# Collective Motion Algorithms for Determining Environmental Boundaries

Daniel Marthaler* and Andrea L. Bertozzi†

*Department of Mathematics and*

*Center for Nonlinear and Complex Systems*

*Duke University, Durham, NC 27708*

April 2, 2003

**Abstract**

This paper develops a mobile robot path planning method for a swarm of homogeneous agents in a stationary environment. The agents cooperatively locate the boundary of a given environmental function in two space dimensions. Applications include tracking oil slicks, algae blooms, and other environmental indicators that occupy a localized area with a well defined boundary. Individual agents are controlled by local rules based on "snake" algorithms from image segmentation. Starting from a partial differential equation (PDE) that arises from a gradient flow energy minimization, a finite difference approximation provides the movement rules for each agent. Each node in the discretization corresponds to an agent in the environment. The communication between agents may be either local or global and the stability of the algorithm using different communication rules is discussed. We obtain bounds on the frequency of communication required for stable implementation of the algorithm using only near neighbor interaction, and also discuss the stability of the algorithm in the case of global communication. We also show through numerical simulations that the cooperative algorithm is robust to failure of some of the agents.

*daniel@math.duke.edu

†bertozzi@math.duke.edu, www.math.duke.edu/faculty/bertozzi

# 1 Introduction

Multi-agent systems have the potential to provide flexible and efficient alternatives to dangerous operations currently performed by humans. De-mining operations [8], mapping and exploration [7], navigation control [1], formation flying [2], and multiple rover planning [3] are a few such examples. We propose a method for a multi-agent system of autonomous vehicles (agents) to perform the exploration of a static environment. The goal of the exploration is for the agents to locate the boundary of an environmental concentration (say an algae bloom or an oil spill). We assume that the agents are equipped with sensors that enable them to determine the local gradient of the concentration. Also, we assume that each agent communicates with other agents in a reliable manner. The frequency and range of communication are two parameters considered.

An analogous problem exists in image segmentation. There, the goal is to find the boundary of an object. This may be accomplished with a model known as a snake [6], 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 [15] under the influence of internal forces and external constraint forces. The internal forces serve to impose a smoothness constraint on the curve, while the external forces allow specification of the type of attribute we wish the snake to locate (in this case, edges).

This paper shows that the snake algorithm from image processing can be adapted for use by autonomous mobile agents to find boundaries. The agents move along a virtual contour following simple rules whose overall dynamics result in convergence onto the boundary of an environmental function. As in the image problem, we use a partial differential equation (PDE) as the main motivation for the boundary tracking part of the algorithm. The snake algorithm moves a curve onto an edge; in our problem, the goal is to move a set of discrete agents onto a boundary. Snake segmentation algorithms have been used for tracking of miniature robots [12]. The method presented here is, to our knowledge, the first use of a deformable model for collective *motion* of vehicles.

The paper is structured as follows: We first review deformable models. Next we present the full motion generating algorithm and new features not present in the image segmentation algorithm. Then, example simulations are presented demonstrating performance. We discuss the concept of agent failure and how the algorithm dynamically reconfigures its state in the presence of agent loss. Finally, future directions are proposed.

## 2 Energy Minimizing Curves in Image Processing

We review the theory of [5]: The deformable contour model is a mapping

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

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 [6]. 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:

$$\begin{aligned} E &: F \to \mathbb{R} \\ E(v) &= \int_\Omega \left[ w_1 |v_s(s)|^2 + w_2 |v_{ss}(s)|^2 + P(v(s)) \right] ds \end{aligned}$$

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:

$$-\partial_s(w_1 v_s) + \partial_{ss}(w_2 v_{ss}) + \nabla P = 0 \tag{1}$$

with periodic boundary conditions.

Equation (1) can be interpreted physically as a mechanical balance.

- The first two terms of equation (1) 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

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

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 [6].

One way to achieve a solution to equation (1) is to preform a time dependent gradient flow of the energy $E$,

$$\boxed{\partial_t v = \partial_s(w_1 v_s) - \partial_{ss}(w_2 v_{ss}) - \nabla P} \tag{3}$$

with steady state solving (1). Equation (3) is the motivation for the boundary tracking feature of the mobile agent algorithm developed in this paper.

# 3 Agent Based Motion via "Virtual" Contours

Equation (3) suggests a starting point for a mobile agent algorithm. Additional dynamics will address such properties as continual self-propulsion of agents, collision avoidance, and sparsing of the group. Self-propulsion is relevant for mobile agents that lack "hovering" capability, such as torpedo AUVs [16]. The connection between the agent motion algorithm and the deformable contour is that the agents are thought to occupy discrete points along a "virtual" contour.

We construct the algorithm in stages. Denote the two coordinates of the position vector $v$ as $v(s) = (x(s), y(s))$, assume $w_1 = \alpha$ and $w_2 = \beta$, are both constants. Then equation (3) has components

$$\begin{aligned} \partial_t x &= \alpha x_{ss} - \beta x_{ssss} - \partial_x P, \\ \partial_t y &= \alpha y_{ss} - \beta y_{ssss} - \partial_y P, \end{aligned} \tag{4}$$

where, analogously to (2), $P = -|\nabla C|^2$ and $C(x, y)$ is an environmental concentration, for example algae, oil, or some chemical for which the agents are equipped with accurate sensors.

The derivatives with respect to the Lagrangian parameter $s$ can be approximated by finite differences:

$$\begin{aligned} \partial_t x_i &= \frac{\alpha}{h^2}(x_{i+1} - 2x_i + x_{i-1}) - \frac{\beta}{h^4}(x_{i-2} - 4x_{i-1} \\ &+ 6x_i - 4x_{i+1} + x_{i+2}) - \partial_x P(x_i, y_i) \\ \partial_t y_i &= \frac{\alpha}{h^2}(y_{i+1} - 2y_i + y_{i-1}) - \frac{\beta}{h^4}(y_{i-2} - 4y_{i-1} \\ &+ 6y_i - 4y_{i+1} + y_{i+2}) - \partial_y P(x_i, y_i) \end{aligned} \tag{5}$$

where the spacing between grid points in the parameter $s$ is $h = 1/N$, the reciprocal of the number of agents $N$, and $(x_i, y_i)$ now denote the $x$ and $y$ spatial coordinates of agent $i$. This means that the agents are thought of as "equally spaced" in the $s$ parameter along the curve. If $s$ were an arc-length parameterization, then the agents would also be equally spaced along the physical contour. The deformable model chosen does not preserve arc-length. However in practice, for sufficiently smooth boundaries, the distance between nodes is observed to vary smoothly. Finally we mention that the coupled system (5) 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. This is discussed in greater detail in Section 4.

For this algorithm to be useful for collective motion of agents, who have only local information about their environment and may also have constraints on their ability to move and stop, we consider two additional dynamics for the algorithm:

## 3.1 Sparsing

Following the argument in [4], we note that in the absence of an external force, or in the event that the external energy is constant for large areas (as it would be in the center of an oil slick), the contour could collapse to a point. Therefore, we adopt a similar inflating force to that seen in [5] but with a biological motivation: we assume that the agents interact through a pairwise repulsive potential of the form

$$\partial_t v_i = \sum_{v_j, j \neq i} \nabla U(v_j, v_i), \quad v_i = (x_i, y_i). \tag{6}$$

The repulsion kernel $U(v, w) = C_r \exp(-|v - w|/l_r)$, with $C_r, l_r \in \mathbb{R}^+$, is the same form repulsion as that in [9] used to model swarming motion in biological systems. In practice (6) requires global communication among agents, so we consider a cut-off distance $R$ above which agents do not interact through this potential. Throughout this paper, we choose $l_r = 0.5$, $R = 1$, and $C_r = 10$ whenever sparsing is used.

## 3.2 Self-Propulsion

Some platforms under development [16] involve agents that do not have the ability to hover, they must continually move with minimal speed $\omega$. Our algorithm is adaptable to such platforms by including rotational self-propulsion of the system. This propulsion is implemented in the model by the addition of the following term:

$$\partial_t v_i = \omega \hat{\tau}_i \tag{7}$$

where $\omega$ is the speed of each agent, and $\hat{\tau}_i = \frac{v_{i+1} - v_{i-1}}{\|v_{i+1} - v_{i-1}\|}$. This last term is an approximation to the tangent to the contour at agent $i$. In the simulations presented in this paper, we choose $\omega = 10$. The underlying motion is that each agent moves at a constant speed tangent to the virtual contour.

# 4 Implementation and Communication

In practice, we suggest a hybrid algorithm in which the group of agents alternates between the boundary tracking motion in (5), the sparsing dynamics in (6) and, if needed, the self-propulsive motion in (7). In the method below, we replace the time derivative $v$, $\partial_t v$ by a finite difference

$$\partial_t v_i \approx \frac{v_{i+1} - v_i}{\Delta t}.$$

The result is a 'split-step' discrete-time implementation in which the "virtual contour" dynamics is advanced over one step, followed by the repulsion and self-propulsion on a second step:

- STEP 1,

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

- STEP 2,

$$v_i^{n+1} = v_i^{n+1/2} + \Delta t \left( \omega \hat{\tau}_i^{n+1/2} + \sum_{v_j, j \neq i} \nabla U(v_j^{n+1/2}, v_i^{n+1/2}) \right).$$

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. Thus we can think of this method as alternating between taking a complete time step for equation (5) followed by a complete time step for methods (6) and (7) combined.

Steps 1 and 2 are alternated with a fixed time step $\Delta t$, resulting in convergence to a hybrid dynamics in the limit as $\Delta t \to 0$. Since the algorithm will be performed by agents, communication must occur at each step, thus it is advantageous to perform the algorithm with as large a time-step as possible. Note that for a localized potential $U$ that cuts off at a finite radius $R$, communication in step 2 will only be required over distances $d < R$. The communication required to perform Step 1 depends on whether the finite differences on the RHS are evaluated at the old time (forward Euler method) or the new time (semi-implicit method). Both situations are now discussed.

## 4.1 Forward Euler (explicit) Update

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

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

Note that agent number $i$ only needs to know the positions of agents $i-2$, $i-1$, $i+1$, and $i+2$ in addition to its own position, in order to move accordingly. Thus, this implementation requires only local communication between agents, not global communication of all agents. However, there is a significant drawback to using this

| Update Type | Time step | Communication |
|---|---|---|
| Forward Euler (Explicit) | $\Delta t \leq \frac{h^4}{8\beta+2\alpha h^2}$ | Local ($i \pm 0, 1, 2$) |
| Semi-implicit | $0 \leq \Delta t$ | Global |

Table 1: Communication frequency and range for stable implementation of the forward and backward Euler methods for the boundary tracking part of the algorithm (Step 1).

method. Stability of the algorithm requires that the time step $\Delta t < h^4/(8\beta + 2\alpha h^2)$, in particular as the number of agents $N$ increases, the algorithm must be updated on a timescale that decreases as $1/N^4$ for $N$ large. This result comes from a von Neumann stability analysis of the method, which we do not derive here but is standard for finite difference schemes of linear partial differential equations [14].

## 4.2 Semi-implicit Update

An implicit update can alternatively be used

$$v_i^{n+1/2} = v_i^n + \Delta t \left( \frac{\alpha}{h^2} \left( v_{i+1}^{n+1/2} - 2v_i^{n+1/2} + v_{i-1}^{n+1/2} \right) \right.$$
$$\left. - \frac{\beta}{h^4} \left( v_{i-2}^{n+1/2} - 4v_{i-1}^{n+1/2} + 6v_i^{n+1/2} - 4v_{i+1}^{n+1/2} + v_{i+2}^{n+1/2} \right) - \nabla P(v_i^n) \right). \quad (8)$$

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 agent to perform a linear algebra computation, which is only $O(N)$ because the matrix is pentadiagonal (e.g. a standard LU decomposition algorithm is very straightforward to use). However, in order for each agent to solve for their next position, they must know the positions of *all* the other agents in order to solve the implicit equations. The benefit of this method is that it is unconditionally stable for arbitrarily large time steps $\Delta t$. In practice, the time-step will be restricted by other factors such as the implementation of the sparsing and self-propulsion parts of the algorithm and desired convergence to the hybrid dynamics. We summarize the time-step constraints and communication constraints in Table 1.

## 4.3 Interpretation of Stability

Since there are two options for the algorithm, the question arises, which one should we use? The answer depends upon how often, how far, and with how many other agents each agent can communicate. With the explicit method, there is an upper bound on the amount of time between agent interaction. This bound is proportional to $1/N^4$ where $N$ is the number of agents. On the other hand, the implicit update has no upper bound on the time

step. This means that we may wait longer between communications, but at the price of having to communicate with all of the other agents. Note that we use the same explicit evaluation of the concentration function for each proposed version of Step 1. Therefore, in practice, when using explicit updates, we may do multiple iterations of the boundary tracking part before taking new sensor readings (corresponding to the evaluation of $P$) or enforcing the anti-collision (sparsing) term.

# 5 Simulations

We consider several simulations that illustrate features and issues with the algorithm. The examples all have $\alpha = 10^{-3}$, $\beta = 10^{-5}$, and concentration function

$$C(x, y) = \tanh\left((F(x, y) - .8)\right), \tag{9}$$

where $F(x, y) = \sum_{i=1}^{4} \exp\left(-((x - x_i)^2 + (y - y_i)^2)\right)$ and $(x_i, y_i) = (1, 0), (0, -.5), (-1.5, .5), (.15, 1)$, for $i = 1, 2, 3, 4$, respectively. This function produces an asymmetric boundary for the agents to locate.

- Figure 1 shows the evolution of agents initially positioned on a circle of radius 0.25. The time step was 0.01, and an implicit update was chosen. The agents move to the boundary and then follow it in a counter-clockwise direction. The "inflating" dynamics due to the sparsing term allows the group to get close enough to the boundary to find it. Upon reaching the boundary, the agents follow it indefinitely. Note that on the left side of the figure, the contour of agents does not lie exactly on the boundary. The sparsing term is responsible for the slight difference, however if this term is significantly decreased then the agents find a sub-optimal solution corresponding to the picture in figure 1 at time 2.5 (bottom left).
- Figure 2 illustrates the instability of the forward Euler method (top two panels) with too large a time step (here $\Delta t = 0.05$) compared with the backward Euler method (bottom two panels). In order to use the forward Euler method we need to choose a much smaller time step (for example $\Delta t = 10^{-3}$).
- Figure 3 shows that the algorithm can perform regardless of whether the initial placement of vehicles in inside or outside the region of interest. In this case, the agents are placed in a circle of radius 2.5.

# 6 Robustness under motion or communication failure

Field deployed robots are susceptible to various environmental and man made hazards. Therefore, it is important to have an algorithm that continues to perform if some percentage of the vehicles fail to communicate or simply

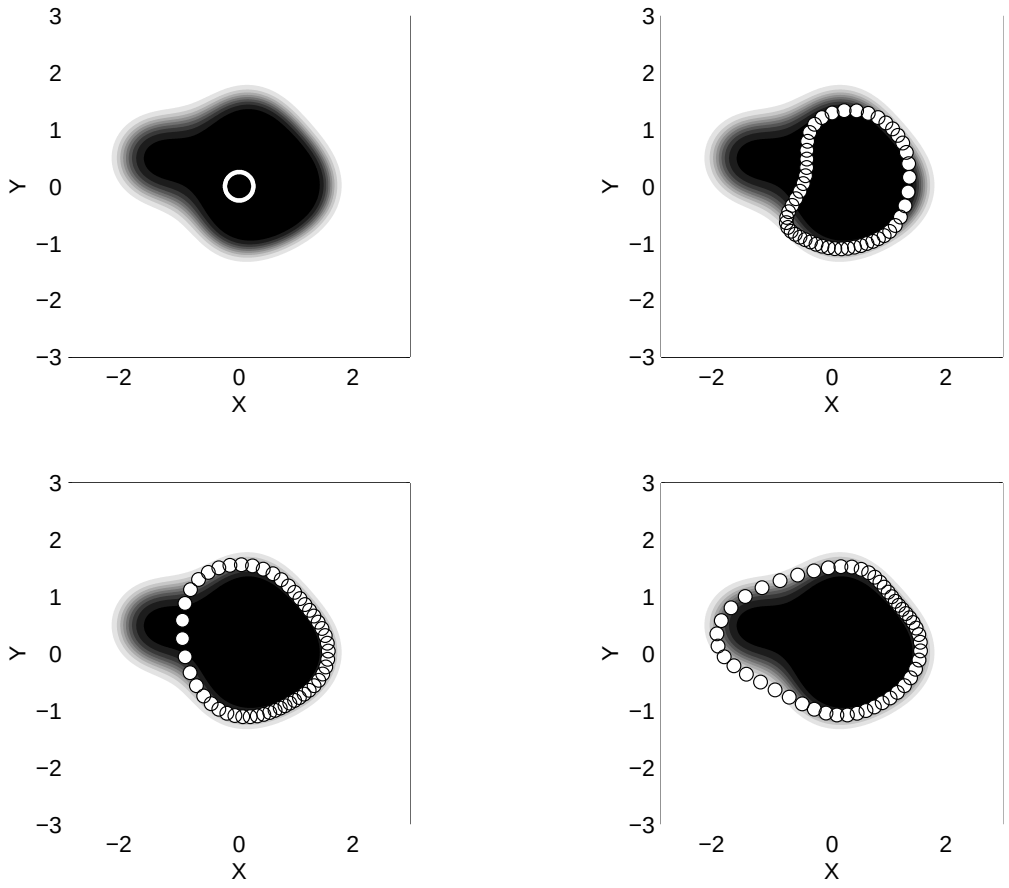


Figure 1: The upper-left panel is the initial orientation of agents in the plane, a circle of radius 0.25. The upper-right, lower-left, and lower-right panels are with $t = 0.5, 2.5$, and 5.0, respectively. The gray scale represents the concentration of the environment.

withdraw from the group. The algorithm must thus be adaptable to one (or more) searcher "deaths." We now demonstrate that the method can dynamically respond to agent disappearance and that this response is robust provided the vehicle deaths do not overly dominate the dynamics.

From the PDE perspective, a searcher death corresponds to the removal of a grid point (coarsening) of the discretization. For the implementation of Step 1 of the algorithm (boundary tracking) we simply treat agent failure by reordering the searchers with the dead one omitted from the list, i.e. we think of the system after vehicle death as a new initial discretization of the PDE (4) with $N - 1$ instead of $N$ nodes. Step 2 does not require any revision other than ignoring the position of the dead vehicle, which must be done if they no longer communication with the group. By the diffusive nature of Step 1, the other agents automatically fill in the hole created by the non-functioning agent. This behavior is further reinforced by the sparsing term in Step 2. To

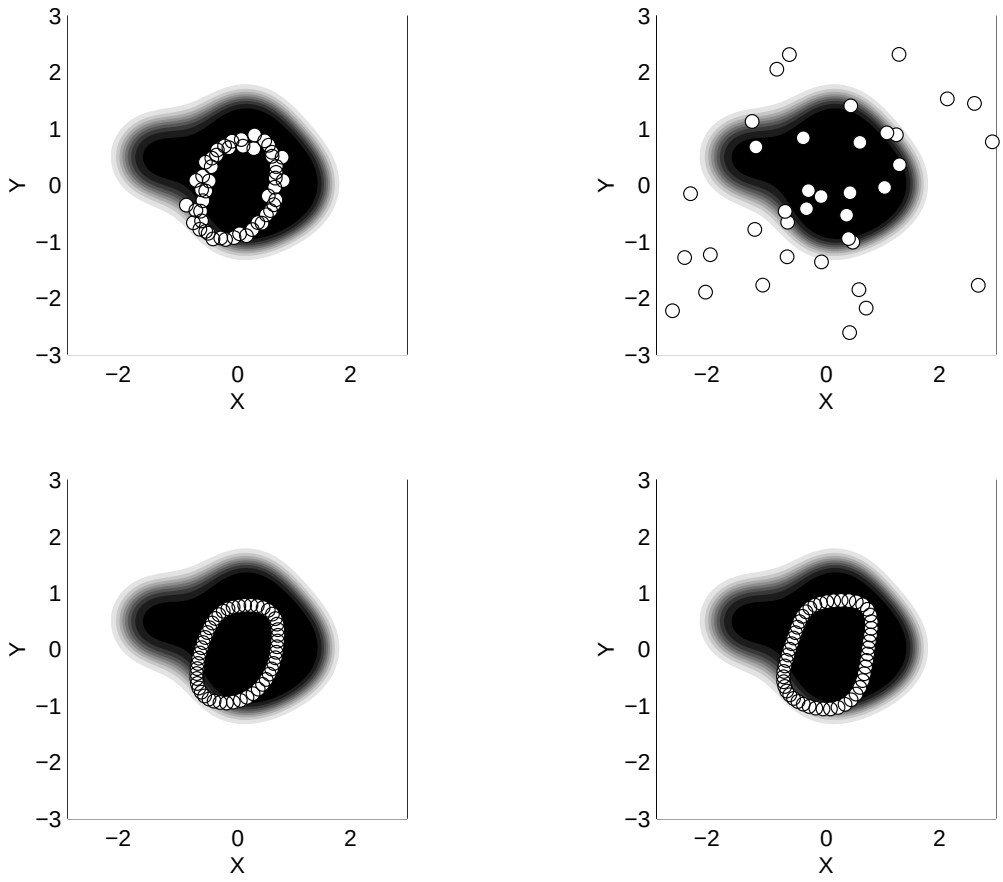


Figure 2: The figure compares explicit vs. implicit time-stepping for early times with $\Delta t = 0.05$, which is unstable for forward Euler but stable with backward Euler. The top two panels show the explicit method at time steps three and four and the bottom two panels for the implicit method.

illustrate this idea, we perform a simulation in which on every time step there is a finite probability that each searcher will fail. The boundary tracking problem is still solved even with significant vehicle loss.

Figure 4 shows an agent disappearing from the group and the resulting readjustment of the contour after agent failure. The three dark circles in the first panel show the agent that is about to fail and its left and right nearest neighbors. The second panel shows one time step later where the failed agent has been removed from the simulation. Ten time steps later, shown in panel three, the neighboring agents have readjusted their positions to "fill in" the gap left by the missing agent. The self-propulsion term rotates all agents around the boundary. Figure 5 shows a simulation with the same parameters from Section 5, in which, on every timestep, each agent has a probability 0.0025 of failure. The dynamics is similar to that shown in Figure 1 and the agents find the boundary despite heavy losses. The three panels show times 0.5, 1.5, and 2.5 at which point the agents have

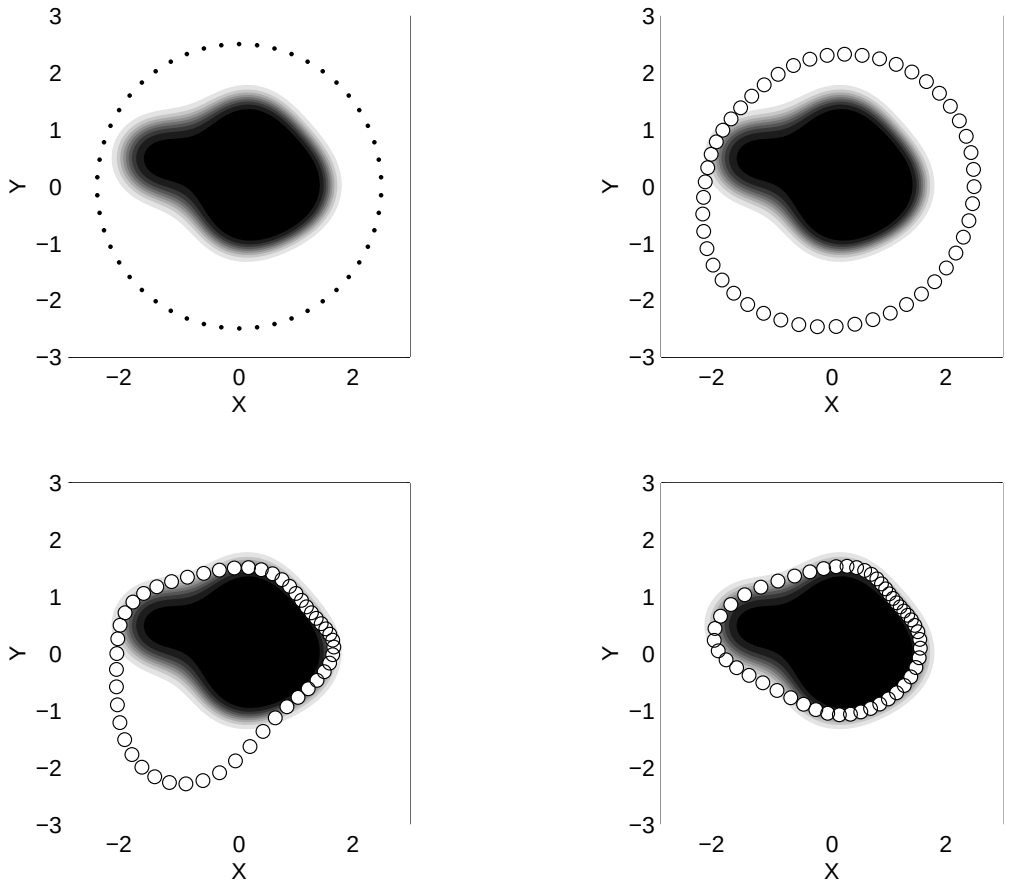


Figure 3: The initial placement of agents is on a circle of radius 2.5 with backward Euler time step, $\Delta t = 0.01$, and the same times as shown in Figure 1.

located the boundary despite over 50% vehicle loss. When the number of agents drops significantly below this, the virtual contour algorithm does not perform well.

# 7 Boundary tracking without communication

The snake algorithm may not perform well under significant vehicle failure due to the fact that a virtual contour is badly approximated by a sparse set of points. In the case where only a few vehicles remain or are initially available, it makes sense to create an algorithm for single agent use or for small groups of agents with no or little communication.

As before, let $C(x, y)$ denote the concentration function of the environment. Also, let $P(x, y) = -|\nabla C(x, y)|^2$, and $v = (x, y)$. Note that the gradient flow $\frac{dv}{dt} = -\nabla P$ yields motion along gradient lines of $P$ while the conservative flow $\frac{dv}{dt} = \nabla^{\perp} P$ denotes motion along level sets of $P$. The boundary of $C$ lies at a minimum of $P$,

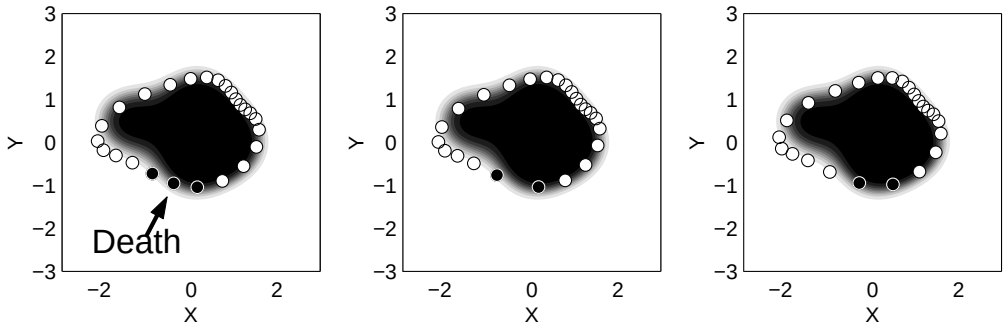


Figure 4: On the left, the three dark circles denote the agent about to fail and its two nearest neighbors. The middle panel shows the gap in the contour after the agent fails, and ten time steps later the right panel shows the readjusted contour and the new positions of the neighbors of the failed agent.

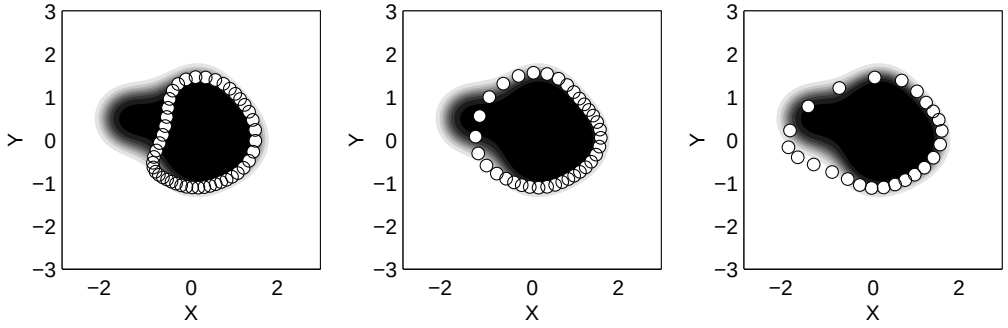


Figure 5: At each time step an agent has probability 0.0025 of failing. The three panels show times .5, 1.5 and 2.5 with agent number decreasing from 45 to 35 and finally 24.

thus the gradient flow will move a single agent to a special point on the boundary, a local minimum of $P$. The conservative flow moves an agent along level curves of $P$. To find and travel around the boundary, we construct an algorithm that includes both effects as well as a backward time "push" term, to avoid becoming stuck in a minimum of $P$:

$$\begin{aligned} \frac{dx}{dt} &= -\frac{\partial P}{\partial x} - \omega \frac{\partial P}{\partial y}/|\nabla P| + \theta n_1 \\ \frac{dy}{dt} &= -\frac{\partial P}{\partial y} - \omega \frac{\partial P}{\partial x}/|\nabla P| + \theta n_2 \end{aligned} \tag{10}$$

where $\vec{n} = [\vec{v} - \vec{v}(t-\delta)]/|\vec{v} - \vec{v}(t-\delta)|$, i.e. the unit displacement vector between current position $\vec{v}$ and position $\vec{v}(t-\delta)$ time units ago. This corrective term serves to add a self-propulsion term of strength $\theta$ to the agent in the direction that it is already going. Figure 10 presents a simulation of (10) with N independent agents starting at random positions in the middle of the domain. The parameters are $\delta = .01$, $\theta = 1.0$, $\omega = 5.0$. Times are at: 0,

.25, 2.0, 3.0 for upper left, upper right, lower left, lower right, respectively. Note that the agents all travel around the perimeter but do not know about each others' behavior. The distribution of agents is clumped, instead of evenly spaced around the boundary, as it is in the cooperative case.

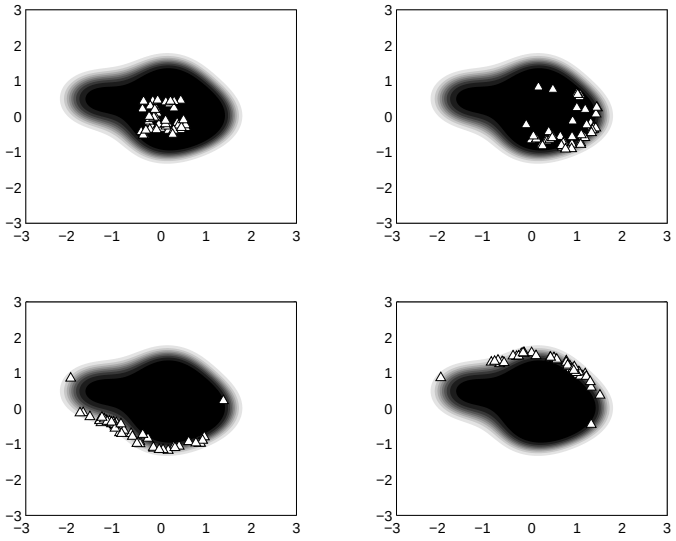

Figure 6: 50 agents independently performing the single-agent algorithm. Times are at: 0, .25, 2.0, 3.0 for upper left, upper right, lower left, lower right, respectively.

# 8 Conclusions and Future Work

We have constructed a class of a boundary finding algorithms for a group of autonomous mobile robots. The communication required for the algorithm can be either be local or global, depending on frequency of interaction. Explicit update equations require very frequent communication for stability of the method, but only near neighbor interactions are required. Implicit updates require less frequent communication although the entire group must relay their positions for this form of the algorithm. Failure of agents may be overcome, at the cost of accuracy. In the case of extreme failure with only a few agents functioning, the single agent method of Section 7 could take over.

There are four main issues that we do not address in this paper but are important for this problem:

- Noise or inaccuracy in the sensor data or communication between agents,
- computing gradients of sensor information,
- changes in topology of the virtual contour,

- time dependent changes in the concentration field.

We address each of these issues and possible solutions for future research. In image processing, typically images have significant noise and the snake algorithms perform quite well in this context. This suggests that sensor noise will be straightforward to address by analogy with the imaging problem. However noise in the communication (i.e. in position of agents) is not an issue for the imaging problem but will be a significant issue for the mobile agent problem. Also note that typically sensors do not measure gradients in concentration, but the concentration itself. So in the field, this quantity will need to be estimated from local information. Regarding topology of the contour, we assume a simple closed curve, however the concentration field in reality may be composed of several disjoint patches. A challenge is to design an algorithm that will automatically split or merge agent contours as required to track the boundaries. This problem has been addressed in the imaging community using level sets [11] or with contours possessing boundaries [10]. It would be interesting to explore both of these methods in this context. Finally time dependent concentration fields are quite interesting. The method presented here should continue to work well when the time scale of the concentration motion is small compared to the reaction time of the virtual contour, and when the topology of the boundary does not change. Rapidly changing concentration fields will require new ideas.

## Acknowledgments

We thank Naomi Leonard and Matthieu Kemp for independently suggesting the problem of boundary tracking using cooperative agents. This research is supported by ARO grant DAAD19-02-1-0055 and ONR grant N000140310073.

## References

[1] J.Desai, V.Kumar, and J.Ostrowski, A Theoretical Framework for Modeling and Controlling Formations of Mobile Robots. IEEE Trans. on Robots and Automations, (under review).

[2] P.Ogren, E.Fiorelli, and N.Leonard, Formations with a Mission: Stable Coordination of Vehicle Group Maneuvers. Proc. 15th Int'l Symposium on Math. Theory of Networks and Systems, (2002).

[3] 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).

[4] L. Cohen and I. Cohen, Finite-Element Methods for Active Contour models and Balloons for 2-D and 3-D Images. *IEEE Transactions on Pattern Anal. and Machine Intelligence* vol. 15 no.11 (1993).

[5] L. Cohen, On Active Contour Models and Balloons. *Comput. Vision, Graphics, and Image Processing: Image Understanding* vol 53 no 2 (1991).

[6] M. Kass, A. Witkin, and D. Terzopoulos, Snakes: Active Contour Models. *International J. of Computer Vision*, 321-331 (1988).

[7] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, Collaborative multi-robot exploration. *Autonomous Robots* vol. 8 no. 3 (2000).

[8] R. Cassinis, Landmine Detection Methods Using Swarms of Simple Robots, *International Conference on Intelligent Autonomous Systems* 6, E. Pagello (Ed.), Venice, Italy (2000).

[9] H. Levine, W.J. Rappal and I. Cohen, Self-organization in Systems of Self-propelled Particles. Phys Rev E, 63, 017101 (2001).

[10] T. McInerney and D. Terzopoulos, T-Snakes: Topology Adaptive Snakes. medical Image Anal. vol. 4 (2000).

[11] Stanley J. Osher and Ronald P. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Springer-Verlag, 2002.

[12] D. P. Perrin, E. Kadioglu, S.A. Stoeter, and N. Papanikolopoulos, Localization of Miniature Mobile Robots Using Constant Curvature Dynamic Contours. Proc. of the 2002 Intl. Cong. on Robotics and Automation. Washington, DC (2002).

[13] J. A. Sethian, *Level Set Methods and Fast Marching Methods : Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, Cambridge University Press, 1999.

[14] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, Chapman and Hall, New York, 1989.

[15] D. Terzopoulos, Regularization of Inverse Visual Problems Involving Discontinuities. IEEE Transactions PAMI-8, p.413 (1986).

[16] `http://www.nektonresearch.com/`