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 inversely 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 in this transform a significant amount of learning is necessary in order 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. Adaptive 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 clear that if communication costs could be ignored the KLT could be used to provide maximum decorrelation for the measurements in a WSN. However, this is not practical given the communication cost involved in obtaining global correlation statistics. As an alternative, the distributed KLT [2] is designed specifically for implementation on a WSN, but requires many backward transmissions and so is also inefficient when considering total communication cost. To avoid such backward transmissions, we restrict ourselves to considering only unidirectional transforms, such that nodes must transmit data towards the sink along a routing tree in the order specified by some transmission schedule. Since data flows only in the direction of the sink, a given node can only use information provided by its descendants in order to achieve decorrelation.

We propose a unidirectional tree-based KLT (T-KLT), where we apply the KLT (whitening) on data collected at each node, i.e., data collected from the node and its descendants. Transform coefficients are then encoded and forwarded to each parent node. Each parent then applies an inverse KLT (coloring) on the KLT coefficients received from each of its children to recover the original data, applies a KLT to this original data and then encodes and forwards the coefficients to its parent. This preserves unidirectional computations and is repeated until all coefficients are collected at the sink. In order for this to work in practice, each node must know the second-order statistics of its sub-tree. Thus, there will be some *learning cost* associated with discovering and disseminating these statistics.

Essentially, each node applies a KLT to the data collected from its sub-tree. Thus, the T-KLT achieves maximal data de-correlation on such data thus leading to fewer bits needed to represent it. This reduces the amount of information nodes must transmit over each hop, thereby reducing the total communication cost in exchange for some added computational cost (which is far lower than the cost of communication). In fact, for a given routing tree, under the constraint of unidirectional computation, T-KLT achieves better de-correlation of the data collected from each node's sub-tree than any other unidirectional transform. Thus, when the cost of training is not considered, T-KLT serves as an upper bound on the performance of any distributed, unidirectional transform.

We now establish some notation. For each node n in an N node network with routing tree $T = (V, E_T)$, $x(n)$ is a single 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), ρ_n denotes the parent of n, C_n is the set of all 1-hop children of n and $|C_n|$ is the number of children of n. Finally, depth(n) denotes the number of hops from node n to the sink.

2.1. Unidirectional T-KLT

The unidirectional T-KLT algorithm has two phases. The *training* phase computes whitening and coloring matrices from the correlation matrix of the network. We assume that each node n has the knowledge of correlation matrix $\mathbf{K}_n$ of Subtree(n) (i.e. of all nodes in its subtree). For each $\mathbf{K}_n$ 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. For the singular case, we use the corresponding reduced rank matrix. Whitening matrix $\mathbf{H}_n$ is used to decorrelate the data transmitted by node n so it is computed and stored in node n. The parent of node n (ρ_n) receives these transmitted coefficients and uses coloring matrix $\mathbf{G}_n$ of node n to recover the original data. Thus, $\mathbf{G}_n$ is computed and stored in node ρ_n. In short, each node n computes a whitening matrix $\mathbf{H}_n$ for its subtree using $\mathbf{K}_n$ and a coloring matrix $\mathbf{G}_{C_n(k)}$ using $\mathbf{K}_{C_n(k)}$ for the subtree of each child $C_n(k)$ of n.

The second phase is *forwarding* and is detailed in Algorithm 1. Data forwarding progresses from nodes of maximum depth down to nodes one hop from the sink, i.e., nodes at depth k forward their data only after all nodes at depth $k+1$ have transmitted theirs. If node n is a leaf node then it does not have any children. Hence $|C_n| = 0$ and Subtree$(n) = n$, thus $\mathbf{x}[n] = x(n)$. The whitening matrix $\mathbf{H}_n$ is just a scalar, yielding coefficient $\mathbf{w}_n = \mathbf{H}_n x(n)$ which is encoded and sent to ρ_n. If node n is not a leaf node, $|C_n| > 0$ and so node n must first receive and decode $\mathbf{w}_{C_n(m)}$ for each child $C_n(m)$. Since node n has stored coloring matrix $\mathbf{G}_{C_n(m)}$ for each child $C_n(m)$ (as done in the *training* phase), it can recover the original data of each child as $\mathbf{x}[C_n(m)] = \mathbf{G}_{C_n(m)} \mathbf{w}_{C_n(m)}$. The original data obtained from all children nodes is then concatenated with the measurement at node n to produce data vector $\mathbf{x}[n]$, i.e., $\mathbf{x}[n] = \left(x(n), \mathbf{x}^t[C_n(1)], \ldots, \mathbf{x}^t[C_n(|C_n|)]\right)^t$, and this vector is whitened as $\mathbf{w}_n = \mathbf{H}_n \mathbf{x}[n]$. Finally, $\mathbf{w}_n$ is encoded and forwarded to next hop in the routing tree. The vector $\mathbf{w}_n$ usually has very few non-zero values as compared to data vector $\mathbf{x}[n]$. Hence the number of bits required to encode $\mathbf{w}_n$ is less than that required for $\mathbf{x}[n]$. This is the source of savings in the T-KLT.

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 using 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 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 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^2 observations of an N dimensional data vector. Thus the advantages of forwarding un-correlated data to the sink may be offset by training cost. 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 1 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**
    for each $m = 1$ to $|C_n|$ **do**
      Receive and Decode $\mathbf{w}_{C_n(m)}$ from m^{th} child of n
      $\mathbf{x}[C_n(m)] = \mathbf{G}_{C_n(m)}\mathbf{w}_{C_n(m)}$
    end for
    $\mathbf{x}[n] = \left(x(n), \mathbf{x}^t[C_n(1)], \ldots, \mathbf{x}^t[C_n(|C_n|)]\right)^t$
    $\mathbf{w}_n = \mathbf{H}_n\mathbf{x}[n]$
    Encode $\mathbf{w}_n$ and Transmit to next hop
  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 such 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 i) they converge to the optimal filters for stationary data, ii) they do not require estimates of data statistics and iii) the filtering done at one node can be replicated at any other node (e.g., the sink) given the same prediction errors and initial prediction filters. As such, we can *apply an adaptive filter at each node* to estimate the optimal prediction filters *without forwarding additional information* to the sink. Note that 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.) However, this method avoids the overhead to communicate data statistics for the T-KLT.

There are a variety of adaptive filters we can choose from, but the step-size parameter μ often must be chosen based on some data dependent parameters to ensure filter convergence. We generally will not know such parameters, so the most suitable choice is a normalized least mean squares adaptive filter since μ need not be specified but is instead adapted as the filter is adapted. Some notation is now established. Suppose nodes measure data at times $t_1, t_2, \ldots, t_M$. Let $x(n,m)$ denote the data at node n captured at time step t_m. The $N \times M$ prediction coefficient matrix 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. The *adaptive filter at each prediction node* n is then computed, from $m = 1$ to $m = M$, via $d(n,m) = x(n,m) - \mathbf{p}_n^t(\mathcal{N}_n, m)x(\mathcal{N}_n, m)$ and the filter update equation $\mathbf{p}_n(\mathcal{N}_n, m+1) = \mathbf{p}_n(\mathcal{N}_n, m) + \mu \frac{x(\mathcal{N}_n,m)d(n,m)}{x^t(\mathcal{N}_n,m)x(\mathcal{N}_n,m)}$.

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. Under that 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. This can be addressed by requiring 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. Each $n \in \mathcal{U}$ must also forward raw data two hops to avoid repeatedly de-coding and encoding coefficients. This results in essentially the same 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 $x(l,m)$.

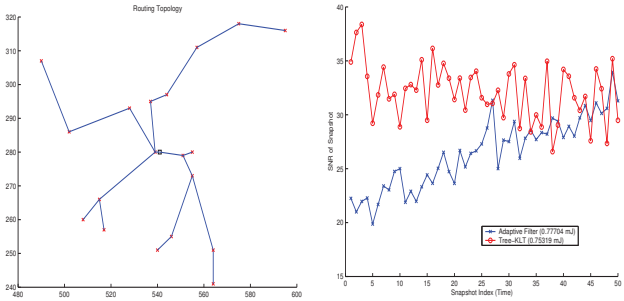
We can also apply these adaptive filters to a simple tree-based DPCM (T-DPCM). As data is forwarded to the sink, each node n will have access to its children's data and so can predict its own data $x(n,m)$ with data from its children then encode and forward the difference, i.e., compute $\hat{x}(n,m) = \sum_{k \in C_n} \mathbf{p}_n(k,m)x(k,m)$, then encode and forward $d(n,m) = x(n,m) - \hat{x}(n,m)$. 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 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. 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.

We also examine the transient behavior of our filter optimization scheme. Fig. 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 method converges to SNR values similar to the T-KLT (i.e., is nearly optimal) in about 40 iterations.



(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.

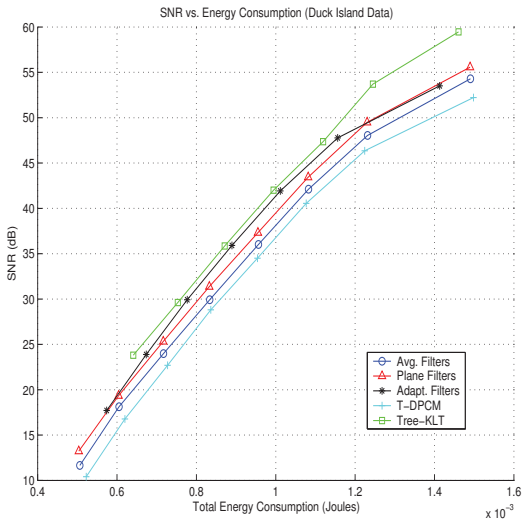


Fig. 2. Performance comparisons.

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*, April 2006.

[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*, April 2006.

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

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