Algorithms for Position and Data Recovery in Wireless Sensor Networks

by Lance Doherty

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Kristofer S. J. Pister
Research Advisor

(Date)

* * * * * * *

Professor Kannan Ramchandran
Second Reader

(Date)

Abstract

Networks of hundreds or thousands of sensor nodes equipped with sensing, computing and communication ability are conceivable with recent technological advancement. Methods are presented in this report to recover and visualize data from wireless sensor networks, as well as to estimate node positions. A communication system is assumed wherein information from sensor nodes can be transferred to a centralized computer for data processing, though suggestions are made for extensions to distributed computation. Specifically, this report presents four topics. First, the notion of using network connectivity to reconstruct node positions via linear or semidefinite programming is explored. Random feasible node placement and bounding methods are both found to increase in precision with the indiviual geographical constraints. Second, the potential effectiveness of two correlation-based sensor data encoding schemes is reported. Blind correlation methods are found to provide meager compression while semi-blind correlation can effectively reduce bandwidth requirements by one-half. Third, trajectory reconstruction through a sparse sensor network is used to track objects with expectation-minimization techniques. Trajectories can be distinguished providing that sufficient spatial or temporal separation exists. Fourth, optical flow algorithms are used to visualize time-varying continuous flow around the network. A qualitative analysis of the reconstructed flow for several case studies suggests a minimal node density as related to flow speeds.

Acknowledgements

This work would not have been possible without the insistence of Kris Pister that I must explore something of my own creation and his patience to wait for it to happen.

Much commiseration and debate with James McLurkin exposed many of the problems discussed.

The optical flow application was first proposed by Dana Teasdale, who also carried out the initial exploration and simulation and contributed to the proofreading and revision of this report.

I am indebted to Veljko Milanović for leading me through the research process with a project on micromachined field emission devices.

Much of the theory for the creative problem solutions described herein stemmed from classes taught by Jana Kosecka, Laurent El Ghaoui and from ideas suggested by Kannan Ramchandran.

Steve Thorne introduced me to the value of a logocentric perspective and the sadness of being teleologically constrained.

Funding for this report came from the Canadian government with an NSERC postgraduate scholarship, and from DARPA through the Smart Dust project.

Table of Contents

Chapter 1 - Introduction

I. THE EMERGENCE OF WIRELESS SENSOR NETWORKS

The maturing of integrated circuitry, microelectromechanical systems (MEMS) and communication theory has fomented the emergence of wireless sensor networks and precipitated the economic and computational feasibility of networks of hundreds or thousands of self-sufficient sensor nodes. Each node has the ability to sense elements of its environment, perform simple computations, and communicate either among its peers or directly to an external observer. Larger node numbers allows for sensing over larger geographical regions with greater accuracy than previously possible.

The work in this report is principally motivated by the BSAC Smart Dust project, aiming to scale sensing communication platforms down to cubic millimeter volume [1]. Combining CMOS technology for processing logic, MEMS corner-cube reflectors [2] for passive communication and thick-film batteries for power, Smart Dust strives to achieve minimal dimension and power consumption with a many-to-one communication paradigm. The most promising methods for short and long range peer-to-peer communications are RF and optical media [3], respectively. Progress is being made to integrate both of these media at the chip level [4], while macroscopic sensor node systems have been demonstrated [5],[6]. Ad hoc routing protocols have been developed – information theoretic bounds are explored in [7] and protocols are detailed in [8],[9].

Judging by the interest shown by the military, academia, and the media, innumerable applications exist for Smart Dust networks. Examples include weather monitoring, security and tactical surveillance, distributed computing, and camel tracking. The research documented in this report develops mathematical constructs to define, explore and apply sensor network theory. The techniques are drawn from well-established fields and applied to this nascent realm.

II. RESEARCH GOAL AND APPROACH

The goal of this research is to apply centralized computation to sensor networks with the philosophy that any information contained in or generated by a sensor network should be accessible to the outside world. McLurkin's report explores many of the same problems from a strictly distributed standpoint [10]. To reject the potential of centralized computation entirely may be to neglect many powerful capabilities. As the networks scale, sheer bandwidth and computational requirements will eventually render impossible the use of an external observer to monitor or completely control sensor networks, but the methods presented herein perform acceptably for networks of hundreds of nodes. Where scaling constraints are apparent, they are explained in detail.

All attempts are made to preserve mathematical rigor, though illustrations and heuristics are sometimes sufficient. It is regrettable that this report deals exclusively with simulation, it is hoped that the algorithms will soon be tested in reality. Errors and shortcomings of the research have been candidly documented; it is the equinoctial union of success and failure that allows researchers to progress and avoid the repetition of colleagues' mistakes.

It is unnecessary that the chapters of this report are read seriatim; aside from the introduction, each chapter presents an independent study with corresponding introductory and conclusive sections. Relevant references are provided throughout the text and detailed at the end of each chapter with the following section providing an overview.

III. RESEARCH OVERVIEW

The material presented in this report may initially seem a collection of disconnected topics related only distantly under the banner of sensor network theory. This is partially true; each chapter grew from the

exploration of a precise and isolated problem. There is still some logic behind this apparent randomness, however, in that the mathematics presented provide a path from network definitions to initialization to application.

A. Preliminaries

The following chapter details the definitions and the fundamental simulation methods that form the basis of the remainder of this report. An abstract definition of a wireless sensor network appropriate for this work is given along with the parameters that serve to compare networks. Methods of network creation for simulation purposes, along with their impact on the network parameters, are discussed. Finally, a brief discussion of homogeneity and symmetry-breaking as related to the research is presented.

B. Position estimation

In Chapter 3, convex optimization techniques are employed to estimate unknown node positions based on known node positions and connectivity-imposed constraints. Linear and semidefinite programming methods based on physical communication geometry provide both feasible solutions and upper bounds on the position estimation problem. The chapter focuses on radial constraints to model broadcast communication systems and angular constraints to model optical communication systems. Within each model, a performance metric is defined to explore the effects of network connectivity and how well the algorithms function in idealized circumstances. The discussion concludes with a set of possible improvements and extensions of the presented methodology.

Given: Node connectivity (not required to be complete)

Output: Node positions with bounded uncertainty

C. Correlation coding

Chapter 4 applies a data-coding scheme that exploits correlation among individual sensor nodes to reduce the overall network communication bandwidth used. Initially, the attempt is made to reduce the message length from most nodes in the network by sending partial sensor data information without any requirement for inter-node communication. The full information can be recovered by considering data from neighboring nodes in the same correlation group. For data-generating functions with low spatial frequency, the proposed coding scheme offers marginal bandwidth reduction with a modicum of error introduction. A similar coding scheme requiring one additional broadcast message per correlation group results in exact message recovery with more substantial reductions in bandwidth.

Given: Node positions and sensor readings

Output: A reduced-bandwidth coding method to communicate all sensor readings

D. Tracking in a sparse network

Chapter 5 applies the expectation-maximization procedure to track objects through a sparse network. An object passing through the sensing radius of a node flags this node for use in the computation. The positions of flagged nodes are used as data to reconstruct the most likely path or paths that passed through the network. The algorithm provides an adjustable threshold for determining the number of paths and for fitting each path to the best member of a specified family of curves. The addition of time stamps to the object sensing events at each node allows for the reconstruction of the spatio-temporal trajectory of the object through the network. Not only does this provide additional logical distance to distinguish between similar trajectories, overall performance is increased as a result.

Given: A set of nodes flagged by an object

Output: The path (time-independent) or trajectory (time-dependent) of the object through the network

E. Visualization of temperature flow

Chapter 6 uses optical flow methods from computer vision to summarize properties of temperature flow in the sensor network. Functions representative of changing temperatures in the network lead to time-

varying sensor data at each node. The presented methods use all sensor readings to determine the qualitative properties of the temperature flow in the environment by developing a node-pixel analogy. The required network parameters for accurate recovery are examined in addition to the type of flow that can be recovered. Function frequency is pivotal for flow recovery, both in the spatial and temporal sense. The former defines the required density of sensors in the network while the latter determines the minimum sampling rate. Canonical two-dimensional flows are examined in detail, forming the basis for arbitrary continuous function recovery.

Given: Sensor readings at known positions and times

Output: A set of vectors describing temperature flow in the network

IV. COMMUNICATION PROTOCOL

This work assumes that there is some working protocol sufficiently competent to permit communication between any node in the network and an external observer. This could be accomplished via individual optical communication from the observer to each node as in the Smart Dust model, through one of countless ad-hoc routing methods, or through clustering. However, the position estimation chapter requires that nodes have some knowledge of their neighbor's location, so the Smart Dust model is not applicable. The issue of which protocol should be chosen or how such methods could be implemented is not considered.

V. REFERENCES

[1] J. M. Kahn, R. H. Katz and K. S. J. Pister, "Mobile Networking for Smart Dust", ACM/IEEE Intl. Conf. on Mobile Computing and Networking, Seattle, WA, Aug. 1999.

[2] V. Hsu, J. M. Kahn, and K. S. J. Pister, "Wireless Communications for Smart Dust", Electronics Research Laboratory Technical Memorandum Number M98/2, Feb. 1998

[3] M. Last, K. S. J. Pister, "2-DOF Actuated Micromirror Designed for Large DC Deflection", MOEMS ' 99, Mainz, Germany, Aug 1999.

[4] B. Atwood, B. Warneke, K. S. J. Pister, "Preliminary circuits for Smart Dust", IEEE Southwest Symposium on Mixed-Signal Design, pp. 87-92, San Diego, CA, Feb. 2000.

[5] S. Hollar, Masters Report, University of California, Berkeley, May 2000.

[6] G. Asada, M. Dong, T. S. Lin, F. Newberg, G. Pottie, W. J. Kaiser, H. O. Marcy, "Wireless integrated network sensors: low power systems on a chip", ESSCIRC '98. Proceedings of the 24th European Solid-State Circuits Conference, The Hague, Netherlands, Sep. 1998.

[7] P. Gupta and P. R. Kumar, "The Capacity of Wireless Networks", to appear in IEEE Transactions on Information Theory.

[8] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols", ACM/IEEE Int. Conf. on Mobile Computing and Networking, Dallas, TX, Oct. 1998.

[9] J. Kulik, W. R. Heinzelman, H. Balakrishnan, "Negotiation-based protocols for disseminating information in wireless sensor networks", ACM/IEEE Int. Conf. on Mobile Computing and Networking, Seattle, WA, Aug. 1999.

[10] J. D. McLurkin, Masters Report, University of California, Berkeley, Dec. 1999.

Chapter 2 - Preliminaries

I. DEFINITION OF A SENSOR NETWORK

For this work, the mathematical abstraction of a sensor network is that of an undirected graph. Nodes of the graph represent sensors in the network while edges represent bi-directional communication links. The latter are characterized by two nodes having the ability to send information to one another and may be defined as having noisy properties. Nodes have computational ability that is physically limited but these limits will not be considered restrictive for the algorithms discussed. Nodes are situated at a physical position (often in the x_1-x_2 plane) and are able to sense physical properties (e.g. temperature, magnetic field) at their location.

To define a sensor network, we require the following information:

- {V,E}: a set of n_v (often abbreviated as n) vertices and n_e edges
- x_i , i =1 to n: the node positions in $\mathbb{R}^2$ or $\mathbb{R}^3$
- T_i , i = 1 to n: the sensor readings at each node in $\mathbb{R}$

The simplification that communication is possible between two nodes if their physical separation is less than the "maximum communication distance" R will be used. A connected graph is one wherein any two nodes have an uninterrupted path between them following edges of the graph. In the physical context, this means that any two nodes can communicate with a finite number of hops between them. Only connected graphs are considered - plural subgraphs with no communication link among them can be considered independently for this work.

To compare one sensor network with another, the following parameters are useful:

- [$x1max$, $x2max$] : A rectangular measure of the size of the area containing the network in $\mathbb{R}^2$
- n : The number of sensor nodes in $\mathbb{N}$
- R : The maximum communication distance in $\mathbb{R}$
- r_{ab} : The distance between node ' a' and node 'b' in $\mathbb{R}^2$
- $2n_e/n$: The average connectivity of a node in the network in $\mathbb{R}$
- D : The distribution of nodes in the network (e.g. random, at grid locations, Gaussian about a center)

II. CREATION OF A SENSOR NETWORK

The method of creation for a sensor network depends on the variables described above. The simplest case is a network with nodes at regular intervals, in which case nodes are simply placed at grid (or hexagon, etc.) locations. A more subtle technique is required to form a network with random node locations while preserving connectedness. Three methods for simulated network creation are presented, each with some restriction on true randomness. However, any of the methods discussed will be considered sufficient for simulating random networks in this study.

A. *Random seed node networks*

One method of network generation places a seed node at the origin. An angle is selected at random in [0 2π] and a distance at random in [0 *R*]. The second node is then placed at the selected angle and distance from the first. The third node is placed according to the same criteria relative to either of its predecessors (selected at random) and so forth. The downside to this method is that nodes situated near the origin (typically those placed first) will generally have increased connectivity relative to those farther away (placed later). This results in a normal distribution of connectivity in the network - in order to simulate fairly, the node names should be shuffled following placement to ensure that an algorithm testing node 1, for example, does not study an artificially high connectivity. This placement method allows for the specification of *n* and *R* with the remaining network properties illustrated in Figure 1. This method generates a distribution Gaussian in the x_1 and x_2 directions with a mean of zero (i.e centered at the initial seed node). For a 100-node network built from random nodes, a standard deviation empirically determined to be near $0.72R$ in both x_1 and x_2 is obtained.

B. *Sequential seed node networks*

A variation places a node exclusively relative to its direct predecessor (i.e. node 4 placed relative to node 3) in a 2D random walk. This has the effect of spreading the network out over a larger geographical area, maintaining a Gaussian distribution centered at the origin and with a standard deviation of $1.55R$. As shown in Figure 1, the mean number of connections per node is reduced by sequentially building the network.

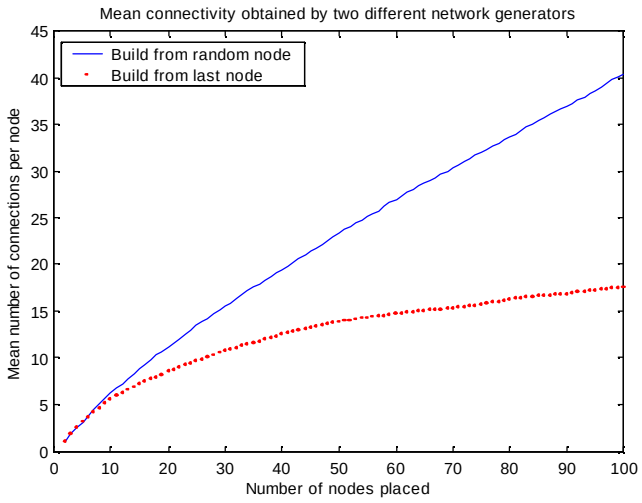


Figure 1. Illustration of the choice between high and low average connectivity for the two methods of generating a connected graph.

C. *Subnetwork extraction*

An alternative method for random placement involves placing nodes one at a time with random x_1 position in [0 x_{1max}] and x2 position in [0 x_{2max}]. From this set, *R* determines the edges of the graph. The largest connected subgraph is then used as the {V,E} set for the sensor network. This allows for specification of the area and *R* but not *n* and generally displays a lower average connectivity than the previous method. The effect of node density on the number of subnets is summarized in Figure 2 and Figure 3.

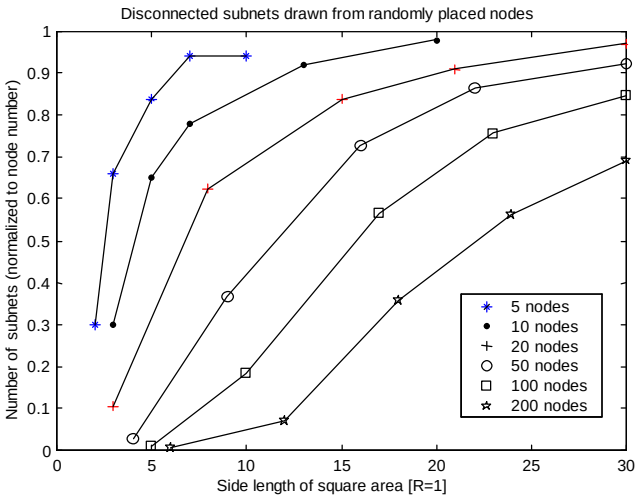


Figure 2. As the size of the placement area increases with the same number of randomly placed nodes, the number of disconnected subnets grows. At the lower extreme, only one subnet is formed with all the nodes being connected. Each data point represents 10 randomly generated networks.

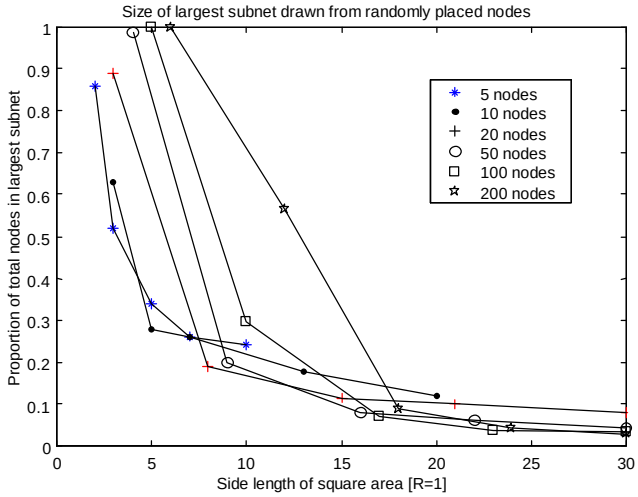


Figure 3. The size of the largest subnet decreases as the dimension of the placement area increases. At the lower extreme, almost all the nodes belong to the largest subnet. Each data point represents 10 randomly generated networks.

The data presented in the previous two figures is expanded for the 20 node case in Figure 4. In this plot, the lines from both previous plots are refined and plotted concurrently.

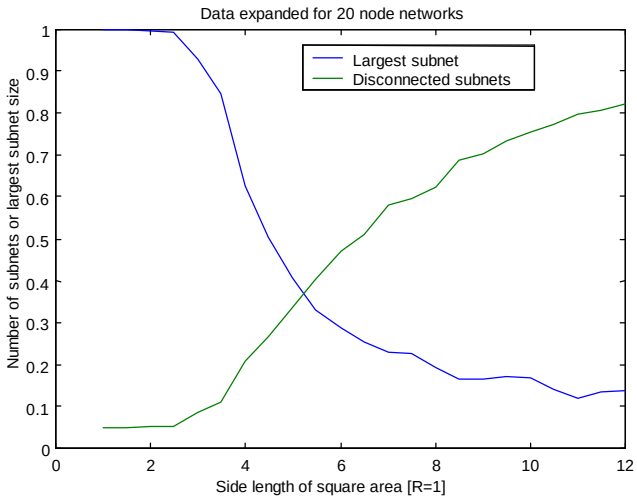


Figure 4. The largest subnet data is examined more closely for a 20 node network. The transition period of the curve is between square side lengths of 4*R* and 8*R*. Below this, almost all nodes are connected in one large network. Above this region, most nodes are isolated.

There is a catastrophic decrease in network connectivity as the area of node distribution increases. An alternate view is that there is a minimum density of randomly distributed nodes that must be present throughout the region to statistically expect to maintain a highly connected network. When random networks are used in this study, they will either be chosen from the *dense* region to the left of the transition or from the *sparse* region in the transition. To the right of the connectivity calamity, the network is too weakly connected to warrant attention. The selection of the largest subgraph introduces some systematic trend in the distribution corrupting it from being totally random.

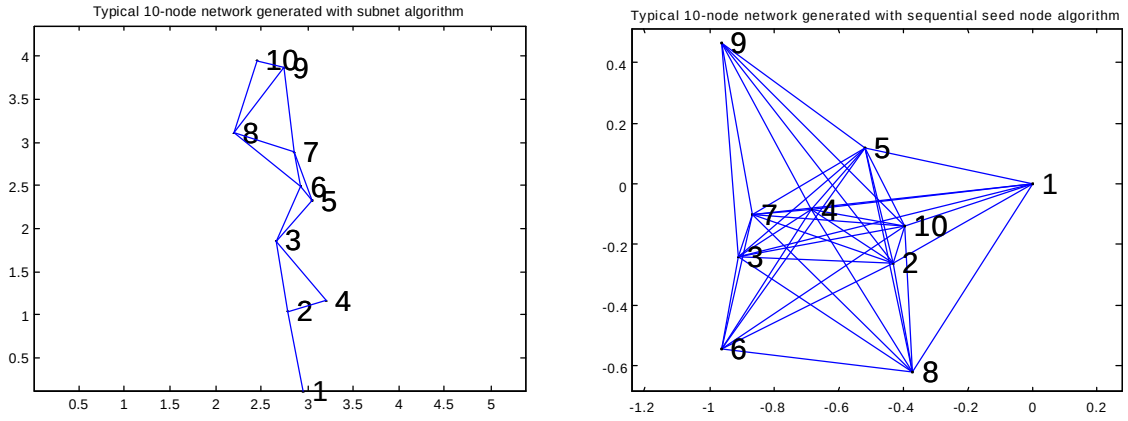


Figure 5. Connectivity obtained with the two different types of network-generation algorithms. In (a), low connectivity is obtained by selecting a 4x4 area with 15 nodes and extracting the largest *R*=1 subnetwork. In (b), the parameters of 10 nodes and *R*=1 are given and the resulting connected network is generated with high connectivity. The regions are drawn with different scale.

III. UNIQUE NODE NAMES

In simulation, there is a temptation to assume qualities of a problem that make it easier to manage; one must take care not to assume away many of the details that make the physical implementation of a problem difficult. One such example in our case is the notion of nodes having a unique name to break symmetry in the network. It is an easy oversight to simply name nodes from 1…*n* in simulation without thought, but solving this in general is perhaps a cumbersome task. A solution is to hardcode a unique name

into each node of the system, but this may be difficult from an organizational perspective. Another acceptable solution is to have the network assign unique names in a distributed sense. To create a minimal spanning tree in an asynchronous network, a problem equivalent to naming nodes, a lower bound on message rounds is $O(n \log n)$ as developed in [11]. A more direct solution is to have nodes pick names at random during their initialization phase.

Providing that a large enough name sample size is taken, the probability of any two nodes selecting the same random name will be small. The issue of randomness should be addressed – taking samples from the least significant bit of any sensor should fulfill this requirement. The size of the name space must be established at start-up, how long should the names be? This problem is isomorphic to the "birthday paradox" [12]. In this problem, the author considers the expected number of people required before any two have the same birthday. In our problem, the expected number of nodes that can be named before a duplicate name is randomly generated from a possible set of m choices is:

$$E\{n\} = 2 + \frac{m-1}{m} + \frac{(m-1)(m-2)}{m^2} + \cdots \frac{(m-1)(m-2)\cdots 1}{m^{m-1}} \qquad (1)$$

For names of length 2^m, a plot of E{n} is shown in Figure 6.

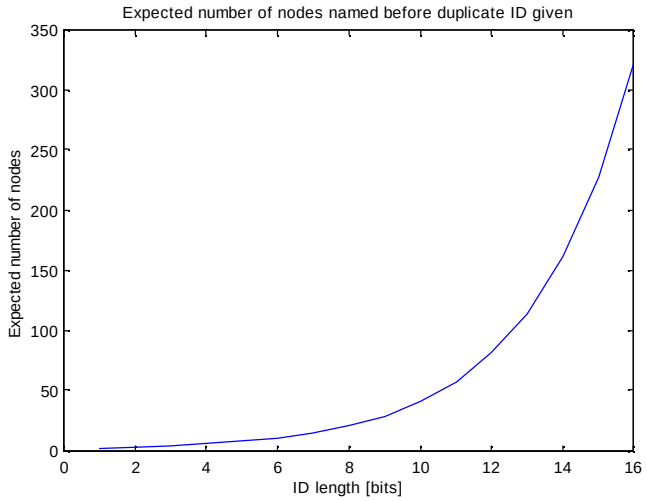


Figure 6. Expected number of nodes required to obtain a duplicate ID in the birthday paradox.

The magnitude of this result is somewhat surprising – from 2^{16} name choices, on the average, it will take only about 320 random selections before a duplicate choice is made. The probability of duplication can be made arbitrarily small with large enough names, but names are so prolific in the network communication (possibly part of every single transmission) that any overhead should be carefully avoided. A more intelligent method might provide some conflict resolution for duplicate names. In this case, we could tolerate larger duplication probabilities – perhaps extending the possibility of a crisis to having 4 or 5 nodes with the same name. The analysis of the birthday paradox to this higher level of collision is developed in [13].

The issue of node names will be disregarded from this point forth – much of the other theory presented may not scale to the upper bound established by a 16-bit name.

[11] N. Lynch, Distributed Algorithms, Morgan Kaufmann, 1996.
[12] M. Klamkin, J. Combin. Theory, vol. 3, pp.279-282, 1967.
[13] P. Flajolet, D. Gardy, L. Thimonier, "Birthday paradox, coupon collectors, caching algorithms and self-organizing search", Discrete App. Math., vol. 39(3), pp.207-229, Nov. 1992.

Chapter 3 - Position Estimation

I. INTRODUCTION

A. Problem definition

In a network of thousands of nodes, it is unlikely that the designer will determine the position of each node. In an extreme case, nodes may be dropped from the air and scatter about an unknown area. To process sensor data, however, it is imperative to know where the data is coming from. Nodes could be equipped with a global positioning system (GPS) to provide them with knowledge of their absolute position, but this is currently a costly (in volume, money, and power consumption) solution. Instead, positional information can be inferred from connection-imposed proximity constraints. In this model, only a few nodes have known positions (perhaps equipped with GPS or placed deliberately) and the remainder of the node positions are computed from knowledge about communication links. A less general attempt at solving for node positions within grid-located known beacons is proposed by Bulusu *et al.* [14].

This chapter describes feasible solutions to the position estimation problem using convex optimization techniques. In the network shown in Figure 7, the solid nodes represent known positions (data) while the open node positions are estimated (variables). If one node can communicate with another, a proximity constraint exists between them. As a physical example, if a particular RF system can transmit 20m and two nodes are in communication, their separation must be less than 20m. These constraints restrict the feasible set of open node positions. Only planar networks are considered, but augmenting the methodology to 3-D is straightforward. In summary:

Given: positions of solid nodes
Find: a possible position for each open node
Subject to: proximity constraints imposed by known connections

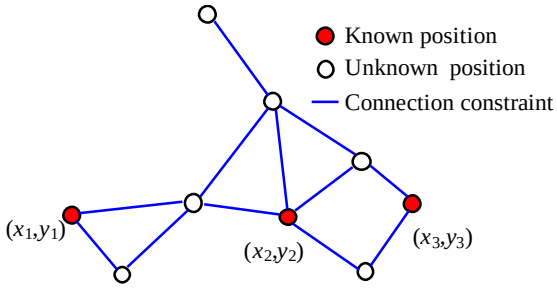


Figure 7. Graph illustrating data and variables as vertices, constraints as edges.

Formally, the network is a graph with n nodes at the vertices (each node having a Cartesian position) and with bi-directional communication constraints as the edges. Positions of the first m nodes are known $(x_1, y_1, \ldots x_m, y_m)$ and the remaining n-m positions are unknown. The feasibility problem is then to find $(x_{m+1}, y_{m+1}, \ldots x_n, y_n)$ such that the proximity constraints are satisfied. Note that there are constraints among open nodes though their positions are unknown.

B. Mathematical tools

Efficient polynomial-time algorithms based on interior-point methods exist for solving linear programs [15] and semidefinite programs [16]. A linear program (LP) is a problem of the form:

$$\begin{aligned} &\textit{Minimize} && c^T x \\ &\textit{Subject to:} && Ax = b \end{aligned} \qquad (1)$$

Geometrically, this amounts to minimizing a linear function over a polyhedron.

A generalization of the LP is the semidefinite program (SDP) of the form:

$$\begin{aligned} &\textit{Minimize} && c^T x \\ &\textit{Subject to:} && F(x) = F_0 + x_1 F_1 + \ldots + x_n F_n = 0 \\ & && Ax = b \\ & && F_i = F_i^T \end{aligned} \qquad (2)$$

The inequality represents a matrix inequality on the cone of positive semidefinite matrices, i.e. the eigenvalues of $F(x)$ are constrained to be nonpositive. This is known as a linear matrix inequality (LMI). Again the objective function must be linear for SDP. Constraints can be stacked in either method. SDP will be sufficient to solve all the numerical problems in this paper though LP will be used where applicable because of superior computational efficiency.

In two dimensions, each node has a position (x,y). For position estimation, a single vector with all the positions is formed representing x in (1) and (2):

$\boldsymbol{x} = [x_1 \; y_1 \ldots x_m \; y_m \ldots x_{m+1} \; y_{m+1} \ldots x_n \; y_n]^T$
The first m entries are fixed as data and the remaining n-m are computed by the algorithm.

In general, efficient computational methods are available for most convex programming problems. Geometrically, a convex set is one for which any two points in the set can be connected with a line entirely contained in the set. Convex constraint models for RF and optical communication systems are presented in section II. Other convex constraints are also computable with the same algorithms.

II. CONVEX CONSTRAINT MODELS

A. Connections as convex constraints

Providing that the connectivity of the network can be represented as a set of convex position constraints, the mathematical methods outlined in section I can be utilized to generate a feasible position for all the nodes in the network. It is sufficient to consider connection constraints individually as both LP and SDP methods allow for multiple constraints to be collected into a single problem. The question becomes: "given the position of node A, what is the set of possible positions for node B?" The remainder of this Section is devoted to two models of this feasible set.

B. RF communication – radial constraints

The RF transmitter of a wireless sensor node can be modeled as having a rotationally symmetric range as illustrated in Figure 9a. This is not an accurate physical representation of what is often a highly anisotropic and time-varying communication range, but we can always use a circle that bounds the maximal range. Furthermore, the proceeding methods apply equally well to ellipses without increased complexity should it become evident that an elliptical communication model is more relevant.

In this symmetric model, a connection between nodes can be represented by a 2-norm constraint on the node positions. More specifically, for a maximum communication range R and node positions a and b, the equivalent LMI is given by:

$$\|a-b\|_2 \le R \Rightarrow \begin{bmatrix} I_2 R & a-b \\ (a-b)^T & R \end{bmatrix} \ge 0 \qquad (3 - \textit{Fixed radial constraint})$$

where I_2 is the 2x2 identity matrix. For a two-dimensional problem, using Schur complements [17], the quadratic inequality transforms into an LMI with a 3x3 matrix in (3). Multiple LMIs can be stacked to form

one large SDP for the entire network. Hence, each bi-directional proximity constraint contributes one convex 3x3 LMI to the system.

A variant of this problem is to use the exact distance between nodes as the outer distance bound in the above LMI:

$$\begin{bmatrix} I_2 r_{ab} & a-b \\ (a-b)^T & r_{ab} \end{bmatrix} \geq 0; \quad r_{ab}\ given \qquad (4 - \textit{Variable radial constraint})$$

Physically, transmitters varying their output power during an initialization phase could obtain an estimate of r_{ab}. If a connection is first obtained at a power P_o, the receiver calculates the maximum possible separation for reception at P_o. This maximum separation $r_{ab} < R$ can be used to determine a tighter upper bound on each individual connection in the network.

Note that neither of the following are convex constraints:

$$\|a-b\|_2 = r_{ab}; \quad \|a-b\|_2 > R \qquad (5)$$

The latter would be useful in "pushing away" nodes that are not connected in the algorithm as in Figure 8. This constraint is not physically realistic either, nodes within a certain range may not be able to communicate due to a physical barrier or transmission anisotropy. Even greater precision can be obtained if some lower bound on separation is known and the former constraint in (5) is formulated as a set of robust convex constraints.

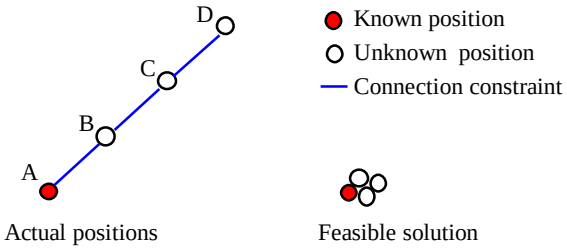


Figure 8. There is no mechanism in the radial constraint model for bounding nodes away from known positions. As illustrated, the entire node chain could feasibly collapse to a point as the unknown nodes B,C and D lie in a feasible set of circles of radii *R*, 2*R* and 3*R* centered at A.

C. Optical communication – angular constraints

In the optical communication realm, consider sensor nodes with laser transmitters and receivers that scan through some angle. The receiver rotates its receiver coarsely until a signal is obtained, and then finely to get the maximum signal strength. By observing the angle at which the best reception occurs, we can estimate the relative angle to the transmitter but not the distance between them. This results in a cone (triangle in 2D) for the feasible set as in Figure 9b. This cone can be expressed as the intersection of three half-spaces, two to bound the angle and one to place a distance limit. The intersection of half-spaces is still an LP.

D. Other constraints

Any combination of the SDP and LP constraints can be used to define individual feasible position sets. Some study was devoted to a quadrant detector scheme (Figure 9c) involving one LMI and two scalar linear constraints. In a more general case, for a trapezoid with variable angles and width (Figure 9d), four linear constraints can approximate a segment of an annulus that might represent uncertain knowledge of both position and angle. There are other non-LMI constraints that may also be useful constraints.

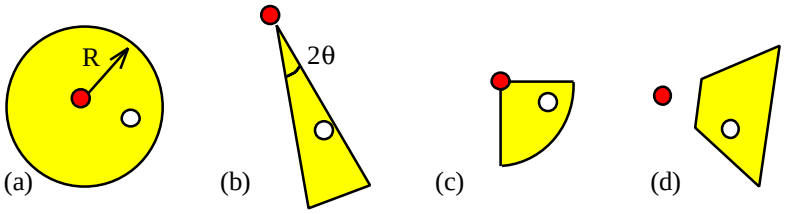


Figure 9. Geometrical interpretation of single constraints. Given that the solid and open nodes are connected, the open node must lie in the shaded region anchored by the solid node position. Constraint shown in (a) radial, (b) angular, (c) quadrant and (d) trapezoid. The outer bound in (b) is optional.

E. Combination of convex constraints

Node positions in the network are often constrained by connections to several other nodes. Satisfying plural constraints means that the feasible set becomes the intersection of the individual constraint sets, necessarily making the feasible region smaller with each added constraint as in Figure 10. The intersection of convex sets is itself a convex set, so our search methodology continues to be sufficient.

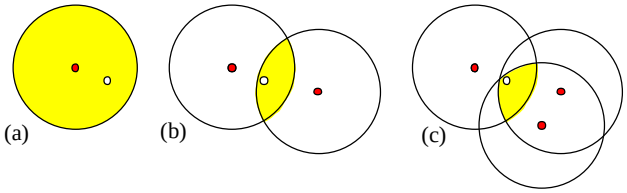


Figure 10. Combination of radial constraints. The shaded region represents the feasible set for the light node, constrained by the dark node positions. From (a)-(c), the intersected constraints yield progressively smaller feasible sets.

Though only connections between known positions and unknown positions are illustrated in Figure 10, unknown-unknown connections are equally important to the problem solution. In this case, both *a* and *b* are variables in the SDP or LP. All unknown positions are solved for simultaneously using a single global program. A summary of the individual constraint methods is given in Table 1.

Method	Allowable angle	Allowable distance	Linear inequalities
Fixed radial	$0 - 2\pi$	Variable	1 LMI
Variable radial	$0 - 2\pi$	Fixed	1 LMI
Angular	Variable	Fixed	3 scalar
Quadrant	One quadrant	Fixed	1 LMI, 2 scalar
Trapezoid	Variable	Variable	4 linear

Table 1. Parameters for each constraint method considered.

III. SIMULATION

A. Test setup

All LPs are solved with the MATLAB optimization toolbox while SDPs are solved with *LMItool* [18]. The position estimation methods outlined in the previous sections are intended for use in networks where each node can communicate bilaterally with each of several adjacent nodes. For the radial constraint model, networks of 15 nodes are generated with the random seed node algorithm from the preliminary discussion. Typically, these networks involve about 130 radial constraints (i.e. the networks have an

average connectivity near 9) which currently pushes the computational limits of LMItool[1]. With the lower computational load of LPs, the angular constraint model is tested on networks of 30 nodes.

B. Performance metric

As mentioned in section I, both the SDP and LP admit linear objectives ($c^T x$) exclusively. However, there is no readily apparent linear objective that would provide any sort of "optimal" solution to the placement problem. Instead, c in (1) and (2) is defined to be either +1 or –1, randomly determined for each entry. This has the effect of selecting a *random* feasible point $\boldsymbol{x}_{est} = (x_{est}, y_{est})$ from the solution space – this point represents a set of (x,y) pairs for each unknown position. The most precise statement of a node's position that can be made is that the node lies somewhere in the intersection of the allowable regions. Providing that these regions are small enough, finding a feasible position for all the nodes may be close enough for the required estimation.

The performance of the algorithm is defined as the mean error from the computed to the actual unknown positions. This mean error provides a measure of the size of the feasible set.

$$error = \frac{1}{n-m} \sum_{i=m+1}^{n} \left\| x_{est}^{i} - x_{real}^{i} \right\|_2 \qquad (6)$$

C. Bounding the feasible set

To effectively exploit the utility of the objective function $c^T x$, the algorithms can be run multiple times. This provides a mechanism for bounding the feasible set with a rectangle parallel to the axes. The computation proceeds as follows for *each* unknown position k:

1) Set $c_{2k-1} = 1$ and all other $c_i = 0$ for c in (1) or (2)
2) Solve the original SDP/LP $\rightarrow$ yields x^k_{min}
3) Set $c_{2k-1} = -1$ and all other $c_i = 0$
4) Solve original SDP/LP $\rightarrow$ yields x^k_{max}
5) Repeat steps 1-4 for c_{2k} to get y^k_{min} and y^k_{max}

This procedure defines the smallest such rectangle that bounds the feasible set as in Figure 11. By selecting the center of this rectangle as the most likely solution, we can expect some improvement in mean error over the randomly selected case.

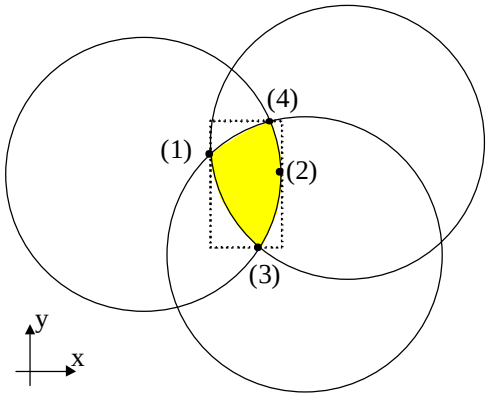


Figure 11. The shaded region represents the feasible set for this problem. The procedure finds points 1-4 that define the tight rectangular upper bound shown in dots. This rectangle runs parallel to the axes shown.

For the price of a $4(n-m)$ times increase in the number of problems solved, an increase in estimation performance and an outer bound on the solution is obtained.

[1] LMItool fails with more than 150 3x3 LMI constraints.

Another possibility is to find the minimum measure elliptical bound on the solution space for each unknown position as discussed in [17]. This does not, in general, provide a tight upper bound due to the problem relaxation, but may provide numerically similar results requiring the solution of a single SDP for each unknown position instead of four.

IV. RESULTS

A. Comparison of two radial constraint methods

We analyze the performance difference between the fixed radius and variable radius RF location methods. Using a 15-node network, the following test is performed:

1) Select nodes 1-3 as known positions
2) Solve for the remaining 12 unknown positions
3) Compute the mean error for these 12 positions from the actual network
4) Increase the number of known positions by 1 (hence decreasing the unknowns by 1)
5) Repeat steps 2-4 until only one unknown remains

Results of these trials are summarized in Figure 12.

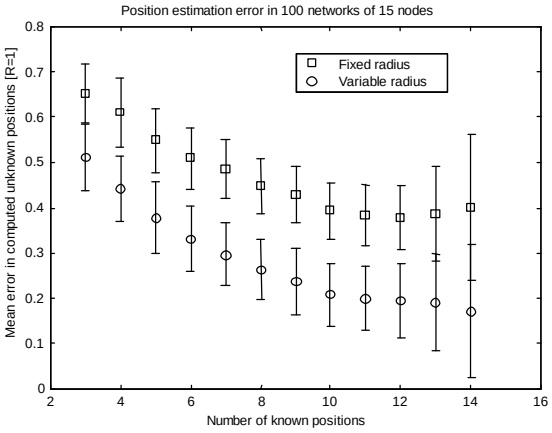


Figure 12. Results of radius location method averaged over 100 different test networks. Error bars indicate one standard deviation from the mean.

As a performance comparison, consider a simple case of two nodes, one position being known. The second node is somewhere within a disc of radius *R* around this node – a point picked at random will have an expected distance error of 2/3 *R*. This is approximately the error for the fixed radius case with a low number of known positions. As data on the positions increases to about 10 known positions, the mean error continues to decrease. With more than 10 known positions, mean error no longer decreases. Providing sufficient positions are known, the connections among unknown positions are equally valuable to decreasing the size of the feasible position set. It is not known why the standard deviation of the unknown positions increases with the number of known positions.

There is a significant performance increase with the variable radius method. This suggests that it is worth the effort to improve distance estimation either by measuring received power directly or by modulating the transmission power through a few discrete steps.

B. Comparison of different angular uncertainties

The same methodology is employed to test the angular estimation model. A visualization of the simulation output is given in Figure 13 with only 10 nodes shown for clarity.

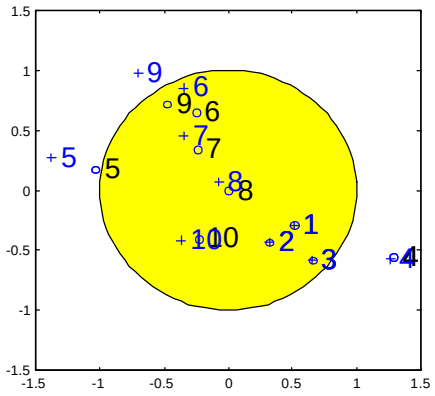


Figure 13. Illustration of a trial with *n* = 10, *m* = 3 and a p/100 half-angle constraint. Circles indicate actual positions, plus signs indicate estimated positions. The shaded region is the communication distance *R* from node 8 illustrating the nodes by which node 8 is constrained. Mean error for this case is 0.19 *R*.

The half-angle is selected from π/10, π/100, π/1000 for the size of the constraint. A varying number of nodes are taken as data as indicated on the horizontal axis of Figure 14 with the mean placement error of the unknown positions represented on a logarithmically scaled vertical axis.

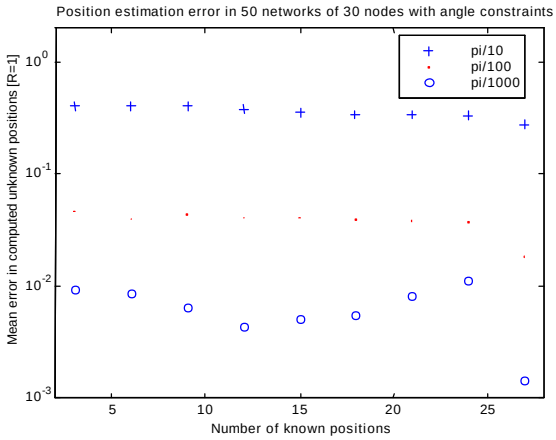


Figure 14. Results of angular estimation method averaged over 50 different test networks.

There is nearly an order of magnitude difference between each level of angular uncertainty. This corresponds roughly to the relative size of the individual feasible areas defined by the three levels. Of significance is the invariance in the mean error as the number of known positions is increased in contrast to the general refinement of the results observed in the radial constraint model. Physically, this indicates that networks employing optical communication and location methodologies require fewer known nodes to effectively place the remaining unknown nodes in the network.

Additionally, the distance limit is not essential to the solution of this set of constraints. To illustrate this, this limit was increased twofold in simulation without significant decay of accuracy 0.0381 *R* versus 0.0397 *R*. For computational purposes, however, an upper bound is useful to avoid numerical problems.

C. Connectivity and mean error

The statement that the intersection of constraints yields smaller feasible sets was made in section II. To verify that the algorithm follows this expected behavior, 100 15-node networks are generated by random seed placement and the connectivity of each node is determined. The variable radius constraint

method is used to place the unknown nodes and the mean error is computed for each node. A summary of the effect that connectivity has on the mean error is given in Figure 15.

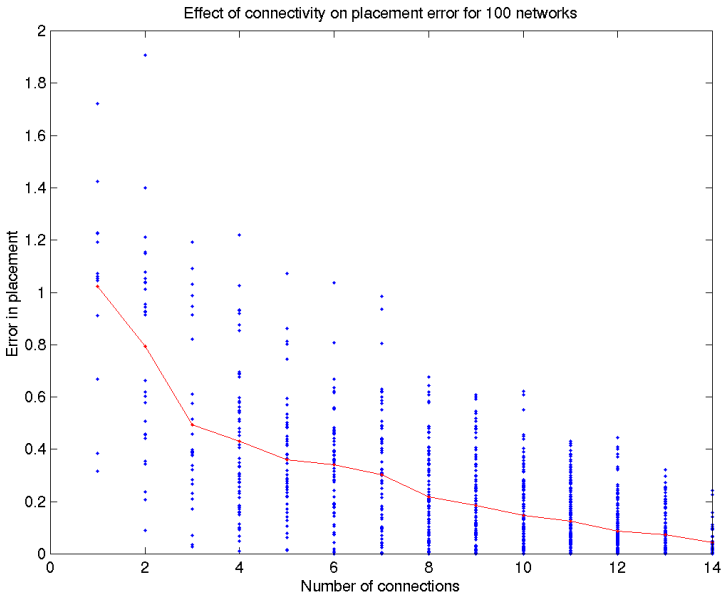


Figure 15. Each dot represents one node with the solid line indicating the mean error for each level of connectivity. All networks are generated with $R = 1$.

As the nodes become progressively more constrained, the mean placement errors decrease. From this data, network density could be designed such that a certain position estimation performance is likely to be achieved. Admittedly, better performance is not without sacrifice – the increased computation required to run the algorithms and communication protocols required to maintain order in such a network may outweigh the aforementioned benefits.

D. Rectangular bounds

The fixed radius estimation method is used to test the validity of the rectangular outer approximation method. For direct comparison with the previous results, the outer bounds are computed for the 15-node networks used previously with 6 known positions. The problem of finding the four Cartesian bounds is formulated as a set of $4(n\text{-}m)$ SDPs discussed in section III. From this rectangular outer bound, the middle is chosen as the "optimal" guess. An example for the angular estimation model is given in Figure 16.

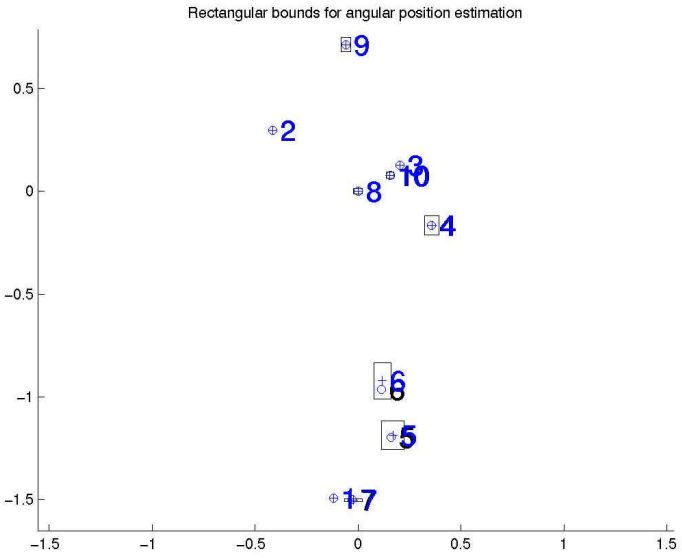


Figure 16. Solution of rectangular bound problem for a 10-node network with 3 known positions (nodes 1, 2 and 3) and an uncertainty of p/100. The rectangles bound the possible positions of nodes 4 through 10 and the estimated positions are at the center of the rectangles.

This example shows how increased connectivity to known positions reduces the feasible set. Nodes 8 and 10 are connected to both 2 and 3 (R=1) and are precisely known. The remaining nodes have connections to only one known position and hence are fitted with larger rectangles. The rectangle centers are surprisingly close to the actual positions and tend to be this way.

Solving these 4(n-m) SDPs and selecting the rectangle center gives position estimates that are nominally slightly better than those picking randomly from the feasible set. Specifically, for the case of m=6 in the previous n=30 analysis, the error is 0.4±0.1 R versus 0.5±0.1 R for the random selection method. The mean area of the bounding rectangles in this case is 1.7±0.8 R^2. The area of the bounding rectangle is correlated with the improvement in performance – smaller rectangles mean that a random point in the feasible set will be closer to the center.

A more significant increase in performance is obtained by using the rectangular center method with angular estimation. In this scenario, mean errors decrease by an order of magnitude under those selecting a random point in the feasible set. Again, however, this involves solving the problem (an LP in this case) 4(n-m) times. The advantage of the accuracy increase must be weighed against the computational increase. Research is underway to determine how the ellipsoidal outer bound compares in terms of performance and computational load.

V. IMPROVEMENTS AND APPLICATIONS

A. Tracking through the sensor network

A specific application of the procedures described is during tracking of an object through the sensor network. The sensing radius can be modeled as in the radial constraint case. If multiple nodes can sense the object, the same set intersection methods via SDP can be utilized to estimate the object's position and provide an upper bound. This is a problem with only one unknown – the position of the tracked object – and m known node positions. The solution should hence be rapid and possibly simple enough to accomplish using the microprocessor of a sensor node. Of course, this can be extended to track k objects concurrently analogous to the k unknown node positions developed previously.

Acoustical data is used by Yao *et al.* [19] to locate an object using sensors. Yao's method uses more specific (phase-related) data and provides more precise results.

B. Hierarchical solution for large networks

With a densely connected 15-node network, the radial constraint method is computationally intensive. This is an inauspicious result for scaling to networks of thousands of nodes. We propose two possibilities: limit the number of constraints on each node or solve the problem hierarchically.

The first option is to impose an upper bound on the number of constraints that will be considered for each node. As illustrated in Figure 15, the performance increase becomes marginal beyond a certain level of connectivity. In essence, we keep the same number of connections in our problem while dividing them up among more nodes. For example, the current limit of 150 radial constraints could solve for networks of 30 nodes if constraints are limited to 5 per node. This does not, however, provide for scaling the solution through orders of magnitude.

The second option is to first divide a large network into smaller subnetworks based on connectivity data – nodes connected to one another will likely be members of the same subnetwork. Position estimation can be carried out for each member of the subnetwork based on an unknown centroid of this region. Following the individual estimations, the subnetwork centroids can be abstracted to nodes in the larger network and placed accordingly with another iteration of position estimation. With plural hierarchical steps, this method scales to arbitrarily large problems.

All methods would benefit from exploiting the sparsity of the connectivity matrices to reduce computational requirements.

C. Setting known positions

The placement of the known positions has not been addressed. This is important, particularly for the radial constraint model. As mentioned in section II, there is no propensity for pushing unknown nodes away from the known positions. Hence, we cannot expect nodes to be constrained outside of the convex hull defined by the known positions. In a network, we would thus benefit from placing nodes around the periphery of the area of interest. It is these nodes that should be GPS-equipped or be manually located at known positions.

With the results obtained for mean error in networks of randomly placed known and unknown positions, it is now possible to compare other node distribution schemes. Intuitively, it seems that networks with nodes around the perimeter or in a regular grid pattern should lead to better position estimation.

D. Implementing continuous distributions

The feasible sets are currently defined as binary fields – either a node is permitted to be at a position (x,y) or it is not. An extension of this theory could be made to continuous distributions. Based on a certain power reading, there may be a Gaussian distribution of the most likely distance between nodes. Solution of the placement problem would then give probabilistic distributions for all unknown node positions. From this, the most likely position can be determined as well as confidence intervals.

E. Erroneous data management

There is currently no faculty for the detection of erroneous connections. If a spurious proximity constraint is fallaciously reported, the algorithm will, in general, fail. Testing for such an error is as difficult as solving the position estimation problem outright.

VI. REFERENCES

[14] N. Bulusu, J. Heidemann, D. Estrin, "GPS-less low cost outdoor localization for very small devices", Technical report 00-729, Computer science department, University of Southern California, Apr. 2000.

[15] N. Karmarkar, "A new polynomial-time algorithm for linear programming", Combinatorica, vol. 4, pp. 373-395, 1984.

[16] Y. Nesterov, A. Nemirovskii, Interior-Point Polynomial Algorithms in Convex Programming, SIAM, 1994.

[17] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, Linear Matrix Inequalities in System and Control Theory , SIAM, 1994.

[18] L. El Ghaoui and J.-L. Commeau, LMItool, http://www.ensta.fr/uer/uma/gropco/lmi/lmitool.html

[19] C. W. Reed, R. Hudson, K. Yao, "Direct joint source localization and propagation speed estimation", 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Phoenix, AZ, Mar. 1999.

Chapter 4 - Correlation Coding

I. INTRODUCTION

The exploitation of correlated data from closely situated sensors to reduce overall transmission bandwidth is the focus of this chapter. In one application scenario for a sensor network, we expect sensors in the same geographical region to have very similar data – the benefits of using a large number of simple sensors instead of a single more complex sensor include higher theoretical SNR and more robust data collection. When correlation exists, it is wasteful to transmit complete data from each node – this chapter outlines a method of transmitting reduced levels of data while maintaining acceptable levels of overall performance.

One proposal to exploit correlation is to perform data fusion in a distributed manner within the network; a selected node queries its neighbors and creates one aggregate reading (ostensibly possessing a higher SNR) for the region. This method is not limited to the fusion of a single type of data, but may instead incorporate readings from an eclectic melange of sensors to come to a conclusion independent of centralized computation. The canonical example for this approach is to fuse temperature, acoustic and video data to detect an intruder [21]. The caveat to fusion is that data is lost before leaving the network. In some situations, this sacrifice is unnecessary.

Instead of fusing data within the network, we consider instead using implicitly assumed correlation statistics to allow for reduced average message lengths and apprising the external controller of *all* sensor readings. Fusion for sensor networks is prodigal – why bother with thousands of readings if only aggregate signals are ever to escape? One benefit of the miniaturization of sensor nodes lies in the ability to resurrect the data in its entirety and to throw out information intelligently once a global perspective is obtained. The gains proposed in this chapter allow for more information to be communicated outside the network without increased bandwidth requirements. Two coding methods are presented. The first is a lossy compression scheme not requiring any inter-node communication. The second requires some inter-node communication but results in more compression and lossless data recovery.

II. MATHEMATICAL BACKGROUND

A. Coding for blind correlation

The sensor information encoding is based on the correlated data coding scheme presented in [20]. In particular, the scalar quantization model is of interest – the model assumes two correlated information strings are to be sent, one from each of two transmitters. If transmitter A sends the full information, the correlation between the messages allows for reconstruction of the both messages even if transmitter B sends an abridged version of its message. For example, A might send 3 bits of information with B sending only 2 bits. The method of minimizing erroneous message recovery for an equal probability of any message is presented below.

For transmitter B, there are $2^3 = 8$ possible messages enumerated as in Figure 17:

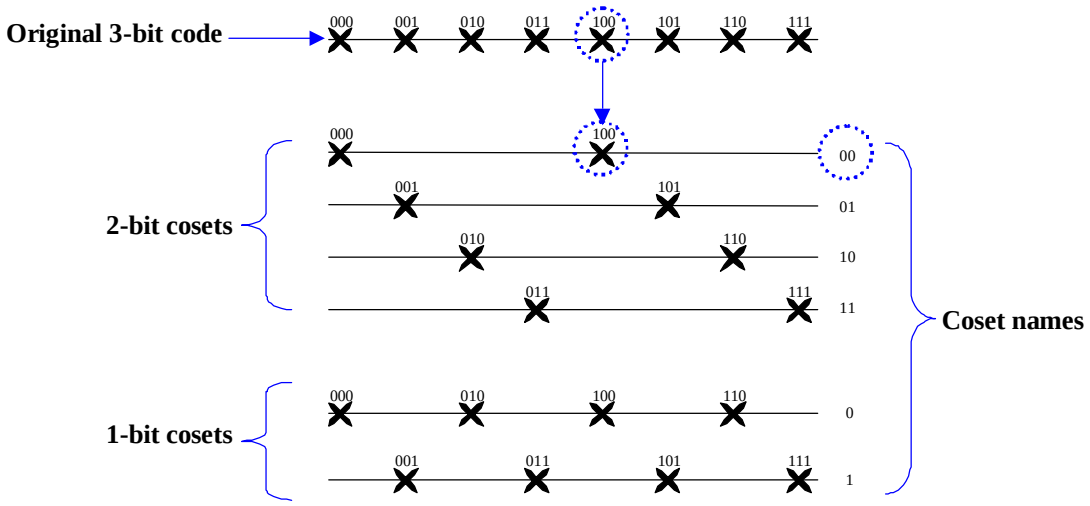


Figure 17. Encoding a 3-bit message as either 2 bits or a single bit. The shorter name of the coset, shown on the right, is transmitted instead of the original 3-bit message. In the illustrated example, the message 100 is encoded as 00.

If B has the message 100 to send with 2-bit encoding, it searches the cosets for 100 and finds it in the 1st, 00. B transmits 00 to the receiver, which then knows that B's original message was either component of the first coset, 000 or 100. The receiver then compares the two options to the 3-bit message sent by A and picks the closer of the two as B's message. For 1-bit encoding, B would send only a '0' and the receiver would then choose the closest of 4 choices to A's message.

It should be emphasized that A and B do not need to communicate in this model, hence the term *blind correlation*. The receiver obtains the messages independently, this is useful to an ad-hoc routing scheme that may have A and B situated in geographic proximity but rarely exchanging messages.

This scheme extends to larger message sizes. For message lengths of k, A sends a message of length k and B sends one of length k-l, $l < k$. The maximum value of l depends on the degree of correlation between the messages. The messages must be correlated to within one-half of the coset member distance to be accurately recovered by the receiver. This distance is defined to be the *differentiation distance* between compressed codewords. A larger value of l will result in a larger quantity of errors, each of equal or greater severity than a smaller value of l. If the noise level is of the same magnitude as this differentiation distance, the data is reconstructed erroneously. This coding scheme should be implemented only when noise is small compared to the level of correlation. To expound, if two nodes are expected to remain within a temperature range $?T$ of each other, the coset size should be chosen such that the differentiation distance is larger than $?T$ and the noise level must be far below $?T$.

This scheme also extends to larger numbers of nodes. In a direct first-order extension, suppose that all of n nodes in a particular region have correlated messages. Only a single node must communicate the full k-bit message; the others transmit (k-l) bit messages. This reduces the overall message length propagated from nk to nk-(n-1)l. As a higher-order application, consider three nodes, A, B and C. Suppose that A and B are sufficiently correlated to permit accurate reconstruction of B's message with k-l_{AB}, but that C is not this correlated to A. Now C might send k-l_{AC} for $l_{AC}<l_{AB}$ bits, but if C is sufficiently correlated to B, define l_{BC} such that $l_{BC}>l_{AC}$. The receiver first decodes B's message based on A, then C's message based on B.

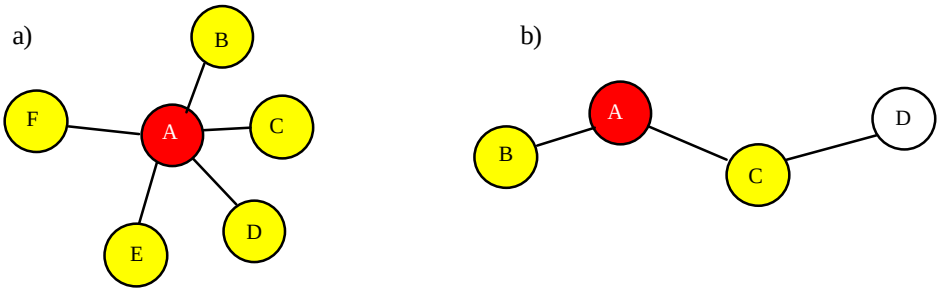


Figure 18. Graphical representation of correlation. Nodes are transmitters with edges indicating sufficient message correlation for reduced message size. In a) the external receiver gets a *k*-bit message from A which is used to decode the (*k*-*l*) bit messages from B through F. In b) the receiver first decodes messages from B and C based on A, then decodes message D based on C.

Of course the coset members have been chosen based on the assumption that all messages are equally probable. More extreme probability distributions on message selection could conceivably result in different coset grouping to minimize the chance of an erroneous decoding at the receiver end.

B. Selecting correlation groups based on connectivity

Now that correlation can be exploited to reduce message size for most nodes in the network, the next issue is how to determine candidates for correlated messages. The metric used for this study is one of connectivity. Just as in node location, we interpret the information that two nodes are within communication range as a suggestion that they are close – now this carries the added suggestion that the nodes will have similar sensor readings. Alternatively we could use the estimated positions from the solution of the location problem and define a certain radius within which we deem sensors will be correlated. Nodes within this radius of each other become "connected" for the purposes of this chapter, a notion then equivalent to the original proposal. Once the overall network connectivity has been determined, an austere heuristic suffices to create the correlation groups:

1. Create two disjoint sets, each containing node names and initialized to the null set: the *parent set* contains one node for each correlation group; the *one-hop set* contains all nodes connected to at least one member of the parent set
2. Select the seed node which can communicate with the external controller as the first element in the set of parent nodes
3. Add all nodes connected to the parent set's recent addition to the one-hop set
4. The next selection to the parent set is the node that will add the most new nodes to the one-hop set. Remove this node from the one-hop set. The new nodes are informed of their parent in the parent set.
5. Repeat steps 2-3 until all nodes are either part of the parent set or the one-hop set

With this construction, the nodes in the parent set will send full *k*-bit messages to the controller. Nodes in the one-hop set will send (*k*-*l*) bit messages and are assumed to be correlated to their parent node in the parent set.

To extend this algorithm to a 2^{nd}-order correlation model (or beyond) as discussed in section II, another set, the two-hop set, must also be maintained with parents in the one-hop set. The 2^{nd}-order correlation between these two sets is exploited after the one-hop to parent set correlation has been used to reconstruct the one-hop data.

This selection procedure is not a perfect algorithm; it does not necessarily return the smallest cardinality parent set possible for the network. However, it is often very close to the limit. A comparison with the brute force combinatorial search for the minimal parent set will be presented in the sequel. Moreover, the parent set is guaranteed to be connected independently of the one-hop set making it a

possible candidate for a communication backbone in distributed routing or other applications. All networks in this chapter use the subnet extraction algorithm for generating connected networks with R = 1.

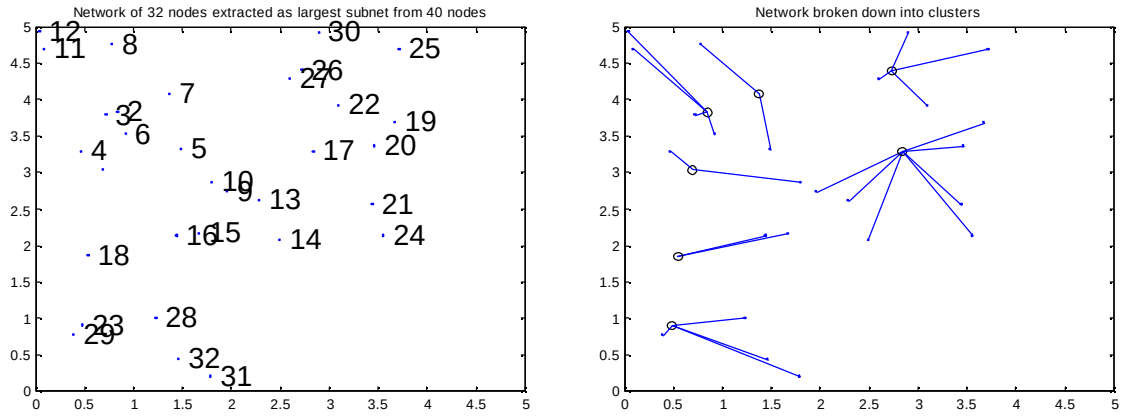


Figure 19. Illustration of the function of the correlation group-forming algorithm, a) shows the node positions and names, b) indicates parent nodes with open circles and one-hop nodes with points. The lines represent the correlation assumption between one-hop and parent nodes.

III. BLIND CORRELATION - SIMULATION AND RESULTS

A. Horizontally increasing temperature

In this simple simulation, the temperature is varied horizontally across a network similar to that in Figure 19. The nodes are placed in a 5x5 square and the temperature at each position is:

$$T(x) = \frac{x}{a} \qquad (1)$$

For simulation purposes, it is beneficial to have temperature in the network limited to the interval [0,1] allowing the sensor encoding for, *e.g*. 8-bits, to be 0 ? 00000000 and 1 ? 111111111. In this case, a =5. In general, larger values of a will give more correlation between the sensor readings.

For each of 2,4,6 and 8 bits, the temperature reading at each node is quantized from the continuous-valued function. At each of the one-hop nodes, the quantized value is placed in the appropriate coset for a compressed code. The compressed code message length is varied from 1 bit to the sensor bits less one. For example, an 8 bit simulation varies compressed code length from 1 to 7. The appropriate coset is the only information kept from each sensor. The quantized value from the parent node of each one-hop node is used to find the closest member of the coset. This member is the best estimate of the temperature and is compared with the original temperature measured at the node's location to obtain a measure of error at each node. Even with perfect reconstruction from the correlated parent value, there is still quantization error in each reading. Results from this experiment are shown in Figure 20.

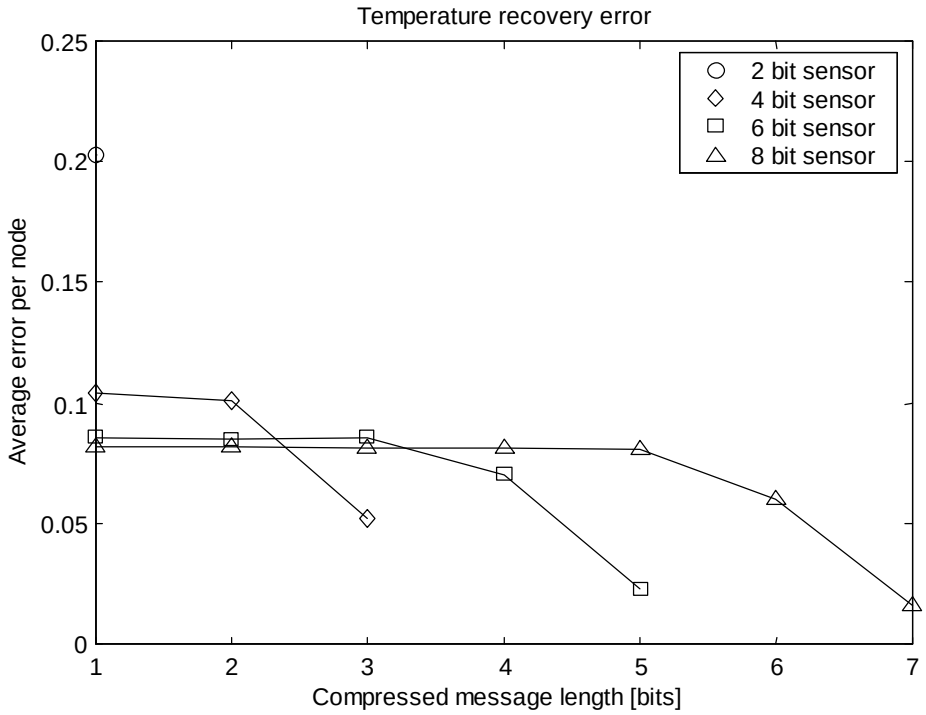


Figure 20. Average temperature error after message compression and correlation recovery. Each data point represents the average of each node for each of 20 networks. For a sensor of *b* bits, the compressed message is at most *b*-1 bits.

As intuition suggests, at each level of sensing precision, the average error decreases as the compressed code length is increased. More surprising is the steady-state value that is approached as the code length is decreased. It appears that a 1-bit message (2 cosets) for an 8-bit sensor is equally effective as a 5-bit message (32 cosets). This suggests that most errors in the network are sufficiently large to impact both of these methods identically. This is possibly due to the fact that errors all result from the same distance gap – in this example, the parent node and one-hop nodes are separated by at most a distance of 1, or by a temperature of 0.2. This maximum temperature difference results in an upper bound on the error and the plateau seen in Figure 20, particularly apparent in the error for the single 2-bit data point.

B. Spatially periodic temperature field

In the previous section we saw that the correlation-coding scheme does allow for accurate reconstruction in a special case. Now we consider what particular property that temperature field possessed and how these results can be generalized. The important parameter of the temperature in this scenario is how rapidly the temperature can change over one communication radius. One means of determining the effect of this spatial variation is by applying a bounded periodic function to the sensor network with varying frequency. A commensurate function mapping to the interval [0,1] is chosen:

$$T(x) = \frac{1}{2}\left[1 + \cos(ax)\right] \qquad (2)$$

Now by varying a, the amount that the temperature can change over a certain distance can be modulated. Results for different values of a are given in Figure 21.

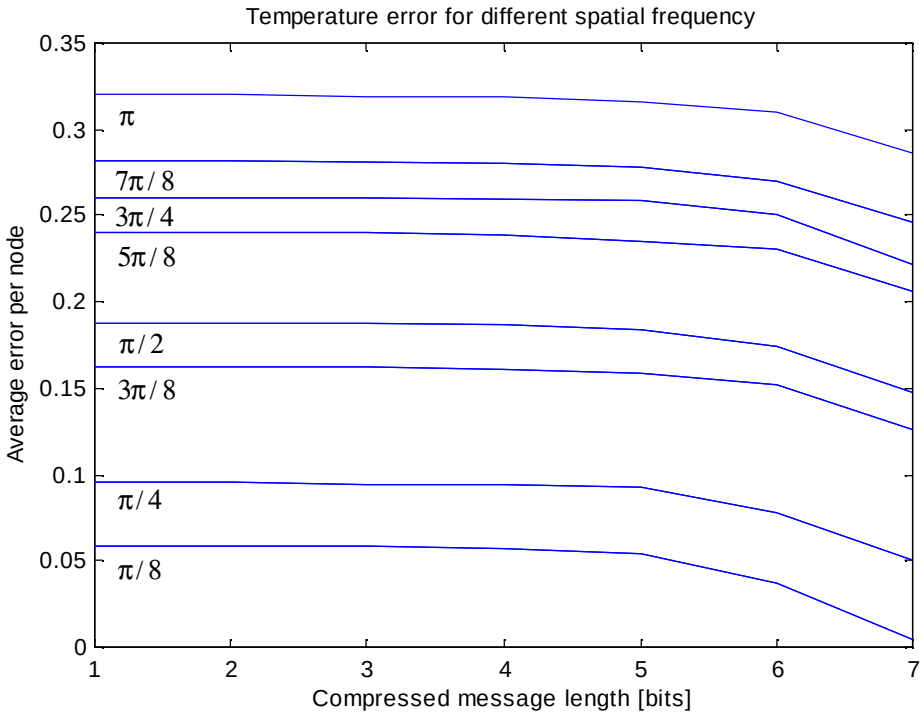


Figure 21. Different spatial frequency of the temperature field leads to varying levels of sensor reading recovery. For each curve, the sensor measures a cosine temperature field with frequency indicated below the curve and with 8-bit precision. Each curve represents the average over 50 subnetworks.

As an aside, consider the selection of two random numbers in the interval [0,1]. The expectation value of the absolute difference of these two numbers is 1/3. This is precisely the value approached by the uppermost curve in Figure 21, indicating that the recovered one-hop readings are as erroneous as would be expected by simply choosing the value randomly. It can hence be stated that in this scenario, correlation-coding is at its theoretic nadir. Values of $a > \pi$ retrace the same curve. Also to note is that all curves would drop to the quantization error value at 8 bits though the algorithm does not return this value.

The plateau at higher levels of compression can no longer be mistaken for an artifact of the first temperature field considered. Surprisingly, the values of the plateau represent *the error that would result if each one-hop node were assigned the temperature reading of its parent*. Hence, sending 5 bits from each one-hop node is scarcely more relevant than ignoring the readings entirely and utilizing only those from the parent nodes. The only performance increases appear to occur for small compression, sending 6 or 7 bits. Depending on the scenario, this may not be significant enough to justify the additional overhead of the correlation-coding techniques. The same phenomenon is observed when the sampling precision is increased to 16 bits.

The lack of effectiveness over a large frequency range is the dirge of the correlation-coding method. This raises the question of what the optimal means of encoding sensor readings is in the network. The blind correlation precept is dismissed to allow for a more complex set of compression techniques. Are better results obtainable if the one-hop nodes are aware of the sensor reading of their parent?

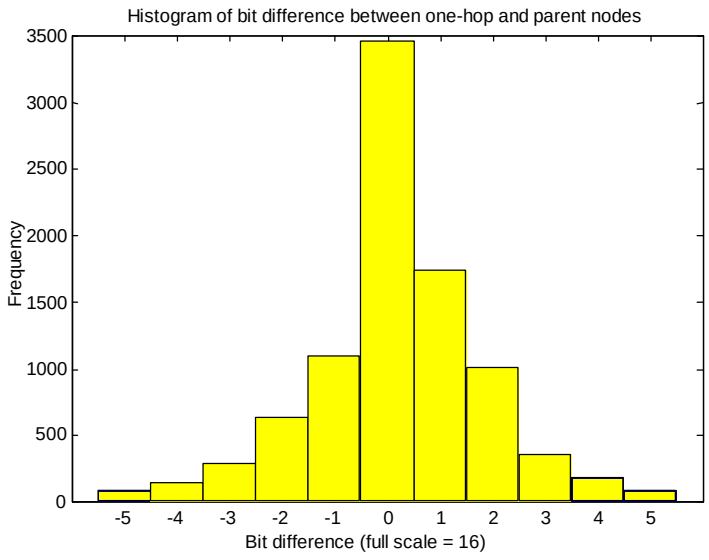
IV. SEMI-BLIND CORRELATION – SIMULATION AND RESULTS

Consider the case where the parent node broadcasts its sensor reading. Every node in the one-hop set will be able to receive this transmission (by definition). Based on this data, the one-hop nodes can encode their sensor readings more effectively than in the blind correlation regime. Still, the parent node

does not need to know the readings of each one-hop sensor in its correlation group, so effectively only one extra broadcast message per correlation group is required to allow for this new set of experiments. This is defined as *semi-blind correlation*.

An effective means of exactly encoding the one-hop node's sensor reading is by representing its difference from its parent's reading. The correlation assumption states that smaller differences will be more likely to occur. In order to code this with minimal message lengths, the more frequent messages should have lesser code lengths. The Huffman code addresses this precise issue. To generate the optimal Huffman code, the relative frequency of codewords must be determined. As a case study, the spatially periodic temperature field with $a = \pi/2$. This is a case that the blind correlation scheme could not adequately improve.

In simulation, each node senses its temperature to 4-bits (to keep the number of codewords manageable). One-hop node quantized values are then compared with the quantized values of their parents. The difference can be any integer in the interval [-15,15] with the limits being attained when readings of 1111 and 0000 are obtained in the same correlation group. Results of this experiment illustrate that the effective range of values is considerably less as shown in Figure 22.



Value	Freq.	Huffman Code	Simple Code
-5	77	1000010	100
-4	144	100000	010
-3	289	11101	000
-2	637	1111	10
-1	1095	110	00
0	3466	0	0
1	1747	101	1
2	1004	1001	01
3	360	10001	11
4	182	11100	001
5	81	1000011	011

Figure 22. The frequency of bit-differences when using a 4-bit code. Values are summed from 500 networks. The table summarizes the frequency of the histogram bins as well as the proposed codewords described below.

The histogram suggests that only codewords in the interval [-5,5] need be considered for the parameter $a = \pi/2$. The asymmetry of the distribution is mildly disconcerting, but is not of current interest, instead we take the distribution as correct and compute an appropriate Huffman code. The code is computed as detailed in [22]. This code has the added benefit of having a unique decoding scheme for an arbitrary concatenation of codewords, despite their varying lengths. If this property is not required, the codeword length can be assigned sequentially from the smallest using a naive code. The major difference is that the number of bits encoded would have to be sent with the simple code. A summary of the results and both possible codes are shown in Figure 22. Of significance is the length of each word. The original system is simulated again with the new codewords. For Huffman encoding, the average codeword length is 2.7 for the one-hop nodes. For the simple code, the average is 1.5. In either scheme, this results in better performance and shorter codewords than possible in the blind correlation process. Note that the particular Huffman code described is not unique – but all such codes are isomorphic. In our example, the simple code requires 2 bits ($\log_2 k$ in general) to identify how many bits are included as data and would jump to 3.5, illustrating the worth of the Huffman code.

V. CONCLUSIONS

This first attempt at coding many sensor readings based on correlation is something of a disappointment. With blind correlation, compression by a single bit is all that is possible in order to maintain a level of error close to the quantization limits. With semi-blind correlation, this can be improved slightly with the use of a minimum length code suitable for the particular expected correlation distribution. The benefit to the latter technique is that if the sensor readings are unexpectedly aberrant, the message length will increase as a survivable consequence. Using the former method, total loss of correlation will not even be apparent and any data collection will be crippled by an information palsy.

The Smart Dust communication scenario is amenable to the semi-blind correlation model with some minor alterations. The interrogator first receives *k* bits of information from a parent node, then interrogates each of the corresponding one-hop nodes *with the parent node's sensor reading.* There is no peer-to-peer communication, but now the one-hop nodes are apprised of the parent value and can encode the difference as detailed in the semi-blind model. There is additional energy expenditure by the sensor nodes to receive this longer interrogative message, however, which may obviate the transmission savings. A more detailed energy analysis is required to characterize the trade-off.

An interesting extension of this work would be to determine exactly how much information is stored by a given number of nodes with given temperature readings. For this study, the search would be for the global minimum sum of bits to communicate all temperatures in the network. From this beginning, the generalization of this analysis to a sensor network-based information theory is a laudable goal, if only to define something called the Shannon-Doherty limit.

Compression algorithms abound have been explored for a variety of purposes, not the least of which are JPEG and MPEG standards for spatial and spatio-temporal compression respectively. Trellis-based coding and symmetric code compression can be used to compress data sequences further [23] in the blind correlation regime. It may be worthwhile to use basis functions as an approximation to a data field with low required bandwidth, the choice of basis functions would have to be appropriate each sensor scenario – e.g. tracking and temperature monitoring may not be possible with the same set of functions. The challenge would be to develop all this in a distributed context, or even an irregular one.

One problem is that even if the message length could be drastically reduced according to the presented methodology, the node names will not be reduced. For example, if 8-bit node names are used, the reduction of the composite name + data might be reduced from 16 bits to 12 bits. The reduction in data size may not be as salient as it initially appears. For communication from nodes to the parent of the correlation group, this may be overcome with channel time-division wherein the parent would know the origin of each message from its arrival time slot.

VI. REFERENCES

[20] S. S. Pradhan, K. Ramchandran, "Distributed Source Coding Using Syndromes: Design and Construction", IEEE Computer Society Conference on Data Compression (DCC'99), Snowbird, UT, pp. 158-167, Mar. 1999.

[21] W. R. Heinzelman, A. Chandrakasan, H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks", Hawaii International Conference on System Sciences, Maui, HI, Jan. 2000.

[22] Grupo de Tratamiento de Imágenes, http://www.gti.ssr.upm.es/~vadis/faq_MPEG/huffman_tutorial.html.

[23] S. S. Pradhan, K. Ramchandran, "Distributed source coding: Symmetric rates and applications to sensor networks", IEEE Computer Society Conference on Data Compression (DCC'00), Mar. 2000.

Chapter 5 - Tracking in a Sparse Network

I. INTRODUCTION

This chapter describes the use of a wireless sensor network for tracking an object through the network and identifying its movement parameters. In this scenario, we are concerned with a sparse sensor scenario – when a small number (usually 0 or 1) sensors can detect an object at any given point in the network. Detection of an object may physically arise from sound, magnetic, light or acceleration sensors. From the geographical location of the sensors observing the object, the trajectory or path of the object is reconstructed.

In the following discussion, a trajectory refers to the time-dependent motion of the object through the network while a path consists solely of the locus of points traversed along the trajectory. The Expectation-Maximization (EM) algorithm is implemented with linear models and augmented with time variables to improve performance.

The number of sensors detecting a path is linear in the total sensor number n and in the sensing radius of each node as illustrated in Figure 24. When selecting the sensing area of a node (or equivalently the node density), it must be large enough so that ample points are available to locate a path while being small enough to reduce the uncertainty in the points. A node detects only that the object is in its sensing radius, not where within this area it lies.

Simulations summarized in this chapter explore the requirements in node density and sensing radius to track objects through the sensor network. The algorithm is intended mainly as a tool to explore the intricacies of the network properties, not as an exposition of itself. Specifically, how much data is required to solve the tracking problem consistently and what type of sensor network can provide this data?

II. MATHEMATICAL BACKGROUND

A. The EM algorithm for path reconstruction

Reconstruction of a single path is accomplished with line-fitting techniques, a least-squares regression is probabilistically optimal in the 2-norm sense. For an object passing through the network in a straight line, linear least squares is applicable. We seek the line that best describes the object's path according to the positions of the sensors that are activated. The abstracted problem equates a path with a line and path-finding is equivalent to line-fitting. Least-squares analysis extends readily to more tortuous curves including arbitrarily large polynomial and basis function fits. Such analysis alone is however not sufficient for identifying plural paths simultaneously.

The EM algorithm provides a mechanism for extending the concepts of least-squares analysis to plural path reconstruction. The algorithm was proposed by Dempster *et al.* [24], and an introduction as applied to line-fitting is given by Weiss [25]. The algorithm alternates between so-called "estimation" and "maximization" steps, each assuming that the other has perfectly solved its part of the problem. The iteration is easily explained through an example – consider two paths through a network each generating a set of disjoint points. To describe the two lines, the parameterization $y = ax + b$ is sufficient for either line. Given the composite set of points $\{(x,y)\}$, we calculate the line parameters a and b for both lines as follows:

1) Start with random a and b for each line
2) Expectation: probabilistically assign each point to the line that best describes it
3) Maximization: apply line-fitting individually to each line given the points found in step 2
4) Repeat steps 2 and 3 until the parameter values converge

This procedure can fit any number of lines simply by assigning a set of parameters to each line. The algorithm as applied in this study requires a priori knowledge of the number of lines.

B. The Expectation step

The expectation step is given a set of l pairs $\{(a,b)\}$ completely describing the l lines. For each point (x,y), a measure of the distance to each of the lines is calculated. In the linear least-squares model, this *residual* from point i to line j is:

$$r_i^j = a^j x_i + b^j - y_i \qquad (1)$$

This is simply the distance along the y-axis from the point to the line. Next, the probabilistic membership of this point in each of the l lines is computed with the *softmin* function:

$$w_i^j = \frac{\exp(-[r_i^j / s]^2)}{\sum_k \exp(-[r_i^k / s]^2)} \qquad (2)$$

where s represents a measure of expected fit and will be discussed in the sequel. Each weight w represents the probability that the point i belongs to the line j – as expected, the weights for a point summed over all lines is unity. For intuition, if a point lies vertically in between two lines, $w_1 = w_2$ and we say, "there is equal probability that the point belongs to either line". If the point is much closer to line 1, $w_1 \sim 1$ and $w_2 \sim 0$ and we say, "the point almost certainly belongs to line 1". The output of the expectation step is the (i x j) component set of weights.

C. The Maximization step

The input to the maximization step is a set of weights. For each line j, a weighted least squares computation is performed. Points that more likely belong to the line j are given higher weights (larger w) in the fitting.

$$\begin{pmatrix} \sum_i w_i x_i^2 & \sum_i w_i x_i \\ \sum_i w_i x_i & \sum_i w_i \end{pmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_i w_i x_i y_i \\ \sum_i w_i y_i \end{bmatrix} \qquad (3)$$

The solution of this linear equation provides the parameters a_j and b_j. After completion of the procedure for every line, the parameters are passed back to the expectation step. Experimentation suggests that five iterations of the EM steps are appropriate for solving the problems at hand – five iterations are used throughout.

D. Determining the number of lines

Determining the number of *models*, independent motions in the image, required to adequately describe a process as applied to video signals is proposed by Weiss [26]. With this methodology, the number of models is dynamically updated as required based on the statistically expected form of the collected data. If a point does not fit a model closely enough according to the image statistics, a new model is introduced to explain this inconsistency. Convergence is not guaranteed; in the pathological case, a distinct model could be assigned to each pixel in the image. In our scenario, we have no such assumptions on data statistics and our procedure is correspondingly ingenuous.

The EM algorithm is first run to full convergence with the assumption that there is one line. The error is computed as the mean of the lowest absolute residual for each node for each iteration. For a number of fitting lines l:

$$E_l = \frac{1}{n} \sum_i \left| \min_j (r_i^j) \right| \qquad (4)$$

The EM algorithm is then run incrementally – first attempting to fit the data to one straight line, then two, and so on. The loop terminates when the addition of another line to the data does not significantly reduce the overall error, at which point we back up a step and take the previous solution. If the difference in E_l and E_{l+1} is less than 10%, the decision is made to pick the l-line fit.

III. SIMULATION

A. Defining the network and determining EM data

In previous testing, it was discovered that random networks and grid-based node placement are equivalent for the path estimation problem. With this in mind, nodes are placed randomly in the prescribed square area. Without loss of generality, we consider only squares of 100x100 units. The parameters required to completely specify the network are:

n – the number of nodes
$\{(x_i,y_i)\}$ – the set of all node positions
s – the sensing radius of each node (sensing area is taken as circular centered at the node)

To generate a set of points corresponding to a sensed path, an object is initialized at the edge at the square area and given a constant "velocity." At each time step, the position of the object is computed and each node is checked to see if the object is within its sensing radius, i.e. if $|x_i - x_{obj}|<s$. If this inequality holds, the node "senses" the object and is flagged. When all objects' paths through the network are complete, the set of flagged node positions is the data input to the EM algorithm. If either one or zero nodes are flagged, line-fitting is not possible and the trial is deemed unsuccessful.

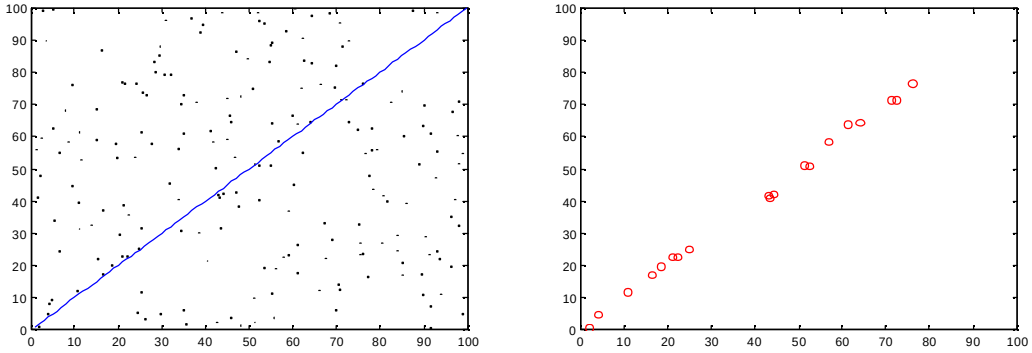


Figure 23. Nodes are flagged by an object traversing the network. In a) the line represents the object's path and the dots are 200 nodes in the network. In b) any node within s=2 of the path is flagged for the curve fitting algorithm.

As stated in section I, the number of nodes flagged is linear in both n and s (Figure 24). As such, the fixed network size is not restrictive; a 50x50 square with s reduced by one half is equivalent.

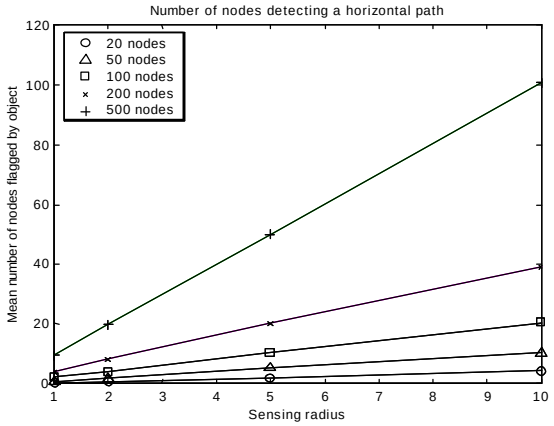


Figure 24. The number of nodes flagged by an object passing through the network is linear in both sensing radius and node density. This same graph format will be used throughout this chapter to summarize data for the same set of variables. Each data point represents 50 simulations.

B. Parameters to test

Specifically, we are interested in how five independent parameters affect the performance of path estimation:

1) *n*
2) *s*
3) The number of paths
4) Individual path characteristics (*a* and *b*)
5) Collective path characteristics (separation, angle difference)

The tests performed will attempt to discover how networks should be designed to obtain sufficient performance in anticipated situations. An example of the EM algorithm output is shown in Figure 25.

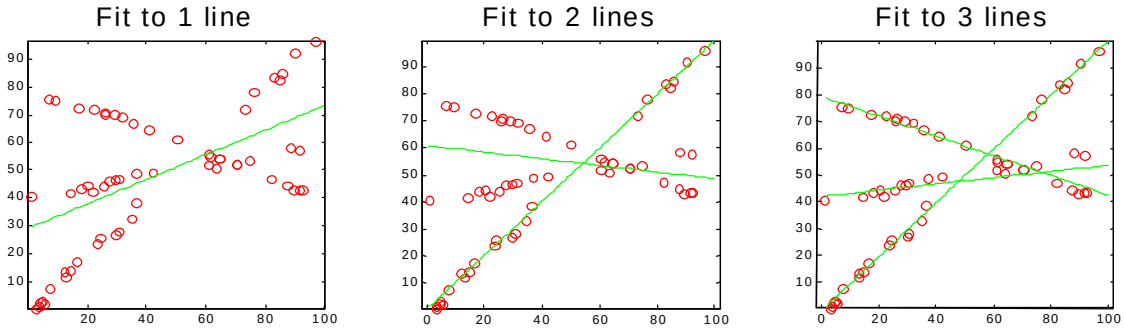


Figure 25. EM algorithm fits 1, 2 and 3 lines (solid line) to points (open circles) generated by 3 distinct paths. An additional line does not significantly reduce the error so the algorithm reports three. Data shown is for *n*=500 and *s*=2. Performance in this case was above average – typical fits are not as accurate.

IV. RESULTS

A. Single path results

Three classes of single paths are considered while varying *n* and *s*: paths along the x-axis, paths along the y-axis and diagonal paths. In the horizontal path trials, the object enters the network at (0,50) and travels in the positive x direction. A "successful" estimate is when the algorithm preferentially fits the points to a single line – the alternatives being that it is unable to fit a line (fewer than two nodes flagged) or that the error criteria prompts the algorithm to fit to multiple lines. Ostensibly, the algorithm will return the same parameters as a simple linear least-squares fit for one line, but it does not know a priori that the search is for only a single line. A summary of the results is plotted in Figure 26. In general, the results show better performance with more nodes.

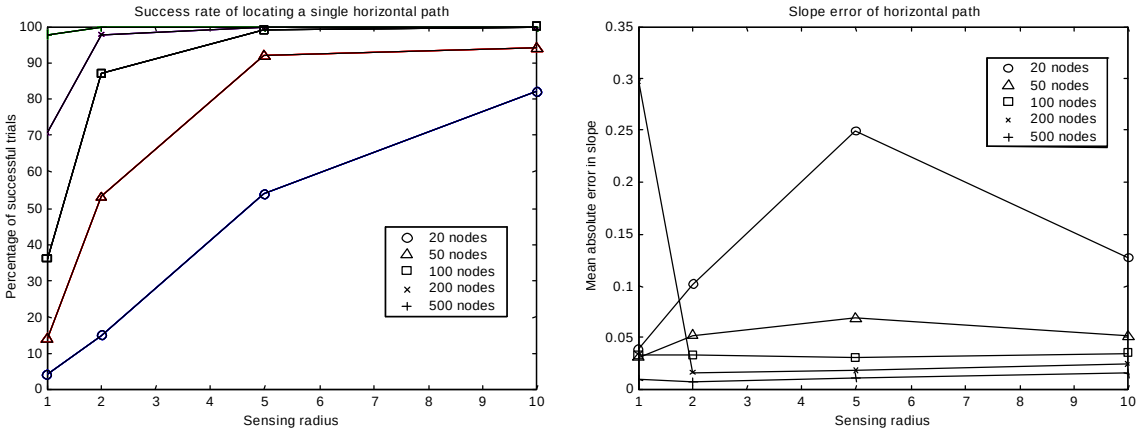


Figure 26. Performance of the algorithm for single horizontal line trials. In a) percentage of trials in which the algorithm correctly returns a single line, b) the error in slope estimation. Each data point represents 100 trials.

It is enlightening to examine the failure modes of the algorithm as identified in Figure 27. At low sensing radii, the path often flags fewer than two nodes and the algorithm cannot identify any lines with this information dearth. At larger sensing radii, the algorithm is more likely to conclude that several lines are present due to the large residual in the data. Perhaps in between s=5 and s=10, there is an optimal sensing radius for the network where the dark part of the bar in Figure 27 is maximized.

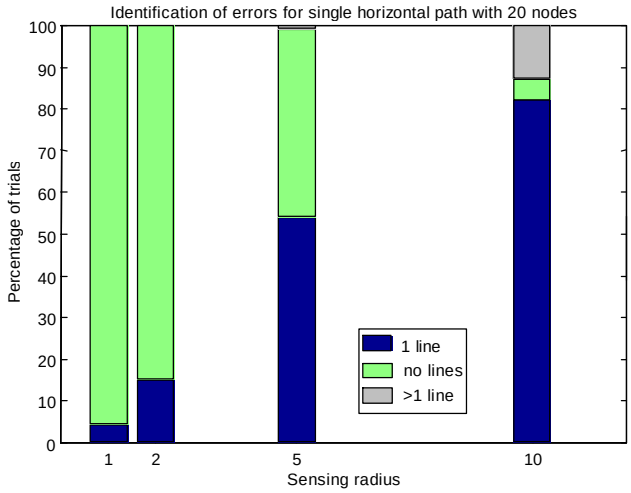


Figure 27. Expansion of data for the 20 node trials in Figure 26. The algorithm either returns the correct number of lines (*i.e.* one), zero lines, or more than one line.

Similar results are obtained for a diagonal trajectory. The success rate and error are slightly improved because the diagonal path through the environment is longer and typically activates more nodes than the horizontal path, thereby providing more data for the EM analysis and greater statistical certainty.

Identification of a single vertical trajectory is less successful. The error in the vertical case is much larger than for a horizontal motion, an artifact of the fitting formula used in the EM algorithm ($y = ax + b$). Numerically, the algorithm preferentially fits the slope to a zero value than to an infinite value. The optimization of the line-fitting algorithm is merely a matter of coordinate choice and is largely peripheral to the current discussion. Therefore, further simulation simply avoids such factious trajectories.

B. Dual path results

The single path analysis does not directly apply to multiple path location. Not only are the individual path characteristics influential to algorithm performance, but the measure of dissimilarity among the paths is pivotal. Intuitively, the closer the paths are geographically, the less chance that the algorithm may be able to distinguish them. The eagerness to divide data into more lines is a function of the expected fit quality s sidestepped earlier. The higher the s, the more likely the algorithm will make all lines exactly the same – in the example of 3 lines, a high s will result in 3 internecine lines with identical slopes and intercepts. As an attempt to quantify some aspects of multiple path behavior, the interrelationship between two paths and the corresponding performance is explored in the following trials. The algorithm again has no knowledge of how many paths have passed through the network.

As seen in single path analysis, diagonal lines are most readily identified. The most conducive intersecting dual path scenario to successful identification should hence be lines with ± 1 slope originating at the corners of the network area. Success requires that two lines are returned and furthermore that their computed slopes are within 20% of the actual values. A summary for this test using the same variable values as in the single-path trials is presented in Figure 28.

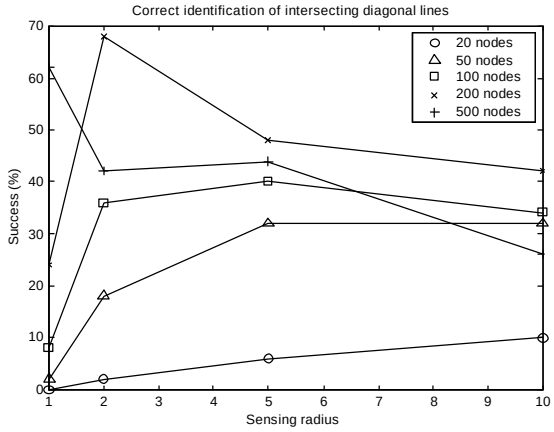


Figure 28. Success rate of accurately parameterizing intersecting diagonal lines. Surprisingly, the best results are obtained for an intermediate node density and an intermediate sensing radius. In all cases, s=10.

The same scenario is used to determine the possibility of changing s to obtain better results. Varying s from 0.1 to 100, it becomes apparent that the initial choice of 10 is fortuitous – there is a range between 3 and 35 outside which the algorithm fails consistently. The peak performance is obtained around the s=10 level, so the plots in Figure 28 reflect the best results.

The next experiment tests how close lines can be and still be separately identified. Horizontal paths through the center of the network are simulated with varying vertical separation. In the event that two lines are detected, the computed gap is compared to the actual path separation. A similar experiment is run for paths intersecting at varying angles. Example trials are illustrated in Figure 29; results from both experiments are shown in Figure 30. The distribution of points in these plots is not yet fully understood, particularly the absence of points near the center of the plot between angles of 50-80 degrees. The angle of 45 degrees corresponds to the n = 200, s = 2 data point of Figure 28.

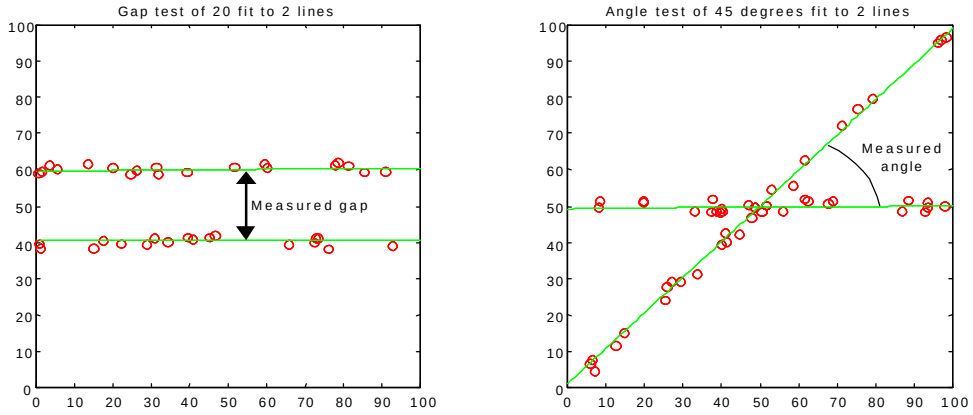


Figure 29. Simulation output for a) gap test and b) angle test. The indicated lines are those returned by the EM algorithm, closely matching the actual parameters in this example. The examples shown use n = 500 for illustrative purposes.

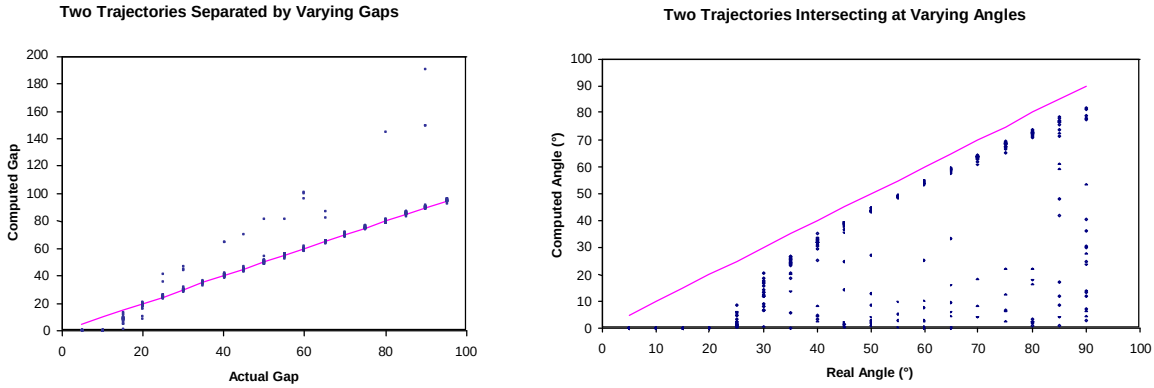


Figure 30. Line dissimilarity affects algorithm performance. In a) horizontal lines separated by less than 20 units are difficult to separate while greater separation leads to a general performance increase and possible gap overestimation. In b) angles of intersection are consistently underestimated and distinction is poor below 25 degrees. Each data point is one trial, all networks have n=200, s=2, s=10. The lines represent target values.

These results give insight into what network parameters should be chosen to distinguish among similar tracked paths. Based on these data, the sensing radius should be chosen lower than one tenth of the desired gap resolution. No corresponding statement can be made to increase angular resolution beyond that shown – presumably smaller sensing radii would yield crisper paths that could more readily be distinguished.

Even with high node density, small sensing radii, and an optimal s, identification is below 70% successful. This suggests that the algorithm is either not suited for the current problem, or that more information is required to improve performance. The additional information comes in the form of adding another dimension to the data, time.

C. Time-enhanced EM solution

With the addition of time to the problem, nodes are not only flagged when the object enters their respective sensing areas, but also record the time at which the object is first sensed. From this data, trajectories are reconstructed – the time-series of the path is now computed. The new problem follows a similar formulation with the additional time data at each flagged node, t_i. Each object location is now characterized by the triple (x_i,y_i,t_i). The models to fit for each line (indexed by j) are now of the form:

$$x^j(t) = x_o^j + V_x^j \cdot t$$
$$y^j(t) = y_o^j + V_y^j \cdot t \qquad (5)$$

Here V_x and V_y represents the x-ward and y-ward velocities, respectively. The intercept with the x-y plane occurs at (x_o,y_o). All four of these parameters are required to characterize the trajectory in three-space. At each data point i, the residual to line j is given by:

$$r_i^j = x_i^j + y_i^j - x_o^j - y_o^j - V_x^j \cdot t - V_y^j \cdot t_i \qquad (6)$$

This is perhaps a naive construct, but it succeeds in representing the discrepancy between the data and the line-fitting. In a perfect fit, the residual vanishes at all location points/times. The corresponding weights are computed from the residual via the *softmin* function in (2).

The maximization step computes the best guess at the four parameters in the now familiar manner. While augmented in dimension, the linear equation takes the same form:

$$\begin{pmatrix} \sum_i w_i t_i^2 & \sum_i w_i t_i \\ \sum_i w_i t_i & \sum_i w_i \end{pmatrix} \begin{bmatrix} V_x & V_y \\ x_o & y_o \end{bmatrix} = \begin{bmatrix} \sum_i w_i t_i x_i & \sum_i w_i t_i y_i \\ \sum_i w_i x_i & \sum_i w_i y_i \end{bmatrix} \qquad (7)$$

The new formulation is transparent to the algorithm which proceeds in exactly the same manner.

Visualization of the results is now more difficult, watching 3D plots on a flat screen is significantly more recondite than the 2D data of Figure 25. Quantification of error is correspondingly mired; does interest lay primarily in reconstructing the time progression of the trajectory or rather in exploiting the additional data to distinguish among previously inseparable trajectories? To an extent, the two are correlated – but the focus will be directed towards improving x-y line-fitting with the knowledge of time.

In short, the addition of time to the algorithm does succeed in improving performance. As a means of comparison with the previous method, we can compare the slope a with V_y/V_x and the intercept b with the $y(t)$ such that $x(t) = 0$. As indicated in Figure 30a, the algorithm performs poorly below a threshold gap distance of 20 between the paths. As an indication of the new possibilities, this gap is reduced to 10 (only 5 times a sensing radius) and additionally, the second trajectory is subject to some time delay before entering the network area. An example output of the algorithm is shown in Figure 31. Results are summarized in Figure 32a. Similarly, the previously undetectable intersection angle of 20° meets time-based improvements in Figure 32b.

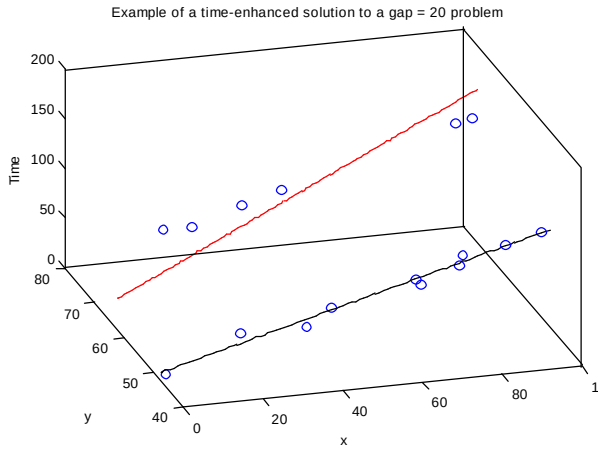


Figure 31. Time-enhanced solution output. The actual trajectory velocites are V_x = [1,1] V_y = [0,0] while the computed velocites are V_x = [0.8,1.0] V_y = [0.1,0.0]. The delay for the second path is set to 50.

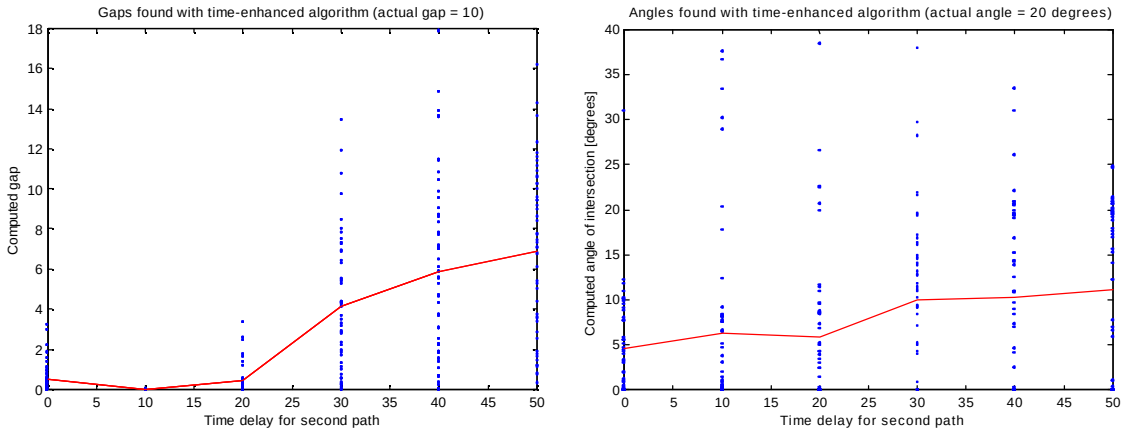


Figure 32. Delaying the second trajectory allows the time-enhanced algorithm to separate two formerly-indistinguishable paths. In a) we test separation identification for a gap = 10 while b) angles of 20 ° are tested. Each point represents one trial and the solid line follows the mean of the data. For all tests, *n*=20, *s*=2 and *s*=10. At delay = 0, the system is equivalent to its predecessor. The trajectories take approximately a time of 100 to pass through the network.

A longer delay between the trajectories gives the algorithm more distinction between the two lines and correspondingly aids performance. Providing that the lines can be distinguished, the estimation ability of the algorithm should apply individually to each trajectory with the level of performance shown in the single path trials.

V. CONCLUSIONS

The EM algorithm with a simple residual function is sufficient for fitting dissimilar paths through a sensor network. Augmenting the model to include time stamping of location data serves to add a degree of dissimilarity to data generated by different trajectories and generally improves performance.

Results are surprising in the sense that there appears to be an optimal node density and sensing radius for detection of multiple lines. It is clear that an infinite node density with vanishingly small sensing radii would provide an exact solution, but when these quantities are bounded, trade-offs do exist between potentially missing a path and overestimating the number of paths.

The least squares model used in this EM algorithm is robust to false alarms in the same manner as it would be for fitting a single line in the presence of outliers. Statistical methods for removing outliers in regression analysis are equally applicable here. No work has been done to improve the either of the residual functions – a preliminary change from vertical distance to perpendicular distance might improve the versatility of the results. Much work has been done to apply and improve the EM algorithm [27]; and the application above is no more than rudimentary.

Questions relating to the requirements of the sensor networks are not fully answered by these simulations. Based on the results, it would appear, for a 100x100 network area, that 200 randomly placed nodes with sensing radii of 2 is sufficient to track single and multiple objects through the network, keeping in mind that the sensing radius and the network dimension scale together. The tracking problem is not always solved with the data provided by such a network, but this may reflect fundamental limits in the problem approach. Nodes with the ability to locate the object within their sensing areas would provide for a better approximation but would require a more complex formulation. The EM algorithm is a subset of the more powerful Bayesian network techniques which would allow for not only positional knowledge, but also the implementation of probability distributions [28] in dense network scenarios.

VI. REFERENCES

[24] A. P. Dempster, N. M. Laird, D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm", Journal of the Royal Statistical Society B, vol. 39, pp. 1-38, 1977.

[25] Y. Weiss, "Motion Segmentation using EM – a short tutorial", http://www-bcs.mit.edu/people/yweiss/tutorials.thml

[26] Y. Weiss, E. H. Adelson, "A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models", IEEE Conference on Computer Vision and Pattern Recognition, pp. 321-326, 1996.

[27] G. J. McLachlan, T. Krishnan, The EM Algorithm and Extensions, John Wiley & Sons, 1997.

[28] B. J. Frey, Graphical Models for Machine Learning and Digital Communication, MIT Press, 1998.

Chapter 6 - Visualization of Data Flow

I. INTRODUCTION

The use of computer vision algorithms to process large amounts of sensor data is the focus of this chapter. In particular, time-varying sensor readings over a two-dimensional area are visualized and summarized using vision techniques. The pervasive example will be one of temperature measurement over a geographical area – based on sensor readings at all locations over a period of time, how can the temperature flow be estimated?

The analogy between sensors and pixels is suggested. While pixels have brightness values, sensors have temperature readings. By interpolating sensor readings to regular grid locations, the network is interpreted as an image. Once the network image is constructed, optical flow algorithms can be applied.

The Lucas-Kanade first-order gradient-based flow recovery algorithm [29] is chosen to estimate temperature flow from the network image. As explained in [30], this algorithm is among the simplest of its kind both conceptually and computationally and provides more accurate results than its peers. It relies on temperature conservation over small patches in the network to constrain the flow solution – this is a reasonable assumption in our physical scenario. Other popular recovery algorithms include Horn-Schunck [31] which maximizes global smoothness and Fleet-Jepson [32] which analyzes differential image phase information. The former tends to provide aesthetically pleasing, yet quantitatively inaccurate flow reconstruction which is not sufficient for this study. The latter requires far more computational complexity and does not offer a significant performance improvement over the selected method [30].

In this chapter, all simulation is carried out assuming that node positions and sensor readings are available and accurate. The motivation is to uncover the limitations imposed by the flow recovery alone and independent of the node location or sensor correlation problems.

II. MATHEMATICAL BACKGROUND

A. From sensor positions to grid locations

In order to process the sensor data with optical flow methods, a regularized set of points is required. This is accomplished using the MATLAB function *griddata* which constructs a surface to pass through all data points (taking position as (x,y) coordinates and sensor readings as the z-coordinate) via Delaunay triangulation. By reading off the z-component of the surface at each grid location, a set of "pixels" with brightness corresponding to the magnitude of the sensor readings is interpolated. Depending on the method employed, the interpolation may allow for discontinuities in the surface itself or its derivatives. The precise method used will be identified when appropriate later in the text.

B. Flow estimation from changes in local image intensity

The foundation of flow recovery as implemented is the conservation of intensity; brightness derivatives are measured and flow is computed. Formally, the total derivative of the image intensity function vanishes at each image location for all time. For an optical flow v, spatial image derivatives f_x and f_y, and temporal image derivative f_t,

$$\begin{bmatrix} f_x & f_y \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} + f_t = 0 \qquad (1)$$

This means that changes in image intensity are produced exclusively through translation of local intensity properties and not by changes in overall image intensity. In actuality, this function will not vanish at all locations and the problem becomes one of minimization in the least-squares sense. The solution of the

least-squares problem can be written as setting to zero the gradient of the squared error in (1). This results in the following linear equation in v:

$$\begin{pmatrix} f_x^2 & f_x f_y \\ f_x f_y & f_y^2 \end{pmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = -\begin{bmatrix} f_x f_t \\ f_y f_t \end{bmatrix} \quad (2)$$

In general, the matrix on the left of (2) can be singular and different flow reconstruction methods have distinct ways of resolving this singularity [30]. The method of Lucas and Kanade [29] actually partitions the space into "patches" consisting of square arrays of several pixels, e.g. 5x5 pixels. The brightness constraint (2) is then applied individually to each patch; the minimization now conserves brightness locally in the optimal least-squares sense.

For i indexed over all pixels in the patch, the following linear equation is solved (with the index suppressed for the derivatives):

$$\begin{pmatrix} \sum_i f_x^2 & \sum_i f_x f_y \\ \sum_i f_x f_y & \sum_i f_y^2 \end{pmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = -\begin{bmatrix} \sum_i f_x f_t \\ \sum_i f_y f_t \end{bmatrix} \quad (3)$$

In this case, there is still the possibility of the matrix being singular. Simoncelli et al. [33] suggest checking the difference in singular values of the matrix to obtain a measure of confidence for the result. If both singular values fall below a certain threshold, the computed flow is discarded for this patch. Additionally, they propose using a weighting function in the matrix to emphasize the more central elements of the pixel array.

C. Computing derivatives

Numerical differentiation techniques tend to amplify noise, though considering the stochastic behavior of the signal and noise in the system can reduce this effect [34]. A method specifically designed for differentiating images to compute optical flow in two dimensions is developed in [35]. In this procedure, two filters are used to spatially differentiate the image. The first is a prefilter based on a discretization of the *sinc* function used to interpolate the image. The second is the derivative filter which closely approximates the derivative of the prefilter. The derivative is computed point-by-point with a two-dimensional convolution of these filters throughout the image. For example, to compute f_x, the image is vertically covoluted with the prefilter and horizontally with the derivative filter. The temporal derivative is one-dimensional and requires only a convolution with the derivative filter to compute f_y. Table 2 below gives the numerical values used for differentiation based on a five-point filter [36].

	-2	-1	0	1	2
Prefilter	0.036420	0.248972	0.429217	0.248972	0.036420
Derivative Filter	-0.108415	-0.280353	0	0.280353	0.108415

Table 2. Numerical filter specification for an estimate based on five points. The derivative is being calculated for the point at location "0".

D. From sensor readings to optical flow

The entire process is characterized by the following steps:

1. Collect sensor data from nodes
2. Interpolate data with Delaunay triangulation
3. Compute spatial and temporal derivatives with matched filters
4. Apply Lucas-Kanade flow estimation algorithm.

The complete process is illustrated in Figure 33.

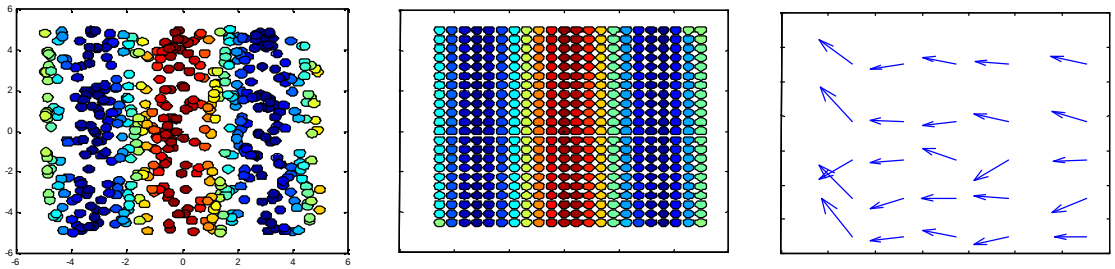


Figure 33. Sensor data leads to flow computation: In a) intensity represents sensor readings at random node positions, b) data is interpolated to regular grid locations, c) after several time steps, flow is calculated using the Lucas-Kanade algorithm on the grid values in b). Simulation has 500 nodes interpolated to a 25x25 grid. Each 5x5 pixel patch from the grid is subjected to the brightness constraint resulting in a total of 25 flow estimates illustrated by the vectors. The actual flow proceeds horizontally to the left. Grid data is not perfect, especially in regions of low node density such as the corners.

III. SIMULATION AND RESULTS

A. Interpolation

The method of interpolation to grid locations can be decoupled from the optical flow estimation itself. To first discover how the number of nodes and the number of grid points as well as the method chosen affect the accuracy of interpolation, some preliminary tests are run. The following temperature field is applied to the sensor network:

$$T(x,y) = \cos(\frac{x+y}{2}) \qquad (4)$$

In general, the temperature field will be a function of time also, *T(x,y,t)*. Nodes are randomly placed computing *T* at each node location and interpolating to a number of grid locations with a method detailed in Section II. The results are summarized in Figure 34.

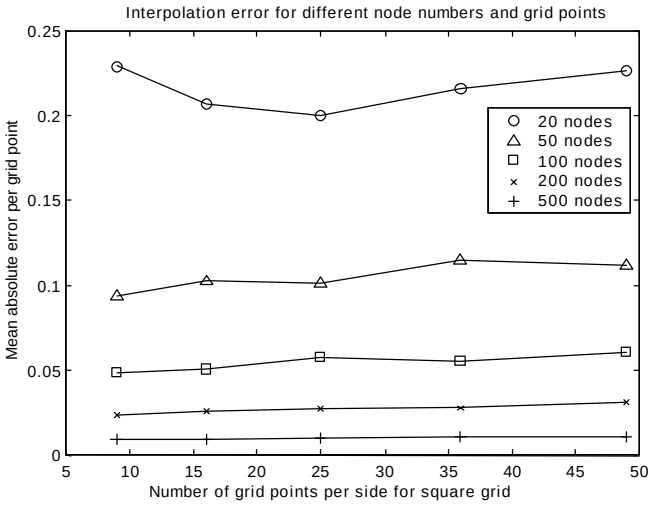


Figure 34. Mean error computed at each interpolated grid point. Linear Delaunay interpolation is used. The error is given in terms of *T* where absolute values of *T(x,y)* are bounded above by 1. Each data point represents the mean of 50 trials.

As evident in Figure 34, the error does not vary significantly with the number of grid points used in the interpolation. The only effect that this number should have on the results is the size of the final flow

data set; if n^2 grid points per side are used, it is natural to choose Lucas-Kanade patches of size n. This provides a total of n^2 total flow measurements (vectors in Figure 33) for the network. For the current study, 25 flow measurements are sufficient and 5x5 pixel patches will be used henceforth, allowing the filters detailed in Table 2 to be used. This patch size is also the one illustrated in Figure 33. As for the number of nodes in the network, it is intuitive that an increase in node number should provide a more accurate interpolation. Increasing up to 500 nodes shows no sign of asymptotic behavior. Again, for the computational complexity associated with 500 nodes, 200 nodes will be sufficient for the remainder of trials.

The type of interpolation used affects the accuracy of the resulting estimated surface. MATLAB offers three Delaunay triangulation methods (‘ nearest’, ‘linear’ and ‘ cubic’) as well as a repudiated algorithm left over from a previous version (‘v4’).

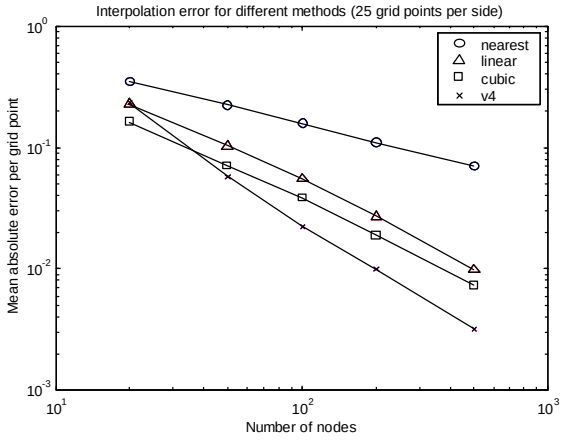


Figure 35. Mean error at each of 25x25 grid points for different interpolation methods. Results are plotted on a logarithmic scale to emphasize differences. Each data point represents the mean of 50 trials.

As indicated by Figure 35, the interpolation method has significant impact how faithfully the T function can be reproduced. Simulation reveals that although the ‘v4’ method offers prime performance, its scaling properties are horrendous. The other methods scale more tolerably to larger networks with computational load increasing for more accurate approximations. With this in mind, the ‘cubic’ Delaunay triangulation method is settled upon. A detailed description of the algorithm can be found in [37].

Another method of interpolation is considered that performs averaging in a distributed sense. Each grid location has an equivalent sensor reading that is equal to the weighted mean of the sensors in its neighborhood. This mean value is reported back to the controller as the pixel brightness for optical flow analysis.

B. Horizontal flow

A simple time-varying temperature function is applied to the network:

$$T(x, y, t) = \cos(x + \alpha t) \qquad (5)$$

where α represents the flow speed - how rapidly the temperature field varies and is the variable of interest in the following study. This field yields a periodic temperature wave moving towards the left and provided the example data for Figure 33. To determine the resolution to which α can be accurately computed, it is varied over a range of values and the flow problem is solved as directed above with $Dt = 1$ between frames. The mean horizontal value of the 25 flow arrows is computed for each trial and this is the computed value of α. In each simulation, five time steps (corresponding to the length of the derivative filter) are used to compute optical flow. Results of these trials are shown in Figure 36.

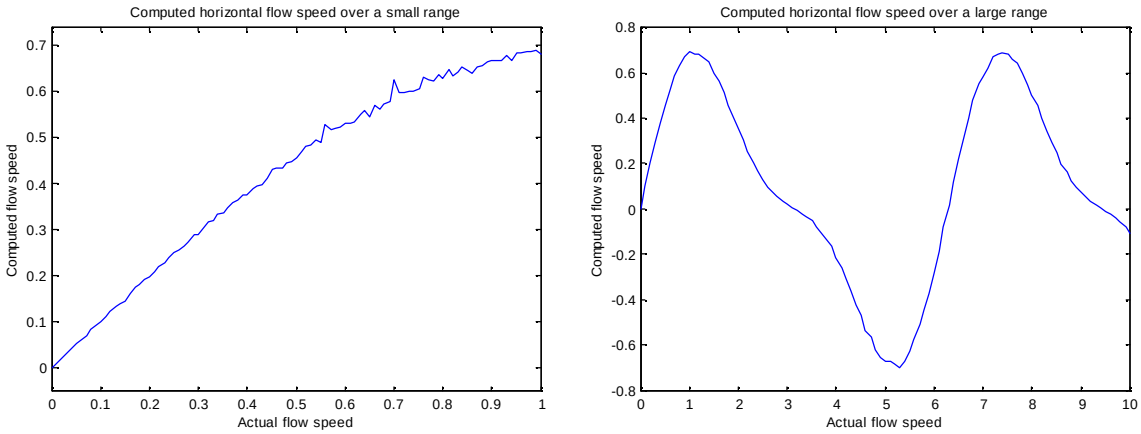


Figure 36. Computed a for a strictly horizontal flow. In a) there is a linear match between the actual and computed flow speeds over a small range before flow speed becomes underestimated. In b) the linearity is clearly broken for sufficiently large a. In particular, for a = p, the flow speed is zero. In all trials, n=200.

For small a, the algorithm effectively estimates horizontal flow speed. However, the range of a is limited due to temporal aliasing. At a value of π, the flow is transparent since $\cos(x + \pi) = \cos(x)$, it is no surprise that the graph has an x-intercept at this point. This phenomenon exposes one limitation of this flow reconstruction technique – flow speeds must be kept well below the frequency of the signals to be accurately measured. One means of bypassing this obstacle is through multi-scale flow recovery, essentially a sampling of the signal at different frequency values [36].

As a comparison to Figure 36a, a network consisting of 625 nodes placed exactly at the grid locations provides the best measure of what can be achieved. The performance of this idealized network is nearly identical to the 200 randomly placed network's performance.

C. Radial flow

In this flow model, the temperature gets equally warmer everywhere with time – but absolute temperatures are lower near the center (5,5) of the simulation area:

$$T(x, y, t) = (x-5)^2 + (y-5)^2 + at \quad (6)$$

For negative a, this has the effect of flowing temperature away from the center of the area. This models a divergent flow emanating from a source located within the network. While this apparently violates the constant temperature constraint upon which our algorithm is based, the recovery of flow vectors proceeds nonetheless.

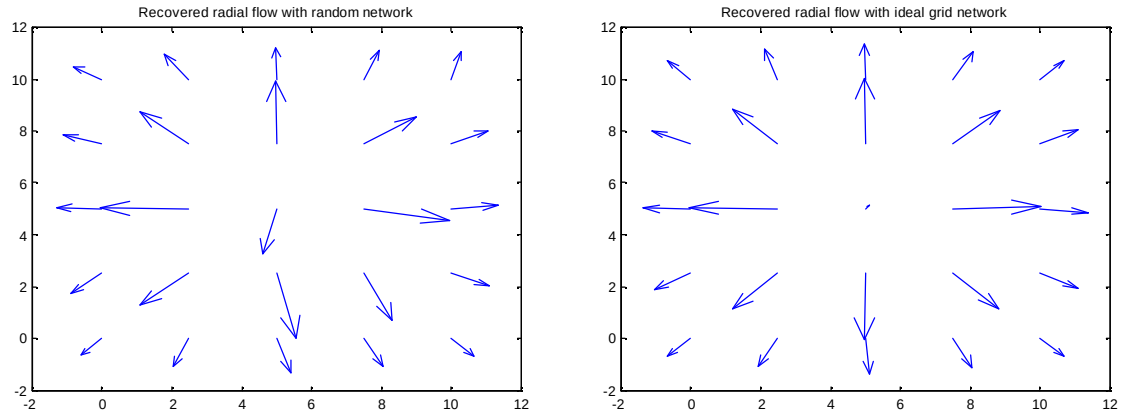


Figure 37. Recovered optical flow at 25 locations for a=-0.1. In a) the random network of 200 nodes has the correct qualitative structure of the flow while in b) the orientation of all arrows is near perfect for nodes at the 625 grid locations.

From Figure 37, it is clear that the random network does not recover flow as well as one with nodes placed at individual grid points. However, this turns out to be only a matter of n.

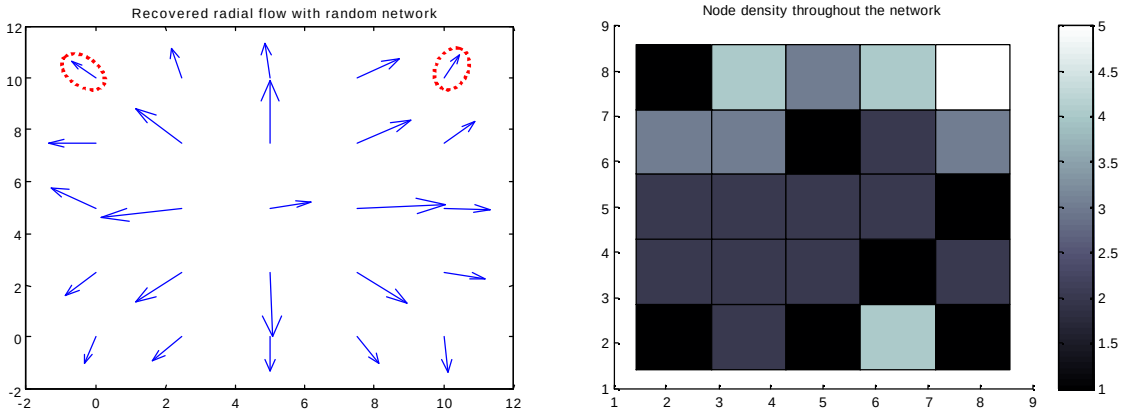


Figure 38. An n=100 random network radial flow recovery in a). In b) the brightness of the region indicates the number of nodes geographically situated therein. The colorbar on the right provides the scale. The circled vectors are discussed in the text.

With better results given by increased overall node densities, it is surprising that the regional node density does not seem to correlate to individual flow arrow accuracy as given in Figure 38. In particular, the circled vector in the top right has high density but large directional error; the node sparsity in the top left gives low directional error.

D. Rotating flow

A rotating flow can be introduced about (5,5) with the function:

$$T(x, y, t) = \cos(\arctan\left[\frac{x-5}{y-5}\right] + at) \qquad (7)$$

For negative a, the rotation is anti-clockwise. In the implementation, there is a discontinuity along the negative x-axis where the *arctan* function jumps between $\pm\pi$, numerically disrupting the flow.

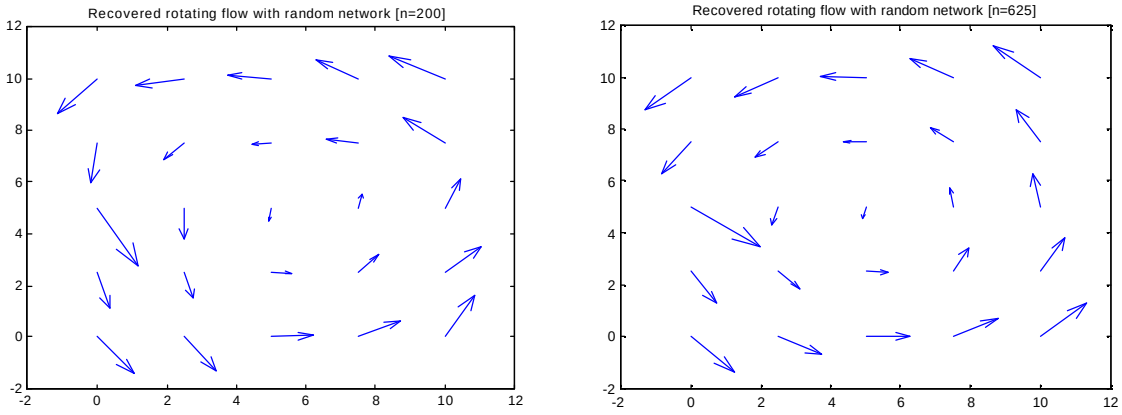


Figure 39. Rotating flow recovery for a) 200 nodes and b) 625 nodes for a =-0.1.

Again, there is qualitative improvement with the increased number of nodes as depicted in Figure 39. It is interesting how the algorithm handles the discontinuity in the flow with minor errors instead of either a disappearingly small arrow or a dwarfingly large one.

E. Arbitrary flow

In two dimensions, flow is characterized by translation, divergence and rotation. The methodology presented can successfully recover all of these flow patterns independently. Combining them linearly yields arbitrary continuous two-dimensional flows; the algorithms presented should recover any continuous flow with components in similar frequency ranges. As a practical example, consider vapors blowing away from a source and detected by chemical concentration sensors at each node. Defining $?$ = arctan[(y-5)/x] to center the source at (0,5),

$$T(x,y,t) = \begin{cases} \exp\left(-q^2\right)\cdot\exp\left(\dfrac{-x}{t}\right) & |q| > p/8 \\ 0 & otherwise \end{cases} \qquad (8)$$

In this model, the wind is blowing towards the right.

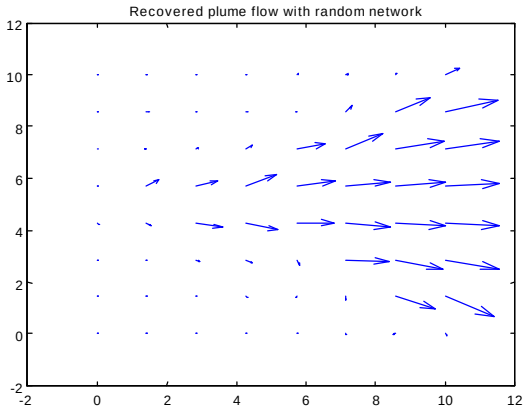


Figure 40. A simulated chemical source is centered at (0,5) with wind blowing the vapors towards the right of the network. In this example, n = 500 and 81 flow vectors are recovered.

F. Computational complexity

The flow recovery algorithm scales gracefully with both the amount of data (number of nodes sensing temperature) and the requested amount of input (number of flow arrows computed). As illustrated in Figure 41, both relations are linear.

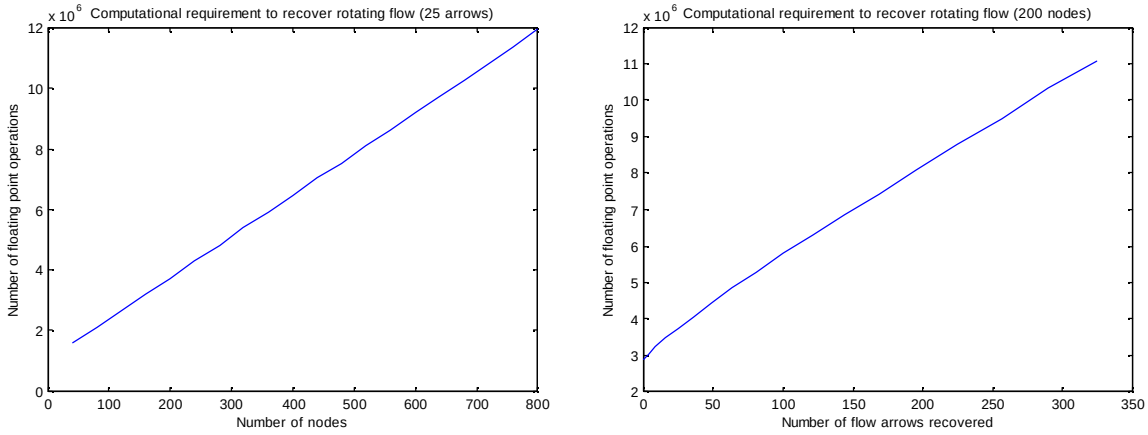


Figure 41. Computational requirements of recovering the rotational flow. In a) the number of nodes is varied for recovering 25 flow arrows. In b) the number of arrows recovered (always a perfect square) is varied while *n*=200 is constant.

Of course as we increase the number of flow arrows computed while keeping the data constant, eventually there are insufficient data for an accurate solution of the problem as exemplified in Figure 42a.

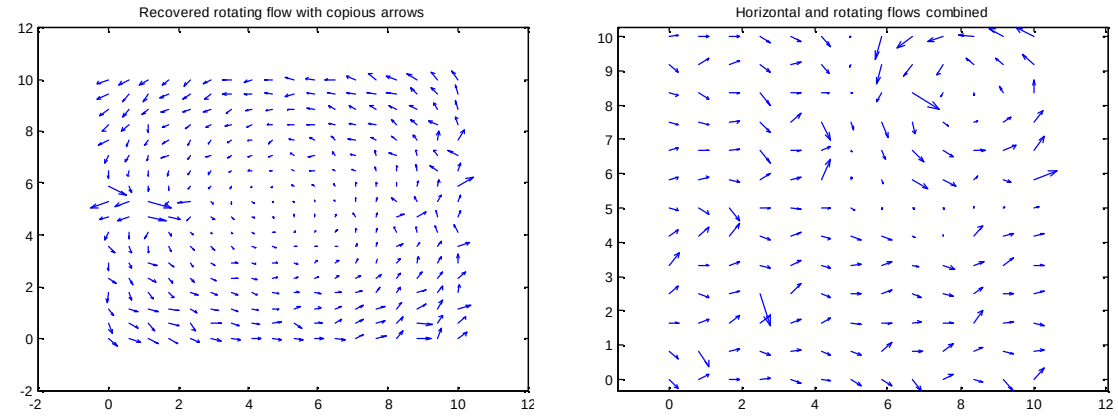


Figure 42. In a) 324 arrows are recovered with *n*=200. The singularity is now very evident and general performance suffers due to a lack of data. In b) a horizontal flow is combined with a rotating flow in the upper-right quadrant.

In Figure 42b, two flow fields are combined by defining a rotating function in the upper-right quadrant and a horizontal flow elsewhere. Qualitative properties are maintained, though the algorithm is less perspicacious around the borders of the two fields.

If both input (node number) and output (flow vector number) are scaled to meet the problem, a more enriching portrayal of the environment can be obtained. To cover a larger area of n^2, the computational requirement will rise as n^4.

IV. CONCLUSIONS

Time-varying temperature flows have been reconstructed using optical flow recovery algorithms. For the recovery of 25 flow arrows, 200 randomly placed nodes according to a uniform distribution are sufficient to qualitatively capture temperature flows. The addition of more nodes to the sensor network improves performance – no asymptotic performance limits are detected up to the tested 500 nodes. Quantitative assessment of the algorithm was attempted only for the simplistic one-dimensional flow model but could be readily extended with an appropriate metric to the other flows discussed. This would provide a definitive means of comparing the first-order gradient algorithm used to other methods of flow recovery.

The recovery of canonical flows in 2-D is successful, as is the combination of flows providing that all components have similar flow speeds. Any such continuous flow in 2-D can be recovered, while discontinuities lead to unpredictable, yet non-disastrous, behavior. Recovery of concurrent and distinct flow patterns could be implemented using coarse-to-fine estimation methods as discussed by Simoncelli [38] if they have significantly different time constants. Performance is mercurial in both simplistic and complex cases.

All of the computation required for flow recovery could conceivably be done within the sensor network itself. The Lucas-Kanade recovery for each flow vector proceeds independently of all other vectors, all required information is contained within the *m*x*m* patch of interpolated pixels. The proposed distributed pixel interpolation method, coupled with the solution of the patch's linear equation solves for flow within each region. The only unresolved issue is differentiation, though this convolution of the pixels could also be done in a distributed sense but would require communication outside the patch. Alternatively, some simulations could be run which rely exclusively on intra-patch data for computation of the derivatives.

V. REFERENCES

[29] B. D. Lucas, T. Kanade, "An iterative image registration technique with an application to stereo vision", International Joint Conference on Artificial Intelligence, pp. 674-679, Vancouver, Canada, 1981.

[30] J. L. Barron, D. J. Fleet, S. S. Beauchemin, T. A. Burkitt, "Performance of optical flow techniques", IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Champaign, IL, pp. 236-242, Jue 1992.

[31] B. Horn, B. Schunck, "Determining optical flow", Artificial Intelligence, vol.17, pp. 185-203, Aug. 1981.

[32] A. Jepson, D. J. Fleet, "Phase singularities in scale-space", Image and Vision Computing, vol.9, no.5, pp. 338-43, Oct. 1991.

[33] E. P. Simoncelli, E. H. Adelson, D. J. Heeger, "Probability distributions of optical flow", IEEE Conference on Computer Vision and Pattern Recognition, pp. 310-315, Maui, HI, June 1991.

[34] B. Carlsson, A. Ahlen, M. Sternad, "Optimal differentiation based on stochastic signal models", IEEE Transactions on Signal Processing, Vol. 39, No. 2, pp. 341-353, Feb. 1991.

[35] E. P. Simoncelli, "Design of multi-dimensional derivative filters", International Conference on Image Processing, vol. 1, pp. 790-793, Austin, TX, Nov. 1994.

[36] E. P. Simoncelli, "Bayesian multi-scale differential optical flow", Handbook of Computer Vision and Applications, pp.397-422, 1999.

[37] T. Y. Yang, Finite Element Structural Analysis, Prentice Hall, pp. 446-449, 1986.

[38] E. P. Simoncelli, "Coarse-to-fine estimation of visual motion", IEEE Workshop on Image and Multidimensional Signal Processing, Cannes, France, 1993.

Chapter 7 - Discussion

This report presented methods for defining sensor networks, developing simulations, estimating node positions, reducing overall communication bandwidth requirements, tracking objects, and recovering data flow. Taken separately, each chapter represents a solution to one facet of a developing sensor network theory. The study of position estimation revealed that given a set of known locations around the perimeter of a sensing area and a sufficiently precise model of network connections, all node positions could be estimated to within a small uncertainty. Message-reduction coding was shown to work efficaciously in the case of semi-blind correlation and marginally in the more restrictive blind correlation model. Expectation-Maximization based line search methods, augmented with a temporal dimension, accurately reconstruct trajectories through the network. The distillation of sensor data into flow arrows was used to visualize qualitative properties of data flow through the network. Each chapter presented some success and some limitations. Each could be improved – the methods applied were taken directly from other fields without attempts at specialization or refinement. A host of similar techniques might be applied anew with similar results.

In each chapter, the network abstraction takes on a certainly different definition. It is hoped, however, that the foundations proposed in the preliminary section provide some structure upon which the further theory and application can grow. Together, the subjects illustrate how a network graduates from initialization to application stages. An extension of this work involves carrying the uncertainty through the chapters, beginning with a network having a communication protocol:

1) Estimate node positions and associated uncertainty
2) Develop correlation-coding using the uncertain positions
3) Apply the tracking algorithm with the positional uncertainty
4) Apply the flow algorithm with both positional and sensing uncertainty

It is not obvious how destructively uncertainty will affect the idealized results. In this case, it would be worthwhile to express node locations as statistical distributions for use in the future algorithms.

There is no current barrier to the implementation of any of the discussed algorithms into hardware. The potential exists for thousands of sensor nodes to be fabricated today. There will be obstacles associated with the transition to reality, but the simulation suggests that the methodology should be sufficiently robust to survive. Work is underway to develop and implement an operating system on sensor network nodes by Culler *et al.* [39]. At this point, the only impediment will be maintaining interest in pursuing such a non-theoretical goal.

The postulate that centralized computation has a place in sensor network is supported by the successes in this report. It is concurrently apparent, however, that some of the algorithms discussed will soon run into computational limits as network size scales higher. At this point, distributing the load within the network itself could spread out the computations, but not beyond a factor of n at each node. The communication savings would be the prime benefit of a distributed implementation. It is not clear how the proposed algorithms could transfer to the local world, but some speculation is offered:

1) The hierarchical node positioning algorithm proposed in Chapter 3 could be run locally according to an unknown local subnetwork landmark. In this scheme, each node in a subnetwork would know its position relative to the landmark, but not realtive to any global perspective. This local information might be all that is required for other network functions.
2) The correlation-coding scheme as presented is still useful for passing reduced-length messages among sections of the network. There will always be some need for message passing over a radius of several hops in the network.
3) The tracking algorithm could be modified with local beamforming to fuse object identification data. In this methodology, nodes make decisions about path parameters within

the network by assimilating data from neighboring nodes. The output of the network would be the computed path itself, not points on a path.

4) The flow algorithm was calculated locally in some experiments. In this model, local derivatives are computed by looking at sensor readings from neighboring nodes. These derivatives can be used to compute flow vectors locally within each section of the network with the same windowing technique discussed earlier. While not quantitatively sound as the presented global counterpart, flow vectors can still be recovered to an acceptable qualitative accuracy.

As stated in the introduction, this report is concerned exclusively with information that is immediately extracted from the network and of some empirical use. The transition to a more academic study – one concerned more with self-organization and self-awareness in a sensor network than with utility – may in turn reveal more robust and less intensive methods for achieving the same goals.

COMPLETE REFERENCE LIST

[1] J. M. Kahn, R. H. Katz and K. S. J. Pister, "Mobile Networking for Smart Dust", ACM/IEEE Intl. Conf. on Mobile Computing and Networking, Seattle, WA, Aug. 1999.

[2] V. Hsu, J. M. Kahn, and K. S. J. Pister, "Wireless Communications for Smart Dust", Electronics Research Laboratory Technical Memorandum Number M98/2, Feb. 1998

[3] M. Last, K. S. J. Pister, "2-DOF Actuated Micromirror Designed for Large DC Deflection", MOEMS ' 99, Mainz, Germany, Aug 1999.

[4] B. Atwood, B. Warneke, K. S. J. Pister, "Preliminary circuits for Smart Dust", IEEE Southwest Symposium on Mixed-Signal Design, pp. 87-92, San Diego, CA, Feb. 2000.

[5] S. Hollar, Masters Report, University of California, Berkeley, May 2000.

[6] G. Asada, M. Dong, T. S. Lin, F. Newberg, G. Pottie, W. J. Kaiser, H. O. Marcy, "Wireless integrated network sensors: low power systems on a chip", ESSCIRC '98. Proceedings of the 24th European Solid-State Circuits Conference, The Hague, Netherlands, Sep. 1998.

[7] P. Gupta and P. R. Kumar, "The Capacity of Wireless Networks", to appear in IEEE Transactions on Information Theory.

[8] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols", ACM/IEEE Int. Conf. on Mobile Computing and Networking, Dallas, TX, Oct. 1998.

[9] J. Kulik, W. R. Heinzelman, H. Balakrishnan, "Negotiation-based protocols for disseminating information in wireless sensor networks", ACM/IEEE Int. Conf. on Mobile Computing and Networking, Seattle, WA, Aug. 1999.

[10] J. D. McLurkin, Masters Report, University of California, Berkeley, Dec. 1999.

[11] N. Lynch, Distributed Algorithms, Morgan Kaufmann, 1996.

[12] M. Klamkin, J. Combin. Theory, vol. 3, pp.279-282, 1967.

[13] P. Flajolet, D. Gardy, L. Thimonier, "Birthday paradox, coupon collectors, caching algorithms and self-organizing search", Discrete App. Math., vol. 39(3), pp.207-229, Nov. 1992.

[14] N. Bulusu, J. Heidemann, D. Estrin, "GPS-less low cost outdoor localization for very small devices", Technical report 00-729, Computer science department, University of Southern California, Apr. 2000.

[15] N. Karmarkar, "A new polynomial-time algorithm for linear programming", Combinatorica, vol. 4, pp. 373-395, 1984.

[16] Y. Nesterov, A. Nemirovskii, Interior-Point Polynomial Algorithms in Convex Programming, SIAM, 1994.

[17] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, Linear Matrix Inequalities in System and Control Theory , SIAM, 1994.

[18] L. El Ghaoui and J.-L. Commeau, LMItool, http://www.ensta.fr/uer/uma/gropco/lmi/lmitool.html

[19] C. W. Reed, R. Hudson, K. Yao, "Direct joint source localization and propagation speed estimation", 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Phoenix, AZ, Mar. 1999.

[20] S. S. Pradhan, K. Ramchandran, "Distributed Source Coding Using Syndromes: Design and Construction", IEEE Computer Society Conference on Data Compression (DCC'99), Snowbird, UT, pp. 158-167, Mar. 1999.

[21] W. R. Heinzelman, A. Chandrakasan, H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks", Hawaii International Conference on System Sciences, Maui, HI, Jan. 2000.

[22] Grupo de Tratamiento de Imágenes, http://www.gti.ssr.upm.es/~vadis/faq_MPEG/huffman_tutorial.html.

[23] S. S. Pradhan, K. Ramchandran, "Distributed source coding: Symmetric rates and applications to sensor networks", IEEE Computer Society Conference on Data Compression (DCC'00), Mar. 2000.
[24] A. P. Dempster, N. M. Laird, D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm", Journal of the Royal Statistical Society B, vol. 39, pp. 1-38, 1977.
[25] Y. Weiss, "Motion Segmentation using EM – a short tutorial", http://www-bcs.mit.edu/people/yweiss/tutorials.thml
[26] Y. Weiss, E. H. Adelson, "A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models", IEEE Conference on Computer Vision and Pattern Recognition, pp. 321-326, 1996.
[27] G. J. McLachlan, T. Krishnan, The EM Algorithm and Extensions, John Wiley & Sons, 1997.
[28] B. J. Frey, Graphical Models for Machine Learning and Digital Communication, MIT Press, 1998.
[29] B. D. Lucas, T. Kanade, "An iterative image registration technique with an application to stereo vision", International Joint Conference on Artificial Intelligence, pp. 674-679, Vancouver, Canada, 1981.
[30] J. L. Barron, D. J. Fleet, S. S. Beauchemin, T. A. Burkitt, "Performance of optical flow techniques", IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Champaign, IL, pp. 236-242, Jue 1992.
[31] B. Horn, B. Schunck, "Determining optical flow", Artificial Intelligence, vol.17, pp. 185-203, Aug. 1981.
[32] A. Jepson, D. J. Fleet, "Phase singularities in scale-space", Image and Vision Computing, vol.9, no.5, pp. 338-43, Oct. 1991.
[33] E. P. Simoncelli, E. H. Adelson, D. J. Heeger, "Probability distributions of optical flow", IEEE Conference on Computer Vision and Pattern Recognition, pp. 310-315, Maui, HI, June 1991.
[34] B. Carlsson, A. Ahlen, M. Sternad, "Optimal differentiation based on stochastic signal models", IEEE Transactions on Signal Processing, Vol. 39, No. 2, pp. 341-353, Feb. 1991.
[35] E. P. Simoncelli, "Design of multi-dimensional derivative filters", International Conference on Image Processing, vol. 1, pp. 790-793, Austin, TX, Nov. 1994.
[36] E. P. Simoncelli, "Bayesian multi-scale differential optical flow", Handbook of Computer Vision and Applications, pp.397-422, 1999.
[37] T. Y. Yang, Finite Element Structural Analysis, Prentice Hall, pp. 446-449, 1986.
[38] E. P. Simoncelli, "Coarse-to-fine estimation of visual motion", IEEE Workshop on Image and Multidimensional Signal Processing, Cannes, France, 1993.
[39] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister, "System architecture directions for networked sensors", submitted for publication.