to provide full geographic coverage of the workspace, other techniques that use feedback to dwell in areas with higher concentrations of the target population can outperform the boustrophedon policy.

We extended the individual movement model into a more general model using a Markov process with transition matrices built by both Monte Carlo simulation and calculations from the environment. We built transition matrices for a defined probability of leaving a cell and devised a method for building these matrices from the desired stationary distribution of the population. Each of these had a single tuning parameter to vary the rate at which the population moved. This was applied to two environments: a uniform distribution within walls and an unbounded area with a normal distribution. The walled environment could have walls that were more attractive to the particles than the interior of the workspace.

We tested several search methods, including pre-planned paths that the robot followed without deviation and feedback-driven paths that allowed the robot to take the

most opportune of the available directions as its next step. Parametric testing varied the time allowed for the robot to run and the speed at which the robot hunted as well as the efficacy of the robot's ability to collect the swarm particles. It also tested the spacing between rows in the pre-planned paths and the swarm's probability of transitioning to another cell or sticking to the walls. In the normal distribution environment, it varied the standard deviation of the distribution.

The results from these tests allow us to find relationships among the performance metrics of the paths. We find that the opportunistic paths are best in the normally distributed population whether they can look forward for one step or two steps. We also find that when working with a pre-planned square spiral path, the robot is much more effective when it has the ability to sense the proportions of the population that are inside or outside the radius of its current turn. As we vary the parameters of the testing, we find that nothing alters the relative effectiveness of these paths and that the effects of the parameters are largely predictable, *e.g.* faster robots collect more particles.

In the sticky wall environment, the relative performance of the paths varies with several parameters. While the two-step greedy path is best for any given combination of parameters, circumstances that lead to a larger fraction of the population around the walls can give an advantage to the wall-following paths over the paths that spend more of their time exploring the interior of the workspace. We also find that pre-planned paths often improve upon the results of an opportunistic path that can only sense the reward for one step at a time, showing that inadequate particle sensors may be worse than no sensors at all.

5.2 Extensions

Exploration of this topic is just beginning; there are a number of avenues in very different directions open to further investigation in future work. The model may be

expanded to three dimensions or may be modified to include obstacles. Additional coverage paths may be compared to the existing algorithms or there may be variation in the starting points of the paths as they repeat. Environments containing a combination of uniformly and normally distributed areas or a group of normal distributions with centers in different areas of the workspace can be explored. When using a greedy algorithm, the number of steps looked forward could be increased to determine if there is a point beyond which further computation has little or no return. A particle motion model that reacts to the robot's motion could be developed. There is no shortage of possibilities for extending this research.

References

[1] Iolaire. (2005) Fishing trawler. [Online]. Available: https://commons.wikimedia.org/wiki/File:Fishing_Trawler.jpg

[2] Google. (2016, Mar.) Google maps. [Online]. Available: http://maps.google.com

[3] O. Sheynin, "Sampling without replacement: history and applications," *NTM International Journal of History & Ethics of Natural Sciences, Technology & Medicine*, vol. 10, no. 1-3, pp. 181–187, 2002.

[4] H. Choset, "Coverage for robotics - a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1-4, pp. 113–126, October 2001.

[5] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[6] *Merriam-Webster's Collegiate Dictionary*, 10th ed. Merriam-Webster, Incorporated, 1999.

[7] H. Choset, "Coverage of known spaces: The boustrophedon cellular decomposition," *Autonomous Robots*, vol. 9, no. 3, pp. 247–253, 2000.

[8] S. Hert, S. Tiwari, and V. Lumelsky, "A terrain-covering algorithm for an auv," in *Underwater Robots*. Springer, 1996, pp. 17–45.

[9] R. N. De Carvalho, H. Vidal, P. Vieira, and M. Ribeiro, "Complete coverage path planning and guidance for cleaning robots," in *Industrial Electronics, 1997. ISIE'97., Proceedings of the IEEE International Symposium on*, vol. 2. IEEE, 1997, pp. 677–682.

[10] S. S. Ge and C.-H. Fua, "Complete multi-robot coverage of unknown environments with minimum repeated coverage," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 715–720.

[11] 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.

[12] 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.

[13] C. Das, A. Becker, and T. Bretl, "Probably approximately correct coverage for robots with uncertainty," in *Intelligent Robots and Systems, 2011. IROS 2011. IEEE/RSJ International Conference on*, vol. 1, 2011, pp. 12–13.

[14] D. W. Gage, "Randomized search strategies with imperfect sensors," in *Optical Tools for Manufacturing and Advanced Automation*. International Society for Optics and Photonics, 1994, pp. 270–279.

[15] 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.

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

[17] P.-M. Hsu, C.-L. Lin, and M.-Y. Yang, "On the complete coverage path planning for mobile robots," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 3-4, p. 945, 2014.

[18] S. K. Sudarshan and A. T. Becker, "Using gradient descent to optimize paths for sustaining wireless sensor networks," in *Wireless and Microwave Circuits and Systems (WMCS), 2015 Texas Symposium on.* IEEE, 2015, pp. 1–6.

[19] D. E. Soltero, M. Schwager, and D. Rus, "Decentralized path planning for coverage tasks using gradient descent adaptive control," *The International Journal of Robotics Research*, vol. 33, no. 3, pp. 401–425, 2014.

[20] J. Nguyen, N. Lawrance, R. Fitch, and S. Sukkarieh, "Energy-constrained motion planning for information gathering with autonomous aerial soaring," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on.* IEEE, 2013, pp. 3825–3831.

[21] A. Clements, "The biology of mosquitoes: sensory, reception and behaviour, vol. 2," *CAB International Publication, London*, 1999.

[22] W. Bidlingmayer, "Mosquito flight paths in relation to the environment. 1. illumination levels, orientation, and resting areas," *Annals of the Entomological Society of America*, vol. 64, no. 5, pp. 1121–1131, 1971.

[23] M. Burbage, S. Manzoor, and A. T. Becker. (2016, Sep.) Mosquito Drone Coverage and Elimination, MATLAB Central File Exchange. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/59142

[24] C. M. Grinstead and J. L. Snell, *Introduction to Probability.* American Mathematical Soc., 2012.

[25] T. Bonald and M. Feuillet, *Network performance analysis.* John Wiley & Sons, 2013.

[26] M. K. Arthur, "Point picking and distributing on the disc and sphere," DTIC Document, Tech. Rep., 2015.