Gradient Calculation in Sensor Networks

Thomas C. Henderson
School of Computing
University of Utah
Salt Lake City, UT, USA
Email: tch@cs.utah.edu

Eddie Grant
Electrical and Computer Engineering
North Carolina State University
Raleigh, NC, USA
Email:egrant@eos.ncsu.edu

Abstract— **Sensor networks are comprised of devices having the ability to communicate, compute and sense the environment. A wide range of information processing tasks have been studied for such networks, including operating systems, issues, architecture optimization, and distributed data processing. In this paper, we analyze and compare four different techniques to estimate the gradient of the function represented by the sensor samples. These include: (GA1) a simple device ID defined direction, (GA2) directional derivative, (GA3) polynomial approximation with a plane, and (GA4) polynomial approximation with a quadratic. We compare these based on density of devices per unit area, and noise in the position and sensed data. The interesting result is that GA3 significantly outperforms the other algorithms, although GA1 performs very well and is much easier to compute than the others.**

I. Introduction

Many advances have been made in sensor network technology and algorithms in the last few years. See [1] for an overview of the state of the art. Work has been done on: architecture [2], systems and security [3], [4], [5], and applications [6]. Our own work has focused on the creation of an information field useful to mobile agents, human or machine, that accomplish tasks based on the information provided by the sensor network [7], [8], [9], [10], [11].

At the most basic level, the devices are distributed in the environment. Consider the following scenario. A set of devices are dropped in a wide geographic area to monitor a toxic gas leak in the air. Mobile robots involved in the containment and cleanup need to follow the chemical gradient to move to the toxin source locations. Thus, the gradient of the concentration scalar field is required. Figure 1 shows an example set of devices with neighbors in the graph defined as those in radio broadcast range.

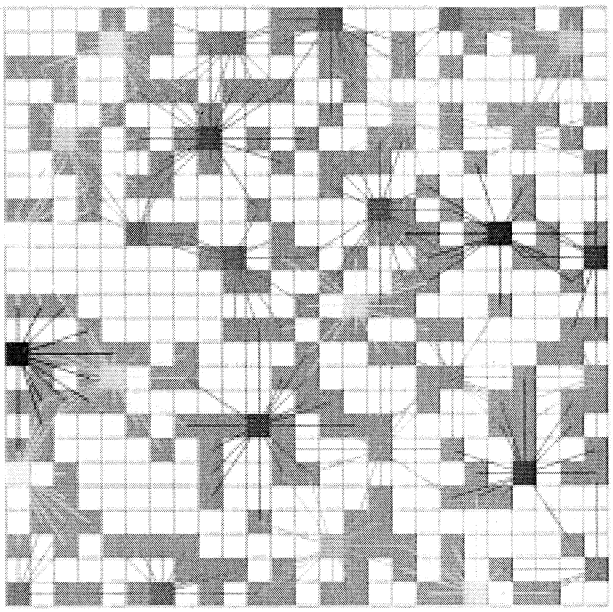

Fig. 1. A Selection of S-Element Devices and Their Neighbors.

II. Gradient Calculation

The **gradient** of f at (x, y) is the vector in $\Re^2$ given by:

$$grad\ \ f = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}) = \nabla f$$

(We follow Marsden [12] for our vector calculus notation.)

A. Coordinate Frame Free Method (GA1)

The gradient is approximated at each device (ID_{device}) as follows:

1) From the neighbors of ID_{device}, the device (ID_{max}) with the maximum sensed data value is determined.
2) The gradient is reported as the pair of device ID's: (ID_{device},ID_{max}).

A mobile agent uses these ID's to move in the gradient direction by moving between the two devices and then moving in the direction of ID_{max}. Note the

method does not require that the (x,y) positions of the individual devices be known. Accurate calibration of sensor networks is a difficult task and getting good position data is very difficult or very expensive [13].

This method is very inexpensive and robust and thus, very attractive if its performance compares well to the other more rigorous approaches. Note that in order to make error comparisons, we use the position given by the actual direction between devices because this is what a mobile agent would use.

B. Directional Derivatives (GA2)

This method requires knowledge of the positions of the devices. The **directional derivative** of f at $\overline{x}$ in direction $\overline{v}$ is given by:

$$d.d. = \frac{d}{dt} f(\overline{x} + t\overline{v}) \mid_{t=0}$$

if it exists. From this, we have that the directional derivative is also defined by:

$$d.d. = \lim_{h \to 0} \frac{f(\overline{x} + h\overline{v}) - f(\overline{x})}{h}$$

We approximate this by:

$$d.d.a. = \frac{f(\overline{x} + h\overline{v}) - f(\overline{x})}{h} \tag{1}$$

Assuming all directional derivatives exist, it is the case that:

$$\overline{D} f(\overline{x})\overline{v} = grad \;\; f(\overline{x}) \cdot \overline{v} = \nabla f(\overline{x}) \cdot \overline{v}$$

so that:

$$d.d. = [\frac{\partial f}{\partial x}(\overline{x})]v_x + [\frac{\partial f}{\partial y}(\overline{x})]v_y \tag{2}$$

where $\overline{v} = (v_x, v_y)$. Combining these, we approximate the gradient at each device, ID_{device}, located at $\overline{e_0}$, as follows:

1) Choose two of ID_{device}'s neighbors, ID_1 and ID_2, located at $\overline{e_1}$ and $\overline{e_2}$, respectively, such that $\angle \overline{e_1 e_0 e_2}$ is as close to a right angle as possible.
2) For the two points, $\overline{e_1}$ and $\overline{e_2}$, solve (1) to get the following pair of equations:

$$d.d.a._1 = f_x e_{1x} + f_y e_{1y} \tag{3}$$

$$d.d.a._2 = f_x e_{2x} + f_y e_{2y} \tag{4}$$

3) Solve (3) and (4) for the two unknowns: f_x and f_y and form the gradient as (f_x, f_y).

C. Polynomial Approximation: plane (GA3)

For each device, the position must be known. To approximate the gradient:

1) From the positions and sensed data values of the n points within broadcast range of the current device (i.e., itself and its neighbors), form the linear system:

$$\begin{pmatrix} f(\overline{e_1}) \\ f(\overline{e_2}) \\ \vdots \\ f(\overline{e_n}) \end{pmatrix} = \begin{bmatrix} 1 & e_{1x} & e_{1y} \\ 1 & e_{2x} & e_{2y} \\ & \vdots & \\ 1 & e_{nx} & e_{ny} \end{bmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} \tag{5}$$

2) Solve (5) for a_0, a_1, and a_2.
3) The gradient is then (a_1, a_2).

D. Polynomial Approximation: quadratic (GA4)

Here we make the assumption that the functional form of the sensed data is:

$$f(x, y) = \frac{D_{max}}{\sqrt{1 + (S_x - x)^2 + (S_y - y)^2}} \tag{6}$$

where D_{max} is the maximum value of the function at the source location (S_x, S_y). In order to set up to solve for the gradient, rewrite (6) as follows:

$$\frac{1}{f(x, y)} = \frac{\sqrt{1 + (S_x - x)^2 + (S_y - y)^2}}{D_{max}}$$

$$\frac{1}{f^2(x, y)} = \frac{1 + (S_x - x)^2 + (S_y - y)^2}{D_{max}}$$

$$u(x, y) = a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 y^2$$

where $u(x, y) = \frac{1}{f^2(x,y)}$, $a_0 = \frac{1 + S_x^2 + S_y^2}{D_{max}^2}$, $a_1 = \frac{-2S_x}{D_{max}^2}$, $a_2 = \frac{-2S_y}{D_{max}^2}$, $a_3 = \frac{1}{D_{max}^2}$, and $a_4 = \frac{1}{D_{max}^2}$.

Then the gradient is found as:

1) From the positions and sensed data of the device and its neighbors, form the linear system:

$$\begin{pmatrix} u(\overline{e_1}) \\ u(\overline{e_2}) \\ \vdots \\ u(\overline{e_n}) \end{pmatrix} = \begin{bmatrix} 1 & e_{1x} & e_{1y} & e_{1x}^2 & e_{1y}^2 \\ 1 & e_{2x} & e_{2y} & e_{2x}^2 & e_{2y}^2 \\ & & \vdots & & \\ 1 & e_{nx} & e_{ny} & e_{nx}^2 & e_{ny}^2 \end{bmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} \tag{7}$$

2) Solve (7) for a_0, a_1, a_2, a_3, and a_4.
3) Then the gradient is given by $(2a_3 x + a_1, 2a_4 y + a_2)$.

Note that $S_x = \frac{-a_1}{2a_4}$, $S_y = \frac{-a_2}{2a_4}$, and $D_{max} = \frac{1}{\sqrt{a_5}}$. Note that the recovery of these parameters is difficult as the a_i's are very sensitive to the data.

III. Simulation Experiments

Our simulation works as follows:

for number of devices = dev_{min} to dev_{max}
  for noise = 0 to $noise_{max}$
    for number of trials = 1 to $trials_{max}$
      Distribute devices uniformly over area
      Select source location for scalar field
      Set values of sensor devices
      for each gradient method
        Calculate the gradient
        Calculate the error
      end
    end
  end
end

This has been implemented in Matlab, and Table I gives the results of the simulations.

The density of the devices was allowed to vary from 1 per unit area to 2 per unit area. The table shows that increasing the number of devices generally improves the quality of the approximation. Two noise levels were evaluator: (1) no noise, and (2) noise with standard deviation 1. This noise is applied to both the position of the devices, as well as to the sensed data values. That is, the device position is normally distributed about the actual position with 0 mean and either standard deviation of 0 or 1. Sensed data is handled in a similar manner.

IV. Conclusion

From Table I it can be seen that GA3 performs significantly better than the other algorithms, even under noisy conditions. Figure 2 shows a sample sensor network with neighbors graph, and Figure 3 shows the gradient approximation by GA3 with an average of 2 devices per unit area and no noise. Moreover, GA1 - which does not use device position information - performs comparable to the other algorithms, and in absolute terms is not so bad (about 16 degrees error under noisy conditions with a couple of devices per unit area). Figure 4 shows the results under the same conditions as GA3 above. Figures 5 and 6 show the results of GA2 and GA4 on the same data. As can be seen, GA4 does a very poor job of approximating the gradient; any time a specific functional form is chosen, it will do poorly if it does not match the actual environment.

In the near future, we intend to implement and test these algorithms on a 100-device sensor network testbed. The design of this testbed is underway by the

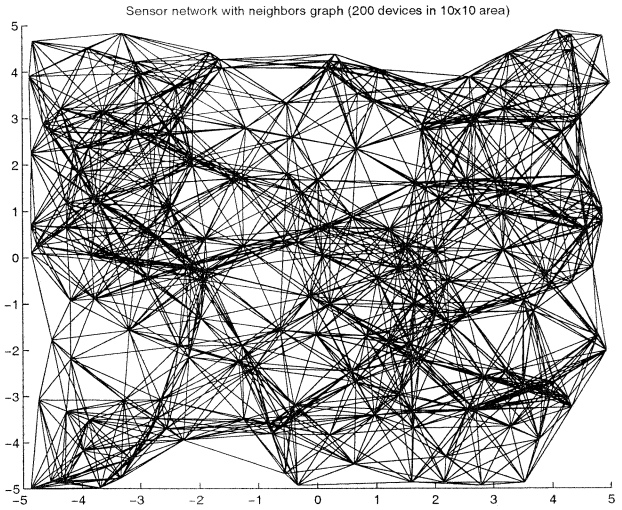

Fig. 2. Sample sensor network with neighbors graph.

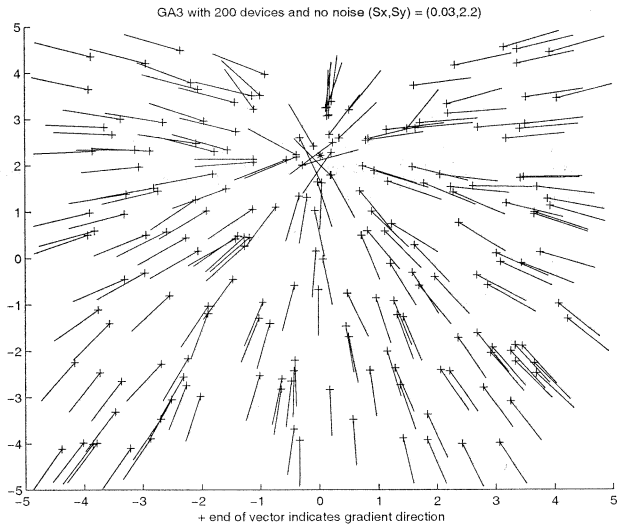

Fig. 3. GA3 gradient approximation with no noise.

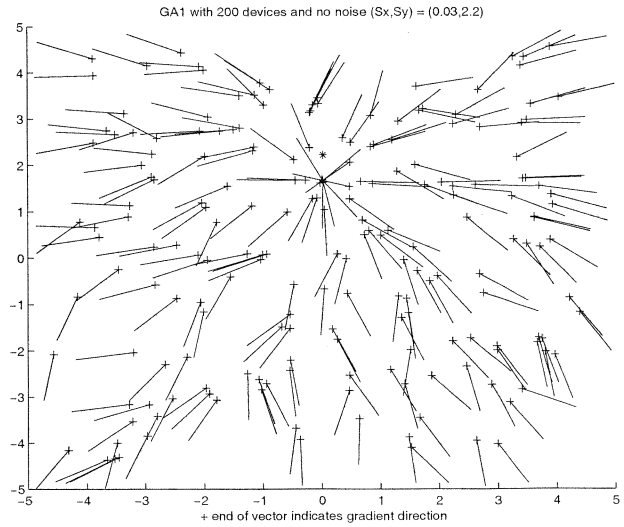

Fig. 4. GA1 gradient approximation with no noise.

TABLE I

SIMULATION RESULTS

Algorithm	*Angle Error (degrees)*	*Angle Error Std*	*Mag Error (pixels)*	*Mag Error Std*
100 devices / 0 data error				
GA1	5.89	0.02	5.85	1.49
GA2	5.39	0.02	10.71	8.40
GA3	1.20	0.01	2.90	0.62
GA4	16.63	0.10	10.70	19.12
200 devices / 0 data error				
GA1	2.92	0.01	5.34	1.34
GA2	6.18	0.01	14.52	21.42
GA3	0.67	0.01	2.61	0.51
GA4	14.83	0.10	6.90	3.54
100 devices / 1 std data error				
GA1	25.79	0.02	6.01	1.43
GA2	20.90	0.06	7.72	2.74
GA3	10.24	0.04	6.49	2.70
GA4	18.59	0.06	56.4	326.94
200 devices / 1 data error				
GA1	15.69	0.03	5.72	1.37
GA2	10.61	0.04	8.67	3.44
GA3	6.58	0.04	5.81	0.97
GA4	17.87	0.08	52.80	150.21

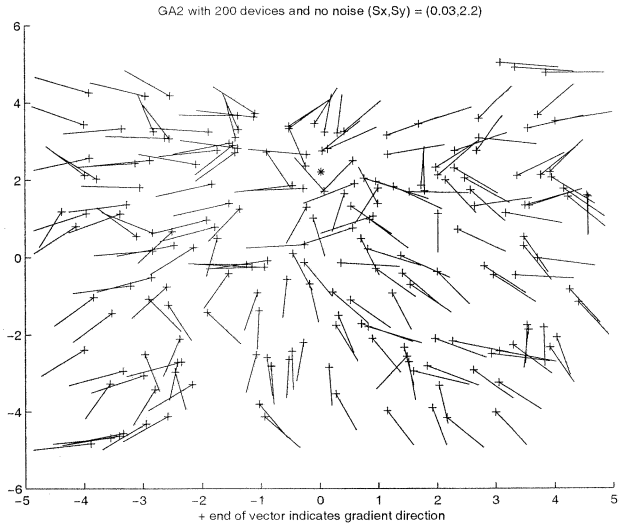

Fig. 5. GA2 gradient approximation with no noise.

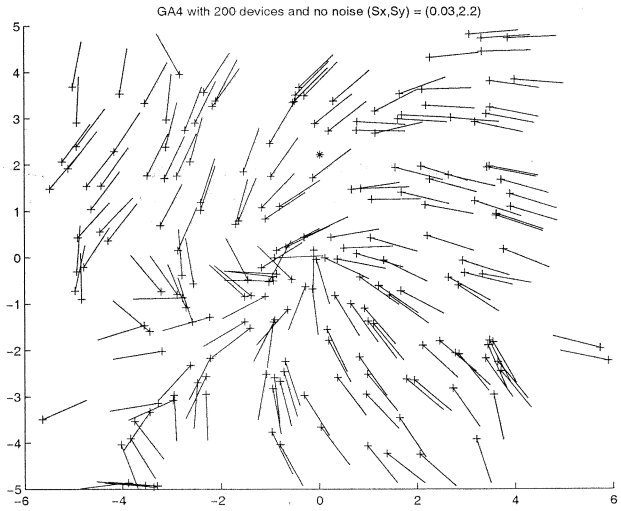

Fig. 6. GA4 gradient approximation with no noise.

authors, and the goal is to construct the testbed this summer.

REFERENCES

[1] F. Zhao and L. Guibas, "Preface," in *Proc of IPSN 2003*, (Palo Alto, CA), pp. v–vi, LNCS, April 2003.

[2] J. Hill and D. Culler, "A wireless embedded sensor architecture for system-level optimization," ece, UC Berkeley, October 2002.

[3] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly resilient, energy efficient multipath routing in wireless sensor networks," *Mobile Computing and Communications Review*, vol. 1, no. 2, 2002.

[4] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: Security protocols for sensor networks," *Wireless Networks*, vol. 8, pp. 521–534, Sept 2002.

[5] L. Zhang, "Simple protocols, complex behavior," in *Proc. IPAM Large-Scale Communication Networks Workshop*, March 2002.

[6] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor netwroks for habitat monitoring," in *WSNA 2002*, (Atlanta, GA), September 2002.

[7] T. C. Henderson, M. Dekhil, S. Morris, Y. Chen, and W. B. Thompson, "Smart sensor snow," *IEEE Conference on Intelligent Robots and Intelligent Systems*, October 1998.

[8] Y. Chen, "Snets: Smart sensor networks," Master's thesis, University of Utah, Salt Lake City, Utah, December 2000.

[9] Y. Chen and T. C. Henderson, "S-nets: Smart sensor networks," in *Proc International Symp on Experimental Robotics*, (Hawaii), pp. 85–94, Dec 2000.

[10] T. C. Henderson, "Leadership protocol for s-nets," in *Proc Multisensor Fusion and Integration*, (Baden-Baden, Germany), pp. 289–292, August 2001.

[11] T. C. Henderson, J.-C. Park, N. Smith, and R. Wright, "From motes to java stamps: Smart sensor network testbeds," in *Proc of IROS 2003*, (Las Vegas, NV), IEEE, October 2003.

[12] J. Marsden and A. Tromba, *Vector Calculus*. New York: W.H. Freeman and Company, 1988.

[13] K. Whitehouse and D. Culler, "Calibration as parameter estimation in sensor networks," in *Proc. WSNA 2002*, (Atlanta), September 2002.