ADAPTIVE DISTRIBUTED TRANSFORMS FOR IRREGULARLY SAMPLED WIRELESS SENSOR NETWORKS

Godwin Shen, Sunil Kumar Narang and Antonio Ortega

Signal and Image Processing Institute
University of Southern California
godwinsh@usc.edu, kumarsun@usc.edu, ortega@sipi.usc.edu

ABSTRACT

We develop energy-efficient, adaptive distributed transforms for data gathering in wireless sensor networks. In particular, we consider a class of unidirectional transforms that are computed as data is forwarded to the sink along a given routing tree and develop a tree-based Karhunen-Loève Transform (KLT) that is optimal in that it achieves maximum data de-correlation among this class of transforms. As an alternative to this KLT (which incurs communication overhead in order to learn second order data statistics), we propose a backward adaptive filter optimization algorithm for distributed wavelet transforms that i) achieves near optimal performance and ii) has no communication overhead in learning statistics.

Index Terms— Adaptive Filters, Data Compression, Wavelet Transforms, Wireless Sensor Networks

1. INTRODUCTION

Wireless sensor devices (sensor nodes) are extremely power limited, especially since most sensor nodes are battery powered. Thus in order to extend the lifetime of a Wireless Sensor Network (WSN) it is important to develop efficient algorithms for data gathering, so that data can be delivered to a sink or base station with high quality while requiring minimal power consumption at the nodes.

In this context, in-network distributed transforms, e.g., [1, 2, 3, 4] and references therein, have long been considered an attractive tool since they exploit the fact that data being gathered has to be routed over multiple hops from sensor to sensor and spatial data correlation exists across sensors. These transforms exploit existing spatial correlation in data in order to reduce the number of bits to be transmitted as data is routed towards the sink.

Our previous work [3, 5] has shown that i) the choice of a routing tree is important, as it affects both the transport costs and the number of bits required to represent the data, and ii) it is more efficient to compute the transform without any "backward" communications, i.e., so that all operations are performed as data flows to the sink, a principle we call *unidirectional computation*. Specifically, the transform in [5] is constructed on a routing tree such that each node uses data from its children and parent to transform its own. Thus, a transmission schedule is defined to allow each node to collect data from its descendants before it processes and forwards its own data. This eliminates "backward" communications.

Given that typically sensors are not necessarily placed on a regular grid, it is useful to consider transforms that can adapt to specific characteristics of the networks. Broadly speaking two types of network-adaptive transforms have been proposed in the literature: those that are "data-dependent", i.e., exploit statistical correlations in the data, and those that are "structure-dependent", i.e., are based on "structural" information about the network (e.g., relative distances between nodes). A recent example of a data-dependent transform is the distributed Karhunen-Loève Transform (KLT) [2], which can be easily applied to WSN. This transform is useful for compression since a KLT achieves maximum data de-correlation [6]. However, the resulting transform is not unidirectional. Instead the network must be separated into clusters, cluster-heads designated in each cluster, then nodes forward data to their cluster-heads that in turn compute the transform and forward coefficients to the sink. This is inefficient since many nodes will need to transmit data away from the sink. This method also incurs a *learning cost* to discover and disseminate the correlation structure.

Examples of the second type of network-adaptive transforms include those that use structural information in the network to design fixed (non-data adaptive) transforms. Thus, there is no learning cost. In the wavelet transform proposed in [4] using wavelet lifting [7], relative location information is used to design prediction filters at nodes that give more weight to data from closer neighbors and less to neighbors further away. However, unidirectional computation is not guaranteed for this transform since the transform is not developed along routing paths. The unidirectional wavelet transform proposed in [5, 8] exploits spatial correlation across neighbors in a routing tree by using prediction filters that employ simple averages.

For coding purposes, in order to maximize overall performance it is important to maximize the amount of data de-correlation. Structure-dependent approaches are only efficient in this sense when the correlation in the data is also structure-dependent, e.g., correlation is proportional to distance. Ideally, we want to construct unidirectional transforms that can be adapted to the underlying data in a distributed manner with no learning cost.

The *main goal* of this work is to develop distributed unidirectional transforms that use spatial data statistics to maximize the amount of data de-correlation with little to no learning cost. This raises one natural question, i.e., how to best use statistics to de-correlate data in the network. Since we wish to minimize the cost of training we require that each node in the network adapt its own coding strategy based only on the data it can observe. Thus, assuming that data is routed to a base station via a routing tree, a given node can only observe its own measurements and those of its descendants in the tree. In the first part of this paper, we assume that each node knows the second order data statistics corresponding to all its descendants in the routing tree. Based on this we develop a *tree-based KLT* that is computed as data is routed towards the sink. This transform gives the best possible representation when we do not allow any backward communications (e.g., we do not allow the

This work was supported in part by NASA under grant AIST-05-0081.

sink to transmit back to the nodes complete second order statistics for the whole network). No such KLT has been proposed to the best of our knowledge. Note that a significant amount of learning must be done to produce a reliable estimate of the necessary statistics.

As a pratical alternative, we consider unidirectional tranforms that use spatial prediction filters to de-correlate data and adapt these filters to data statistics over time in a distributed manner with virtually no learning cost. Techniques that adapt the filters used in wavelet decompositions using lifting have been proposed for image processing applications. In [9], both the length of each prediction filter and the filter coefficients are adapted to data by solving a least squares problem. However, when applying this idea to a WSN the sink must know the coefficients and lengths of the filters that nodes use in order to invert the transform and so nodes must transmit these values along with the data, resulting in some communication overhead. The method in [10] adapts filters with no learning cost by using an adaptive filter run over consecutive predicted pixels with a fixed weight vector applied to a fixed set of neighbors. Since nodes in a WSN will generally have a different number of neighbors, a fixed weight vector cannot be applied. Instead, we use prediction filters that can be applied to an arbitrary set of neighbors and use adaptive filtering to tailor these filters to data statistics at each node (over time) using data from the node and its neighbors. Thus, we can achieve distributed filter adaptation with virtually no learning cost.

This paper is organized as follows. Section 2 describes the tree-based KLT. Section 3 provides an algorithm to estimate the optimal lifting filters with no learning cost. Section 4 evaluates the performance of the proposed algorithms. Section 5 concludes the work.

2. TREE-BASED KLT

It is well known that the KLT can maximally de-correlate any signal for which the second order correlation structure is known. The obvious advantage of such a transform is improved coefficient encoding. In order to use the KLT for distributed compression in WSN, the ideal scenario would be to first compute the correlation matrix of the entire network and then whiten the measurements of each node using the KLT. However, this requires a great deal of inter-node communication since each node must exchange data with every other node. The distributed KLT [2] is designed specifically for implementation on a WSN, but requires many backward transmissions and so can be inefficient when considering total communication cost. To avoid such backward transmissions, we only consider unidirectional transforms. Thus, nodes must transmit data towards the sink along a routing tree in the order specified by some transmission schedule. For example, we may schedule transmissions so that a node will only transmit once it receives data from all of its descendants. This means that data correlation can only be exploited on the sub-trees formed at each node along routing paths. By applying a KLT to the measurements collected up to each node, we can maximally de-correlate the data of that node's sub-tree. Thus, this tree-based KLT (T-KLT) is the best among all unidirectional transforms in terms of maximizing data de-correlation. It also serves as an upper bound on the performance of any distributed, unidirectional transform.

The following notation will be used in the mathematical formulation. For a given node n in an N node network with routing tree $T = (V, E_T)$, $x(n)$ is its measurement and Subtree(n) is the set of nodes in the sub-tree below n including node n itself. Moreover $\mathbf{x}[n]$ is the data vector containing all measurements from Subtree(n). $\mathbf{K}_n$ is the correlation matrix of the nodes in Subtree(n) and child(n) is the vector containing all 1-hop children of node n. Finally, let depth(n) denote the number of hops from node n to the sink.

2.1. Unidirectional T-KLT

The unidirectional T-KLT algorithm has two phases. In the *training* phase, we find the whitening and coloring matrices as detailed in Algorithm 1. For a given correlation matrix $\mathbf{K}_n$ for each node n we compute the matrix $\mathbf{E}_n$ of eigenvectors that diagonalizes it, i.e., $\mathbf{E}_n^T \mathbf{K}_n \mathbf{E}_n = \mathbf{\Sigma_n}$, where $\mathbf{\Sigma_n}$ is the diagonal matrix of eigenvalues of $\mathbf{K}_n$. The whitening matrix for this node in the non-singular case is $\mathbf{\Sigma_n^{-1/2}} \mathbf{E_n^T}$ and the corresponding coloring matrix is $\mathbf{E}_n \mathbf{\Sigma_n^{1/2}}$ These are denoted as $\mathbf{H}_n$ and $\mathbf{G}_n$ respectively. If $\mathbf{K}_n$ is singular with rank $m < |\text{Subtree}(n)|$, we take $\tilde{\mathbf{E}}_m$ ($\tilde{\mathbf{\Sigma}}_m$) as the m eigenvectors (diagonal matrix of eigenvalues) corresponding to the m non-zero eigenvalues in $\mathbf{\Sigma_n}$. The whitening matrix is then $\tilde{\mathbf{\Sigma}}_m^{-1/2} \tilde{\mathbf{E}}_m^T$ and the coloring matrix is found using the singular value decomposition of $\tilde{\mathbf{\Sigma}}_m^{-1/2} \tilde{\mathbf{E}}_m^T$. So for each node n, we compute $\mathbf{H}_n$ for n using $\mathbf{K}_n$ and $\mathbf{G}_{C_n(k)}$ using $\mathbf{K}_{C_n(k)}$ for every child $C_n(k)$ of n. As we shall discuss in the next subsection, this requires a lot of inter-node communication and increases the cost of operation.

The second phase is *forwarding*. Encoded coefficients $\mathcal{W}$ are forwarded from leaf nodes all the way up to the sink node as explained in Algorithm 2. For example node n receives encoded coefficients $\mathcal{W}_{C_n(k)}$ from each of its child $C_n(k)$ and reconstructs original data $\mathbf{x}[C_n(k)]$ using the corresponding coloring matrix. It then combines data from all children nodes with its own measurement $x(n)$. This combined $\mathbf{x}[n]$ is then whitened using $\mathbf{H}_n$ and sent to next hop node in routing tree.

Algorithm 1 Tree-KLT Training Algorithm

for $k = \max(\text{depth}) : -1 : 1$ **do**
  $\mathcal{I}_k = \{m \in V : \text{depth}(m) = k\}$
  for each $n \in \mathcal{I}_k$ **do**
    Compute $\mathbf{K}_n = \mathbf{E}_n * \mathbf{\Sigma_n} * \mathbf{E_n^T}$
    $\mathbf{H}_n = \mathbf{\Sigma_n^{-1/2}} * \mathbf{E_n^T}$
    Compute $C_n = \text{child}(n)$
    Compute $nC = \text{size}(C_n)$
    for each $k = 1$ to nC **do**
      Compute $\mathbf{K}_{C_n} = \mathbf{E}_{C_n} * \mathbf{\Sigma_{C_n}} * \mathbf{E_{C_n}^T}$
      $\mathbf{G}_{C_n(k)} = \mathbf{E}_{C_n} * \mathbf{\Sigma_{C_n}^{1/2}}$
    end for
  end for
end for

2.2. Learning Cost for T-KLT

Training could be done by forwarding raw measurements to the sink for a certain period of time. During this period each node trains its correlation matrix based on the measurements it receives from its sub-tree. The training phase could also be done by an online estimation of correlation matrices at each node and forwarding those matrices to the next hop node along with the transformed coefficients. In either case there is some time-lag before each node can efficiently start de-correlating data. In the latter case there is an additional overhead of sending correlation matrices that rapidly grow in size near the sink. Moreover the cost of training grows quadratically with the increase in the size of the network, i.e., in order to estimate an $N \times N$ correlation matrix we require at least N observations of an N dimensional data vector. Thus the advantages of forwarding un-correlated data to the sink are eclipsed by the cost of training. In addition, the algorithm undergoes transform and inverse transform at each node in the routing path and so it also suffers from the propagation of quantization error.

Algorithm 2 Tree-KLT Forwarding Algorithm

for $k = \max(\text{depth}) : -1 : 1$ **do**
  $\mathcal{I}_k = \{m \in V : \text{depth}(m) = k\}$
  for each $n \in \mathcal{I}_k$ **do**
    Compute $C_n = \text{child}(n)$
    Compute $nC = \text{size}(C_n)$)
    for each $m = 1$ to nC **do**
      Receive $\mathcal{W}_{C_n(m)}$ from m^{th} child of node n
      $\mathbf{x}[C_n(m)]. = \mathbf{G}_{C_n(m)} * \mathcal{W}_{C_n(m)}$
      $\mathbf{x}[n] = [x(n), \mathbf{x}[C_n(1)], ..., \mathbf{x}[C_n(nC)]$
      $\mathcal{W}_n = \mathbf{H}_n * \mathbf{x}[n]$
      Transmit $\mathcal{W}_n$ to next hop
    end for
  end for
end for

3. DISTRIBUTED FILTER OPTIMIZATION

As a practical alternative to the T-KLT, we propose a method for adaptively changing the predictive filtering operations used within standard coding schemes. This adaptation is performed in a distributed manner with virtually no learning cost. We first discuss how to find optimal spatial prediction filters, particularly, ones that minimize the energy in each residual prediction error. Suppose we are given an arbitrary encoding scheme that uses a prediction step to de-correlate data (such as DPCM or a lifting transform). For each predicted node n, we want to find a linear estimate of $x(n)$, given by $\hat{x}(n) = \sum_{i \in \mathcal{N}_n} \mathbf{p}_n(i)x(i)$ for some set of neighbors $\mathcal{N}_n$, that minimizes the residual prediction error $d(n) = x(n) - \hat{x}(n)$. The optimal solution for each predicted node n is the vector $\mathbf{p}_n^*$ that minimizes $\mathbb{E}[d^2(n)]$, e.g., the Wiener-Hopf solution [11], and is a function of the correlation $R_X(i,j) = \mathbb{E}[x(i)x(j)]$ between nodes $i, j \in \mathcal{N}_n$.

As discussed in Section 2.2, estimation of these statistics is costly in terms of delay, computation and communication. Alternatively, we can use adaptive filters to estimate the optimal spatial prediction filters over time with no learning cost since they i) converge to the optimal filters for stationary data, ii) do not require estimates of data statistics and iii) are such that the filtering done at one node can be replicated at any other node (e.g., the sink) given only the residual prediction errors and the same initial prediction filters. As such, we can apply an adaptive filter at each node to estimate the optimal prediction filters. Note that no information need be sent to the sink, but it still takes time for the filters to adapt to the data well enough to produce good predictions. Thus, there will be a small learning cost for nodes to initially "train" their filters and also to "re-train" their filters when data statistics change (i.e., the overall encoding rate will be higher during training periods, during which filters have not yet converged to a state that matches current data statistics.)

There is a variety of adaptive filters we can choose from, but for each method the step-size parameter μ often must be chosen based on some data dependent parameters to ensure filter convergence. Since we generally will not have prior knowledge of those data dependent parameters, the most suitable choice for this application is a normalized least mean squares adaptive filter since μ need not be specified but is instead adapted as the filter is adapted.

Some additional notation is now established. Suppose each node measures data at times $t_1, t_2, \ldots, t_M$. Denote by $x(n,m)$ the data at node n taken at time step t_m. The linear prediction coefficient matrix ($N \times M$) for node n is given by $\mathbf{p}_n$, where column i, i.e., $\mathbf{p}_n(:,i)$, is the prediction vector at the i-th time step at node n. Using the formulation in [11], we have the following adaptive filter at each prediction node n run over time from $m = 1$ up to $m = M$:

- $\hat{x}(n,m) = \mathbf{p}_n^T(\mathcal{N}_n, m)x(\mathcal{N}_n, m)$
- $d(n,m) = x(n,m) - \hat{x}(n,m)$
- $\mathbf{p}_n(\mathcal{N}_n, m+1) = \mathbf{p}_n(\mathcal{N}_n, m) + \mu \frac{x(\mathcal{N}_n,m)d(n,m)}{x^T(\mathcal{N}_n,m)x(\mathcal{N}_n,m)}$

This technique can be easily applied to the unidirectional lifting transform in [8]. In this transform, at each level in the wavelet decomposition nodes are split into even and odd sets, $\mathcal{P}$ and $\mathcal{U}$, respectively, along some tree T. Then, for each node $n \in \mathcal{P}$ the prediction filter $\mathbf{p}_n(:,m)$ is applied to data from its neighbors to form a prediction $\hat{x}(n,m)$ and the detail coefficient $d(n,m) = x(n,m) - \hat{x}(n,m)$ is computed and forwarded to the sink. Similarly, for each $n \in \mathcal{U}$, an update filter is applied to data at n using detail coefficients from $\mathcal{N}_n$ to generate smooth coefficient $s(n,m)$.

Minor modifications of the algorithms in [8] are needed to maintain unidirectional computation. In particular, under this unidirectional transform, each odd node uses data from its parent and children for prediction but only receives data from its children. This is problematic since an adaptive filter can only be run when all data it uses is available. To address this issue, we simply require that each $n \in \mathcal{P}$ forward raw data one hop to its parent, at which point all data used to adapt $\mathbf{p}_n$ is available. Also, each $n \in \mathcal{U}$ must forward raw data two hops to avoid repeatedly de-coding and re-encoding coefficients. This results in essentially the same local communication cost as that in [8]. The sink can reverse this processing by inverting the smooth coefficients to get $x(n,m)$ for each $n \in \mathcal{U}$, and then can run the adaptive filter for each $l \in \mathcal{O}$ using $d(l,m)$ to find the prediction filter used at each time step m, at which point it can invert the prediction step to recover $x(l,m)$ for each $l \in \mathcal{O}$.

We can also apply these adaptive filters to a simple tree-based DPCM (T-DPCM). As data is forwarded to the sink, each node n will have access to its children's data and so can predict its own data $x(n,m)$ with data from its children then encode and forward the difference. Hence, we compute $\hat{x}(n,m) = \sum_{k \in C_n} \mathbf{p}_n(k,m)x(k)$, then encode and forward the difference $d(n,m) = x(n,m) - \hat{x}(n,m)$. As in the wavelet encoding case, each node will also forward raw data one step for the reasons discussed above. The prediction vectors $\mathbf{p}_n(:,m)$ are then adapted over time using the formulation given above and the sink would reconstruct the original data in a manner similar to that done for the wavelet transforms.

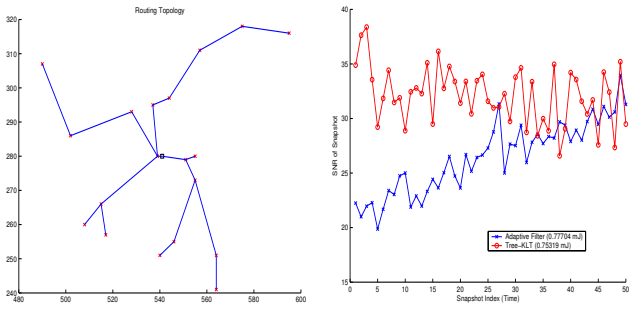
4. EXPERIMENTAL RESULTS

The tree-based wavelet transform [8] is used to compare the "structure-dependent" filter designs of [4, 8] against our distributed optimization method. We also compare these against the T-KLT and T-DPCM with adaptive filters. The adaptive filters are run over time using data collected at each node and at neighboring nodes. The sequential entropy coding scheme in [12] is used to encode the transform coefficients of each node. Energy consumption is modeled as in [13], where the cost to transmit k bits a distance D is equal to $C = c_p k D^2$ Joules with c_p a constant of proportionality. Furthermore, in those models the energy consumed in receiving k bits and performing computations is negligible compared to transmission costs and so we assume zero cost for reception and computation.

For our experiment, we use a set of empirical data taken from 19 sensors from a habitat monitoring deployment [14] on the Great Duck Island. The routing topology is shown in Figure 1(a). The performance curves in Figure 2 compare the trade-off between total

energy consumption (in Joules) and reconstruction quality, i.e., SNR, for each method. T-DPCM with adaptive filters has the worst performance because most nodes only use data from at most one neighbor to de-correlate their own data. In the case of the wavelets, each node uses data from its parent and children. The average prediction filters of [8] obviously have the worst performance among wavelet-based approaches. The planar prediction filters of [4] outperform the average filters and the distributed filter optimization scheme we propose here is second only to the T-KLT. However, in practice there will be significant learning costs that we do not account for here. Also note that the T-KLT suffers from quantization error propagation, i.e., more distortion at lower costs (coarser quantization), unlike the other methods. Due to a lack of space, we have omitted results for the artificial data used in [5, 8]. The relative performance is very similar for that data, except that the T-KLT does much better than our optimization scheme since that data is actually spatially stationary.

We also examine the transient behavior of our filter optimization scheme. Figure 1(b) shows SNR values of the reconstructed data at each measurement time with comparable energy consumption for the T-KLT and our method. Our proposed method converges to SNR values similar to the T-KLT in about 40 iterations and so converges to near optimal performance with very little learning cost.



(a) Shortest Path Routing Tree (b) Transient Behavior of Method

Fig. 1. Duck Island Network Experiment. Here x's denote nodes, lines between x's denote forwarding links and the square is the sink node.

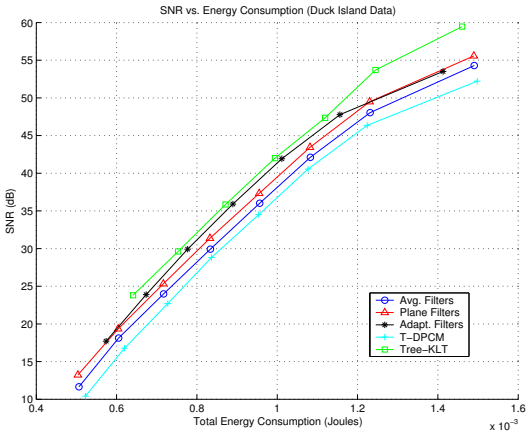


Fig. 2. Performance comparisons.

5. CONCLUSIONS

Optimal unidirectional transforms have been developed for WSN. A tree-based KLT is presented that achieves the maximum decorrelation among all possible unidirectional transforms on a routing tree but incurs significant learning overhead to estimate the necessary statistics. As an alternative, we also proposed a filter optimization method for lifting transforms and T-DPCM that achieves performance close to the T-KLT with virtually no learning cost.

6. REFERENCES

[1] D. Ganesan, D. Estrin, and J. Heidemann, "Dimensions: Why do we need a new data handling architecture for sensor networks?," *ACM SIGCOMM Comput. Commun. Rev.*, Jan. 2003.

[2] M. Gastpar, P. Dragotti, and M. Vetterli, "The distributed Karhunen-Loève transform," *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5177–5196, December 2006.

[3] A. Ciancio, S. Pattem, A. Ortega, and B. Krishnamachari, "Energy-efficient data representation and routing for wireless sensor networks based on a distributed wavelet compression algorithm," in *IPSN '06*, New York, NY, USA, 2006, pp. 309–316, ACM Press.

[4] R. Wagner, R. Baraniuk, S. Du, D.B. Johnson, and A. Cohen, "An architecture for distributed wavelet analysis and processing in sensor networks," in *IPSN '06*, New York, NY, USA, 2006, pp. 243–250, ACM Press.

[5] G. Shen and A. Ortega, "Joint routing and 2D transform optimization for irregular sensor network grids using wavelet lifting," in *IPSN '08*, St. Louis, April 2008.

[6] H. Stark and J.W. Woods, *Probability and Random Processes with Applications to Signal Processing*, Prentice Hall, 3rd edition, 2002.

[7] W. Sweldens, "The lifting scheme: A construction of second generation wavelets," Technical report 1995:6, Industrial Mathematics Initiative, Department of Mathematics, University of South Carolina, (ftp://ftp.math.sc.edu/pub/imi_95/imi95_6.ps), 1995.

[8] G. Shen and A. Ortega, "Optimized distributed 2D transforms for irregularly sampled sensor network grids using wavelet lifting," in *Proc. of ICASSP'08*, Las Vegas, April 2008.

[9] R. L. Claypoole, R. G. Baraniuk, and R. D. Nowak, "Adaptive wavelet transforms via lifting," in *Proc. of ICASSP'98*, Seattle, May 1998.

[10] O. N. Gerek and A. E. Cetin, "Adaptive polyphase subband decomposition structures for image compression," *IEEE Transactions on Image Processing*, vol. 9, no. 10, pp. 1649–1660, October 2000.

[11] S. Haykin, *Adaptive Filter Theory*, Prentice Hall, 4th edition, 2004.

[12] A. Kiely and M. Klimesh, "Generalized Golomb codes and adaptive coding of wavelet-transformed image subbands," *JPL IPN Progress Report*, June 2003.

[13] A. Wang and A. Chandraksan, "Energy-efficient DSPs for wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 19, no. 4, pp. 68–78, July 2002.

[14] H. M. on Great Duck Island. Online data-set located at http://www.greatduckisland.net, ," .