Gradient-based Assisted Navigation and Control for Autonomous UAV Swarms

Bryan Kate
bkate@eecs.harvard.edu
Harvard University

Peter Bailis
pbailis@eecs.harvard.edu
Harvard University

Abstract

We present a system and corresponding algorithms for directing the flight of autonomous aerial vehicle swarms in order to perform long-distance navigation and control. Our solution relies on a static deployment of embedded devices in the environment, each capable of radio communication with the mobile vehicle within a local radius. As the vehicle moves through the environment, it detects the presence of one or more artificial fields projected by the nodes from which it computes a gradient value. The vehicle ascends or descends this gradient based on simple local rules, using radio signal strength as a rudimentary metric for progress. Our solution does not require localization or sophisticated sensors; it requires only radio capabilities. We characterize the feasibility of our system in both real-world tests and simulation. We show that in simulation the paths taken by vehicles are roughly twice the length of the optimal path when using randomized movement algorithms.

1 Introduction

As the sizes of unmanned aerial vehicles continue to shrink, there are increasingly many difficult design tradeoffs. Energy becomes a primary constraint, limiting the capabilities of the robotics. Micro-scale UAVs will have limited sensors and computational ability, and balancing the tradeoffs between functionality and vehicle scale is a non-trivial task. The Harvard RoboBees project [13] is building a swarm of micro-insect UAVs and is faced with many of these problems.

In this paper, we propose a solution for UAV swarm control that requires only local radio communication by leveraging a system of deployed ground nodes. The nodes broadcast artificial spatial field values via radio messages. Coupling these values with the received signal strength of the message allows swarm agents to perform coarse-grained bearing control without relying on any sort of localization. Multiple such fields can be combined in order to build more complex behaviors, and we can construct complicated global behaviors while limiting the computational and communication costs at a local agent-based scale.

We present and evaluate a series of algorithms that allow us to use RF gradients in practice. We present an algorithm that allows an agent to determine whether it is making progress while moving through a field; this informs another set of algorithms that dictate how the agent should actually move throughout the environment. We examine the feasibility of using signal strength as a coarse-grained estimator of progress using commodity radios in a representative configuration in an outdoor environment and conclude that it is adequate given appropriate smoothing techniques. We implement our algorithms in simulation and obtain an average path efficiency of approximately two times that of the optimal distance using only RF bearing estimation. Finally, although our future hardware platform for UAV exploration is currently under development, we perform limited field experiments with human subjects that show promise for more algorithmic control strategies in practice. We conclude that properly constructed RF gradients are both feasible in practice and a convenient abstraction for controlling UAV swarms with limited sensing capabilities.

The structure of this paper is as follows: In Section 2, we motivate the need for RF bearing and explain the RoboBee project's goals. In Section 3, we describe the system of embedded nodes, propose a novel algorithm for determining progress within a gradient, and discuss possible movement algorithms. In Section 4, we evaluate the feasibility of using signal strength indicators for ranging in a real environment, we evaluate our gradient algorithms in simulation, and present limited evidence from human-based trials. In Section 5, we discuss possibilities for future research in this area. In Section 6, we discuss related work and how our proposed system differs from prior gradient-based approaches. We conclude in Section 7.

2 Motivation

The RoboBees project is building a swarm of biologically inspired micro-robotic aerial vehicles. Over the next four years, the team will design and build an aerial robot system at a previously unexplored micro scale. There are myriad challenges involved at each level, from the mechanics and electronics to coordination of behavior. Enabling these robots to cooperate on several complex tasks such as pollination, search and rescue, and environmental mapping will be difficult.

Undoubtedly, the final RoboBees platform will be limited. It should fly and have moderate control capabilities, but will likely have limited sensors and poor reliability. In addition, the robots will not likely know their location within an environment. They will be equipped with the fewest number of sensors and computation required in order to achieve their goals. These constraints are not artificial; at a micro-scale, increased sensor capability or computation power translate to shorter flight times due to increased weight. In order to control and coordinate these agents at scale, we face many of the problems inherent in traditional distributed systems, but we expect the magnitude of problems to be much greater: partial knowledge, agent failure, and limited connectivity will be the norm, and we must design for this worst case.

In order to perform meaningful tasks with these robots, we require a new abstraction. We need to influence agent movement while allowing agent behaviors to be expressed and computed simply. We cannot depend on localization or complex sensors. Ideally, we will be able to reason about the global behavior of the system instead of programming agents individually. By macroprogramming the RoboBees and using global-to-local control techniques, we can scale the number of agents without incurring significant additional overhead. Perhaps most importantly, we need an abstraction layer that will work in practice and not simply in simulation. We are building a real-world system, and need to account for real-world constraints like energy, inaccuracy, and failure while providing a usable abstraction for algorithm designers and system users.

3 Design

The scale of a RoboBees swarm, predicted to be on the order of magnitude of one thousand autonomous vehicles, prevents us from implementing a system that requires even modest amounts of direct communication between devices. Instead, we choose to focus on decentralized systems that rely on shared infrastructure and indirect communication. Subsequently, we allow individual vehicles to make local decisions and carry out tasks without the limitation and responsibility of maintaining connections to other bees in the swarm.

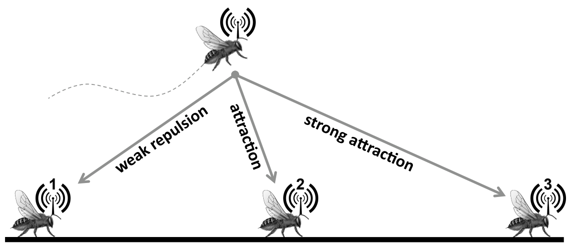

Figure 1: A RoboBee vehicle traversing unknown terrain by ascending a virtual gradient being broadcast by devices embedded in the environment.

Our motivating problem is one of environment exploration and resource exploitation. We envision a scenario in which the bees are deployed into an unfamiliar area and are tasked with finding resources in the environment (e.g. flowering plants) and directing the swarm to exploit that resource accordingly (e.g. cross-pollinating patches of flowers). Since the environment is unfamiliar, using a map that is pre-computed offline is not possible. Furthermore, it is difficult to collaboratively construct a map when the robotic platform lacks reliable sensing, odometry, and localization capabilities. However, we want to exploit individual knowledge in a way that is meaningful to the swarm as a whole. We propose a solution in which online decisions about long-distance travel are influenced by information collected from the environment and controlled by high level agent behaviors that are triggered in the presence of this information. We believe that controlling the cues that are available to agents and the corresponding reactive tasks is a scalable, global-to-local solution to defining swarm behavior.

We propose a system that uses devices embedded in the environment to provide the aforementioned cues via radio signals. In addition, the embedded devices can act as a repository for information, allowing passing vehicles to deposit data that may be relevant to others, such as the amount of energy it takes to reach a destination or the presence of a valuable resource. By embedding these devices, we establish a shared infrastructure that obviates the need for extensive inter-vehicle communication. Through this infrastructure we can build primitives like routing, virtual fencing, and signposts that can be used to define agent behavior. For the scope of this project we assume that the embedded devices are statically deployed, though one could envision a system in which the swarm dynamically deployed the infrastructure as it explored a space, with some bees landing to fill that role.

Additionally, we assume that the embedded devices are not networked together, though that capability could enable a host of useful system properties stemming from the ability to disseminate information.

Our solution relies on radio communication between the flying vehicles and the embedded ground nodes. More specifically, we rely on the ability to detect if a vehicle is moving closer to or further from a radio source. While radio signal strength is theoretically predictable as a function of distance, it is well known that such measurements are not constant in reality and can vary by device and environmental conditions [14]. It is important to emphasize that we are not proposing a precise method for navigation in a small space. Rather, we are aiming to quickly guide bees to a remote destination (such as across a large field) where other mechanisms can take over to perform detailed tasks. As such, we believe we can provide a solution in spite of the inherent variability of radio communications. At this point in time we do not attempt to provide an optimal gradient-following algorithm (in terms of distance or energy) for the vehicles and instead focus on finding a reasonable algorithm as a proof-of-concept, though this is a topic of future work.

3.1 Gradient Abstraction

We propose a gradient abstraction to leverage this embedded mesh of ground nodes. Each node broadcasts at least one artificial spatial field "value" that is assigned at a global system level. Computing the gradient values from the artificial fields and combining them with signal strength measurements gives the observer a bearing within the gradient without reference to actual node topology. An observer can tell its relative distance from different points in the field, localizing it within the global field space, but not within absolute space. However, we do not require absolute localization; as long as the observer can detect progress within a field (whether following increasing gradient values, decreasing gradient values, or a level set within the gradient), then it can traverse it. The gradient abstraction allows a local interpretation of global goals and desired behavior.

To illustrate the gradient abstraction, we provide three sample scenarios:

- **Movement to and from the hive.** In the first panel of Figure 2, we show a gradient that allows an agent to move towards or away from the hive. At the hive, the gradient value is zero, and the gradient value increases proportionally as one moves away from the hive. An agent can move away from the hive by ensuring it moves towards higher gradient values, or can return by moving towards the hive until it reaches the gradient value of zero.
- **Movement along a path.** In the second panel of Figure 2, we show a gradient that allows an agent to move along a path. Along the path, the gradient value is nonzero, and off of the path, the gradient value is zero. By adding a monotonically increasing value to each node along the path, we can construct a gradient that allows the agent to perform simple path-following.
- **Virtual Fencing.** In the third panel of Figure 2, we show a gradient that warns the agent of areas where it should not go. There are two strongly negative areas of the gradient where the agent should not fly (perhaps obstacles exist or the area is previously explored). The negative values form a "virtual fence" for the swarm.

We can also consider combining several of our gradients based upon the agent behavior. Consider an scenario in which we want the agent to travel away from the hive along a path but avoid certain obstacles. We could multiply the values from the fields shown in the first two panels of Figure 2 to get a monotonically increasing path away from the hive. We could then add or multiply the resulting field with the values of the field in the third panel. By programming the agent to move towards increasing gradient values, the agent could complete a task that requires it to move away from the hive along a path while avoiding fenced regions.

Similarly, agents should be able to modify parts of the field as they encounter it. In a system designed for environmental mapping, once an agent has visited a node, the node broadcasts that it has already been visited by expressing a negative field value. In another system designed for search and rescue, agents could stigmergically deposit information that other agents could use as a trail.

3.2 Gradient Progress Evaluation

Agents need a means of evaluating their progress within an artificial field. As they fly, they will receive signals from the ground nodes which will provide them feedback about their latest movements. Because we do not have means for localization, we cannot determine a movement vector for the agent even with the help from the ground nodes. Similarly, because we do not give strict requirements for the ground node topology, ranging and triangulation will not be helpful. Instead, we want a localization-free algorithm that will give us our relative (not absolute) progress within the gradient.

Given a radio message from a ground node, we have several pieces of information that we can use. First, we have the message itself, which will contain the field value and associated metadata. For example, we will be able to determine which node sent the message by attaching

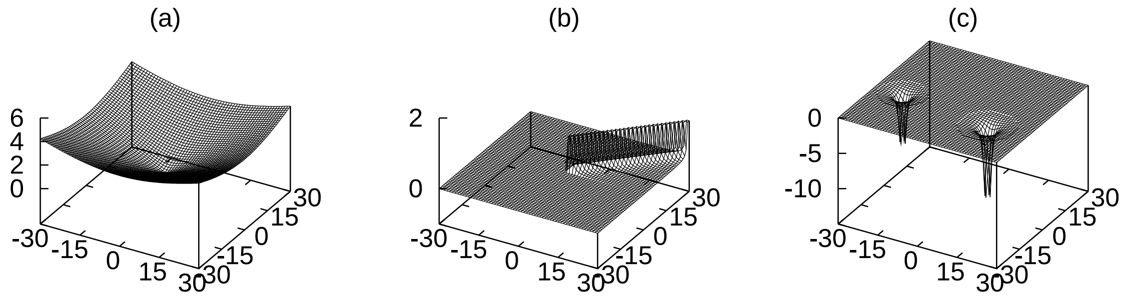

Figure 2: Example fields that can may combined and followed by aerial vehicles.

a sender ID. We can also determine the received signal strength (RSS) of the message, which is known to degrade inversely with the distance from the sending radio. This can provide some information about the direction of travel over time relative to the transmitter. Finally, we also have information about which nodes we have recently heard from. We can compare the current set of broadcasts to the previous set, which may give us an indication of progress.

We measure relative progress instead of absolute progress. Our progress evaluation is expressed as a difference from the previous position instead of an absolute metric of how far along we are in the gradient. Although there may be absolute characteristics inherent in the field (the distance from the hive, for example), due to the complexity of RSS measurements we have decided it is better to tell the agent that they made a "good step" rather than provide an absolute measure (ex. a percentage of the goal) as a means of evaluating progress.

We model progress through the field as a response to attractive and repulsive forces on the agent. We define the agent's current field level at any point in time as the value of the node with the highest RSS. If ascending the gradient, the agent wants to move away from nodes with lower field values – the repulsive force – and move towards nodes with higher field values – the attractive force. As such, it is important to ensure that the ground nodes are deployed with overlapping radio ranges so that there is more than one force guiding the vehicle.

The rate at which an agent evaluates its progress is implementation and deployment-specific and depends on several factors. An agent needs a representative sample of messages from broadcasting nodes in order to make an informed decision. If an agent is moving very quickly, it may need updates much quicker than a slower moving agent. With more signals to analyze, an agent can perform better noise reduction and make more accurate decisions. The rate at which the ground nodes broadcast, however, is not necessarily fixed; there is no reason why an agent could not broadcast itself in order to request messages from the nodes around it. We do not explore these ideas here, but the broadcast rate, communication protocol, and update step size are important consideration for system builders.

For a given evaluation period e, assume an agent receives a set of signals $S_e = s_i, ..., s_n$ such that s_i is from node i with field value $f(i)$. We define RSS of message s_i as $r(s_i)$. This RSS can be an average of many values or may be a smoothed value; we propose a sliding window of values as a first approach. The agent considers node c, the node with the largest RSS value in S its current node and considers $f(c)$ as its current field value. The agent then compares its current state to its previous state, S_{e-1}, by taking the relative change $\Delta(n)$ in RSS between S_e and S_{e-1} for each node n ($\Delta(n) = r(s_i) - r(s_i^*)$ such that $r(s_i^*)$ is the message from $i \in S_{e-1}$). If there is no message from i in S_{e-1}, then we assign $\Delta(n) = k$ where k is a small constant. If there are multiple nodes with a single gradient value, we only consider the node with the highest RSS from the set.

In order to evaluate our progress, we weight the differences in RSS by the field-distance of the node from the current node c. Our progress function $P(S_e, S_{e-1})$ is defined as $\sum_{s_i \in S_e} p(s_i)$ such that:

$$p(s_i) = \begin{cases} \frac{\beta}{(i-c)^3} * \Delta(i) & : i < c \\ \gamma * \Delta(i) & : i = c \\ \frac{\alpha}{(i-c)^3} * \Delta(i) & : i > c \end{cases}$$

If the progress function is positive, then the agent made progress within the field. Otherwise, the agent did not make progress.

For an example, consider an agent that previously heard from nodes 1, 2, and 3 with signal strengths -30, -25, and -40 respectively. For simplicity, assume $\alpha = \beta = \gamma = 1$. Now, in the next time period, it hears from the same nodes, but the RSS values are now -40, -10, and -30. $\Delta(1) = -40 - (-30) = -10, \Delta(2) = -10 - (-25) = 15$ and $\Delta(3) = -30 - (-40) = 10$. The current node is node 2, so $p(1) = \frac{\beta}{(1-2)} * -10 = 10, p(2) = \gamma * 15 = 15$, and $p(3) = \frac{\alpha}{3-2} * 10 = 10$.

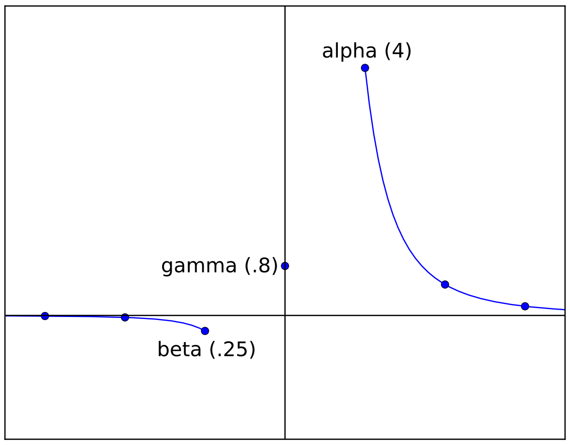

Figure 3: The piecewise function used to compute $p(s_i)$. The X axis represents the distance in the artificial field from that of the closest node ($i - c$), and the corresponding point on the Y axis is multiplied by $\Delta(i)$. The function is designed to provide attractive and repulsive forces to guide the motion of the vehicle as it encounters embedded devices.

Therefore, our progress function evaluates to 45, and the agent has made progress in moving towards values higher in the field by ascending the gradient.

Intuitively, α controls the agent's attraction towards higher field values, β controls the agent's repulsion away from lower field values, and γ controls the agent's attraction to its current node. Note that without α or β, the agent would simply hone in on a node and stop moving. In practice, β is small, α is large, and γ is in-between. This asymmetry, depicted in Figure 3, is intended to bias the agent in one direction while traveling through the field.

This evaluation function is a first step towards an appropriate gradient progress evaluation function. We focused on finding a function that would allow us to take into account all nodes within radio range instead of simply performing iterative homing. Ideally, future functions would better integrate noise smoothing by the use of Kalman filters or other filtering techniques. In practice, however, our algorithm works reasonably well (see Section 4).

3.3 Movement Algorithms

Using the metric defined above, it is possible for an agent to determine how it has moved, relative to an arbitrary spatial field, over some time interval. Given this information, what action should the agent take to reach its goal? To begin, we consider stateless approaches - those that do not rely on tracking prior actions to make a decision. Our agents evaluate their progress periodically, so a decision that can influence the direction of travel is made at a regular interval and the results of that action remain until the next update. In addition, we assume that the agents are capable of level flight, effectively reducing the problem to two dimensions. Finally, we restrict the agent to movement in the forward direction at a constant velocity and turns about a single axis (left and right).

The first movement algorithm we considered is essentially gradient descent. At each update the agent can take one of two actions:

1. Continue moving in the current direction.

2. Make a random turn prior to moving forward.

The first action is taken if progress has been made since the last time step ($P > 0$), and the second is taken if no progress is made ($P \leq 0$).

The second movement algorithm we tried, chemotaxis, is inspired by the movement of bacteria toward chemical food sources [7]. In the wild, bacteria are capable of detecting chemical gradients but have no means of determining a direction to the food source. As such, the bacteria implement a biased random walk strategy to locate the source of the chemical. Agents implementing the chemotaxis algorithm can perform the same set of actions listed above, but are governed by a different set of rules. Specifically, the agent can choose the first action (move forward for a complete time step) only if progress was made ($P > 0$) after tumbling (took the second action in the prior step). If progress was made but the agent had not previously tumbled (took the first action in the prior step) and the prior step was a full step then it executes the first action for a partial time step ($t_{+1} = t + stepsize * bias : 0 < bias < 1$). This is referred to as a *biased* step because it allows an agent to take advantage of the progress and continue moving in the same direction as opposed to tumbling at each time step (as would be the case in a pure random walk). Finally, if progress was not made ($P \leq 0$) or if the agent took a biased step during the previous period, then the second action is taken. This strategy forces the agent to make random turns more often (every other time step if progress is being made) but is less prone to getting stuck in local minima.

Figures 4 and 5 show simulated traces of agents using the gradient descent and chemotaxis (20% bias) algorithms, respectively. A detailed description of the scenario is given in Section 4.2. From these traces we can see that the resulting aggregate motion of the gradient descent algorithm is dominated by long, straight segments separated by abrupt changes in direction, whereas the chemotaxis algorithm has shorter, curved sections and more gradual turns - a direct result of frequent tumbling.

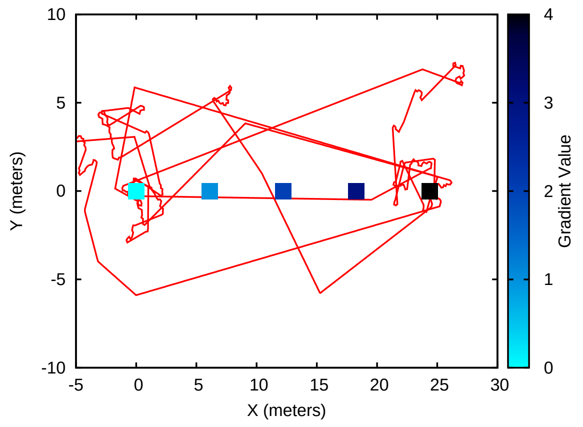

Figure 4: Simulation of an agent moving according to the gradient descent algorithm. Embedded nodes are represented by squares and the agent's path is denoted by the solid line.

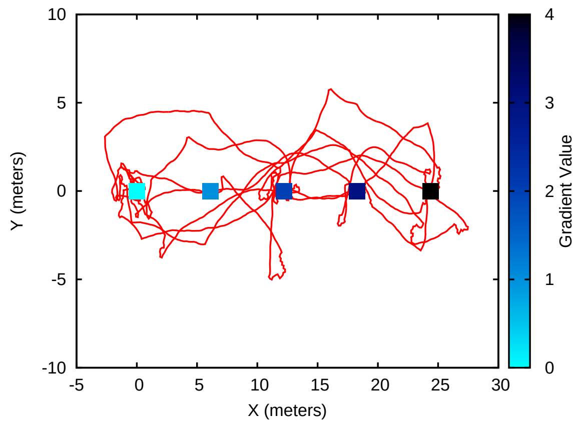

Figure 5: Simulation of an agent moving according to the chemotaxis algorithm (20% bias). Embedded nodes are represented by squares and the agent's path is denoted by the solid line.

In both movement algorithms we restrict the angle for a random tumble so that the agent does not completely reverse direction when it stops making progress. This limits the amount of possible negative backtracking caused by tumbles in the chemotaxis algorithm and generally encourages forward progress, especially as it is unlikely that an agent needs to turn around completely. Likewise, the agent is slower to react when a drastic course correction is required. It is empirically evident that the former point generally dominates the latter.

Both of these algorithms are relatively simple and are stateless. Using these algorithms we can arrive at our destination but not in the most efficient manner. We are considering several improvements as future work, including making more informed decisions by tracking past actions and reasoning about the overall direction traveled before turning. Using this information we hope to eliminate the random element and choose more efficient paths.

4 Evaluation

We evaluated the feasibility of using radio signals and artificial gradients through real-world and simulated experimentation. We first explored the possibility of using commodity hardware in an outdoor environment by running a characterization experiment. We then implemented our algorithms in simulation and examined the effectiveness across different parameters and behaviors. Although our ultimate goal is to use a helicopter testbed to approximate our RoboBees, because the hardware is not yet operational we could not rigorously validate our simulation results. We did, however, perform limited human trials that give anecdotal evidence that a RF-based gradient navigation system can work in practice.

4.1 Characterization Experiments

We explored the correlation between received signal strength and distance in practice. Theoretically, the signal strength for a line-of-sight transmission should drop off with the square of the distance r from the radio ($\frac{1}{r^2}$). However, the signal attenuation is much more severe in reality due to equipment imperfections, deployment details, and environmental factors. Although this relationship has been studied extensively in the literature [14], we wanted to examine its accuracy in our operating environment with a reasonable deployment of commodity radios. To this end, we devised a characterization experiment to collect data on the received signal strength indicator (RSSI) - a measure of signal strength provided by the radio chip with each signal reception.

Figure 6 provides an aerial view of our experimental setup. We chose a space that is largely open (with the exception of one tree) and established a grid measuring 80 feet in width and 70 feet in length, marking measurement locations every 10 feet. We then deployed six TelosB motes (battery-powered embedded devices used in sensor networks containing 802.15.4 wireless radios and built-in omnidirectional antennas) into the space in a cross formation, co-locating two of the devices at the center. We elevated the motes 18 inches off the ground on cardboard boxes and oriented them to face in the same direction (along the Y axis of our grid). Each mote ran a program that broadcast a message at -25 dBm every 500 ms (with random jitter of 100 ms to reduce congestion) that contained the mote ID and a sequence number. A mote connected to a laptop was used as a receiver and collected data at each point in our grid (also on a card-

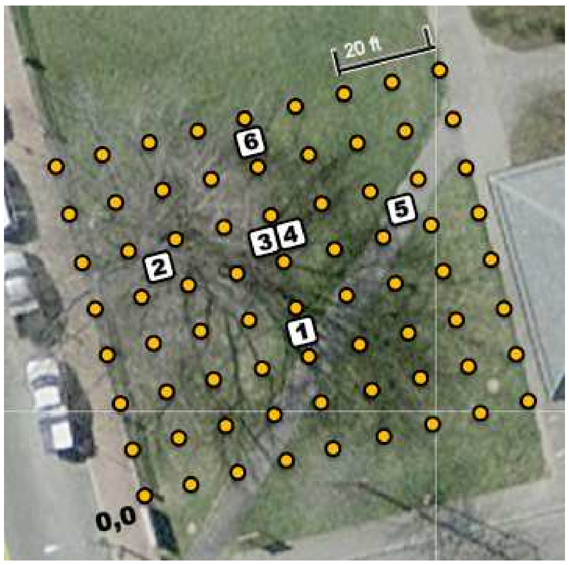

Figure 6: A satellite image of the experiment landscape. Broadcasting motes are denoted by numbered squares and measurement points (where the receiver is placed) are denoted as orange circles. The origin of the measurement grid is in the lower left corner.

board box 18 inches from the ground). At each point the receiving mote captured and recorded all messages and corresponding RSSI values for 30 seconds.

With this setup we were able to compute statistics related to the received signal strength of messages sent from each mote. Figure 7 shows a heatmap of the mean RSSI of messages from the mote with ID 6. As expected, there is a clear trend indicating that the signal is stronger as the distance between the transmitter and receiver shrinks. However, there is a great deal of variation in the readings, causing false peaks and valleys. The significant result from this experiment is that RSSI may be used to indicate progression toward or away from a transmission source in practice, provided that some weight is given to the longer trend as opposed to changes over shorter time scales. Additionally, this experiment highlights the importance of smoothing data and algorithmic robustness to false peaks in the gradient.

4.2 Simulation

Though the ultimate goal of this project is to implement our system on the helicopter testbed, we created a simulation environment in which our ideas could be rapidly prototyped. Our simulator not only allowed us to quickly realize our ideas, it provided a means for exploring the parameter space of our algorithms in a repeatable way. Since the helicopter platform is a work in progress, most of the core results in this paper are validated through simulation.

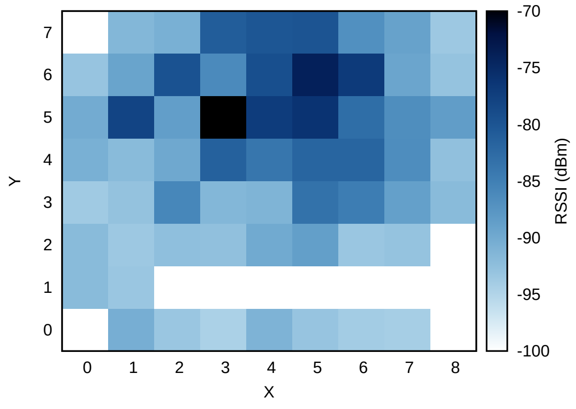

Figure 7: A heatmap showing the average RSSI of packets transmitted by the mote with ID 6 (located at (4,6.5) in the grid) and received at each point in the collection grid. Note that data is missing for the cells (2-8,1).

Our simulator, written in Java, provides a 3D continuous virtual world. The position, orientation, physical extent, and kinematics of objects in our simulation are handled by an embedded 3rd-party rigid body physics engine, JBullet [8]. We build upon this base by providing a model for radio communication, virtualized sensing equipment, an abstraction for flight control, and an extensible interface for user-supplied agent behavior. Using the modeling framework provided, we constructed two bee models - one that is stationary and acts as a beacon, and one that collects beacon information (virtual field values) and uses it to navigate in the virtual world.

An important part of creating a simulation environment is anchoring it to reality. Results obtained through simulation will always be idealized and limited to the fidelity of the models. However, if the fidelity is increased, users can be more confident in the validity of their algorithms and accuracy of their performance. As such, we focused on modeling RF communications with improved fidelity since RSSI is such an integral part of our solution. We modeled the radio from the TelosB motes, the Chipcon CC2420, using the transceiver properties from the data sheet [4]. We assume that the built in antenna provides 0 dBi gain and is omnidirectional (modeled as an isotropic antenna). We created a functional representation of the radio's receiver by mapping the measured signal to noise ratio (SNR) to a packet reception rate (PRR). This mapping function was generated from empirical evidence [9] and is now incorporated into the TinyOS simulator, TOSSIM [10].

Since the largest variation in RSSI comes from path loss, we focused our attention on correctly modeling signal attenuation in open, outdoor environments. There are many models for path loss based on theoretical physics

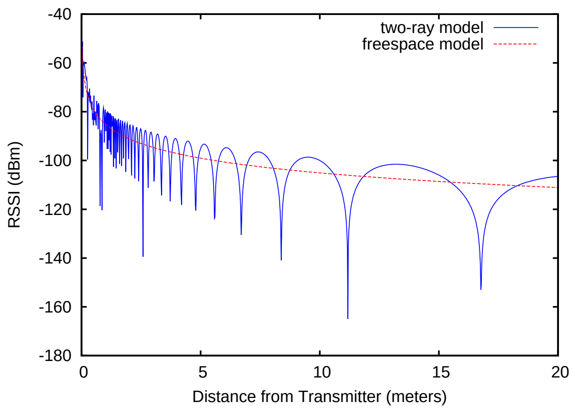

Figure 8: Theoretical path loss models for a radio transmission at -25 dBm in the 802.15.4 (2.4 GHz) spectrum. The transmitter and receiver are each elevated 18 inches above the ground plane.

Param	**Value**	**Description**
α	4.0	See Section 3.2
β	0.25	See Section 3.2
γ	0.8	See Section 3.2
velocity	0.5 ($\frac{m}{s}$)	Forward vel. of bees
altitude	0.3 (m)	Altitude of bees
run length	0.25 (s)	Movement time step
tumble bias	$\pm\frac{\pi}{16}$ (rad)	Max. tumble angle
history	0.5 (s)	Past signal queue len.
window	4	Sliding window size
beacon power	-7 (dBm)	Ground node Tx power
beacon rate	20 (Hz)	Beacon broadcast rate
noise floor	-100 (dBm)	Mean RF noise level
noise σ	10 (dBm)	RF noise variance
total time	600 (s)	Scenario length

Table 1: Non-varying simulation parameters used in all experiments.

and empirical data, but we chose two that best represented our target environment. The first is the free space model, a simple, monotonically decreasing approximation based on the inverse square law. The second is the two-ray (plane earth) model, in which signal reception is modeled with two rays - a direct line-of-sight connection and a reflection off the surface of the earth [12]. Figure 8 shows the behavior of the two models for a transmission at -25 dBm. Notice that the two-ray model accounts for constructive and destructive interference (caused by the phase shift of the reflected ray) and exhibits a fluctuating pattern. This behavior is evidenced in the real world, so it was important to capture in the simulator so that appropriate smoothing mechanisms could be developed to compensate for the variation. In addition to path loss, the communications model must also account for background noise. We are currently modeling noise using a Gaussian distribution around a mean noise floor that is independently distributed between measurements. We are not modeling channel contention or the effects of simultaneous transmission. As an extra measure, we would like to use the data collected in our characterization experiments to improve the fidelity of the propagation and noise models, but this is left as future work.

We performed a number of simulated experiments in which our bees navigated through the the virtual environment. We propose two deployments of embedded nodes, a straight line and a series of concentric circles. In the straight line scenario, nodes are placed every 20 feet along the positive X axis (five in total) and are broadcasting artificial field values that are monotonically increasing. The agent's task is to follow the gradient from its starting position at the origin to the end of the line, reverse direction, and repeat. In the concentric circle scenario, rings of nodes are placed at 20 foot intervals from the origin. Each ring is approximated by a regular decagon, with every node in the ring broadcasting identical artificial field values. The field values increase monotonically from the origin with each level of rings. In total, 41 nodes are needed to complete this deployment. In both scenarios the distance between nodes was selected such that the radio signals overlap, enabling our progress function to be evaluated effectively.

There are a number of parameters that are fixed for each experiment. The free space propagation model, as opposed to the two-ray model, was used in all runs. Our trivial smoothing and thresholding approaches could not adequately deal with the peaks and nulls of the fluctuating RSSI values while using the tw-ray model. A more robust approach is anticipated as future work. Table 1 lists the most important variable values used in the simulated experiments. Some values, like α, β, γ, and the tumble bias were obtained with a coarse, gridded search of the parameter space. Note that many of the values are codependent (such as velocity, time step, beacon rate, history length, and window size) and must be adjusted together.

We began experimenting with the straight line scenario. The purpose of these experiments is twofold; first, we verify that the progress evaluation function can be used to ascend and descend an artificial gradient and second, we evaluate the effectiveness of the two movement algorithms. Figures 4 and 5 depict two runs of this scenario in which the bees move according to the gradient descent and chemotaxis algorithms, respectively. A cursory evaluation indicates that that the progress evaluation mechanism and simple movement algorithms are sufficient for accomplishing the task - the bee is able to ascend and descend the gradient several times before the

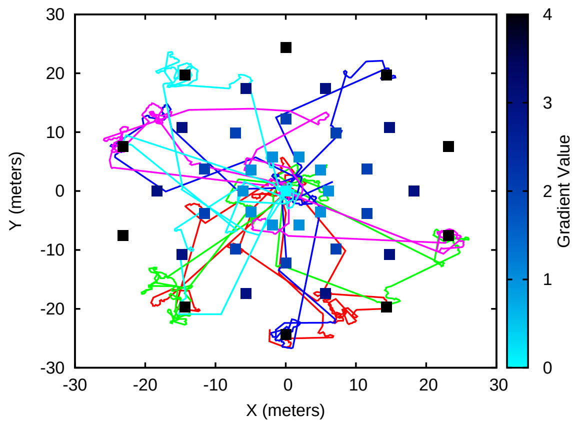

Figure 9: Simulated agents (five in total) navigating through a landscape using the gradient descent algorithm. The embedded devices produce an artificial field that is monotonically increasing outward from the origin.

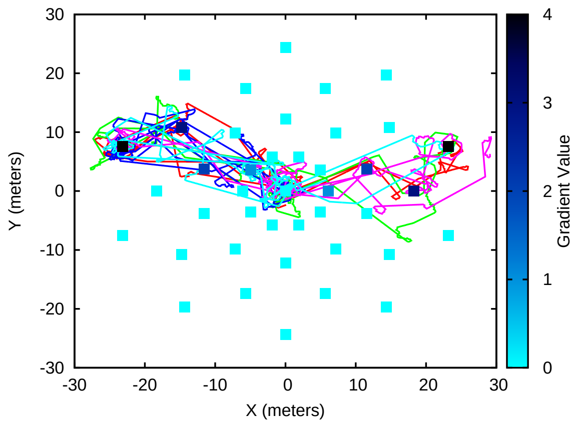

Figure 10: Simulated agents (five in total) navigating through a landscape using the gradient descent algorithm. The embedded devices produce an artificial field that is constant with the exception of two monotonically increasing paths anchored at the origin.

scenario finishes. However, this process is by no means efficient. While the gradient is attracting bees to move along the X axis, there is no guidance (with the exception of fading signal strength) in the Y axis. Despite this inefficiency and the ”hovering” behavior displayed around the goal, the path length is not prohibitively long. We ran the simulation 100 times using the gradient descent algorithm and found that the mean path length is about 1.88 times the optimal (straight line) path length. While there is room for improvement, the results are not bad for such trivial algorithms. In addition, we ran a series of experiments using the chemotaxis algorithm, varying the run bias from 10% to 90%. The mean path length over 100 executions for each bias variation ranged from 1.84 to 1.89 times the optimal length. This indicates that these is no real difference between the two movement algorithms, and, surprisingly enough, that changing the bias has little effect on the overall performance of chemotaxis. Of course, not all paths are equal - some have more straight sections and others may be dominated by twists and turns (an important distinction for a UAV platform in which turning is a difficult maneuver). If we classify portions of the paths into useful and superfluous segments, we may find that one algorithm outperforms the other.

We now turn our attention to the concentric circle scenario. The impetus for these experiments is to see how the progress evaluation function behaves when embedded nodes are distributed throughout the environment and there are multiple nodes broadcasting the same virtual field value. We present two variations of this scenario; the first is a discrete approximation of a ”funnel” in which the field is monotonically increasing uniformally in all directions away from the origin, and the second is a level field with two preferred paths away from the origin. Figure 9 depicts the path traces of five bees navigating the space according to the same ascend-descend-repeat behavior used in the line scenario. Each bee starts with a random orientation; this initial orientation along with the random tumbles taken by the bee during flight determine which outer node(s) it visits. The average path length over 100 executions is about 2.11 times the the optimal length. The increased overhead is likely caused by diversions in the path caused by confusion between multiple nodes broadcasting the same artificial field value. Figure 10 depicts the path traces of five bees using the same behavior in an environment where there are two preferred paths. Such a scenario might come about if bees had the ability to ”mark” nodes as they fly, similar to how ants use pheromones to mark trails. In the presence of marked nodes the bees would follow the path (by multiplying the ”funnel” field values with the ”path” field values to create a new field) and would otherwise explore the space or switch to another task. We ran this experiment 100 times, and the average path length is approximately 2.5 times the optimal length.

There are numerous other experiments we could run with the simulator. Our immediate future work will focus on modeling RSSI fluctuations more accurately, implementing better smoothing mechanisms, stigmergically constructing preferred paths, and developing methods for combining fields. The ultimate goal of the simulator is to work out the subleties in our algorithms prior to implementing them on a UAV testbed.

4.3 Human Trials

Since our UAV testbed is not yet running, we satisfied our curiosity about how this mechanism functions in the real world by conducting human trials. In these thoroughly unscientific experiments, a human subject attempted to move through an outdoor space by interpreting feedback provided by a laptop. The subject was deprived of sight and could only hear auditory cues through headphones. These audible cues corresponded to the periodic evaluation of the progress function, indicating if the subject gained or lost ground since the last update. Three TelosB motes were set up in the space approximately 20 feet apart to form a line. The experiment was deemed a success if the human subject, carrying a laptop connected to a receiving mote, moved to within 5 feet of the end node after being randomly oriented and walking more than 60 feet.

These experiments provided much needed (videotaped) comic relief, but no deep insights. Some of the participants were able to reach the goal, but it is unclear if this is due to random chance. Our main takeaway from these experiments is that our algorithm may work in practice, but because our test subjects were unable to follow a rigorously defined algorithm and often chose to ignore our auditory cues, this evidence is anecdotal at best.

5 Future Work

This project forms a solid foundation for significant future work. We have validated the feasibility of using artificial gradients for coarse-grained navigation and have formulated basic algorithms to accomplish this on a simple agent, however there are many opportunities for future research.

First and foremost, we would like to implement our algorithms using real UAVs. We have a set of eFlite Blade MCx remote control helicopters with custom sensor boards capable of running TinyOS that we would like to use to test our algorithms. Real algorithmic control (as opposed to human control) of our agents should give us a better understanding of how the algorithms work in practice. Once the TinyOS radio driver for the RF231 radio is ported to our robotic platform, we should easily be able to start experimenting.

We have presented only one of many possible gradient progress evaluation metrics. We need to explore metrics that take into account more state and that possibly give the agent additional information about its location within the field. It would also be useful to characterize for which fields we can have an absolute measurement of progress instead of a relative metric. Additionally, we would like to explore better RF signal smoothing, whether in the form of Kalman filters, more aggressive signal collection and averaging, or another technique from the literature.

Our current system design uses statically deployed nodes. We would like to explore methods for building fields at runtime using techniques from the sensor network and networked robotics literature. We would also like to consider mutable fields and inter-node communication which would facilitate more complex behaviors such as state propagation back towards the hive. We also envision using a higher-level programming language or mini-language in order to express our gradients more efficiently. We could also compile existing languages like Proto into TinyOS modules we could deploy on our system.

We hope to continue this work throughout the summer in order to answer many of these questions and test our system using real hardware. We believe the system will prove to be beneficial for the RoboBees project.

6 Related Work

We are by no means the first group to propose the use of embedded sensors to assist mobile robots with navigation through unknown territory. There are several notable projects in which this concept is demonstrated through simulation and prototype platforms.

The work of Corke et al. [5] aimed to guide a mobile agent to a destination through a potentially hostile environment, such as a forest fire. This system relied on the deployment of an embedded sensor nodes into the target environment. For their solution to work, the nodes must be localized (by GPS or other means) and form an ad-hoc sensor network once deployed. In the forest fire example, the embedded nodes would be responsible for sampling the environment and maintaining a temperature gradient that could be used to navigate resources toward the fire or evacuate personnel safely. With the embedded network in place, a distributed computation can determine the most efficient route through the environment while avoiding dangerous regions. The coordinates of the sensor nodes on the path are transmitted to the helicopter, which is equipped with a GPS device. We aim to achieve a less precise form of guidance by allowing the vehicle to make local decisions based on sensed information from passive (non-planning) nodes. In addition, our solution to this problem cannot rely on localizing the embedded nodes or autonomous agents.

Another project by Dantu et al. [6] used passive embedded sensor nodes to supply gradient information to a mobile robot. In this work, the sensor nodes are randomly deployed into the environment according to a minimum required density. The nodes are localized, but do not form a network. Each node is capable of sampling the environment and responding to queries from a mobile agent. The authors propose a control law that allows a mobile robot to descend the gradient formed by the sensed values or to trace a level set (contour) in the field. To determine the correct course of action, mobile robots query sensor nodes in the local neighborhood around its current position. The nodes return a position and sensor value, which can be used to compute the robot's next movement. Iteratively applying this technique allows the robot to move along (or orthogonally to) the gradient of the sensor field, which may fluctuate over time. Though randomly deployed in the environment, the embedded nodes are localized so that the absolute direction for the robot to travel can be accurately computed.

Batalin et al. [3] showed that a robot can use node-wise navigation to traverse an unknown environment. In this project, the continuous world is discretized to form a graph of nodes. As such, a network of embedded devices is deployed such that each node is aware of the relative direction to its neighbors. At the outset of a journey, the destination of the robot is given to the node closest to the robot's starting position. A distributed computation is performed by the embedded nodes to determine the transition weights in the graph that will move the robot to the goal along the shortest path. The robot then navigates to a destination by following hop-by-hop instructions provided by the nodes it encounters. Since the robot in this work has no localization capabilities, it relies on radio signal strength to determine if it has reached a node in the graph. The authors present a statistical mechanism, known as Adaptive Delta Percent, that outperforms simple thresholding mechanisms when making these decisions. We aim to explore this approach in our forthcoming helicopter testbed. This work is very similar to our own in that the capabilities of the embedded nodes and mobile robot are severely restricted. However, we are pursuing a solution that does not require a careful deployment (to form a well known graph) and a networked population of embedded nodes to compute movement plans.

Our solution relies on the construction and detection of artificial computational fields in the environment. This concept is directly related to amorphous computing, in which computation is defined in terms of a continuous amorphous medium that fills space. Each point in the amorphous medium has the ability to propagate, sense, and react to fields defined by the space. These concepts have been demonstrated in simulation and expressed in amorphous computing languages like Proto [1]. Proto allows users to manipulate fields (through combination, restriction, and other operations) to produce new fields. If certain particles in the medium actuate according to a sensed field it is possible to guide their movement by manipulating the field over time. This concept has been extended and applied to a multirobot system, Protoswarm [2], which is capable of simple coordinated behavior like dispersion and clustering. The amorphous computing abstraction is of limited usefulness to our project since Proto assumes the ability to localize points in the medium and Protoswarm bots have the ability to determine the relative position of neighbors. However, there is a great deal to learn from the field operations defined in Proto, which can be applied by our mobile agents when interpreting the signals from embedded nodes.

A similar approach to distributed motion coordination is used in the Co-Fields [11] model, in which a coordination field is created as a combination of fields generated by each participant in the system. Like many other systems, the Co-Fields model assumes a dense network of localized nodes spread throughout the environment. Despite this drawback, the literature provides some useful insight into constructing coordination fields for various tasks.

7 Conclusion

We believe that our efforts in this project have been largely beneficial. Through simulation we have shown that the concept of assisted navigation using RF transmitters embedded in a target environment to be feasible. Furthermore, we have shown that knowledge of position in the environment is not a prerequisite for offering guidance. Our system is capable of assisting a mobile agent by broadcasting an arbitrary number of artificial fields. Unlike prior systems that rely on real-world sensors to create a gradient from physical phenomena, our system is extensible. Though we cannot provide precision guidance, we are confident that the course bearing we offer to mobile agents is useful for long distance movement.

This solution is attractive to the RoboBees project not only for its ability to operate within the given constraints. We believe that the ability to define agent behavior in terms of operations on fields will greatly simplify the task of programming and managing swarms. In addition, the presence of the embedded nodes provides an indirect means of communication between bees - a potentially scalable solution that is also energy conscious. We hope to advance these concepts in the coming months and realize this potential.

References

[1] BACHRACH, J., AND BEAL, J. Programming a sensor network as an amorphous medium. *Distributed Computing in Sensor Systems (DCOSS)* (2006).

[2] BACHRACH, J., MCLURKIN, J., AND GRUE, A. Protoswarm: A language for programming multi-

robot systems using the amorphous medium abstraction. In *AAMAS '08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems* (2008), vol. 3, pp. 1175–1178.

[3] BATALIN, M., SUKHATME, G., AND HATTIG, M. Mobile robot navigation using a sensor network. In *IEEE International Conference on Robotics and Automation* (2004).

[4] *Chipcon AS SmartRF CC2420 Preliminary Datasheet (rev 1.2)*, 2004.

[5] CORKE, P., PETERSON, R., AND RUS, D. Networked robots: Flying robot navigation using a sensor net. *Proceedings of the 11th Annual International Symposium of Robotics Research* (2003).

[6] DANTU, K., AND SUKHATME, G. Detecting and tracking level sets of scalar fields using a robotic sensor network. In *IEEE International Conference on Robotics and Automation* (2007).

[7] DHARIWAL, A., SUKHATME, G., AND REQUICHA, A. Bacterium-inspired robots for environmental monitoring. In *IEEE International Conference on Robotics and Automation* (2004).

[8] JBullet. http://jbullet.advel.cz.

[9] LEE, H., CERPA, A., AND LEVIS, P. Improving wireless simulation through noise modeling. *Proceedings of the 6th international conference on Information processing in sensor networks* (2007).

[10] LEVIS, P., LEE, N., WELSH, M., AND CULLER, D. Tossim: accurate and scalable simulation of entire tinyos applications. *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems* (Nov 2003).

[11] MAMEI, M., AND ZAMBONELLI, F. Field-based approaches to adaptive motion coordination in pervasive computing scenarios. *Handbook of Algorithms for Mobile and Wireless Networking and Computing* (2004).

[12] RAPPAPORT, T. *Wireless Communications: Principles and Practice*, second ed. Prentice Hall, 2002.

[13] RoboBees. http://robobees.seas.harvard.edu.

[14] WHITEHOUSE, K., KARLOF, C., AND CULLER, D. A practical evaluation of radio signal strength for ranging-based localization. *ACM SIGMOBILE Mobile Computing and Communications Review* (2007), 41–52.