Determining Environmental Boundaries: Asynchronous Communication and Physical Scales

Andrea L. Bertozzi[1,2], Mathieu Kemp[3], and Daniel Marthaler[1]

[1] Department of Mathematics, UCLA, Los Angeles, CA 90095
bertozzi@math.ucla.edu,daniel@math.ucla.edu
[2] Department of Mathematics and Physics, Duke University, Durham, NC 27708
[3] Nekton Research, Durham NC 27713 fruitbat@nektonresearch.com

Summary. This paper considers a recently proposed model for the self-organizing, decentralized, real-time motion planning for a swarm of homogeneous mobile robots in a stationary environment. The model allows the robots to cooperatively locate the boundary of a given environmental function in two space dimensions using a combination of sensing and communication. Starting from a partial differential equation (PDE) used in image processing for edge detection, a finite difference approximation provides the movement rules for each robot. We consider physical parameters for a specific platform of underwater vehicles. We design the algorithm to function with asynchronous communication and noisy position information. We present numerical simulations illustrating the stability and performance of this system.

1 Introduction

The incentive to substitute robots for humans is particularly strong in hazardous environments. Three areas where this substitution is successfully taking place are de-mining operations [7], military urban operations [20], and aerial surveillance [31]. A natural extension that builds on this early success is the deployment of coordinated mobile sensors. Multi-agent systems offer the promise of a leap in mapping and exploration [17], navigation control [21], formation flying [29], multiple rover planning [12], and military planning [1].

Here, we explore the problem of locating the boundary of a physical phenomenon using multiple vehicles. An important application is the monitoring of harmful algae blooms (HAB), whose impact on coastal areas has increased over the past decade [2]. One of the key obstacles to being able to forecast HABs is their vast size and rapid dynamics which renders traditional techniques ineffectual. We propose to address this using a team of mobile sensors that can move to the edge of the plume and create an image of its perimeter.

Other cooperative boundary tracking (edge detection) algorithms exist in the literature [9, 15]. In [9], the algorithm is implemented with a *static* sensor field. We

V. Kumar, N. Leonard, A.S. Morse (Eds.): Cooperative Control, LNCIS 309, pp. 25–42, 2004.

adopt a self-organizing, distributed approach for boundary tracking with a *dynamic* sensor field. Some efforts have modeled distributed systems inspired by biology [3, 6, 10, 33], fluidics [4, 8, 21, 39], and economy-based [18, 32, 34, 38] concepts. We consider a method used in image processing.

An analogous problem of locating boundaries exists in image segmentation. There, the goal is to find the edges of an object in an image. This may be accomplished with a model known as a snake [23], or energy-minimizing curve. This process simulates elastic material which can dynamically conform to object shapes in response to internal or external forces. The basic snake model is a controlled continuity spline [37] under the influence of internal forces and external constraint forces. The internal forces impose a smoothness constraint on the curve, while the external forces specify the attribute we wish the snake to locate (in this case, edges).

We assume knowledge of the robots' absolute positions (we discuss the issue of uncertainty in Section 6 and the issue of relative positions in Section 3.). Our algorithm is high level and requires only the following platform features, in addition to position estimation:

1. ability of each agent to perform a 'move to' function, to move to a specified new position on command,
2. ability of each agent to obtain position information about other agents,
3. a sensor for determining environmental concentration at the agent's location and a method for estimating the local gradient of the concentration (either through a combination of local motion and sensing or from information received by other agents).

Function 3 is perhaps the most difficult although approaches have been proposed and studied for both groups of agents and single agents using bio-inspiration [5, 19, 28, 30].

The algorithm is relevant for both holonomic and nonholonomic vehicles provided that they can perform Function 1 over a sufficiently large scale. That is, the typical length-scale of the environment is large compared with the typical length-scale on which range of motion is constrained. Interesting points for further study are cases where this assumption breaks down (confined areas, massively high numbers of robots, etc.). We do explicitly address one motion issue in the algorithm: the inability to hover which is quite common in both underwater and air vehicles. Specifically, we add a component to the motion that is along a level curve of the environmental concentration, so that when the group reaches the boundary they do not have to hover, they can continuously travel around the boundary in either a clockwise or counter-clockwise direction.

2 Modeling

2.1 Phenomenon scale

Although the approach we propose is not limited to HAB monitoring, this problem is an excellent case-study for clarifying key algorithmic issues. The objective is to characterize the shape, content, and evolution of an environmental phenomenon. The scales of interest are size (1km for HAB) and time scale (1hr) [2]. To characterize the phenomenon, we use a team of autonomous underwater vehicles (AUVs), each

equipped with an appropriate chemical sensor plus navigation sensors, and communication hardware. The relevant vehicle parameters are AUV speed, navigation accuracy, and communication rate [13].

2.2 Hardware

The specifics of sensing, navigation, and communication methods are often neglected in the design of multi-agent algorithms. This type of encapsulation is not always appropriate, and for plume monitoring these issues are in fact core concerns.

- **Communication.**
  AUVs use one of two communication methods: Radio (RF) above the horizon, or acoustically while submerged. RF has a long communication range but it is only available at the surface. Acoustic communication is always available, but it has a short range (500m). Also, it is noisy, it has low bandwidth (100bps), and the bandwidth must be shared between vehicles. For HAB monitoring, the range of the phenomenon and the subsurface nature of the plume dictates the use of RF.
  There are two important consequences to using RF: The first is that the availability of communication is episodic. In contrast to, say, a telephone line, one cannot assume that a vehicle can transmit whenever it needs to. To transmit, a vehicle must first surface. The second consequence is that communication is asynchronous. The variance of surfacing time is large for a single vehicle; for multiple vehicles, the effect is compounded. Vehicles can therefore not be expected to be at the surface at the same time.
- **Navigation.**
  A vehicle's position is never known with certainty. With GPS for example, the (non-WAAS) positional accuracy is 25m RMS. Algorithms that rely on gradient estimation must incorporate this early into the design.
- **Sensing.**
  As with positional sensors, environmental sensors have limited resolution and dynamical range. This is a limiting factor when the concentration is smaller than the resolution or when the signal is larger than the sensor range, since in these cases the environment appears locally homogeneous.

We formulate a dimensionless model based on parameters from the Ranger AUV [13].

The relevant scales for the problem are as follows: the robots travel with a characteristic speed of 1 m/s. The typical radius of a plume boundary is roughly 1km. The position estimation is done with standard GPS which has an error of approximately 25m. Communication is performed with radio which happens whenever a vehicle comes to the surface at intervals of 30 seconds or more.

2.3 Dimensionless parameters

For the model we consider the following characteristics dimensions: the characteristic velocity $\bar{V}$ is 1 m/s and the characteristic length $\bar{L}$ 1 km. This gives a characteristic macro-timescale of 1000 seconds which we denote by $\bar{t}$. There is another timescale in the problem which is the time between surfacing. We denote this timescale by $\bar{\tau}$.

We can now formulate a dimensionless version of the problem. We denote the dimensionless parameters as follows: $\tilde{x} = x/\bar{L}$, $\tilde{v} = \bar{v}\bar{t}/\bar{L}$, $\tilde{t} = t/\bar{t}$, and $\tilde{\tau} = \bar{\tau}/\bar{t}$, where x is the spatial position. This means that in the dimensionless model, robots move with unit speed to find a plume of unit length diameter. They can surface after 0.03 dimensionless time units. There is also a micro-timestep (δt) that denotes the interval at which the agent senses the environment. In our simulations we take this to be 0.001 which is one second of real time. Finally the noise in the system due to GPS inaccuracy of position is 0.025 dimensionless spatial units.

For the remainder of this paper, we consider dimensionless units and we drop the $\tilde{}$ for simplicity of notation.

3 The Algorithm

We break the algorithm down into a series of components, each of which functionally involves moving a vehicle based on sensing information from the environment, position information, and other information relevant to the performance of the method. In the following subsections we discuss the components of the algorithm. They include an anti-collision/inflation mechanism, a component of the motion related to sensing, and a component related to communication and cooperation.

In this paper we derive these rules by first describing an instantaneous velocity field for the robots and then discussing how to implement this in practice by specifying a position to advance to at each communication step and in between the steps.

To begin, the motion is comprised of two distinct parts: sensing and communication. The sensing portion will involve an estimate of the local gradient of the environmental concentration. Such estimates have been used previously by AUVs [14, 25].

3.1 Sensing

Let $C(x, y)$ denote the concentration function of the environment at the robot's position (x, y). Consider a function $P(x, y) = f(C(x, y))$ that achieves a minimum at the boundary of the environmental concentration. One such example is $P = -(C_0 - C)^2$, where C_0 is a designated boundary concentration. Let $v = (x, y)$ denote the position vector. It is well known that motion according to the simple rule $\frac{dv}{dt} = -\nabla P$ results in a gradient descent toward a local minimum of the function P. Also, the instantaneous motion rule $\frac{dv}{dt} = \nabla^{\perp} C$ results in travel along level sets of of the concentration C. For a function P as described above, the instantaneous velocity field

$$\begin{array}{l} \frac{dx}{dt} = -\partial_x P - \omega \frac{\partial_y C}{|\nabla C|} \\ \frac{dy}{dt} = -\partial_y P + \omega \frac{\partial_x C}{|\nabla C|} \end{array} \tag{1}$$

results in a composite motion of an agent toward the boundary plus motion along level curves of the concentration function C. The constant ω determines the speed at which the agent traverses the boundary once it arrives there. The subscripts denote partial differentiation. Figure 1 shows a collection of 25 independent agents moving

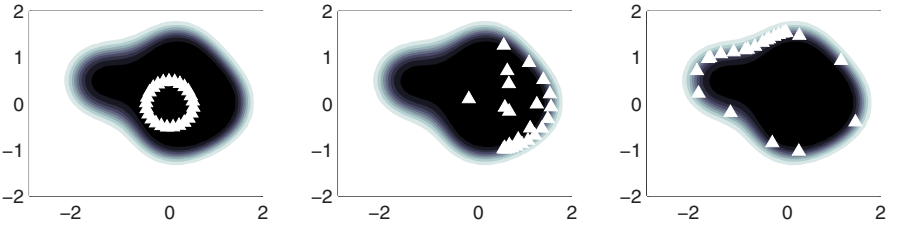

Fig. 1. 25 independent (non-communicating) agents moving to and following the boundary according to (1). Panels (left to right) show times 0, 1, and 3. Here $\omega = 2$, $C_0 = 0.01$.

according to the rule (1). They begin in a circle and independently move outward to find and follow along the boundary. In this simulation, $P = -(C - C_0)^2$,

$$C(x, y) = \tanh\left(F(x, y) - \frac{3}{4}\right) \tag{2}$$

$$F(x, y) = \sum_{i=1}^{4} \exp\left(-((x - x_i)^2 + (y - y_i)^2)/\sigma^2\right) \tag{3}$$

where $(x_i, y_i) = (1, 0), (0, -\frac{1}{2}), (-\frac{3}{2}, \frac{1}{2}), (\frac{3}{20}, 1)$, for $i = 1, 2, 3, 4$, and $\sigma = 1.0$. Because the agents do not communicate, a group of them will tend to bunch up along the boundary. Moreover this 'single-agent' method is susceptible to sensor noise [27]. Both issues can be dealt with by including interactions between agents that damp out noise and diffuse agent positions around the boundary. We introduce this through a virtual contour below.

3.2 Motion of the virtual contour

Another component of the instantaneous velocity vector is a function of the position of other nearby robots along a 'virtual contour'. This part of the algorithm is motivated from the image snake [23]. The appendix discusses the mathematical formalism behind the snake algorithm. The effect of the communication is for the robots to spread themselves out to reach all parts of the boundary. Coupling the motion (1) from Section 2.2.1 together with diffusive based communication yields the following PDE:

$$\begin{aligned} \partial_t x &= \alpha x_{ss} - \beta x_{ssss} - \partial_x P - \omega \frac{\partial_y C}{|\nabla C|} \\ \partial_t y &= \alpha y_{ss} - \beta y_{ssss} - \partial_y P + \omega \frac{\partial_x C}{|\nabla C|} \end{aligned} \tag{4}$$

where the partial derivatives with respect to s denote differentiation along the virtual contour. Here s is a smooth Lagrangian parameterization of the virtual contour. This PDE can be viewed as the rule prescribing the desired instantaneous direction in which to move the curve and hence the agents that are positioned along the curve. It is a combination of sensing and communication. The parameters α and β are constants (although they could be position dependent in a more general model) and determine the relative strengths of two kinds of diffusive motion along the contour. Equation (4) with $\omega = 0$ is precisely the basic snake algorithm from image processing (see the Appendix).

3.3 Discretization of the virtual contour

In reality, there are only a finite number of agents occupying discrete points on the virtual contour. It is therefore natural to approximate the partial derivatives with respect to s in (4) by finite differences at the robot positions. That is, approximate at the ith robot location

$$x_{ss} \sim (x_{i+1} - 2x_i + x_{i-1})/h^2, \quad x_{ssss} \sim (x_{i-2} - 4x_{i-1} + 6x_i - 4x_{i+1} + x_{i+2})/h^4 \quad (5)$$

$$y_{ss} \sim (y_{i+1} - 2y_i + y_{i-1})/h^2, \quad y_{ssss} \sim (y_{i-2} - 4y_{i-1} + 6y_i - 4y_{i+1} + y_{i+2})/h^4 \quad (6)$$

These are standard centered finite difference operators for which $h = 1/N$, N being the number of agents. Here we assume a fixed ordering for the robots around the virtual contour. We can then make this substitution in (4) to obtain an instantaneous velocity vector $(\partial_t x_i, \partial_t y_i)$ at each agent location (x_i, y_i):

$$\partial_t x_i = \frac{\alpha}{h^2}(x_{i+1} - 2x_i + x_{i-1}) - \frac{\beta}{h^4}(x_{i-2} - 4x_{i-1} + 6x_i - 4x_{i+1} + x_{i+2}) - \partial_x P(x_i, y_i) - \omega \frac{\partial_y C(x_i, y_i)}{|\nabla C(x_i, y_i)|} + \text{Sparsing}(x_i), \quad (7)$$

$$\partial_t y_i = \frac{\alpha}{h^2}(y_{i+1} - 2y_i + y_{i-1}) - \frac{\beta}{h^4}(y_{i-2} - 4y_{i-1} + 6y_i - 4y_{i+1} + y_{i+2}) - \partial_y P(x_i, y_i) + \omega \frac{\partial_x C(x_i, y_i)}{|\nabla C(x_i, y_i)|} + \text{Sparsing}(y_i). \quad (8)$$

Note that there is an additional part of the motion which we denote by "Sparsing". This is also done on the communication step and has two functional roles: (1) providing the vehicles with an anti-collision mechanism and (2) given the group a virtual 'inflationary force' in order for the contour to expand outward in the case of constant concentration (i.e. starting from a point inside the plume). The coupled system (7-8) of ordinary differential equations is very stiff. This leads to restrictions on how frequently and how far each vehicle must communicate with other vehicles in order to result in stable collective motion. In previous papers [27, 26] we reviewed two different communication scenarios, one in which synchronous communication is performed less frequently across the entire grouping and one in which only near neighbor communication is performed rather frequently. In this manuscript we consider a different option, that of asynchronous communication involving a central command post.

Note that (7-8) is a coupled set of ODEs for the positions of the agents; such ODE-based motion rules arise in control theory models for cooperative motion of robots for formation flying [22] and coordinated control of groups [24].

4 Discrete time steps

In practice, we can not prescribe an instantaneous cooperative velocity rule without requiring instantaneous and constant communication between the agents. Instead we propose an algorithm in which communication occurs at discrete time intervals spaced apart by a time-step τ. We have modified our previous synchronous algorithm to perform asynchronously. The basic idea is that the motion planning steps that

require communication are only updated as each agent surfaces. At this time, a new velocity vector is prescribed that incorporates the communication step of the algorithm. This velocity vector is incorporated into the sensing part of the algorithm that is updated continuously while the vehicle is underwater.

4.1 Temporal operator splitting in numerical analysis

A well-known method in computational science for time stepping the system (7-8) is called operator splitting. The basic idea is as follows: given a differential equation of the general form

$$u_t = F(u) + G(u)$$

where F and G are functionals (possibly involving differential or integral operators), the solution can be approximated by alternating small time steps of the separate equations

$$u_t = F(u) \quad and \quad u_t = G(u).$$

The simplest form of operator splitting is the Trotter product formula [36] which computes identical timesteps of each equation. This method is typically first order in time, meaning that the difference between the time stepped solution and the solution of the continuous equation scales like $O(\Delta t)$. A more complicated form of time-stepping is Strang splitting [35] which involves an initial half step of the first equation followed by alternating full steps of each with a final half step of the first equation. Strang splitting is second order in time.

In previous papers we considered simple Trotter splitting as follows: Replace the time derivative of the position $v(t)$, denoted $\partial_t v(t)$ by a finite difference

$$\partial_t v \approx \frac{v^{k+1} - v^k}{\Delta t}. \tag{9}$$

where the $k+1$ superscript denotes the position at the next time.

If we let $v_i(t) = (x_i, y_i)$ denote the spatial coordinates of the ith robot, then the result is a 'split-step' discrete-time implementation in which the communication dynamics is advanced over one step, followed by the sensing on a second step.

- STEP 1, (Communicate)

$$v_i^{n+1/2} = v_i^n + \Delta t \left(\frac{\alpha}{h^2} (v_{i+1} - 2v_i + v_{i-1}) - \frac{\beta}{h^4} (v_{i-2} - 4v_{i-1} + 6v_i - 4v_{i+1} + v_{i+2}) \right). \tag{10}$$

- STEP 2, (Use sensors and do not hit anything)

$$v_i^{n+1} = v_i^{n+1/2} + \Delta t \left(-\nabla P(v_i^n) + \omega \frac{\nabla^\perp C(v_i^n)}{|\nabla C(v_i^n)|} + \text{Sparsing}(v_i^n) \right). \tag{11}$$

The superscript $n+1$ denotes the position at the next time. We use $n+1/2$ to denote the intermediate time for the split step. The choice of splitting serves to illustrate two basic alternatives for the motion, based on ideas from numerical analysis for which we have well developed theory.

In [27, 26], Steps 1 and 2 were alternated with a fixed time step Δt, resulting in convergence to a hybrid dynamics in the limit as $\Delta t \to 0$. Two different forms of the communication step were considered.

Near Neighbor Communication (Explicit Update)

Using an explicit update, step 1 can be solved by the explicit formula

$$v_i^{n+1/2} = v_i^n + \Delta t \left(\frac{\alpha}{h^2} \left(v_{i+1}^n - 2v_i^n + v_{i-1}^n \right) - \frac{\beta}{h^4} \left(v_{i-2}^n - 4v_{i-1}^n + 6v_i^n - 4v_{i+1}^n + v_{i+2}^n \right) \right). \tag{12}$$

Note that robot i only needs to know the positions of robots $i-2$, $i-1$, $i+1$, and $i+2$ in addition to its own position, in order to move accordingly. Thus, this implementation requires only local (near-neighbor) communication between robots. However, there is a significant drawback to using this method. Stability of step 1 requires that the time step $\Delta t < h^4/(8\beta + 2\alpha h^2)$ [27], in particular as the number of robots N increases, the algorithm must be updated on a timescale Δt that decreases as $1/N^4$ for N large. This results in an overall algorithm in that requires $O(N^5)$ computations as the number of robots increases. The number of communications required scales like $O(1)$ per robot at each time-step.

All-to-all Communication (Implicit Update)

An implicit update can be used for solving step 1:

$$\begin{aligned} v_i^{n+1/2} = v_i^n + \Delta t \Big(& \frac{\alpha}{h^2} \left(v_{i+1}^{n+1/2} - 2v_i^{n+1/2} + v_{i-1}^{n+1/2} \right) \\ & - \frac{\beta}{h^4} \left(v_{i-2}^{n+1/2} - 4v_{i-1}^{n+1/2} + 6v_i^{n+1/2} - 4v_{i+1}^{n+1/2} + v_{i+2}^{n+1/2} \right) \Big). \end{aligned} \tag{13}$$

Note that in this situation, the new positions v_i^{n+1} must be determined implicitly by solving the coupled systems of linear equations above. This requires each robot to perform a linear algebra computation, which is $O(N)$ in complexity because the matrix is pentadiagonal (e.g. a standard LU decomposition algorithm is very straightforward to use). However, in order for each robot to solve for their next position, they must know the positions of *all* the other robots in order to solve the implicit equations. The benefit of this method is that it is unconditionally stable for large time steps Δt [27] independent of the number of agents. Thus the total computational complexity scales as $O(N)$ for N agents. Note that the above step necessarily requires synchronous communication.

4.2 Asynchronous Communication

In this paper we consider a platform that uses asynchronous communication. Thus we must adapt the above synchronous algorithm. For asynchronous communication, we again base our algorithm on ideas from operator splitting in numerical analysis, however we have to work with constraints that do not commonly arise in numerical solution of PDEs. These constraints are as follows: for the communication part of the algorithm, when a vehicle surfaces, it gets a GPS read on its own position, however the data regarding positions of other vehicles is time-lagged back to the last time each vehicle surfaced. Thus, we can not directly perform the step in (4.1) above. Moreover there is another constraint that the timescale τ in between surfacing is

much larger than the timescale on which the sensors may be sampling the environment (and thus on which the sensing part of the algorithm could be updated). Thus we need to consider an algorithm with two different timescales and asynchronous communication.

Our approach is as follows. Motivated by operator splitting, we consider a multistep method in which each vehicle continuously moves according to a sensing algorithm of the form

$$(v_i)_t = -\nabla P(v_i) + \omega \frac{\nabla^\perp C(v_i^n)}{|\nabla C(v_i^n)|} + \mathrm{Comm}(v_i, t_n), \quad t \in \left[t_n^i, t_n^i + \tilde{\tau}\right] \tag{14}$$

where t_n is the nth surfacing time for vehicle i and $\tilde{\tau}$ is the surfacing time interval. The additional velocity vector $\mathrm{Comm}(v_i, t_n)$ is what arises from a semi-implicit timestep (of size τ) for solution of the split part of the PDE, namely the parts that require communication.

In order to complete the algorithm, we have to define $\mathrm{Comm}(v_i, t_n)$. When a vehicle surfaces, it only has information about other vehicle positions at previous surfacing times. This is enough to make an extrapolated guess of the location of the agents at the current time. Let $\hat{v}_j(t)$ denote the guess for the jth vehicle at time t based on the previous two location positions. In this case we make a linear extrapolation using the formula

$$\hat{v}_j(t) = v(t_{n-1}) + \frac{t - t_{n-1}}{t_{n-1} - t_{n-2}}(v(t_{n-1}) - v(t_{n-2}))$$

where for simplicity of notation t_{n-1} and t_{n-2} denote the previous two surfacing times for the jth vehicle.

Now we can compute a change in the position of vehicle v_i due to the communication terms in the PDE over the timestep τ. Note that for the Ranger platform, surfacing intervals force τ to be greater than 0.03 which means that we have to evaluate the stiff part of the problem implicitly. This is done by solving the implicit system of equations

$$\begin{aligned} v_i^{new} = \hat{v}_i + \tau \Big(& \frac{\alpha}{h^2}(v_{i+1}^{new} - 2v_i^{new} + v_{i-1}^{new}) \\ & - \frac{\beta}{h^4}(v_{i-2}^{new} - 4v_{i-1}^{new} + 6v_i^{new} - 4v_{i+1}^{new} + v_{i+2}^{new}) \Big) + \tau \left(S(v_i^{old}) \right). \end{aligned} \tag{15}$$

where S denotes the sparsing term previously implemented in Step 2.

The communication velocity $\mathrm{Comm}(v_i, n)$ used in (14) is then the differential position $D_i = v_i^{new} - v_i$ divided by the communication time $\tilde{\tau}$. The integrated steps described above are consistent with a discrete time implementation of equations (14) using asynchronous communication. In practice we believe that it will be simpler to have each robots travel at a constant speed while updating its direction vector based on sensing and communication information. The simulations described in the next section take this into account by normalizing the speed in (14) to be one.

5 Cooperative motion simulations

We now show some simulations that illustrate the algorithm using an environmental function $P = -|C - C_0|^2$. The parameters α and β were set to be .01 and .0001, respectively. The sparsing for the ith robot is a repulsive force of the form

$$\nabla_i \sum_{j=1}^{N} C_r \exp\left(-|v_i - v_j|/l_r\right), \tag{16}$$

where v_k is the two-dimensional position coordinate for robot k. The concentration function, $C(v)$, is

$$C(x,y) = \tanh\left(F(x,y) - \frac{4}{5}\right) \tag{17}$$

$$F(x,y) = \sum_{i=1}^{5} \exp\left(-((x-x_i)^2 + (y-y_i)^2)/\sigma^2\right) \tag{18}$$

where $(x_i, y_i) = (1,1), (2,-1), (-\frac{3}{2}, \frac{1}{2}), (\frac{3}{20}, 1), (\frac{3}{2}, -\frac{1}{2})$ for $i = 1, \ldots, 5$, and $\sigma = 1.0$.

5.1 Internal initialization

The N agents are dropped off from a boat in the plume. Upon insertion in the water, agent k moves at an angle $(2 * \pi/N) * k$ until all agents are inserted. When an agent surfaces, it checks to see how many other agents have surfaced. Until all are in the water, the agents continue in their initial direction. Once all agents are present, then both the communication and sensing steps are implemented, i.e. the algorithm is turned on.

The following simulations each have 25 agents. The repulsion strength from (16), C_r ,is 10 for the first .78 dimensionless time units of the simulation. After that, it drops to 1. The repulsion length, l_r, is 1. The spatial resolution is .025 and each simulation has variable surfacing times.

5.2 Effect of surfacing time

The majority of the dynamics of HAB occurs underwater. Therefore, we expect that the AUVs will spend most of their time there. They will periodically surface to get a position fix, and to implement the communication part of (14). Below, we show four simulations showing agent positions during a simulation. The only difference between the figures is the amount of time between surfacing for each agent. The surfacing times were .03, .045, .06, and .12, respectively. As one can see, the coverage of the algorithm deteriorates as the surfacing time increases. This makes sense, as it is the communication step of the algorithm that prevents behavior seen in Figure 1. So the more each agent surfaces, the less the full algorithm acts less like a single agent.

5.3 Effect of initialization

In the previous figures, the initialization procedure was to drop the agents off in the plume, have them move at a specific angle until all vehicles were inserted, then begin the algorithm. We change this procedure in the figure below to demonstrate the impact of the initial conditions on the success of the algorithm. Figure 6 has the same parameters as that of Figure 2 except the initialization procedure. Here, instead of each agent moving at a prescribed angle, each vehicle sits and waits until all AUVs are inserted. Such a scenario might take place in the case of agents being

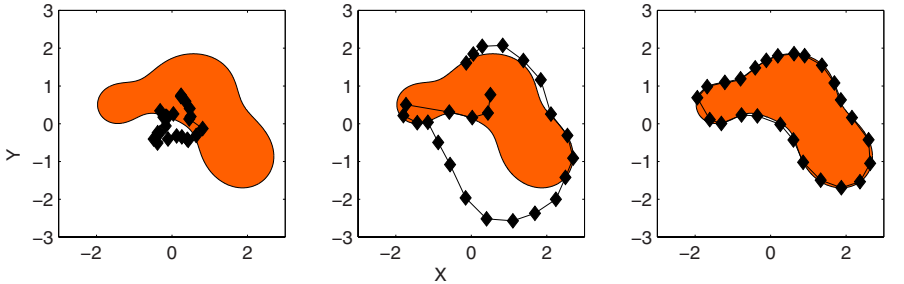

Fig. 2. 25 agents with surfacing time of .03 moving to and following the boundary according to (14). Panels (left to right) show times .14, .885, and 5.385. Here $\omega = -2$, $C_0 = 0.01$.

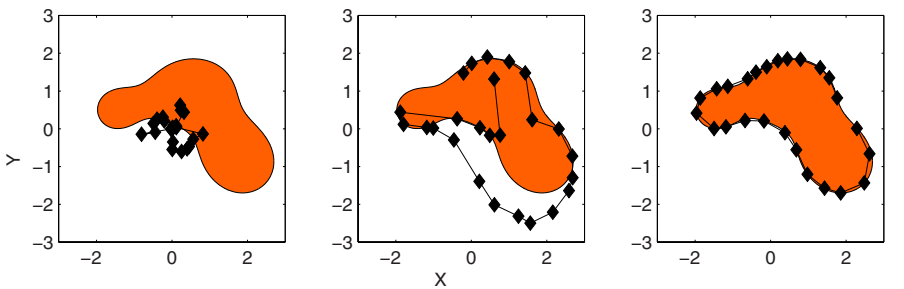

Fig. 3. 25 agents with surfacing time of .045 moving to and following the boundary according to (14). Panels (left to right) show times .14, .885, and 5.385. Here $\omega = -2$, $C_0 = 0.01$.

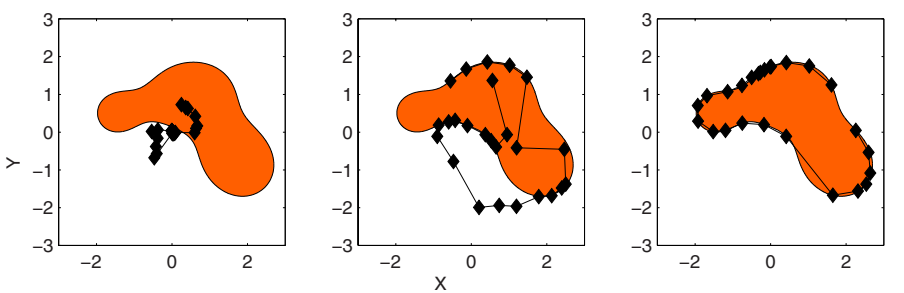

Fig. 4. 25 agents moving with surfacing time of .06 to and following the boundary according to (14). Panels (left to right) show times .14, .885, and 5.385. Here $\omega = -2$, $C_0 = 0.01$.

released from a moving platform such as a boat. The first panel of Figure 6 shows this case in which the agent positions lie on a line at the beginning of the algorithm.

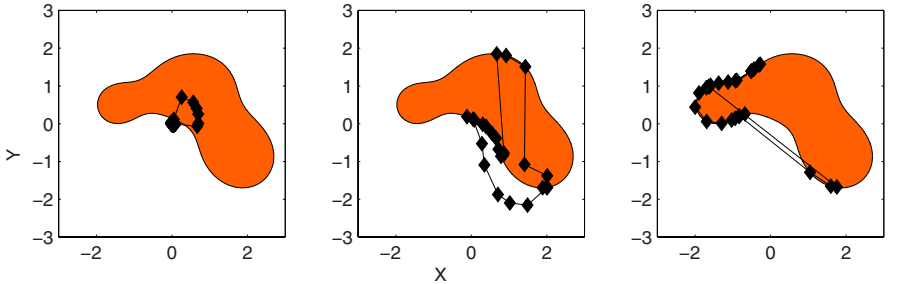

Fig. 5. 25 agents with surfacing time of .12 moving to and following the boundary according to (14). Panels (left to right) show times .14, .885, and 5.385. Here $\omega = -2$, $C_0 = 0.01$.

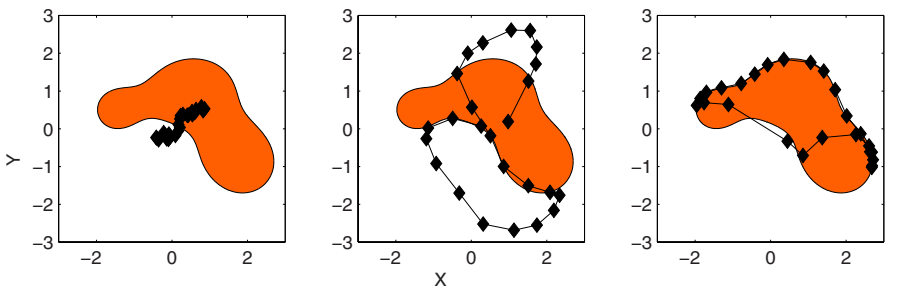

Fig. 6. 25 agents released along a line and following the boundary according to (14). The surfacing time is 0.03. Panels (left to right) show times .14, .885, and 5.385. Here $\omega = -2$, $C_0 = 0.01$.

The different initialization procedures determine the amount of inflation that the algorithm needs to function well. The more scattered the agents are initially, the less inflation they need as the algorithm progresses. Widely scattered initial conditions require less frequent surfacing time than those that are initially clustered. This is because the inflation term is implemented at the surface, so to inflate, the agents must communicate.

5.4 Effect of position noise

The following figure shows the error of the agents' positions as a function of GPS spatial resolution. The resolution was implemented in the following way: Upon surfacing, when the agent reports its position to the surface node, it reports its true position + (Noise Coefficient)*h, where h is a uniform random variable in $[-\frac{1}{2}, \frac{1}{2}]$. The error plotted is calculated in the following way: At the final timestep, each agent calculates the concentration at its point. The average of the absolute value of the difference between these values and the "true" level set value is the error. This process is repeated 100 times, and the average value is the one plotted in the figure. Barely visible are the error bars showing the variance of the error. Note that this is

a single agent error, it does not measure how well the agents are distributed along the boundary. In particular it does not differentiate between the bunched agents in Figure 5 and the well-distributed agents in Figure 2.

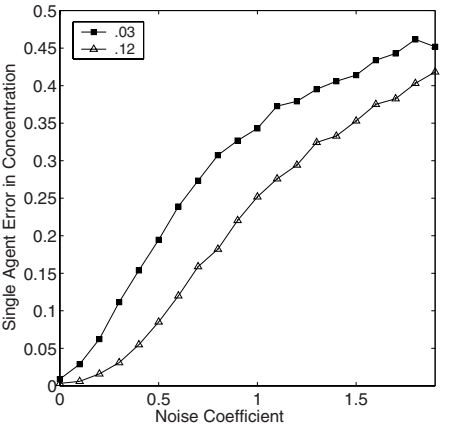

Fig. 7. The figure is a plot of the single agent boundary "errors" as measured by concentration. We compare two different surfacing times, .03 and .12, respectively. The error bars are so small as to be (almost) invisible.

As can be seen from Figure 7, the larger surfacing time interval actually has less concentration error. We believe this is related to the fact that our simulation only considers the error when an agent reports its position, therefore the less this occurs (the less an agent surfaces), the less the error. In the field, there is error in both the sensing and communication steps.

Figure 8 shows the effect of changing the relative navigational accuracy, as would be the case if the plume were smaller. The left panel has no noise, while the right panel has a noise coefficient of 1.5. The surfacing time for both simulations is .03.

6 Discussion and Future Work

The Princeton group [16] has designed a multi-vehicle solution to the related problem of tracking large-scale ocean phenomena. The elegant solution they propose is a deployment in formation and the use of this formation to extract the front gradient. The gradient is then used to direct the group towards the front.

Similar to the problem of detecting environmental boundaries, the control algorithm is derived using a continuous formulation, (ODEs there and PDEs in our case). This is most naturally formulated on a time grid, but as we also found, synchronous updating is impractical since the vehicles can only communicate while at the surface. To address this, they propose a solution similar to the one we adopted, i.e. a surfacing vehicle determines the location of the other vehicles by extrapolation. Likewise, they found that the algorithm is robust to latencies. An interesting

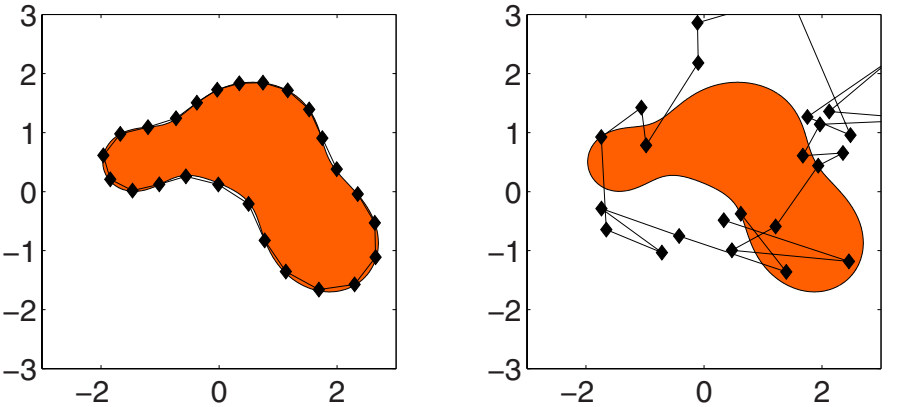

Fig. 8. The left panel shows the final positions ($t = 5.28$) with no noise. Notice that the boundary level set is tracked and "locked on." The right panel is the same simulation with an error coefficient of 1.5. In this case, the agents are widely scattered. The error, as computed in figure 7 is around 0.4, showing that this is a "large" value of error.

question is whether the robustness to latency that is found in these two problems shares a common origin.

We believe that to implement this algorithm in the field we will require a very good method for gradient estimation. Adaption of the algorithm to cooperatively use sensor data is one approach that could be combined with biomimetic motion or the motion of agent clusters. The version of the algorithm presented here only uses the sensor data for motion of the particular agent on which the sensor is located.

Although we use this method for perimeter detection, there are other closely related applications. One of the simplest and most useful extensions is for characterizing the horizontal and vertical structure of a plume. To extract the horizontal structure, the target level set is slowly varied between two extremes, enabling the construction of a topographical map. The same idea is used to get the vertical structure: the target depth gradually changes and constant depth slices can be assembled into a 3D map.

One of our observations is that the dynamics of the virtual contour may depend on the initial location of the agents, the magnitude of the inflation forces, and the snake parameters. In order to spread the vehicles out, we use an ad-hoc method where the inflation parameter is set high initially and low thereafter. Alternatively we could have changed the snake parameters. We are currently designing a more elegant method for achieving this, where the algorithm parameters are automatically set using inputs from the proximity to the perimeter and from vehicle clustering.

Regarding positional accuracy we find that the tracking performance actually improves with fewer surfacing events. This is because we only consider positional error introduced at the surfacing time through a GPS reading. In the single-agent part of the algorithm, we do not consider errors in the gradient estimation. This part of the algorithm allows the agent to converge to an environmental level-set. On the other hand, the vehicles tend to clump when they act only individually.

An effective multi-agent algorithm should meet two criteria: (1) they must work even if only one agent remains and (2) the communication should accelerate performance. Here we clearly meet the first criterion. The second criteria is also met when we consider additional sensor error that will naturally be introduced in the field. In a previous paper [27] we show that errors arising from sensor readings are significantly reduced by the multi-agent method over the single agent method. This is because the diffusive nature the virtual contour equation damps out high frequency spatial noise very effectively.

Acknowledgments

This research is supported by ARO STTR grant DAAD19-03-C-0075, ARO grant DAAD19-02-1-0055, and ONR grant N000140310073.

Appendix: Energy Minimizing Curves in Image Processing

We review the theory of [11]: The deformable contour model is a mapping

$$\Omega = \mathbb{S}^1 \to \mathbb{R}^2, \quad s \mapsto v(s) = (x(s), y(s)) \tag{19}$$

where $\mathbb{S}^1$ is the periodic unit interval. For the purpose of this paper, we assume periodic boundary conditions; other boundary conditions are possible and used in image processing problems [23]. A deformable model is defined to be a space of admissible deformations F and functional E to minimize. This functional represents the energy of the model and has the following form:

$$E : F \to \mathbb{R} \tag{20}$$

$$E(v) = \int_\Omega \left[w_1 |v_s(s)|^2 + w_2 |v_{ss}(s)|^2 + P(v(s)) \right] ds \tag{21}$$

where the subscripts denote differentiation with respect to the Lagrangian parameter s, and P is the potential function associated with the environment. In imaging, P is a function of the image data. For example, if we want the snake to be attracted to boundaries, then P would depend upon the gradient of the image intensity. The mechanical properties of the contour are specified by the functions w_j. Their choice determine the elasticity and rigidity of the curve.

If v is a local minimum of E, it satisfies the associated Euler-Lagrange equation:

$$-\partial_s(w_1 v_s) + \partial_{ss}(w_2 v_{ss}) + \nabla P = 0 \tag{22}$$

with periodic boundary conditions.

Equation (22) can be interpreted physically as a mechanical balance.

- The first two terms of equation (22) impose the regularity of the curve. The functions w_1 and w_2 impose elasticity and rigidity of the interpolated curve through the agents.

- In image segmentation, one chooses P as

$$P(v) = -|\nabla I(v)|^2 \tag{23}$$

where I denotes the image intensity function. This choice causes the contour to be attracted to sharp gradients, i.e. boundaries in the image. A thorough treatment of the relationship between potential functions and desired goals is discussed in [23].

One way to achieve a solution to equation (22) is to perform a time dependent gradient flow of the energy E,

$$\boxed{\partial_t v = \partial_s(w_1 v_s) - \partial_{ss}(w_2 v_{ss}) - \nabla P} \tag{24}$$

with steady state solving (22). Equation (24) is the motivation for the boundary tracking feature of the mobile agent algorithm developed in this paper.

References

1. See for example the documents on future naval combat capabilities, downloadable from http://www.onr.navy.mil/sci_tech/engineering /334_shiphull/other/fnc_support.htm.
2. http://www.cbl.umces.edu/Research/data-flow.html.
3. R.C. Arkin. Cooperation without communication: Multi-agent schema-based robot na vigation. *Journal of Robotic Systems*, 9(3):351–364, 1992.
4. K. Bohringer, R. Brown, B. Donald, J. Jennings, and D. Rus. Distributed robotic manipulation: Experiments in minimalism. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*, 1995.
5. C.J. Brokaw. Chemotaxis of bracken spermatozoids. In *Ph.D. thesis. Cambridge University*, 1958.
6. R.A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.
7. R. Cassinis. Landmine detection methods using swarms of simple robots. In E. Pagello, editor, *International Conference on Intelligent Autonomous Systems*, volume 6, Venice, Italy, 2000.
8. D. Chevallier and S. Payandeh. On kinematic geometry of multi-agent manipulating system based on the contact force information. In *The 6th International Conference on Intelligent Autonomous Systems (IAS-6)*, pages 188–195, 2000.
9. K. Chintalapudi and R. Govindan. Localized edge detection in sensor fields. *Ad-hoc Networks Journal*, 2003.
10. V. Cicirello and S. Smith. Insect societies and manufacturing, 2001. The IJCAI-01 Workshop on Artificial Intelligence and Manufacturing: New AI Paradigms for Manufacturing.
11. L. Cohen. On active contour models and balloons. *Comput. Vision, Graphics, and Image Processing: Image Understanding*, 53(2), 1991.
12. T. Estlin, G. Rabideau, D. Mutz, and S. Chien. using continuous planning techniques to coordinate multiple rovers. *Elec. Trans. on AI*, 4:45–57, 2000.
13. B. Hobson et al. Field results of multiple autonomous underwater vehicle deployments. In *Proceedings of the 13th Int. Symp. on Unmanned Untethered Submersible Technology*, 2003.

14. E. Burian et. al. Gradient search with autonomous underwater vehicles using scalar measurements. In *Proceedings of IEEE OES AUV Conference*, 1996.
15. J.T. Feddema et al. Analysis and control software for distributed cooperative systems. *Proceedings of DARPA ITO Conference on Software for Distributed Robotics*, 2001.
16. Edward Fiorelli, Pradeep Bhatta, Naomi Ehrich Leonard, and Igor Shulman. Adaptive sampling using feedback control of an autonomous underwater glider fleet, 2003. Proceedings of the 13th Int. Symp. on Unmanned untethered submersible technology.
17. D. Fox, W. Burgard, H. Kruppa, and S. Thrun. Collaborative multi-robot exploration. *Autonomous Robots*, 8(3), 2000.
18. M. Golfarelli, D. Maio, and S. Rizzi. A task-swap negotiation protocol based on the contract net paradigm, 1997. Technical Report CSITE, No. 005-97.
19. H.C.Crenshaw and L. Edelstein-Keshet. Orientation by helical motion-ii. changing the direction of the axis of motion. *Bull. Math. Biol.*, 55:213–230, 1993.
20. IRobot. Packbot. http://www.packbot.com.
21. J.Desai, V.Kumar, and J.Ostrowski. Controlling formations of mobile robots. In *Proceedings of the IEEE International Conference on Robots and Automation*, 1998.
22. E.W. Justh and P.S. Krishnaprasad. A simple control law for uav formation flying, 2002. http://techreports.isr.umd.edu/TechReports/ISR/2002/TR_200 2-38/TR_2002-38.pdf.
23. M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International J. of Computer Vision*, pages 321–331, 1988.
24. N. E. Leonard and E. Fiorelli. Virtual leaders, artificial potentials and coordinated control of groups. In *Proc. 40th IEEE Conf. Decision and Control*, pages 2968–2973, 2001.
25. N.E. Leonard. Schooling by design: Coordinated multi-vehicle dynamics. plenary lecture, siam conference on applications of dynamical systems, 2003.
26. D. Marthaler and A.L. Bertozzi. Collective motion algorithms for determining environmental boundaries, 2003. UCLA CAM Report 03-17, http://www.math.ucla.edu/applied/cam/index.html.
27. D. Marthaler and A.L. Bertozzi. Tracking environmental level sets with autonomous vehicles. In S. Butenko, R. Murphey, and P.M. Pardalos, editors, *Recent Developments in Cooperative Control and Optimization.* Kluwer Academic Publishers, 2003.
28. R.S. Payne. Acoustic location of prey by barn owl. *Journal of Experimental Biology*, 54:535, 1971.
29. P.Ogren, E.Fiorelli, and N.Leonard. Formations with a mission: Stable coordination of vehicle group maneuvers. In *Proc. 15th Int'l Symposium on Math. Theory of Networks and Systems*, 2002.
30. M. Poteser, M. Pabst, and K. Kral. Proprioceptive contribution to distance estimation by motion parallax in a praying mantis. *Journal of Experimental Biology*, 201:1483, 1998.
31. UAV USAF Predator. http://www.af.mil/news/factsheets/ RQ_1_Predator_Unmanned_Aerial.html.
32. T. Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings, Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pages 256–262, 1993.

33. M. Schneider-Fontan and M.J. Mataric. A study of territoriality: The role of critical mass in adaptive task division. In *Proceedings from Animals to Animats 4, Fourth International Conference on Simulation of Adaptive Behavior (SAB-96)*, pages 553–561. MIT Press/Bradford Books, 1996.
34. A. Stentz and M.B. Diaz. A free market architecture for coordinating multiple robots, 1999. CMU-RI-TR-99-42, Robotics Institute, Carnegie Mellon University.
35. Gilbert Strang. On the construction and comparison of difference schemes. *S.I.A.M. J. Numer. Anal.*, 5(3), September 1968.
36. Michael E. Taylor. *Partial Differential Equations III.* Springer-Verlag, 1996.
37. D. Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Transactions PAMI*, 8, 1986.
38. M.P. Wellman and P.R. Wurman. Market aware agents for a multi-agent world. *Robotics and Autonomous Systems*, 24:115–125, 1998.
39. J. Zhang, M. Ferch, and A. Knoll. Carrying heavy objects by multiple manipulator s with self-adapting force control. In *The 6th International Conference on Intelligent Autonomous Systems (IAS-6)*, pages 204–211, 2000.