Collaborative Searching Through Swarming

Wangyi Liu, Martin B. Short, Andrea L. Bertozzi

***Abstract*— This paper presents a searching algorithm for a group of agents moving in a swarm and sensing potential targets. The objective of the algorithm is to use these groups 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.**

I. INTRODUCTION

Collaborative sensing has long attracted research interest. Researchers have variously investigated scenarios where sensors require localization [13], where they are used to control collaborative movement [10], used to detect a scalar field [12], or are needed to perform a collaborative task [11]. In addition, using such collaborating sensors to detect targets within an area has been studied in reference to the "mine counter-measure" problem [19], referring to the specific military task of locating ground or water-based mines. In this paper, we develop an algorithm for a specific type of mine counter-measure problem, in which a number of independent agents are tasked with determining the locations of targets within a fixed area using noisy sensors that detect a scalar quantity emmitted by the targets, but only when within a fixed distance r_s from a target. We control the motion of the agents through an algorithm that causes the individuals to form distinct swarms, and present techniques and filters that allow for the use of the noisy data to locate targets. This approach is partly inspired by biological examples such as birds, which have long been known to form flocks when moving and searching for food [18].

We assume here a sensing radius r_s much smaller than the domain size but comparable to swarm size. Different assumptions, however, may require different algorithms. For example, in [20] there is an infinite sensing radius, but sensing is limited by obstacles, and in [23] is a scenerio where communication between agents is not automatic.

A. Scenario Description

We consider a simple scenario in which there are M targets in an area S, each emmitting a scalar signal $g(r)$ that decays with distance and effectively drops to zero at some radius r_s. We have N agents, each with a sensor that can detect the signals of the targets, albeit with (perhaps very strong) noise. If an agent receives signals from multiple targets at the same time, only the sum of these signals is returned as sensor data. For simplicity, we suppose the agents take sensor readings once every time step, which is long enough that noise is independent between two time steps.

The algorithm needs to accomplish three basic tasks: filter the noisy sensor data into a usable form, control the coordinated movement of the agents using this data, and use the data to determine when a target has been acquired and where that target is located.

B. Structure of this paper

The algorithm is described in the next three sections: Section II focuses on the techniques we use to process sensor data, Section III describes the general movement control of the agents, and Section IV describes the method for locating the targets. The evaluation of our algorithm is presented in section V, which is followed by conclusion and ideas for future research.

II. SENSOR DATA PROCESSING

Due to the existence of noise in the sensor data and the finite sensing radius of targets, we employ two separate filters to our data: a Kalman filter and a cumulative sum (CUSUM) filter. The former is used to simply reduce the initial noise in the data while the latter is specifically suited to determining whether or not an agent has entered into the sensing radius of a target.

We now write our model for the sensor output as a mathematical formula. As described in Section I, the formula models scalar sensors with signals that depend only on distance from a target. Given the M targets at positions y_j and an agent i with current position $x_i(t_k)$ at timestep k, the sensor reading $s_i(t_k)$ is given by

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

where $n_i(t_k)$ denotes sensor noise, and $g(r)$ again is the signal pattern emitted by a target. For simplicity, we have assumed that $g(r)$ is isotropic, decaying, is the same for all targets, and has a cutoff at radius r_s.

A. Kalman Filter

Before we use the sensor readings to try to estimate the position of nearby targets or control motion, we pass the data through a Kalman filter. Since the signal is presumed to vary smoothly as the agents move throughout the environment, a Kalman filter is a natural choice to eliminate much of the initial noise within the sensor readings. This filter takes the

This paper is supported by ARO MURI grant 50363-MA-MUR

W. Liu, M.B. Short, A.L. Bertozzi are with the Department of Mathematics, University of California Los Angeles, Los Angeles, CA 90059, USA. Email: {bobbyliu, mbshort, bertozzi}@ucla.edu

raw signal s_i of agent i at time t_k, and converts it into a filtered signal f_i according to

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

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

Here $R_i(t_k)$ is the square of noise strength, which is supposed to be known or estimated by the agents. $Q_i(t_k)$ is the square of estimated change of the signal between two time steps, for which we can give a fixed value or estimate the value using the current velocity of the agent. $P_i(t_k)$ represents roughly the variance of the signal. The output f_i of this Kalman filter is then used in target estimation, as described later.

B. Threshold Check and the CUSUM Filter

Before attempting to locate targets, each agent needs to determine whether or not it is receiving a true target signal, rather than simply noise. Equivalently, each agent needs to know if it is within the sensing radius of a target at any given time t_k. This knowledge is then used both in controlling the movement of the agents and in determining when to begin attempting to estimate target locations. In order to determine the sensing status of an individual, we employ a CUSUM filter, as this type of filter is well-suited to determining abrupt changes of state [7], and has been used in the similar task of boundary tracking [5][21]. 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

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

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 output 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 some target, so the mean is larger than zero though the distribution is still a Gaussian. If we estimate the mean of state g_1 as $2B$, then

$$Z(t_k) = \log\left[\frac{e^{-[s(t_k)-2B]^2/2\sigma^2}/(\sigma\sqrt{2\pi})}{e^{-s(t_k)^2/2\sigma^2}/(\sigma\sqrt{2\pi})}\right] \tag{6}$$

$$= \frac{-[s(t_k) - 2B]^2}{2\sigma^2} + \frac{s(t_k)^2}{2\sigma^2} \tag{7}$$

$$= \frac{2B}{\sigma^2}[s(t_k) - B] \tag{8}$$

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

$$U_i(t_k) = \max(0, s_i(t_k) - B + U_i(t_{k-1})) \tag{9}$$

$$L_i(t_k) = \min(0, s_i(t_k) - B + L_i(t_{k-1})) \tag{10}$$

$$b_i(t_k) = \begin{cases} 1 & b_i(t_{k-1}) = 0,\ U_i(t_k) > \bar{U} \\ 0 & b_i(t_{k-1}) = 1,\ L_i(t_k) < \bar{L} \\ b_i(t_{k-1}) & \text{otherwise.} \end{cases} \tag{11}$$

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

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, sensor data $s_i(t_k)$ tend 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 data 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, and the agent leaves to find further targets.

III. MOVEMENT CONTROL OF AGENTS

We have chosen to control the movement of our agents by breaking up our total agent population N into a number of distinct, leaderless "swarms". This is done for a variety of reasons. First, it increases robustness, as any individual

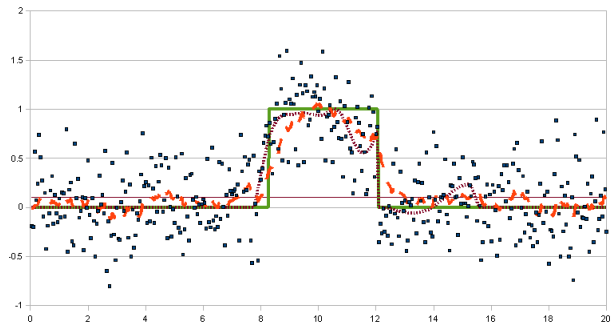

Fig. 1. Sample filter results from one agent within a simulation as a function of time. The fine-dashed purple line represents the true signal that should be detected by the agent. The blue dots are the actual signal detected by the agent at each time step (i.e., the purple curve plus noise). The green 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.

swarm member is not critical to the functioning of the swarm as a whole. Second, since we imagine that any sensor data acquired by these agents is local in space, a swarm provides a method of extending the effective sensing area to that of the whole swarm. Third, a swarm of nearby agents may use their combined measurements to decrease sensor noise. Fourth, 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 pinpointing phase, respectively. For a general idea of the two types of motion, see Fig. 2.

A. The Swarming Model

There are a variety of mathematical constructs that lead to agent swarming (see for example [3], [4] and [6]). Here we choose a second-order control algorithm similar to that described in [1] and [2], which has been successfully implemented as a control algorithm for second order vehicles on real testbeds[8],[9]. In this system, each agent of the swarm is subject to self-propulsion, drag, and attractive, repulsive, and velocity alignment forces from each other agent. The position and velocity of an individual agent i are governed by

$$\frac{dx_i}{dt} = v_i \tag{12}$$

$$m_i \frac{dv_i}{dt} = (\alpha - \beta|v_i|^2)v_i - \nabla_i U(x_i) + \sum_{j=1}^{N} C_o(v_j - v_i) \tag{13}$$

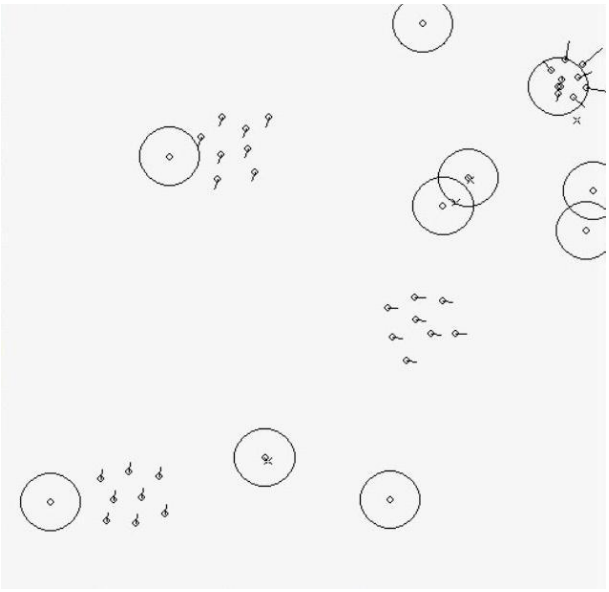

Fig. 2. A screenshot from the simulation. Four swarms with eight agents each are used. Three of them are in the searching phase, and the upper right swarm is in the pinpointing phase. Large circles around targets denote the sensing radius. Small crosses are already registered targets.

where

$$U(x_i) = \sum_{j=1}^{N} C_r e^{|x_i - x_j|/l_r} - C_a e^{|x_i - x_j|/l_a} \tag{14}$$

Depending upon the various values of the parameters, the swarms can undergo many complex motions [1], two of which are flocking and milling. In addition, in some cases the swarms can alter motions spontaneously [22]. For our purposes, we simply alter the parameters as necessary to accomplish the type of motion currently desired.

B. Searching Phase

In this phase, the agents move together in a single direction as a regularly-spaced group moving with a fixed velocity. Since the agents know nothing about the location of targets, a random search is chosen here. Specifically, we use a Levy flight, which is known to be optimally efficient under certain search conditions [16], and is the same movement that some birds employ [18]. To accomplish this type of search, we simply command the swarm to turn by some random angle after flocking for some random length of time. For a Levy flight, the time interval Δt between two turns follows the heavy-tailed distribution

$$P(\Delta t) \sim \Delta t^{-\mu} \tag{15}$$

where μ is a number satisfying $1 < \mu \leq 3$. The value of μ should be chosen optimally according to the scenario in question [16]. For destructive searching, μ should be as close to 1 as possible. For nondestructive searching, 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.

C. Pinpointing Phase

When enough agents agree that a target is nearby (see section III, CUSUM filter), the pinpointing phase begins. In this phase, we want the agents to move only toward 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 the pinpointing with zero velocity. In addition, data from those agents that are within the sensing radius are used to continually estimate the position $\bar{y}$ of the target (see section IV) and the swarm members then try to move towards it, thus causing other agents not yet in the sensing radius to move closer to the target as well.

To make the agents move towards the target, we add an additional potential to equation (14) of the form

$$U_c = C_c(x_i - \bar{y})^2/2 \tag{16}$$

where $\bar{y}$ is the estimated position of the target. The full control equations in the pinpointing phase are therefore Eq. 12 and

$$m_i \frac{dv_i}{dt} = -\beta|v_i|^2 v_i - \nabla_i U(x_i) \tag{17}$$

where

$$U(x_i) = \frac{1}{2}C_c(x_i - \bar{y})^2 + \sum_{j=1}^{N} C_r e^{|x_i - x_j|/l_r} - C_a e^{|x_i - x_j|/l_a} \ . \tag{18}$$

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

$$E = \frac{1}{2}\sum_{i=1}^{N} m_i v_i^2 + \sum_{i=1}^{N} U(x_i) \tag{19}$$

serves as a Lyapunov function, so the collective tends to minimize this quantity. That is,

$$\dot{E} = -\beta \sum_{i=1}^{N} v_i^4 \leq 0 \ . \tag{20}$$

Hence, the velocities will eventually reach zero (due to the drag) and the swarm members will spatially order themselves so as to minimize the potential energy, forming a regular pattern centered on 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 upon the value of C_c, with high C_c yielding less oscillation. However, since the potential being minimized now includes a term that is effectively attracting all of the agents toward the center of mass, the overall swarm size will be smaller than it was before the pinpointing 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 causing the swarm to become too small.

IV. LOCATING TARGETS

During the pinpointing 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" above). The agents then use a least-squares algorithm to give an estimate $\bar{y}$ of where the target is located via

$$\bar{y} = \min_y \sum_{k=1}^{N'} \left[g(|y - x(t_k)|) - f(t_k)\right]^2 \ , \tag{21}$$

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

Solving this least square minimization is quite straightforward, but this technique requires certain assumptions. First, it assumes that the form of $g(r)$ is know by the agents. For certain classes of targets and scalar fields, we believe this assumption is fair. Second, it supposes 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 task at hand, other methods such as gradient estimation could be employed.

If the estimated position of the target stabilizes, 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. We modify (1) to read

$$s_i(t_k) = \sum_{j=1}^{M} g(|y_j - x_i(t_k)|) + n_i(t_k) - \sum_{j=1}^{M'} g(|\bar{y}_j - x_i(t_k)|) \ , \tag{22}$$

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

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

V. PERFORMANCE

Two main criterion for evaluation of our algorithm are efficiency and accuracy. For the algorithm proposed here, the two criterion are roughly determined by two different phases: efficiency is mainly related to the swarming phase, while accuracy is mainly related to the pinpointing phase. To evaluate the performance of the algorithm numerically, we use the following values: the average time needed for the group to pinpoint one target (average time), the average distance between the located targets and their estimated 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.

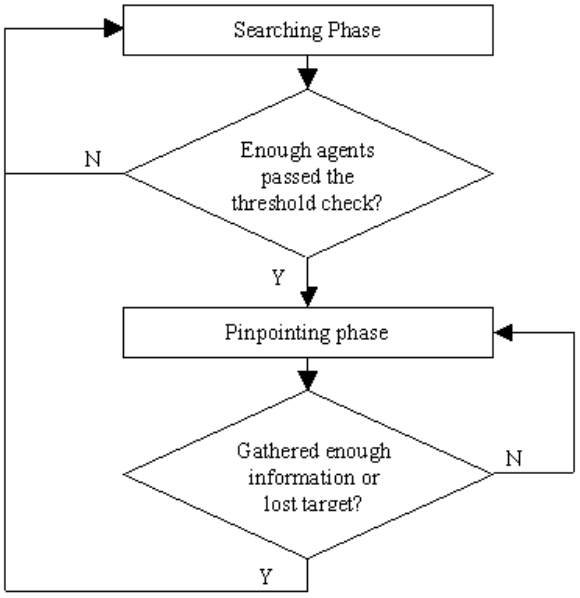

Fig. 3. A simple flowchart of the algorithm.

We run 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 the peak signal-to-noise ratio being about 10.5dB. We then run two different cases. In the first case, there are 20 targets and we restrict the duration of the simulation with the main goal of measuring efficiency. In the second case, we distribute only 5 targets on the board and use a much longer time limit with the main goal of measuring accuracy. In either case, the simulation ends when time runs out or when all targets are found. For each case, we do 100 trials and calculate the average of the statistics.

Since we may have multiple groups, it is important to decide how they cooperate with one another. We try two different policies here. 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 subregion, remaining within that area the entire time. The other policy allows all groups to search the entire region independently. In the results, we denote the use of divide-and-conquer tactics with an asterisk.

Another important factor in these measures is how many swarms we divide the agents into, or equivalently, the size of each swarm. We therefore present the results for various subdivisions. The final results are listed in Tables I and II, with their associated plots presented in Figs. 4 and 5.

From the results we can see that the size of the swarms works as a balance between accuracy and efficiency. As we may have guessed, using larger swarms makes the results more accurate, while using multiple swarms makes the searching more efficient. To have an acceptable error and false register rate, groups of at least four agents should be used. This is perhaps due to the fact that at least three agents

TABLE I

CASE 1: 20 TARGETS, TIME LIMIT 50.0. ASTERISKS DENOTE THE USE OF DIVIDE-AND-CONQUER TACTICS.

#Swarms	Size	Average time	Average error	False register
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%

TABLE II

CASE 2: 5 TARGETS, TIME LIMIT 200.0. ASTERISKS DENOTE THE USE OF DIVIDE-AND-CONQUER TACTICS.

#Swarms	Size	Average time	Average error	False register
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%

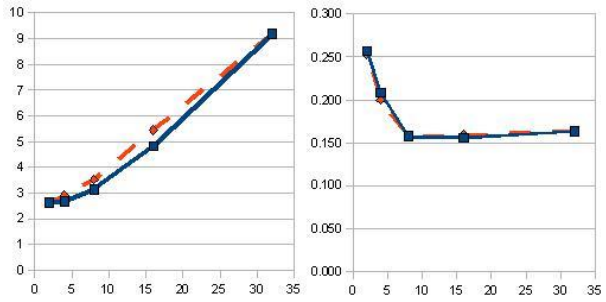

Fig. 4. Average search time (left) and average error (right) for case 1: 20 targets, time limit 50.0. The blue line is for the divide-and-conquer method, and the red dashed line is the result without divide-and-conquer.

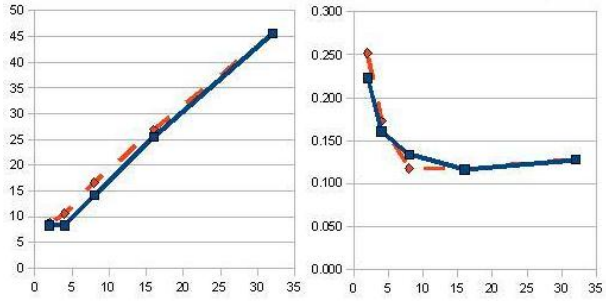

Fig. 5. Average search time (left) and average error (right) for case 2: 5 targets, time limit 200.0. The blue line is for the divide-and-conquer method, and the red dashed line is the result without divide-and-conquer.

are needed to locate a target using triangulation. Also, divide-and-conquer tactics seem to work well for this searching scenario.

VI. CONCLUSION AND FUTURE RESEARCH

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.

There are many openings for future research in this area. First, we can use alternative methods in some parts of the algorithm. A potential change is to use a compressed sensing method [17] for target locating, which enables us to find multiple targets at the same time. Another interesting modification is to use an anisotropic Levy search [15] and take previously covered paths into account. Also, different scenarios can also be evaluated, which might lead to different results for accuracy and efficiency, or even suggest the use of new algorithms. For example, we can extend the 2D problem to 3D, as would be the case for underwater searches. Or, perhaps a model for the detected signal is unknown, in which case we will need different formulations to estimate the target location. Finally, apart from numerical simulations, we also plan to do experiments on a testbed where we have small robotic vehicles as agents. This will provide an actual evaluation of the algorithm in the presence of real sensor noise, which may not be Gaussian in nature.

VII. 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

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

[2] Y.L. Chuang, Y.R. Huang, M.R. D'Orsogna, and A.L. Bertozzi, Multi-vehicle flocking: Scalability of cooperative control algorithms using pairwise potentials, *IEEE International Conference on Robotics and Automation* 2292-2299 (2007).

[3] E. W. Justh and P. S. Krishnaprasad, Equilibria and steering laws for planar formations, *Systems & Control Letters*, 52(1):25-38 (2004).

[4] T. Vicsek, A. Czirok, E.B. Jacob, I. Cohen, and O. Shochet, Novel Type of Phase Transition in a System of Self-Driven Particles, *Physics Review Letter*, 75(6):1226-1229 (1995).

[5] Z. Jin and A.L. Bertozzi, Environmental Boundary Tracking and Estimation Using Multiple Autonomous Vehicles, *IEEE Conference on Decision and Control*, 4918-4923 (2007).

[6] R. Sepulchre, D.A. Paley and N.E. Leonard, Stabilization of Planar Collective Motion with Limited Communication, *IEEE Transactions on Automatic Control*, 53(3):706-719 (2008).

[7] E. S. Page, Continuous inspection schemes, *Biometrika*, 41(1-2):100-115 (1954).

[8] B.Q. Nguyen, Y.L. Chuang, D. Tung, C. Hsieh, Z. Jin, L. Shi, D. Marthaler, A.L. Bertozzi, and R.M. Murray, Virtual Attractive-Repulsive Potentials for Cooperative Control of Second Order Dynamic Vehicles on the Caltech MVWT, *American Control Conference, Proceedings of the 2005*, 1084-1089 (2005).

[9] K.K. Leung, C.H. Hsieh, Y.R. Huang, A. Joshi, V. Voroninski and A.L. Bertozzi, A Second Generation Micro-vehicle Testbed for Cooperative Control and Sensing Strategies, *American Control Conference, Proceedings of the 2007*, 1900-1907 (2007).

[10] S. D. Bopardikar, F. Bullo, and J. P. Hespanha, A Cooperative Homicidal Chauffeur Game, *IEEE Conference on Decision and Control*, 4857-4862 (2007).

[11] S. L. Smith, and F. Bullo, The Dynamic Team Forming Problem: Throughput and Delay for Unbiased Policies, *Systems & Control Letters*, 2008, Submitted.

[12] C. Gao, J. Cortes, and F. Bullo, Notes on Averaging over Acyclic Digraphs and Discrete Coverage Control, *IEEE Conference on Decision and Control*, 44(8):2120-2127 (2008).

[13] F. Bullo and J. Cortes, Adaptive and Distributed Coordination Algorithms for Mobile Sensing Networks, *Lecture Notes in Control and Information Sciences*, Springer Verlag, 2005

[14] I.D. Couzin, J. Krause, R. James, G.D. Ruxton, N.R. Franks, Collective Memory and Spatial Sorting in Animal Groups, *J theor. Biol.*, 218(1):1-11 (2007).

[15] D. Marthaler, A.L. Bertozzi, and I.B. Schwartz, Levy Searches Based on a priori Information: The Biased Levy Walk, UCLA Center for Applied Mathematics Report 04-50, 2004.

[16] G.M. Viswanathan, S.V. Buldyrev, S. Havlin, M.G.E. da Luzk, E.P. Raposok, and E. Stanley, Optimizing the success of random searches, *Nature*, 401(6756):911-914 (1999)

[17] J.F. Cai, S. Osher, and Z. Shen, Linearized Bregman Iterations for Compressed Sensing, *Preprint(UCLA CAM report 08-06)*, 2008.

[18] J. Travis, Do Wandering Albatrosses Care About Math?, *Science*, 318:742-743 (2007).

[19] B. Cook, D. Marthaler, C. Topaz, A.L. Bertozzi, and M. Kemp, Fractional bandwidth reacquisition algorithms for VSW-MCM, Multi-Robot Systems: From Swarms to Intelligent Automata, Volume II, 2003, Kluwer Academic Publishers, Dordrecht, A.C. Schultz et al. eds. 77-86.

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

[21] A. Chen, T. Wittman, A. Tartakovsky, and A.L. Bertozzi, Image segmentation through efficient boundary sampling, *8th international conference on Sampling Theory and Applications*, 2009, Submitted.

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

[23] R. Olfati-Saber, Distributed Tracking for Mobile Sensor Networks with Information-Driven Mobility, *Proc. of the 2007 American Control Conference*, 2007