MULTISCALE COLLABORATIVE SEARCHING THROUGH SWARMING

Wangyi Liu, Martin B. Short, Andrea L. Bertozzi
Department of Mathematics, University of California, Los Angeles, CA 90095-1555, U.S.A.
{*bobbyliu, mbshort, bertozzi*}*@math.ucla.edu*

Yasser E. Taima
Department of Electrical Engineering, University of California, Los Angeles, CA 90095-1594, U.S.A.
yasser.taima@engineering.ucla.edu

Keywords: Mobile Robots, Autonomous Agents, Searching, Swarming, Target Detection.

Abstract: This paper presents a multiscale searching and target-locating algorithm for a group of agents moving in a swarm and sensing potential targets. The aim of the algorithm is to use these agents to efficiently search for and locate targets with a finite sensing radius in some bounded area. We present an algorithm that both controls agent movement and analyzes sensor signals to determine where targets are located. We use computer simulations to determine the effectiveness of this collaborative searching. We derive some physical scaling properties of the system and compare the results to the data from the simulations.

1 INTRODUCTION

Collaborative sensing has long attracted research interest. Researchers have investigated scenarios where sensors require localization (Bullo and Cortes, 2005), where they are used to control collaborative movement (Bopardikar et al., 2007), detect a scalar field (Gao et al., 2008), or perform a collaborative task (Smith and Bullo, 2009). Using such collaborating sensors to detect and locate targets within an area has been studied in reference to the "mine countermeasure" problem (Cook et al., 2003), the specific military task of locating ground or water-based mines.

In this paper, we develop a multiscale search and target-locating algorithm for a type of mine countermeasure problem in which a number of independent agents are given the task of determining the precise location of targets within a domain. The algorithm is designed to handle problems where the scale of the target sensing radius is much smaller than the domain size. The focus of this work is to identify optimality of the algorithm as a function of the swarm size, the number of agents per group and the distribution of resources into different groups.

We assume a connected domain with no holes, and use noisy sensors that detect a scalar quantity emitted by each target, but only when the agent is within a fixed distance r_s from a target. We control the motion of the agents through a model that makes the individuals form distinct swarms, and present filtering techniques that allow for locating targets despite noisy data. The inspiration for this approach comes in part from biology, as in the example of birds forming flocks when flying and searching for food (Travis, 2007).

Next, we analytically derive some of the system's main scaling properties, such as the relationship between the swarm size, the distance between agents and the target sensing radius, and compare to the experimentally recorded data. We conclude that our analytical approach closely matches the data from the simulations.

We assume a sensing radius r_s much smaller than the domain but comparable to or less than the swarm size. Other assumptions, however, may require different algorithms. For example, in (Burger et al., 2008) the sensing radius is infinite, but sensing is limited by obstacles, and in (Olfati-Saber, 2007) communication between agents is not always possible.

1.1 Scenario Description

We consider M targets in a two-dimensional simply-connected domain, that is, a flat enclosed area with no holes, and N agents able to move freely within the domain. Each target emits a radially symmetric

scalar signal $g(r)$ that decays with the distance r from the target, and drops to zero at some r_s, the target's sensing radius. Agents detect this signal with an additional Gaussian, scalar white noise component added. If an agent is within the sensing radius of multiple targets, it detects only the sum of the individual signals, again with a noise component added. We suppose that an agent takes sensor readings at regular intervals (once per "time step") spaced such that the noise between time steps can be assumed independent.

The algorithm accomplishes three tasks: filtering the noisy sensor data, controlling the coordinated movement of the agents based on this data, and determining when a target has been acquired and where it is located.

1.2 Structure of the Paper

The algorithm is described in the next three sections: Section 2 focuses on the techniques we use to process sensor data, Section 3 describes the general movement control of the agents, and Section 4 describes the method for locating a target. The algorithm is evaluated in Section 5. Scaling properties are derived and checked against simulations in Section 6, which is followed by a conclusion and ideas for future research in Section 7.

2 SENSOR DATA PROCESSING

Due to noise in the agent sensor readings and the sensing radius r_s being finite, we employ two distinct filters to the data from the readings: a Kalman filter and a cumulative sum (CUSUM) filter. The Kalman filter reduces or eliminates the noise in the data while the CUSUM filter is specifically suited to determining whether or not an agent is within the sensing radius of a target.

The sensor data model follows as a mathematical formula. As explained in Section 1, the formula describes sensor readings as the sum of scalar signals that depend only on the distance from a target, together with a noise component.

Given the M targets at positions y_j and an agent i with current position $x_i(t_k)$ at timestep k, the agent sensor reading $s_i(t_k)$ is given by

$$s_i(t_k) = \sum_{j=1}^{M} g(|y_j - x_i(t_k)|) + n_i(t_k) \ , \qquad (1)$$

where $n_i(t_k)$ denotes sensor noise and $g(r)$ is the signal strength at a distance r from the target. For simplicity, we have assumed that $g(r)$ is isotropic, smooth, decaying, the same for all targets, and has a cutoff at r_s.

2.1 Kalman Filtering

Before using the agent sensor readings to locate targets or control agent motion, we pass the sensor readings through a Kalman filter. Since the signal from the target is presumed to be varying smoothly with the distance to the target r (up to the cutoff point r_s) as the agents navigate the environment, a Kalman filter is a natural choice to eliminate or reduce noise in the sensor readings. The Kalman filter takes the sensor reading $s_i(t_k)$ of agent i at time t_k, and converts it into the filtered data $f_i(t_k)$ according to

$$P_i(t_k) = \frac{P_i(t_{k-1})R_i(t_k)}{P_i(t_{k-1}) + R_i(t_k)} + Q_i(t_k) \ , \qquad (2)$$

and

$$f_i(t_k) = f_i(t_{k-1}) + \frac{P_i(t_k)(s_i(t_k) - f_i(t_{k-1}))}{P_i(t_k) + R_i(t_k)} \ . \qquad (3)$$

Here $R_i(t_k)$ is the square of the noise amplitude, known or estimated by the agent, and $Q_i(t_k)$ is the square of the estimated change of the signal amplitude between two time steps, either fixed beforehand or estimated using the current velocity of the agent (in this paper, it is fixed beforehand). $P_i(t_k)$ is roughly the variance of the sensor reading's amplitude. The output $f_i(t_k)$ of this filter is then used in target locating, as described in Section 4.

2.2 Threshold Check and the CUSUM Filter

Before attempting to locate targets, an agent needs to determine whether or not it is receiving an actual signal, rather than just noise. In other words, an agent needs to determine whether it is within the sensing radius of a target at each time-step t_k. This information is then used both in controlling the movement of the agents, and in determining when to begin estimating a target's location. In order to determine the sensing status of an individual agent, we employ a CUSUM filter, as this type of filter is well-suited to determining abrupt changes of state (Page, 1954), and has been used in the similar task of boundary tracking (Jin and Bertozzi, 2007; Chen et al., 2009). In essence, the filter keeps a sort of running average of the signal, and notes when this average seems to have changed by rising above a certain threshold, indicating that the agent is now within the sensing radius of a target. As the noise is effectively summed up by the filter, it tends to cancel out.

In the original form of the CUSUM filter, we imagine a sensor that returns a sequence of independent observations $s(t_1)...s(t_n)$, each of which follows one of two probability density functions: a pre-change function g_0 and a post-change function g_1. The log-likelihood ratio is

$$Z(t_k) = \log[g_1(s(t_k))/g_0(s(t_k))] \,, \tag{4}$$

and we define the CUSUM statistic as

$$U(t_k) = \max(0, Z(t_k) + U(t_{k-1})), \;\; U(t_0) = 0 \,. \tag{5}$$

We then choose a threshold $\bar{U}$, and when $U(t_k) \geq \bar{U}$ for the first time, the algorithm ends and we declare that the state has changed from g_0 to g_1. The threshold should be chosen so as to minimize both false-alarms (these happen more frequently for small $\bar{U}$) and time to detection (this gets larger as $\bar{U}$ increases).

In our system, we choose the special case where sensor reading follows a Gaussian distribution. In the pre-change state, the agent is outside the sensing radius of any target and senses only noise, which we model as a Gaussian with zero mean. In the post-change state, the agent enters the sensing radius of a target, so the mean is larger than zero though the probability distribution is still Gaussian. If we estimate the mean of state g_1 as $2B$, then

$$\begin{aligned} Z(t_k) &= \log\left[\frac{e^{-[s(t_k)-2B]^2/2\sigma^2}/(\sigma\sqrt{2\pi})}{e^{-s(t_k)^2/2\sigma^2}/(\sigma\sqrt{2\pi})}\right] \\ &= \frac{-[s(t_k)-2B]^2}{2\sigma^2} + \frac{s(t_k)^2}{2\sigma^2} \\ &= \frac{2B}{\sigma^2}[s(t_k) - B] \,. \end{aligned} \tag{6}$$

We also modify the algorithm so that it can detect status changes both into and out of detection zones. Thus, we implement two filter values: $U_i(t_k)$ to determine when an agent has entered a zone, and $L_i(t_k)$ to determine if they have left a zone. We also define a binary function $b_i(t_k)$ which denotes the status of an agent; $b_i(t_k) = 1$ denotes that the agent is near a target and $b_i(t_k) = 0$ means otherwise. These filter values all start at zero, and are updated according to

$$U_i(t_k) = \max(0, s_i(t_k) - B + U_i(t_{k-1})) \,, \tag{7}$$

$$L_i(t_k) = \min(0, s_i(t_k) - B + L_i(t_{k-1})) \,, \tag{8}$$

and

$$b_i(t_k) = \begin{cases} 1 & b_i(t_{k-1}) = 0, U_i(t_k) > \bar{U} \\ 0 & b_i(t_{k-1}) = 1, L_i(t_k) < \bar{L} \\ b_i(t_{k-1}) & \text{otherwise.} \end{cases} \tag{9}$$

In addition, when the status of agent i changes, we reset the corresponding U_i or L_i to zero.

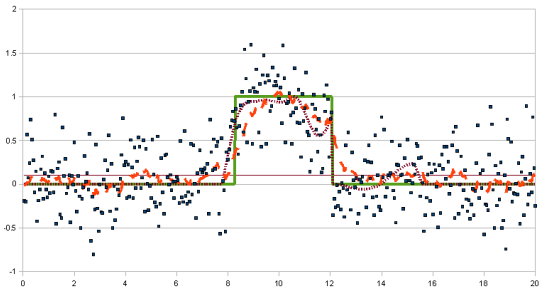

Figure 1: Sample filter results from one agent within a simulation as a function of time. The fine-dashed purple line represents the true signal that should be detected by the agent. The blue dots are the actual signal detected by the agent at each time step (i.e., the purple curve plus noise). The green line is the signal status returned by the CUSUM filter, with a thin crimson line representing the value $B = 0.1$. The sparsely-dashed red curve is the result of the Kalman filter.

Recall that B is a sensor value that is less than the predicted mean when inside a sensing radius, and $\bar{U}$ is our chosen detection threshold. So, when the agent is near a target, the sensor reading $s_i(t_k)$ tends to be larger than B, causing $U_i(t_k)$ to grow quickly until it is larger than $\bar{U}$, indicating a change in status. The converse is true if an agent leaves the sensing region of a target. The values of the various parameters of the filter are problem-specific, and should be estimated in a manner that minimizes both false-alarms while keeping the average time to detection as low as possible, as mentioned above.

An example of sensor reading from an agent within our current simulations can be seen in Fig. 1. The Kalman filter does a good job of reducing initial noise, bringing the sensor readings much closer to the true values. Near the middle of the plot, the agent enters into the sensing radius of a target, and this is reflected by a transition within the CUSUM filter from $b = 0$ to $b = 1$. There is, as expected from the behavior of CUSUM, a slight delay between when the agent actually enters into the radius and when this transition of b occurs. After spending some time within the sensing radius, the estimated target location stabilizes, the agent begins to subtract the true signal from its measurements (this will be explained below), and the agent leaves to find further targets.

3 AGENT MOVEMENT CONTROL

We have chosen to control the movement of our agents by breaking up our total agent population N

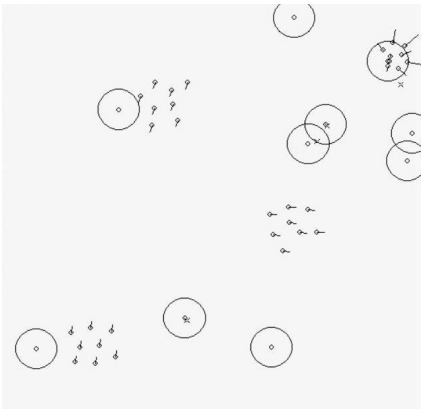

Figure 2: A screenshot from the simulation. Four swarms with eight agents each are used. Three of them are in the searching phase, and the upper right swarm is in the target-locating phase. Large circles around targets denote the sensing radius. Small crosses are already registered targets.

into a number of distinct, leaderless "swarms". This is done for a variety of reasons. Firstly, it increases robustness, as any individual swarm member is not critical to the functioning of the swarm as a whole. Secondly, since we imagine that any sensor data acquired by readings from these agents is local in space, a swarm provides a method of extending the effective sensing area to that of the whole swarm. Thirdly, a swarm of nearby agents may use their combined measurements to decrease sensor noise. Finally, the swarm provides the ability to locate targets via triangulation or gradient methods. Each of the various swarms may search within a different region of space if a divide-and-conquer tactic is desired, or each swarm may be free to roam over the entire region. In the following two sections we mainly focus on the control of one swarm.

Since the agents have a limited sensing radius, we choose to employ two different phases of swarm motion. When there are no targets nearby, the agents should move through the space as quickly and efficiently as possible, performing a simple flocking movement as legs of a random search. After a signal is sensed via the CUSUM filter, the agents should stop, then slowly move around the area, searching for the exact point of the nearby target. We call these two phases the searching phase and the target-locating phase, respectively. For a general idea of the two types of motion, see Fig. 2.

3.1 The Swarming Model

There are a variety of mathematical constructs that lead to agent swarming (see for example (Justh and Krishnaprasad, 2004), (Vicsek et al., 1995), and (Sepulchre et al., 2008)). Here we choose a second-order control algorithm similar to that described in (D'Orsogna et al., 2006) and (Chuang et al., 2007), which has been successfully implemented as a control algorithm for second-order vehicles on real testbeds (Nguyen et al., 2005; Leung and Bertozzi, 2007). In this system, each agent of the swarm is subject to self-propulsion, drag, and attractive, repulsive, and velocity alignment forces from each of the other agents. The position and velocity of an individual agent i are governed by

$$\frac{dx_i}{dt} = v_i \,, \tag{10}$$

and

$$m_i \frac{dv_i}{dt} = (\alpha - \beta |v_i|^2) v_i - \nabla U(x_i) + \sum_{j=1}^{N} C_o (v_j - v_i) \,, \tag{11}$$

where

$$U(x_i) = \sum_{j=1}^{N} C_r e^{-|x_i - x_j|/l_r} - C_a e^{-|x_i - x_j|/l_a} \,. \tag{12}$$

Depending upon the various values of the parameters, the swarms can undergo many complex motions (D'Orsogna et al., 2006), two of which are flocking and milling. In addition, in some cases the swarms can alter motions spontaneously (Kolpas et al., 2007). For our purposes, we simply alter the parameters to obtain the type of motion desired, as described in (D'Orsogna et al., 2006).

3.2 Searching Phase

In this phase, the agents move together in one direction as a uniformly-spaced group travelling with a fixed velocity. Since the agents know nothing about the location of targets, a random search is chosen here. Specifically, we use a Lévy flight, which is optimally efficient under random search conditions (Viswanathan et al., 1999), and is the same movement that some birds employ (Travis, 2007). To accomplish this type of search, we simply command the swarm to turn by some random angle after flocking for some random amount of time. For a Lévy flight, the time interval Δt between two turns follows the heavy-tailed distribution

$$P(\Delta t) \sim \Delta t^{-\mu} \,, \tag{13}$$

where μ is a number satisfying $1 < \mu \leq 3$. The value of μ should be chosen optimally according to the scenario in question, as in (Viswanathan et al., 1999). For destructive searching (where targets, once located, are

no longer considered valid targets), μ should be as close to 1 as possible. For non-destructive searching (where located targets remain as valid future targets), the optimal $\mu \sim 2 - 1/[\ln(\lambda/r_s)]^2$, where λ is the mean distance between targets and r_s is the sensing radius.

3.3 Target Locating Phase

When enough agents agree that a target is nearby (see Section 2.2), the target-locating phase begins. In this phase, we want the agents to move only towards the target, so we remove the velocity alignment force ($C_o = 0$), disable self-propulsion ($\alpha = 0$), and issue a halt command so that all agents begin target locating with zero velocity. In addition, data from agents within the sensing radius is used to continually estimate the position $\bar{y}$ of the target (see section 4), and the agents in the swarm then try to move towards it, thus attracting other agents in the swarm not yet in the sensing radius to move closer to the target as well.

To make the agents move towards the target, we add another potential in Eq. 12,

$$U_c = C_c(x_i - \bar{y})^2/2 \,, \tag{14}$$

where $\bar{y}$ is the estimated position of the target. The full control equations in the target-locating phase therefore become Eq. 10 and

$$m_i \frac{dv_i}{dt} = -\beta |v_i|^2 v_i - \nabla U(x_i) \,, \tag{15}$$

where

$$U(x_i) = \frac{1}{2} C_c (x_i - \bar{y})^2 + \sum_{j=1}^{N} C_r e^{-|x_i - x_j|/l_r} - C_a e^{-|x_i - x_j|/l_a} \,. \tag{16}$$

To show that this system converges to a stationary swarm centered on the target, we note that the total energy of the target locating system,

$$E = \frac{1}{2} \sum_{i=1}^{N} m_i |v_i|^2 + \sum_{i=1}^{N} U(x_i) \,, \tag{17}$$

serves as a Lyapunov function, so that the collective tends to minimize it. That is,

$$\dot{E} = -\beta \sum_{i=1}^{N} |v_i|^4 \leq 0 \,. \tag{18}$$

Hence, velocities will eventually reach zero (due to drag) and the swarm members will spatially reorder themselves so as to minimize the potential energy, forming a regular pattern centered at the target position. This stationary state serves as a spiral sink, however, so the swarm tends to oscillate about the target position for some amount of time that depends on the value of C_c, with a high C_c yielding less oscillation. However, since the potential being minimized now includes a term that is effectively attracting all of the agents towards the center of mass, the swarm will be more compact than it was before the target locating potential was added, so too large of a C_c will make the swarm smaller than desired. In practice, then, we want to make C_c just large enough to minimize the oscillations in space without making the swarm get too compact.

4 LOCATING TARGETS

During the target-locating phase of motion, all agents of the swarm that are within the sensing radius keep a common register of all of their positions and signal readings made since entering the radius (see "Threshold Check", Section 2.2, above). The agents then use a least-squares algorithm to give an estimate $\bar{y}$ of where the target is located via

$$\bar{y} = \min_{y} \sum_{k=1}^{N'} [g(|y - x(t_k)|) - f(t_k)]^2 \,, \tag{19}$$

where N' is the number of sensor readings in the common register.

Solving this least-squares minimization is quite straightforward, but the technique requires certain assumptions. Firstly, it is assumed that the form of $g(r)$ is known by the agents. For certain classes of targets and scalar fields, we believe this assumption to be fair. Secondly, it is assumed that only one target is nearby, or that one target is much closer to the agents than any other target. When the sensing radius is small compared to the average distance between targets, these assumptions should hold true. If, however, these assumptions are invalid for the particular system at hand, other methods such as gradient estimation could be employed.

If the estimated position of the target stabilizes, it is considered to have been located, and the agents register the position of the target and return to the searching phase. The model signal $g(r)$ from the registered target will be subtracted from further sensor readings so that it is not detected again, a form of destructive searching. We thus modify Eq. 1 to read:

$$s_i(t_k) = \sum_{j=1}^{M} g(|y_j - x_i(t_k)|) + n_i(t_k) - \sum_{j=1}^{M'} g(|\bar{y}_j - x_i(t_k)|) \,, \tag{20}$$

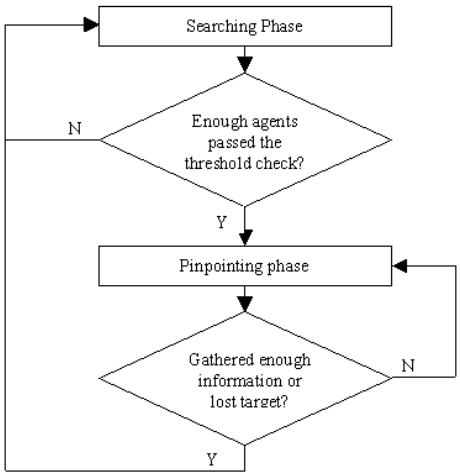

Figure 3: A simple flowchart of the algorithm.

where M' is the total number of registered targets. Note that the positions of these targets may or may not be accurate, due to noise and other errors. If, instead of the estimated target location stabilizing, the agents lose track of the target, they simply return to the searching phase without registering the target.

For a general idea of the entire algorithm, see Fig. 3.

5 PERFORMANCE EVALUATION

Two main criteria for the evaluation of this algorithm are efficiency and accuracy. The two criteria are roughly determined by the two different phases: efficiency is mainly related to the swarming phase, while accuracy is mainly related to the target-locating phase. To evaluate the performance of the algorithm, we used the following measurements: the average time needed for the group to locate one target (average time), the average distance between the actual and estimated target positions (average error), and the percentage of registered positions that are not within any actual sensing radius (false register). Note that the false registers are not included in the average error calculation.

We ran computer simulations of the algorithm in a dimensionless 20 by 20 area, with a total of 32 agents and a dimensionless sensing radius of 1. The signals have a Gaussian form, with a peak signal-to-noise ratio of about 10.5 dB. Two cases were considered. In the first case, there were 20 targets and we restricted the duration of the simulation, the main goal being that of measuring efficiency. In the second case, we distributed just 5 targets randomly, and used a much longer time limit, with the main goal of measuring accuracy. In either case, the simulation ends either when time runs out or when all targets are found. For each case, we performed 100 trials and calculated the average of the measurements.

Table 1: Case 1: 20 targets, time limit 50.0. Asterisks denote the use of the divide-and-conquer tactic.

Swarms	Agents/ swarm	Average time	Average error	False reg.
1	32	9.17	0.163	9.77%
2*	16	4.83	0.155	8.40%
2	16	5.45	0.159	11.90%
4*	8	3.15	0.158	8.68%
4	8	3.52	0.16	10.59%
8*	4	2.67	0.208	9.91%
8	4	2.9	0.200	11.73%
16*	2	2.64	0.257	15.59%
16	2	2.64	0.253	15.17%

Since we may have multiple groups, it is important to decide how they cooperate with one another. Here, we try two different policies. One is a simple divide-and-conquer tactic where we divide the whole region into subregions before the simulation, and each group is in charge of a single sub-region, remaining within that area the entire time, and performing a Lévy flight search pattern within its designated sub-region. The other policy allows all groups to search the entire region independently. In the results, we denote the use of the divide-and-conquer tactic with an asterisk (*).

An important factor that influenced these measurements is how many swarms we divided the agents into, or equivalently, the number of agents in each swarm. We therefore present the results for various sub-divisions. The results are listed in Tables 1 and 2, with their associated plots presented in Figs. 4 and 5.

From this data we can see that the number of agents in the swarm works as a balance between accuracy and efficiency. As could have been anticipated, larger swarms give more accurate results, while multiple, smaller swarms make the search more efficient. To have an acceptably low error and low false target registration rate, groups of at least four agents should be used. This is perhaps due to the fact that at least three agents are needed to locate a target, using triangulation. Also, we note that the divide-and-conquer tactic seems to work better for this search scenario.

Table 2: Case 2: 5 targets, time limit 200.0. Asterisks denote the use of the divide-and-conquer tactic.

Swarms	Agents/ swarm	Average time	Average error	False reg.
1	32	45.53	0.128	10.76%
2*	16	25.51	0.116	8.06%
2	16	26.89	0.117	8.95%
4*	8	14.22	0.134	8.96%
4	8	16.64	0.118	8.79%
8*	4	8.35	0.161	7.24%
8	4	10.58	0.172	8.97%
16*	2	8.31	0.223	11.97%
16	2	8.91	0.252	13.79%

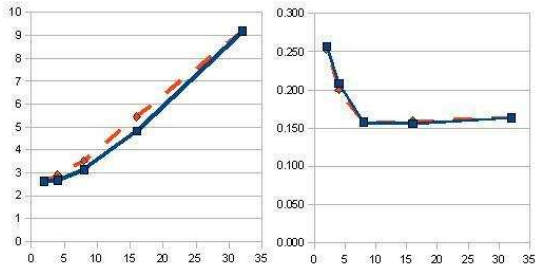

Figure 4: Average search time (left) and average error (right) as a function of the number of agents in each swarm for case 1 (20 targets and time limit 50.0). The blue line is for the divide-and-conquer tactic and the red dashed line is for the result without divide-and-conquer.

6 SCALING PROPERTIES

Having noted the results above, one may wonder how these are affected by the various scales present in the system, such as the swarm size, distance between agents, target sensing radius, etc. Below we present some arguments for determining optimal search parameters given these scales.

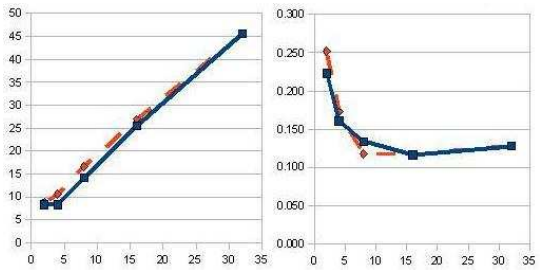

Figure 5: Average search time (left) and average error (right) as a function of the number of agents in each swarm for case 2 (5 targets and time limit 200.0). The blue line is for the divide-and-conquer tactic, and the red dashed line is the for result without divide-and-conquer.

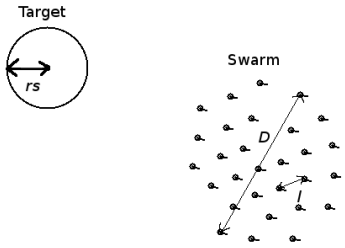

Figure 6: Scales influencing target locating time: the swarm diameter D, inter-agent length l, and sensing radius r_s.

6.1 Estimating the Swarm Diameter

We first define a measure for the swarm size, the swarm diameter $D = \max(|x_i - x_j|)_{i=1}^{N}$, where N is the number of agents in the swarm. Let us also define the inter-agent distance $l = |x_i - x_j|$ for any two nearest-neighbor agents i, j.

For the remainder of this section (and for the results in Fig. 7), we choose the parameters of motion so that the system is either in regime VI (catastrophic) or VII (H-stable) as defined in (D'Orsogna et al., 2006), with the swarms flocking naturally in VII, and in VI due to the velocity alignment term C_o in Eq. 11. Under these regimes, D and l stabilize after a transient period, so for the purposes of this section we will consider them to be constant in time. In such a stable swarm, agents are uniformly distributed in space along a hexagonal pattern, so that the swarm diameter D and inter-agent length l are related geometrically as follows: since the area occupied by a single agent in the swarm is $A_a \approx \pi l^2/4$ and the total swarm area is $A_s = \pi D^2/4 \approx N A_a$, then

$$D \simeq \sqrt{N} l \ . \tag{21}$$

Thus, D scales with l, and for $N = 16$ (as used in Fig. 7) we get $D \simeq 4l$. Since l is approximately the distance that minimizes the inter-agent potential of Eq. 12, we can easily adjust the swarm diameter D by varying the system parameters.

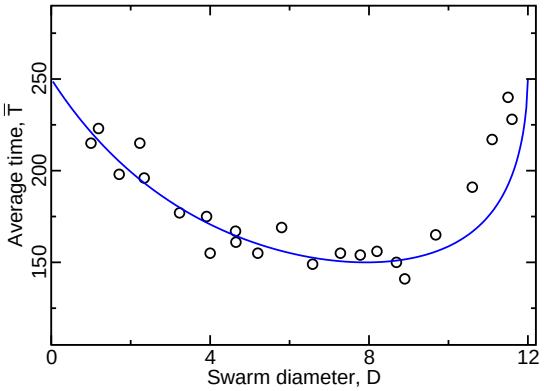

Figure 7: Average time to reach a target $\bar{T}$ as a function of the swarm diameter D for $r_s = 3.0$. Values of D were obtained over a range $C_r = 2.0 - 12.0$, $l_r = 0.2 - 0.7$, $C_a = l_a = 1.0$. Averaging was carried out over 200 simulated trials, yielding the numerical results (circles); the theoretical results of Eq. 26 with the best fitting τ are shown as a line. The number of agents $N = 16$, and $D_{\text{opt}} \approx 8$.

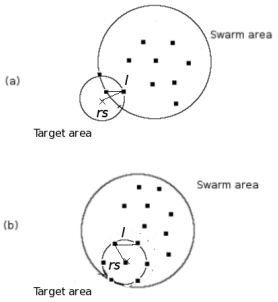

Figure 8: Sensing configurations for the case when 4 or more agents are required to sense the target before it can be detected. (a) Though there is overlap between the swarm and target, too few agents can sense the target for it to be detected. (b) The largest inter-agent distance l while still allowing for detection, $l = r_s$.

6.2 An Upper Bound on the Optimal Swarm Diameter

Consider a setting with one target of sensing radius r_s and one swarm of diameter D, as in Fig. 6. We measure the average time to locate the target $\bar{T}$ by starting the swarm at the center of the search field at t_0, placing the target at a random location within the field, allowing the simulation to run until time T when the target is found, and averaging these T values over many runs of the simulation. As D grows, we observe (see Fig. 7) that $\bar{T}$ decreases until an optimal swarm diameter D_{opt} is reached, after which $\bar{T}$ increases again, growing without bound. We wish to explain this, by first finding an upper bound on D_{opt}.

Let us begin by fixing the swarm consensus percentage at 25%; i.e., the swarm of agents decides a target is present when 1/4 of the agents or more are within the sensing area of a target, $A_t = \pi r_s^2$ (see Fig. 8). Clearly, the borderline case between detection and non-detection occurs when the target area is completely subsumed within the swarm area, yet there are only just enough agents (25% of the total) within the target sensing radius to detect it. If we assume a constant density of agents in the swarm, this scenario occurs when the target area is 1/4 of the swarm area. So, it must be the case that

$$D \leq 4r_s \,, \tag{22}$$

or else the target will not be detected at all. This condition therefore gives an upper bound for D_{opt}. The condition (22) can also be written in terms of l, in which case $l \leq r_s$; the borderline case is illustrated in Fig. 8(b). In Fig. 9, snapshots from an actual simulation show how the swarm flies over the target without being able to detect it in a case when condition (22) is violated.

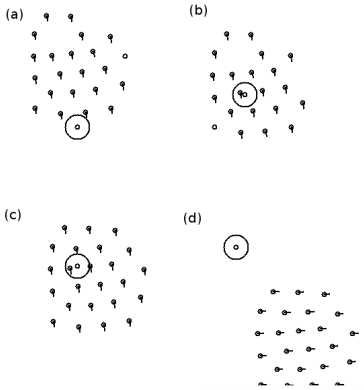

Figure 9: When r_s is too small compared to D, the swarm does not detect the target. The snapshots (a)-(d) show the swarm flying over the target without locating it. Here $N = 24$, $r_s = 1.5$, required percentage for consensus is 25%, and D stabilizes at ≈ 13.5.

6.3 An Approximation for the Optimal Swarm Diameter

Now that we have an upper bound on D_{opt}, we assume that condition (22) is met and look for approximate expressions for D_{opt} and $\bar{T}$. First, we note that the

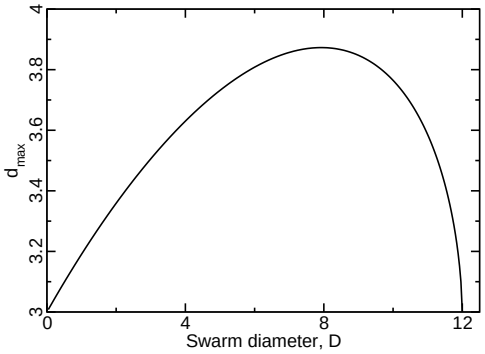

Figure 10: Parameter $d_{\max}$ versus D for $r_s = 3.0$. Note that $d_{\max}$ increases at first, as the growing size of the swarm allows it to be further away while still easily satisfying (22). However, after the peak at $D_{\text{opt}} \approx 8$, the condition (22) becomes the limiting factor, requiring greater overlap between the two areas for detection to occur.

area of overlap A_o between the target area and swarm area, when the centers are separated by a distance d, is given by

$$A_o = r_s^2 \arccos\left[\frac{z}{r_s}\right] + \frac{D^2}{4}\arccos\left[\frac{2(d-z)}{D}\right] - z\sqrt{r_s^2 - z^2} - (d-z)\sqrt{D^2/4 - (d-z)^2}\,, \quad (23)$$

where

$$z \equiv \frac{r_s^2 - D^2/4 + d^2}{2d}\,. \quad (24)$$

Eq. 23 is valid for $|r_s - D/2| \le d \le r_s + D/2$. Now, as above, we require that A_o be at least equal to 25% of the swarm area in order for the target to be detected. Thus, we obtain an implicit equation for the maximum separation $d_{\max}$ between the center of the swarm and the target location such that the target is detected:

$$A_o(r_s, D, d_{\max}) = \pi D^2/16\,. \quad (25)$$

The parameter $d_{\max}$ will depend therefore upon r_s and D. At least in terms of the time spent within the searching phase of the algorithm, the shortest time $\bar{T}$ until detection ought to occur when, for a given r_s, D is chosen such that $d_{\max}$ is maximized (see Fig. 10), giving the largest effective target size to hit; hence this D should be D_{opt}. Furthermore, we expect a scaling law such that the time to detection is roughly given by

$$\bar{T} \approx \tau\left[\frac{A_{\text{field}}}{\pi d_{\max}^2} - 1\right]\,, \quad (26)$$

where τ is a characteristic timescale and A_{field} is the total area of the search field.

We have experimentally verified this scaling, with numerical results usually quite close to the theoretic values, as illustrated by the example with $r_s = 3.0$ in Fig. 7. We do note that the actual time to detection is a bit above the theory for $D > D_{\text{opt}}$, presumably due to our assumption of constant density in deriving Eq. 25; that is, (especially for large D) the area of overlap between target and swarm may be sufficient, but still not contain at least 25% of the agents, causing the time to detection to be above that expected.

7 CONCLUSION

We considered a mine counter-measure type scenario using multiple agents that move cooperatively via swarming. The agents use a variety of signal filters to determine when they are within sensing range of a target and to reduce noise for more accurate control and locating of targets. We explored the parameter space through simulations, determining optimal values for some of the search parameters. We derived scaling properties of the system, compared with the data from simulations, and found a good experimental-analytical fit.

There are many openings for future research in this area. First, we could use alternate methods in some parts of the algorithm. A potential change is to use a compressed sensing method (Cai et al., 2008) for target locating, which would enable us to find multiple targets at the same time. Another interesting modification would be to use an anisotropic Lévy search and take previously covered paths into account. Different scenarios could also be evaluated, which might lead to different results for accuracy and efficiency, or even suggest using new algorithms. For example, we could extend the two dimensional problem to 3-D, as would be the case for underwater targets. Or, perhaps the model for the detected signal is unknown, in which case we would employ a different method to estimate the target positions. Finally, apart from numerical simulations, we plan to do experiments on a real testbed, with small robotic vehicles as agents. This would provide an evaluation of the algorithm in the presence of real sensor noise, which may not be entirely Gaussian in nature.

ACKNOWLEDGMENTS

The authors would like to thank Alex Chen and Alexander Tartakovsky, whose suggestions on the use of the CUSUM filter were very helpful.

REFERENCES

Bopardikar, S. D., Bullo, F., and Hespanha, J. P. (2007). A cooperative homicidal chauffeur game. In *IEEE Conference on Decision and Control*.

Bullo, F. and Cortes, J. (2005). Adaptive and distributed coordination algorithms for mobile sensing networks. In *Lecture Notes in Control and Information Sciences*. Springer Verlag.

Burger, M., Landa, Y., Tanushev, N., and Tsai, R. (2008). Discovering point sources in unknown environments. In *The 8th International Workshop on the Algorithmic Foundations of Robotics*.

Cai, J., Osher, S., , and Shen, Z. (2008). Linearized Bregman iterations for compressed sensing. In *UCLA CAM report 08-06*.

Chen, A., Wittman, T., Tartakovsky, A., and Bertozzi, A. (2009). Image segmentation through efficient boundary sampling. In *8th international conference on Sampling Theory and Applications*.

Chuang, Y., Huang, Y., D'Orsogna, M., and Bertozzi, A. (2007). Multi-vehicle flocking: Scalability of cooperative control algorithms using pairwise potentials. In *IEEE International Conference on Robotics and Automation*.

Cook, B., Marthaler, D., Topaz, C., Bertozzi, A., and Kemp, M. (2003). Fractional bandwidth reacquisition algorithms for vsw-mcm. In *Multi-Robot Systems: From Swarms to Intelligent Automata*, volume 2, pages 77–86. Kluwer Academic Publishers, Dordrecht.

D'Orsogna, M., Chuang, Y., Bertozzi, A., and Chayes, L. (2006). Self-propelled particles with soft-core interactions: Patterns, stability and collapse. In *Physical Review Letters*, 96(10):104302.

Gao, C., Cortes, J., and Bullo, F. (2008). Notes on averaging over acyclic digraphs and discrete coverage control. In *IEEE Conference on Decision and Control*.

Jin, Z. and Bertozzi, A. (2007). Environmental boundary tracking and estimation using multiple autonomous vehicles. In *Proceedings of the 46th IEEE Conference on Decision and Control*.

Justh, E. and Krishnaprasad, P. (2004). Equilibria and steering laws for planar formations. In *Systems & Control Letters*, 52(1):25–38.

Leung, K., Hsieh, C., Huang, Y., Joshi, A., Voroninski, V., and Bertozzi, A. (2007). A second generation micro-vehicle testbed for cooperative control and sensing strategie. In *Proceedings of the 2007 American Control Conference*.

Kolpas, A., Moehlis, J., and Kevrekidis, I. (2007). Coarse-grained analysis of stochasticity-induced switching between collective motion states. In *Proceedings of the National Academy of Sciences USA*, 104:5931–5935.

Nguyen, B., Yao-Ling, C., Tung, D., Chung, H., Zhipu, J., Ling, S., Marthaler, D., Bertozzi, A., and Murray, R. (2005). Virtual attractive-repulsive potentials for cooperative control of second order dynamic vehicles on the caltech mvwt. In *Proceedings of the American Control Conference*.

Olfati-Saber, R. (2007). Distributed tracking for mobile sensor networks with information-driven mobility. In *Proceedings of the 2007 American Control Conference*.

Page, E. (1954). Continuous inspection schemes. In *Biometrika*, 41(1-2):100–115.

Sepulchre, R., Paley, D., and Leonard, N. (2008). Stabilization of planar collective motion with limited communication. In *IEEE Transactions on Automatic Control*.

Smith, S. L. and Bullo, F. (2009). The dynamic team forming problem: Throughput and delay for unbiased policies. In *Systems & Control Letters*.

Travis, J. (2007). Do wandering albatrosses care about math? In *Science*, 318:742–743.

Vicsek, T., Czirok, A., Ben-Jacob, E., Cohen, I., and Shochet, O. (1995). Novel type of phase transition in a system of self-driven particles. In *Physical Review Letters*, 75:1226–1229.

Viswanathan, G., Buldyrev, S., Havlin, S., da Luzk, M., Raposok, E., and Stanley, E. (1999). Optimizing the success of random searches. In *Nature*, 401(6756):911–914.