Using more realistic data models to evaluate sensor network data processing algorithms

Yan Yu, Deborah Estrin, Ramesh Govindan, Mohammad Rahimi
Email: yanyu/destrin@cs.ucla.edu, govindan@usc.edu, mhr@cens.ucla.edu

ABSTRACT

Sensor network research is still in its infancy. Few real systems are deployed and little experimental data from sensor networks is available to test proposed protocol designs. Due to lack of experimental data and sophisticated models derived from such data, most data processing algorithms from the sensor network literature are evaluated with data generated from simple parametric models.

We identify a few widely-studied classes of problems that are potentially sensitive to data input: Statistics estimation of the field data; Data compression; and Field estimation. We use them as examples to investigate the dependency of algorithm performance on data.

For each class of problem, given the selected problem and algorithm instance, we systematically study how the algorithm performance varies across a range of data input. We also demonstrate how different data input can change the algorithm performance dramatically, the performance comparison between two algorithms may even change depending on the different data inputs.

In the end, we propose our synthetic data generation framework and recommend evaluating algorithms across a wide range of data input.

1 Introduction

Sensor network research is still in its infancy. Few real systems are deployed and little experimental data from sensor networks is available to test proposed protocol designs. Due to lack of experimental data and sophisticated models derived from such data, most data processing algorithms from the sensor network literature are evaluated with data generated from simple parametric models. For example, it is common that data collection and estimation algorithms are evaluated with uniform or Gaussian data input [3, 11]. Similarly, a random Gaussian field model is typically used in the analytical and simulation studies to evaluate the compression and source coding algorithms [5].

Many of these data processing algorithms are sensitive to data input. If this sensitivity only produces small perturbations on the algorithm performance, simple parametric models used in the simulation helps to clarify the algorithm behavior without the distraction from the details of data input. Unfortunately, the type of the data input used in the evaluation often significantly affect the algorithm evaluation results, as elaborated by the following examples:

In the first example, we demonstrate that for a given performance accuracy requirement, the performance of one algorithm may appear to be acceptable for certain data set, but unacceptable for another input data set.

We evaluate a uniform sampling based median computation algorithm [10] (Section 3.1) against 4 data sets: data generated from uniform, Gaussian, and Bimodal distribution; and a one-day snapshot from the S-Pol radar data set[1]. Our performance metric here is communication cost, which is proportional to the sampling rate. To achieve the same precision, the particular experimental data set we used and the bimodal distributed data require an order of magnitude more samples than Gaussian and uniformly distributed data. Specifically, in order to obtain similar precision as uniformly distributed data at the cost of 1% sampling rate, it requires 8 - 10% sampling rate in the case of the experimental data. Furthermore, to obtain similar precision as a normal distributed data set at the cost of 1% sampling rate, the experimental data requires 30% sampling rate.

The significantly higher cost in the case of experimental data set and bi-modal distributed data may suggest that uniform sampling based median computation approach is not feasible for these two data distributions. The results from using our experimental data suggest that the median is under-represented, thereby the median estimated from uniform sampling will not work well in practice.

In the second example, we will show that the performance comparison between two algorithms may change depending on the different data inputs.

As reported in Section 5.1, we compared two field estimation algorithms, Adaptive Sampling vs. Raster Scan against two types of data: (1) data generated from simple parametric models and (2) data collected from a lab environment. For each data distribution, we compare the reconstruction accuracy of Adaptive Sample vs. Raster Scan under the same cost budget, *i.e.*, the same number of samples. In terms of Mean Squared Error (MSE) in the reconstruction results, for data generated from linear and quadratic models, Adaptive Sampling is orders of magnitude better than raster scan; whereas for a data set collected from the lab environment, Adaptive Sampling is worse than raster scan.

Given that different data inputs can change algorithm performance dramatically, evaluating the system with data representing a diverse range of real-world scenarios is essential. Most available

[1] S-Pol (S band polar metric radar) data were collected during the International H 2O Project (IHOP; Principal Investigators: D. Parsons, T. Weckwerth, et al.). S-Pol is fielded by the Atmospheric Technology Division of the National Center for Atmospheric Research. We acknowledge NCAR and its sponsor, the National Science Foundation, for provision of the S-Pol data set.

experimental data from remote sensing or in-situ instrumentation are collected from a regular grid. Therefore, they cannot be directly used to evaluate the sensor network algorithms since the deployed sensor networks are most likely in an irregular topology. In the end, we propose to generate synthetic data from models derived from the experimental data.

Organization of the paper. We identify a few widely-studied classes of problems that are potentially sensitive to data input: Statistics estimation of the field data; Data compression; and Field estimation. We use them as examples of how to systematically study the dependency of algorithm performance on data. We present our evaluation results on statistics estimation and field estimation in Section 3, and 5 respectively, and refer readers to [16] for results on data compression algorithms. In the end, we propose our synthetic data generation framework and recommend evaluating algorithms across a wide range of data input.

2 Data distributions and algorithm performance

In this section, we identify a few widely-studied classes of problems that are potentially sensitive to data input:

- Statistics estimation of the field data, *e.g.*, median, percentile, or density estimation, among which we studied median and percentile estimation in this paper.

- Data compression, *e.g.*, distributed source coding, entropy coding, or coding schemes that exploit the spatial and temporal correlation in the data. We studied an instance of wavelet compression, and joint entropy coding algorithm.

- Field estimation. There has been extensive interest in sampling and reconstruction of a physical field [12, 15, 2, 9] in recent sensor network literature. We investigated fidelity driven sampling in our case study.

We would like to clarify that providing a complete taxonomy of sensor network algorithms that are potentially sensitive to data input is not our intention. Instead, we investigate a few widely studied algorithms and use them as examples of how to systematically study the dependency of algorithm performance on data. We use the above three classes of problems as our case studies for the following reasons:

1. Many sensor network systems are deployed to monitor and understand the physical environment. Due to energy constraints, most of the time, sensor network applications can not afford to transmit every bit of sensed information back to the base station. Therefore, statistics summary, data compression, and efficient field estimation algorithms are essential to building a long-lived system.

2. The algorithms in the above three categories all need to exploit the statistics and redundancy in the data in some manner and therefore are potentially sensitive to data input.

For the first two classes of problems, given the selected problem and algorithm instance, we studied how the algorithm performance varies across a wide range of data input. In the third category of problems, we studied the Fidelity Driven Sampling algorithm. Given its complexity, instead of a systematic study, we use a few data sets to demonstrate its sensitivity to data input and how it may even change its performance comparison results relative to a simple alternative, namely, raster scan.

Due to the space limitations, in the next two sections, we present results on the first and third class of problems, and refer readers to [16] for results on the second class of problem.

3 Statistics estimation problem

We start with a definition of the statistics estimation problem we consider here. A random process, Z, which consists of a set of random variables $Z(u)$, one for each sensor location u in the deployment area A, denoted as $\{Z(u), \forall u \in A\}$. Assuming Z is a stationary process, the problem is to estimate some statistics (*e.g.*, the median) of $Z(u)$ ($u \in A$) given a realization of $\{Z(u)\}$ (*i.e.*, a time snapshot of sensor readings) at locations u_i, $i = 1, ..., n$, $u_i \in A$.

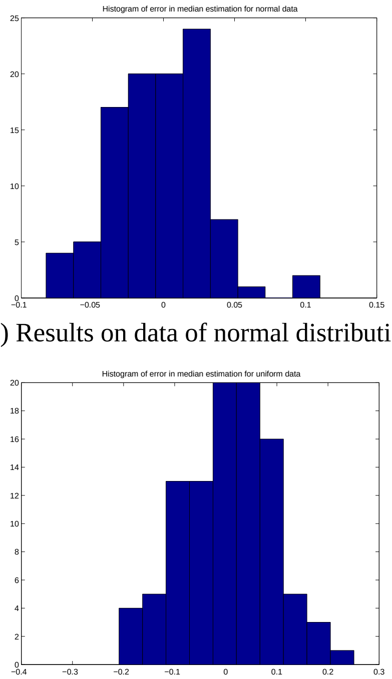
In this section, we consider the median and percentile estimation problems; and two order-statistics estimation algorithms, namely, uniform sampling based approach, and Power-Conserving Computation of Order-Statistics proposed in [7]. We investigated how the performance of these algorithms is affected by data characteristics across a wide range of parameters.

3.1 Uniform sampling-based median computation

For simplicity of illustration, here we consider an instantiation (a snapshot) of random process $Z(u)$ as deterministic. Thus, given a snapshot of sensor readings at each node, $z(i)$, $i = 1$, ...,n, the real median $M(Z)$ is defined as $z(\frac{n+1}{2})$, if n is even, or $(z(\frac{n}{2}) + z(\frac{n}{2}+1))/2$ if n is odd. The estimated median is written as $\hat{M}(Z)$.

There are variations of uniform-sampling based approach to estimate $M(Z)$. Without loss of generality, in this paper we consider a specific uniform sampling based median computation algorithm as follows: Suppose there is a single sink in the deployed sensor network, and each sensor node has the same probability (*e.g.*, 1%) of sending its reading back to the sink. The sensor value is forwarded along the shortest path tree from the sensor source to the sink. An intermediate node on the shortest path tree will not prune samples, instead it simply relays these packets back to the sink. Suppose p samples are transmitted back to the sink, s(1),..., s(p), we compute the median, M(S) of s(1),..., s(p) as an estimate of $M(Z)$.

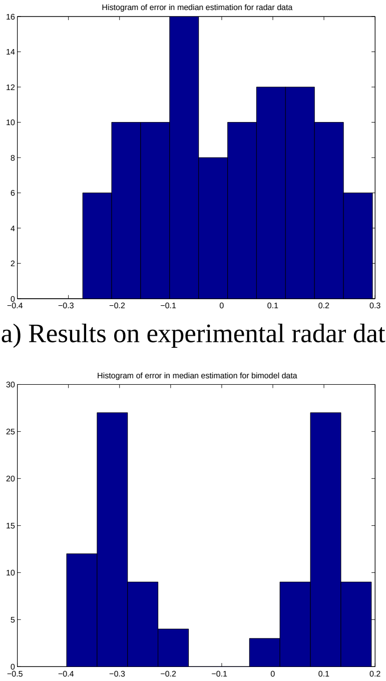
Before systematically evaluating how the algorithm performance are affected by different data input, we use four example data sets to demonstrate that different data input can dramatically affect the algorithm performance. Specifically, we consider data sets generated from uniform, Gaussian, Bi-modal distribution, and S-Pol radar data set. The S-Pol radar data provided by NCAR records the intensity of reflectivity in dBZ, where Z is proportional to the returned power for a particular radar and a particular range. We selected 259 time snapshots across 2 days in May 2002 in our study. Each snapshot of data in our study is a 60 x 60 spatial grid data with 1 km spacing.



(a) Results on data of normal distribution

(b) Results on data of uniform distribution

Figure 1: Histogram of estimation error on data of normal or uniform data distribution is close to the normal bell shape (x-axis: normalized estimated median error; y-axis: number of occurence)



(a) Results on experimental radar data

(b) Results on data of bimodal distribution

Figure 2: Histogram of estimation error for radar data and bimodal distribution is wide spread out or bi-modal, and the average estimation error is also larger than that in Gaussian and uniform distribution. (x-axis: normalized estimated median error; y-axis: number of occurence)

We define the performance metric of the median computation algorithm to be **normalized estimated median error**, which is *the difference between the estimated median, $\hat{M}(Z)$, and the real median, $M(Z)$*, normalized by *the range of entire sample values*. Normalization was introduced to make comparing results across different data sets meaningful. Median is often used as a robust estimator instead of mean. Therefore, we defined the error metric in terms of value, as opposed to position. [2]

The histogram of normalized estimation errors against the four data sets mentioned above was presented in Figures 1 and 2. For normal data (Figure 1(a)) and uniform data input (Figure 1(b)), the estimation error is close to the normal bell shape; whereas, for bimodal data (Figure 2(b)) and the radar data (Figure 2(a)), Both the mean and variance of error are higher than for the other distributions. Above we use four data sets as an example to illustrate to what extent the uniform sampling-based median computation algorithm can be sensitive to data distribution. Next, we systematically (using statistics tools) investigate how the algorithm performance varies across a wide range of data distribution.

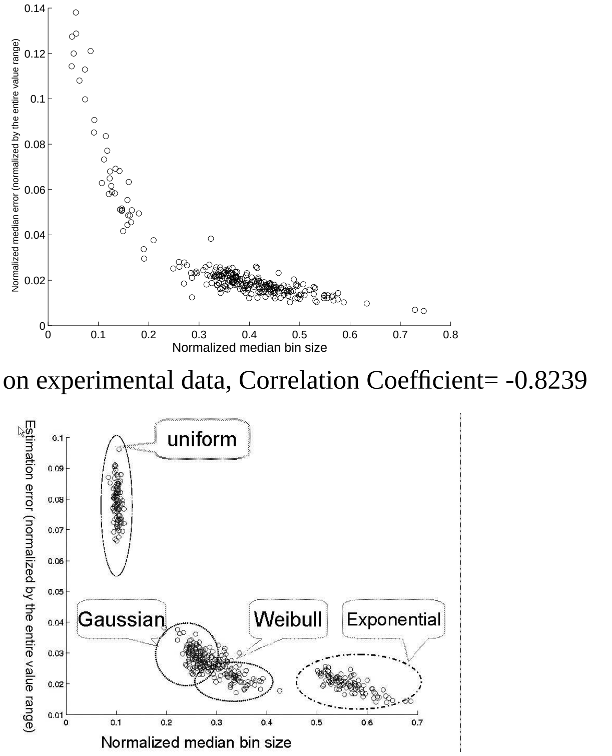
Our statistical analysis consists of three key steps. First, we define our performance metric to be *normalized estimated median error*. Second, we identify the data characteristic that is most relevant to the median computation algorithm to be *normalized median bin size*. We bin the entire sample set into a fixed number (*e.g.*, 10) of equally spaced containers. *Median bin* is defined as the container that includes the median. Let n denote the total number of samples, and m denote the number of samples in the *median bin*, the **normalized median bin size** is defined as m/n, *i.e.*, the ratio of the size of the median bin relative to the size of the entire data set.

As a final step, we study how the algorithm performance changes with data input across a wide range of parameters. We evaluate the algorithm against two types of data: simulated data generated from Gaussian, Exponential, and Weibull distributions; and 259 snapshots of the experimental radar data. Note that data generated from simple parametric models have been widely used in previous algorithm evaluations. We vary the parameters in the above model in a wide range: in Gaussian distribution, we fix the mean, vary the standard deviation from 1 to 100; in Weibull distribution, we fix the scale parameter, vary the shape parameter from 1 to 100; and in Exponential distribution, we vary the exponential

[2] If the estimation error metric is defined in terms of order (*i.e.*, let p denote the real position of the estimated median $\hat{M}(Z)$ in the original data set, $n/2$ is the position of the real median, the estimation error is then defined as $p - n/2$), we can prove (Appendix A) that uniform-sampling based median computation is not sensitive to data distribution. Intuitively, uniform sampling was applied in the spatial dimension, when the error metric is defined in the same dimension, *i.e.*, position in an ordered list or space, the estimation error will not be sensitive to the underlying data distribution.

changing rate λ from 0.1 to 10.

Figure 3 shows the scatter plot of *normalized estimated median error* vs. *normalized median size*. Each point in the graph corresponds to one data set. The x-axis is the *normalized median bin size of the data*, the y-axis is the *normalized estimation error* averaged from 100 runs of algorithm on the corresponding data set.



(a) Results on experimental data, Correlation Coefficient= -0.8239

(b) Results on data generated from Gaussian, uniform, Weibull and exponential distributions, Correlation Coefficient=-0.7818

Figure 3: the scatter plot of normalized estimated median error vs. normalized median size: the normalized median error is well correlated with the normalized median bin size; with increasing normalized median bin size, the estimation error decreases. Further, the experimental data covers a super set of all 4 families of data distributions and more.

As shown in Figure 3, for both experimental data and data generated from parametric models, the algorithm performance (*i.e.*, the normalized median error) is well correlated with our defined data characteristic, namely, *the normalized median bin size* . With increasing normalized median bin size, the estimation error decreases. Intuitively, under uniform sampling, with increasing normalized median bin size, more samples from the median bin will appear in the final sample set at the sink; therefore, there is a higher chance that a sample from the *median bin*, will be selected as the estimated median at the sink, as oppose to samples from other bins. Because samples from the same bin are close in value, the *estimated median* will be close to the real median in value.

We also compute the correlation coefficient of *the normalized median error* and *the normalized median bin size*. It is -0.8239 and -0.7818 for the data in Figure 3(a) and 3(b) respectively. The fact that the correlation coefficients being close to -1 also indicates the strong correlation between them.

We would like to point out another interesting observation from Figure 3: In terms of normalized median bin size, experimental data encompasses a super set of all four families of data distributions (as mentioned above, for each distribution family, we vary the parameter across a wide range). This may suggest that, it would be difficult to cover a wide range of data characteristics if we only use data generated from simple parametric models in our algorithm evaluation. The experimental data sets cover a wide range of data characteristics not covered by any single distribution. We believe this result strongly suggest the importance of experimental data in algorithm evaluations.

3.2 Uniform sampling-based percentile computation

Without loss of generality, we define percentile as follows: for an ordered data set, $z(i)$, $i = 1$, ..., n, its *p* percentile is defined as $z(\lfloor p * n \rfloor)$. Similar to median computation, in uniform sampling based percentile computation, each sensor node has the same probability of sending its reading back to the sink. An intermediate node simply relay the sample along the shortest path tree back to the sink. If eventually t samples are transmitted back to the sink, $s(1)$,..., $s(t)$, we compute the p percentile of $s(1)$,..., $s(t)$ as an estimate of the p percentile of the original sample set, $z(1)$, ..., $z(n)$.

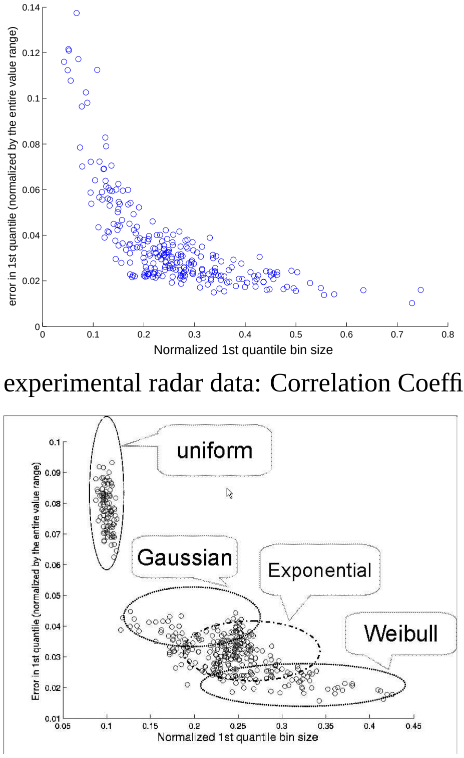
Similarly we identify the data characteristics as *normalized p-percentile bin size*: the entire sample set is divided into a fixed number of equally spaced containers. The *p-percentile bin* is defined as the container that includes the *p-percentile*. Let *n* denote the total number of sensor readings, and *p* denote the number of samples in the *p-percentile bin*; then the *normalized p-percentile bin size* is defined as p/n.

Here we presented results on *1st-quartile* (*i.e.*, 25-percentile). We evaluate the algorithm against the same data used in the median computation. Figure 4 shows the scatter plot of normalized estimation error vs. normalized 1st-quartile bin size. We observe the same algorithm performance behavior as in the case of median computation (Figure 3), i.e., the estimation error increases when the size of the bin that includes the corresponding quartile decreases, which indicates that the corresponding quartile is less represented. We also studied the third quartile estimation, and obtained similar results as in Figure 4. We leave out the details here due to the limit of space.

3.3 Summary

Given *median estimation* and *percentile estimation* problems, and the *uniform sampling-based approach* with different parameters corresponding to these two problems, we above demonstrated the key steps to systematically study how the algorithm performance changes with various data input:

- Define a performance metric for the particular problems and algorithms analyzed.



(a) Results on experimental radar data: Correlation Coefficient=-0.6796

(b) Results on data generated from Gaussian, uniform, Weibull and exponential distributions, Correlation Coefficient=-0.8729

Figure 4: First quartile computation results: the scatter plot of normalized estimation error vs. normalized 1st-quartile bin size. The estimation error increases when the size of the 1st-quartile bin decreases. Again, experimental data covers a super set of all four families of data distributions.

- Identify the set of data characteristics most relevant to the algorithm in the study, *e.g.*, normalized estimation error in the above case study.

- Statistically study how the algorithm performance varies with changing data characteristics. Scatter plots and correlation coefficients were used to identify the correlation between the performance metric and data characteristics in the study.

To demonstrate that it is applicable to the scope beyond the uniform sampling-based median computation, we applied the above proposed statistical analysis to Power-Conserving Computation of Order-Statistics proposed in [7]. We used the same problem definition, performance metric, and data characteristics as defined in Section 3.1, and evaluated PCCOS with the radar data set. Similar to the case of the uniform sampling-based approach, we observed similar correlation between the estimation accuracy and the normalized median bin size (Figure 5). It is evident that the variance of estimation error is larger when the normalized median bin size is small (in the range between 0.1 and 0.3).

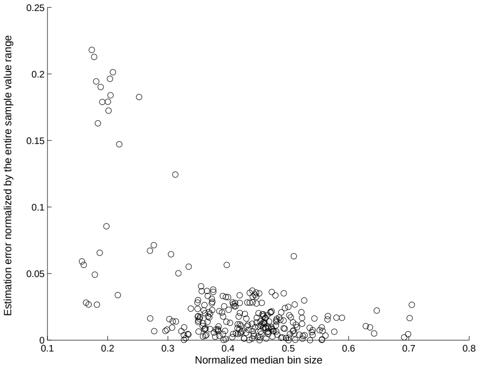


Figure 5: Median computation results from PCCOS algorithm on radar data: scatter plot of estimation error vs. normalized median bin size indicates strong correlation between them. correlation coefficient = -0.58

4 Data compression problem

Since it is prohibitively expensive to transfer raw sensor data in sensor networks, compression is often performed in the temporal domain or in the spatial domain to reduce the communication cost. In this paper, we consider spatial compression. Given a snapshot of sensor readings, $\{Z(u)\}$ (*i.e.*, at locations u_i, $i = 1, ..., n$, $u_i \in A$, a compressed representation of $\{Z(u_i)\}$, instead of $\{Z(u_i)\}$ itself, were transfered to the sink. Depending on whether we can reconstruct the original sensor readings $\{Z(u_i)\}$ without error, compression algorithms can be classified into lossless and lossy compression. In this section, we consider two compression algorithms that are intuitively sensitive to data, an instance of wavelet compression algorithm and joint entropy coding algorithm. They fall into lossy compression and lossless compression algorithms category respectively. We quantitatively demonstrate how the algorithm performance changes with data of various spatial correlations.

4.1 Wavelet Compression Algorithm

In this section, we study an instance of wavelet compression algorithms. As illustrated in Figure 6, it is composed of three steps: (1) we apply wavelet decomposition to the 2D spatial signal. Details on how to implement this in a distributed fashion is out of the scope of this paper. For example, [13] provide a distributed algorithm of wavelet decomposition in the spatial dimension. The results presented here are acquired from applying the Matlab built-in wavelet decomposition function to a 2D spatial data set. Without loss of generality, we use the ”sym2” filter. (2) After wavelet decomposition, most of the energy concentrates on small percentage of coefficients; therefore, we use threshold cutting to discard insignificant coefficents. In the reconstruction phase, these insignificant coefficients were set to 0. (3) We apply run-length coding to the wavelet coefficients that are above a certain threshold.

Among these three steps, wavelet coefficients thresholding are lossy compression, the other two steps are loseless transformation. The diagrams on the bottom of Figure 6 illustrate the decompression steps. In the reconstruction phase, the discarded wavelet coefficients are set to a default value, 0; therefore, the reconstructed wavelet coefficients won't be exactly the same as the original wavelet coefficients. We define the **MSE** to be the **mean squared difference** between the reconstructed data set and the original data set.

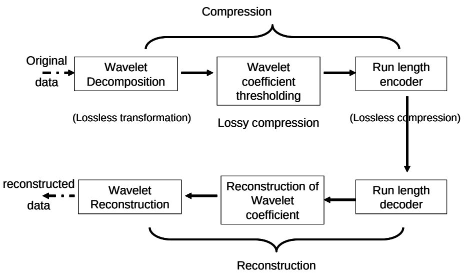


Figure 6: Illustration of wavelet compression and decompression procedure

In step (2), the threshold used to select the wavelet coefficients is proportional to the reconstruction mean squared error; this threshold can be used to adjust the reconstruction fidelity (*i.e.*, mean squared error) requirement. Assuming that uniform quantization is applied to the output of the run-length encoding, the amount of data output of the run-length encoding is proportional to the communication cost measured in bit-hops count. Since the communication cost in transfering the compressed data depends on the distance from the sensor source to the sink, our communication cost metric, the amount of the data output of the run-length encoding, is in terms of spatial bit rate. In our study, we fix the threshold used in selecting the wavelet coefficients, and examine the amount of output data of the run-length encoding.

Following the statistical analysis described in Section 3.3, we first define the performance metric of the wavelet compression algorithm to be the **communication cost** that is measured by *the number of elements in the output from the run-length encoding*.

Second, we identify the relevant data characteristic to be the *spatiall correlation*. In geostatistics, *Variogram* has been used to characterise the spatial correlation in the data. Briefly, the variogram (also called semivariance) of a pair of points x_i and x_j is defined as $\gamma(x_i, x_j) = \frac{1}{2}\{Z(x_i) - Z(x_j)\}^2$. To characterise the spatial correlation between data points separated by various distances, for an isotropical data set, *variogram* can be defined as a function of *separation distance* between two points. To simplify our analysis, we use the variogram value at unit separation distance to represent the spatial correlation in the data. The radar data set under study is from a grid topology, thus, the variogram value at unit separation distance is equivalent to the expected difference between neighboring nodes. This variogram value is identified as the relevant data characteristic in our study.

As a final step, we study how the algorithm performance changes with data input across a wide range of parameters. We evaluate the wavelet comrepssion algorithm against 259 snapshots of the radar data.

In the scatter plot of *communication cost* vs. *spatial correlation*(Figure 7), each point in the graph corresponds to one snapshot of the radar data. The x-axis is its *variogram value at unit separation distance*, which is used to characterize spatial correlation of the data. A small variogram value indicates a strong spatial correlation in the data, or vice versa. The y-axis is the number of elements in the data output from the run-length encoding, which is directly proportional to the communication cost in transfering the compressed data to the sink.

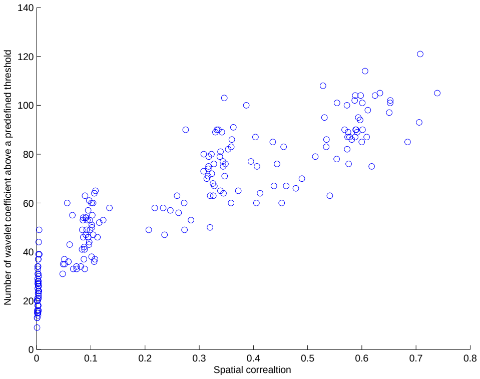


Figure 7: Scatter plot of the communication cost in wavelet compression versus spatial correlation, which is measured in terms of variogram value at unit distance. Communication cost increases with increassing variogram values. Correlation coefficient = 0.9069

As shown in Figure 7, the communication cost increases with the increasing variogram value (*i.e.*, less spatial correlation). In wavelet compression, the wavelet decomposition basically removes the spatial redundancy. Stronger spatial correlation will lead to more spatial redundancy, thereby, wavelet compression will have a better chance to compress the data, as a result, the communication cost will decrease. We also compute the correlation coefficient of the *communication cost* and *the variogram values*. Again, the correlation coefficient, 0.9069, being close to 1 indicates the strong correlation between them.

4.2 Joint entropy coding algorithm

Compared to coding each data point separately, joint entropy coding improves the coding efficiency by exploiting the correlation between sensor readings at neighboring nodes, and coding data from a local neighborhood together. In our case study, we compare two coding policies: (1) apply joint entropy coding to two neighboring nodes; (2) apply entropy coding at each node separately. To evaluate the gain of the joint entropy coding over independently entropy coding at each node, we define the performance metric of the joint entropy coding algorithm to be *the joint entropy of sensor readings at two neighboring nodes* normalized by *Sum of the corresponding marginal entropy*. This normalized communication cost metric is inverse to the *efficiency of the joint entropy coding*.

Since joint entropy = marginal entropy - mutual information (*i.e.*, $H(X,Y) = H(X) + H(Y) - I(X;Y)$ [14]), and mutual information can be captured by the spatial correlation in the data, we identify the relevant data characteristic to be the spatial correlation in the data. Similar to adding normalization to the performance metric, we introduce a normalization factor to the variogram value defined in Section 4.1. We define the *normalized spatial correlation* to be the *variogram value at unit separation distance normalized by the variance of the data*. The normalization introduced here removes the effect of data being measured in different units, and makes data of different magnitudes comparable.

Given the performance metric and relevant data characteristic identified above, we study how the algorithm performance changes with different data input. Again, we use 259 snapshots of radar data in our evaluation. In the scatter plot of *normalized communication cost* vs. *normalized spatial correlation* (Figure 7), the x-axis is its *variogram value at unit separation distance normalized by the variance of the data*, which is used to characterize the spatial correlation in the data. The y-axis is the *joint entropy normalized by the sum of the marginal entropies*, which is directly proportional to the spatial bit rate in sending the compressed data to the sink.

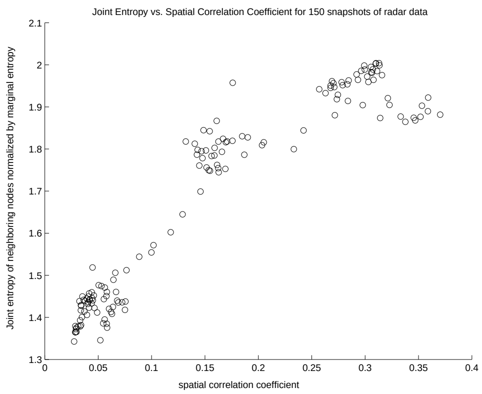


Figure 8: Scatter plot of the normalized communication cost versus normalized spatial correlation. Communication cost increases with increassing variogram values. Correlation coefficient = 0.9355

As shown in Figure 8, the communication cost of joint entropy coding increases with the increasing normalized spatial correlation (*i.e.*, less spatial correlation). Both the scatter plot and the correlation coefficient (0.9355) indicates that the efficiency of the joint entropy coding algorithm is well correlated with the spatial correlation in the data.

5 Field estimation

Densely deployed sensor networks provide unprecedented capability for environmental monitoring. The objective of the *field estimation* is to reconstruct a map of the physical field at the sink. Due to the energy constraints in sensor networks, field estimation algorithms resort to efficient sampling and data transportation techniques to reduce the communication cost and provide a high-fidelity reconstruction of the field at the same time.

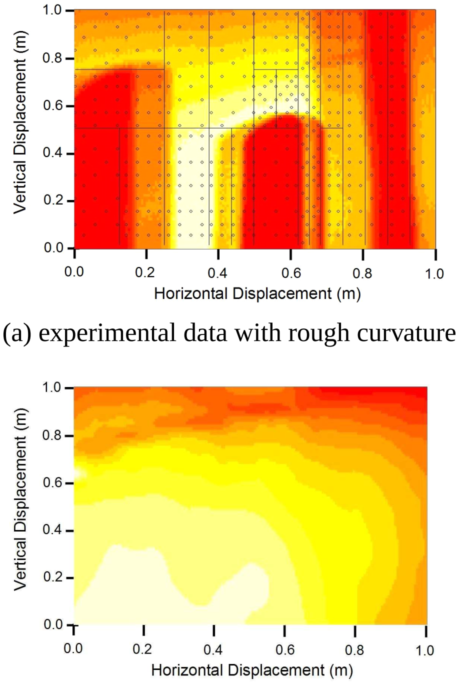
5.1 Adaptive sampling

Fidelity Driven Sampling (proposed in [12, 1]) exploits mobile sampling to first stratify the environment into regions requiring varying degrees of sample density, then samples in these regions. Fidelity Driven Sampling (FDS) maintains an estimate of the field being observed. Using this estimate, the FDS identifies regions or strata exhibiting a high degree of misfit. At each step in the sampling process, FDS adds points to that stratum with the largest error. In so doing, FDS attempts to reduce the mean squared error at each sampling point by adjusting point density and location. The algorithm continues adding points to poor fitting strata until either an overall sample budget is exhausted or a desired fidelity limit is achieved. A simple alternative to Adaptive Sampling is to raster scan the field with the fixed resolution.

Following the Fidelity Driven Sampling operation (or raster scanning data acquisition), the returned variable field with its distribution of sample points is then supplied to a local polynomial interpolation algorithm and returns a reconstruction of the environmental field. We define the performance evaluation metric as the Mean Squared Error (MSE) between this reconstructed field map and the ground truth. We plot the Mean Squared Error achieved in Adaptive Sampling or Raster Scan against the total number of samples collected by the Adaptive sampling or Raster scan (Figures 10 and 11). The MSE represents the quality of the reconstructed field map, and the cost or delay to achieve this reconstruction is proportional to the number of samples. Thus the performance curves in Figures 10 and 11 reflect the quality vs. delay or cost trade-offs. The lower the curve, the more desirable the performance is.

The Adaptive Sampling algorithm can be evaluated using simulated data generated from linear, quadratic, and cubic models [12]. As shown in Figure 10, when evaluated with data simulated from simple models, both Adaptive Sampling and Raster Scan deliver a very small MSE. Further, in most cases, the Adaptive Sampling performs several magnitudes better than Raster Scan in terms of MSE for the same budget (which is measured in the *number of samples*). However, this conclusion does not hold when evaluated with experimental data collected from a lab environment.

As discussed in [1], Adaptive Sampling is evaluated by subjecting the algorithm to environmental variable fields having two extremes in their “curvature” characteristics. For one limit, the environmental variable field was created by placing many obstacles in the illumination field (Figure 9(a)) to emulate the most complex patterns observed in the natural environment. In addition, we created a low curvature field by casting only diffuse shadowing on the transect (Figure 9(b)). This latter case is characteristically similar to the least complex fields observed under clear forest canopy structure. In both cases, the “ground truth” was obtained by measurements from exhaustively moving the node at its highest resolution through the variable field. As shown in Figure 11, when evaluated with the experimental data, the MSE obtained from Adaptive Sampling is very close to or worse than the MSE obtained from Raster Scan.



(a) experimental data with rough curvature

(b) experimental data with smooth curvature

Figure 9: Experimental data collected from a lab environment

5.2 Summary

In summary, compared with using data simulated from simple models, evaluating the Adaptive Sampling algorithm with experimental data changes its performance comparison results relative to Raster Scan. This may suggest that the evaluation results from data input based solely on simple parametric models may be misleading. Evaluating algorithms using experimental data with various features helps identify the regime of the parameter space where the algorithm may perform well compared to other alternatives. Due to the complexity of the physical phenomena, it is impossible to represent all possible data input using only parametric models. Experimental data guides our efforts to the portion of the parameter space that matters in practice.

The dramatically different performance caused by different data inputs may not be unique to the Adaptive Sampling algorithm alone. For example, the Backcasting algorithm proposed in [15] shares a similar spirit as the Adaptive Sampling in that they both adjust the sampling density based on the initial coarse reconstruction of the field map. Therefore, the Backcasting algorithm might be subject to the same problem, *i.e.*, sensitivity to the environmental field being sensed. In [15], the algorithm was evaluated using a simulated piecewise smooth field with a single edge. Evaluating the algorithm using experimental data with different features as in Figure 9 may help us understand the algorithm's different performance under various types of environmental fields.

6 Proposed algorithm evaluation with more realistic data input

6.1 Synthetic data generation

The huge parameter space of data input makes exhaustive exploration of parametric models impractical. By driving simulations from previously collected experimental data, we focus our testing on the part of parameter space that matters in reality. Unfortunately, collecting new experimental data sets is expensive, time-consuming, and often presents technical challenges in itself. Thus, it is advantageous to develop methods of applying previously collected data sets in the development of new algorithms.

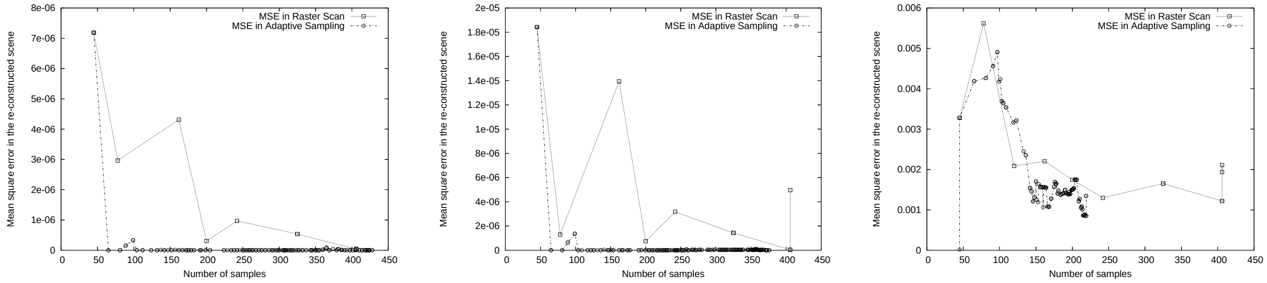
This application of old data to new problems presents several challenges. First, existing experimental data is often collected from regular grids, whereas real deployments may have an irregular topology. Second, the important features of the data are application dependent. Leveraging the existing experimental data we propose to generate synthetic data of irregular topology from modeling the experimental data. Our proposed synthetic data generation techniques attempt to approximate the experimental data in terms of distribution, spatial correlation, or other features of interest.

We first sketch out some principles in our proposed synthetic data generation approach:

1. Our approach is experimentally oriented. Due to reasons stated above, we recommend generating synthetic data from models driven by the experimental data, rather than from purely parametric models.

2. The selection of data features to model should be driven by the relevant applications. Since the underlying physical phenomena it represents are inherently different, data from different application domains, or data from different modalities may dramatically differ from each other in terms of distribution, spatial correlation, frequency spectrum, *etc*. For example, environmental weather data could be very different from seismic data in many characteristics that are relevant to the algorithm in the study.

3. The specific synthetic data generation technique used should adapt to the application and algorithm in the study. Here we propose a methodology to generate synthetic data, as opposed to recommending a single synthetic data generation algorithm or a single synthetic data set. Our synthetic data generation tool-box introduced in [17] includes eight spatial interpolation algorithms, which in turn will generate eight different data sets based on one single experimental data set. Which synthetic data set is more desirable depends on the specific application and algorithm in the study. For example, in evaluating a wavelet compression algorithm [17], *spatial correlation* was identified as the data characteristics essential to wavelet compression. We then select the synthetic data

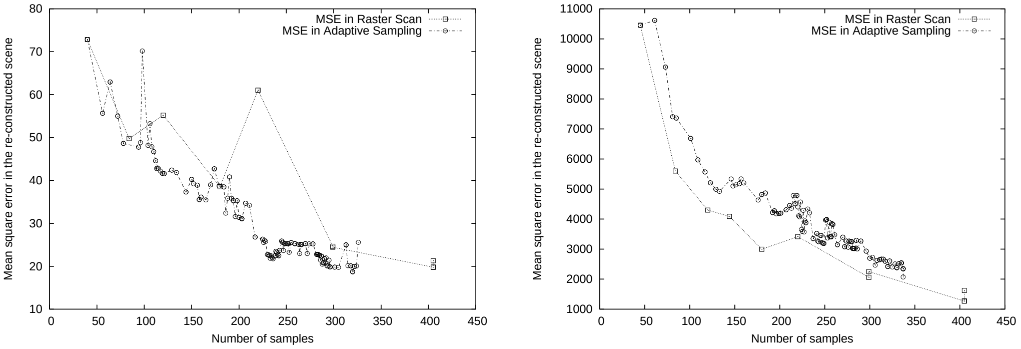


(a) Result on linear data

(b) Result on quadratic data

(c) Result on cubic data

Figure 10: comparison of Adaptive Sampling vs. Raster Scan on data generated from linear, quadratic, cubic models. Both Adaptive Sampling and Raster Scan deliver very small MSE; however, Adaptive Sampling performs several magnitudes better than Raster Scan for the same cost, as measured in the number of samples.



(a) Result on data with smooth curvature

(b) Result on data with rough curvature

Figure 11: Comparison of Adaptive Sampling vs. Raster Scan. Results on data collected from a lab environment. The MSE achieved by Adaptive Sampling is very close to or worse than Raster Scan.

set that can best match the experimental data in terms of the *spatial correlation*. Identifying relevant data characteristics (or in other words, the evaluation metric for synthetic data generation techniques) requires knowledge of the algorithm being evaluated.

Next we briefly describe the procedure to generate irregular topology data based on the experimental data (please refer to [17] for details).

1. Generate ultra fine-grained data using the spatial modeling and interpolation techniques discussed in [17]. The objective in this step is to create a grid topology at a much finer granularity than our target topology.

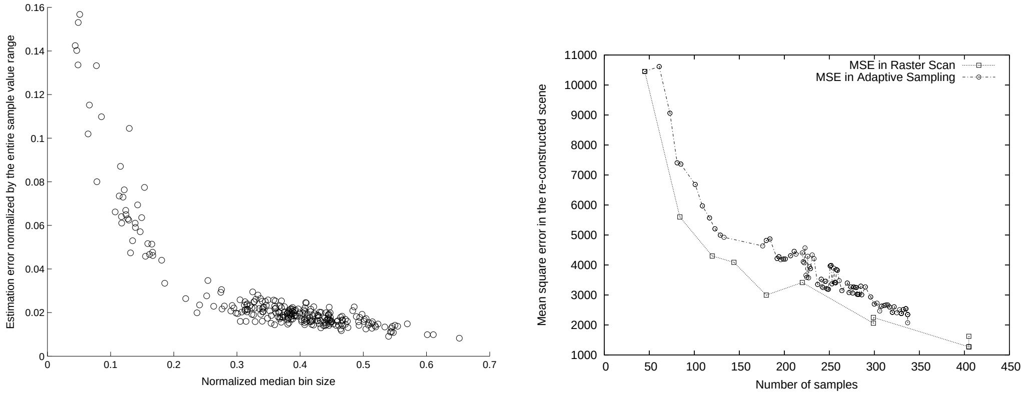
2. Overlay the target topology on top of the ultra fine-grained data obtained from above. Each node in the target topology is assigned a value from the nearest grid.

Among synthetic data sets generated from different spatial interpolation techniques in step 1, we select the one that can best match the original experimental data in terms of whatever data characteristics are of interest.

We evaluated the median computation algorithm discussed in Section 3.1 using our synthetic data, which is generated based on the radar data. Figure 12(a) verifies that the median computation algorithm evaluated with our synthetic data exhibits similar behavior as that evaluated with the experimental data, and our synthetic data covered a similar range of data distribution as that of the experimental data. In addition, we also applied the same synthetic data set to the Adaptive Sampling algorithm. The evaluation results (Figure 12(b)) provide similar insight as the evaluation using the data collected from the lab environment (Figure 11(b)): the MSE achieved by Adaptive Sampling is slightly worse than Raster Scan.

6.2 Algorithm evaluation across a wide range of parameters

Whenever possible, it is desirable to reduce the dependency of algorithm performance on data. As demonstrated in Section 3.1, the uniform sampling-based median computation algorithm is sensitive to data inputs. Next, we introduce *selective sampling based*



(a) Median computation results, similar to Figure 3(a), estimation error decreases with increasing normalized median bin size

(b) Adaptive Sampling results: similar to Figure 11(b), MSE achieved by Adaptive Sampling is worse than Raster Scan

Figure 12: Evaluation results using synthetic data generated from the radar data: the algorithm exhibits similar behavior as the evaluation results using the experimental data.

median computation. We are *not* trying to propose the best available median computation algorithm. Instead, it serves as an example algorithm that explicitly takes data distribution into consideration, and reduces the dependency of the algorithm performance on data distribution.

We first describe a primitive (called *selective sampling*) used in the algorithm. Basically, it selects m samples from a sample set S, where $m \leq |S|$. The procedure is as follows: (1) Sort the samples in S; (2) Divide S into m equal-ranged containers, *i.e.*, S is divided into S_1, S_2, ..., S_m; (3) from each bin S_i, select the median of S_i (denoted as M_i) to represent S_i, with weight W_i equal to the number of elements in S_i. After this step, we have m pairs (W_1, M_1), (W_2, M_2), ..., (W_m, M_m).

Next we briefly describe the *selective sampling* based median computation algorithm:

- At the leaf node, each node reports its sensor reading, s. This sensor reading is sent along the shortest path tree back to the sink.

- At an intermediate node A, suppose A has k children. Each child reports a list of (W_i, M_i). Each list has at most k tuples, where k is determined by the communication budget. Combine the inputs from all child nodes together, treat each (W_i, M_i) as W_i copies of M_i, Use the primitive *selective sampling* described above to output a new list of (W_i', M_i'). Note that the *sort* operation in *selective sampling* is *not* computationally intensive since it is *merge sort* from k sorted list, as opposed to sort from scratch, where k is the number of children at an intermediate node.

- At the sink, combine inputs from all its children. Suppose that there are m tuples in total, (W_1, M_1), (W_2, M_2), ... (W_m, M_m). When computing the median from these m tuples, we treat each (W_i, M_i) as W_i copies of M_i.

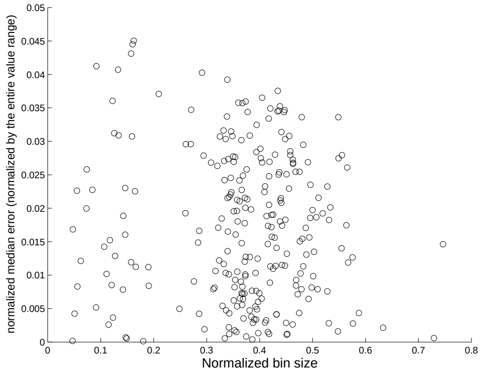


Figure 13: Results from Selective Sampling algorithm evaluated with the radar data set: normalized estimation error vs. normalized median bin size indicates no correlation between them

We evaluate the *selective sampling based median computation algorithm* using the radar data set. The scatter plot of the normalized estimation error vs. the normalized median bin size (Figure 13) indicates no correlation between them. This may suggest that the algorithm is not sensitive to data distribution.

Even though we demonstrated that a simple improvement to the uniform sampling based approach reduces the algorithm's sensitivity to data distribution, it may be difficult or impossible to design such an improvement for other types of problems. Compared to field estimation or other problems in data processing, median computation is relatively simple. However, when it is difficult to design an algorithm that can explicitly remove its sensitivity to data input, we recommend evaluating algorithms with data across a range of parameters, and investigate how the algorithm's performance changes with different data characteristics. The parameter of interest could be data distribution, spatial correlation, or other data characteristics. The synthetic data generation approach discussed in Section 6.1 can be used to generate realistic data sets with a wide range of parameters.

The statistical approach introduced in Section 3.1 can be used to systematically study how the algorithm behaves in different parts of the parameter space. The challenge of the systematic study is to identify what characteristic of the data set best defines the data dependency for a given algorithm. For example, in the median or percentile computation, we identified *the size of the bin that includes the median or the corresponding percentile* to be the data characteristic of interest. In the case of wavelet compression, spatial correlation is identified to be of major interest; in joint entropy coding, we identify the interesting characteristic to be the normalized spatial correlation coefficient. Identifying the relevant set of data characteristics usually requires fair understanding of the algorithm under evaluation.

7 Related work

In the context of the Internet research, Floyd *et al.* [4] use examples drawn from Active Queue Management and TCP variants to illustrate the problems caused by inappropriate models. It identifies the need of a richer understanding of the range of realistic models, and the relevance of different model parameters to network performance. It proposes to build models based on the existing measurement results, and to generate new empirical results when needed.

In [17], we propose synthetic data generation techniques to support irregular topology sampling in sensor networks, In this paper, we expanded the previous work in the following aspects:

- In [17], we pointed out the problem, without elaboration, that sensor network algorithms are potentially sensitive to data inputs. In this paper, we use the case studies drawn from statistic estimation and field estimation problem to systematically investigate how the algorithm performance varies across a wide range of data input.

- In addition to recommend generating synthetic data based on modeling the experimental data, we introduce another principle in our synthetic data generation framework, an application and algorithm dependent approach: the initial experimental data should be drawn from relevant applications; The specific synthetic data generation technique and corresponding evaluation metric used should tailor to the application and algorithm under study;

The work in [8] propose a mathematical model to capture the spatial correlation in sensor network data, and to generate large synthetic traces from a small experimental trace. This synthetic data generation technique can be easily incorporated into our proposed synthetic data generation framework. As pointed out in [17], we do not recommend using it to generate traces of finer granularity due to the fact that we lack ground truth data to verify that the synthetic data match the statistics of the experimental data at that fine scale.

8 Conclusion and future work

In this paper, we identified a few widely-studied classes of problems that are potentially sensitive to data input: Statistics estimation of the field data; Data compression; and Field estimation. We use them as examples of how to systematically study the dependency of algorithm performance on data. We also demonstrated how different data input can change the algorithm performance dramatically, the performance comparison between two algorithms may even change depending on the different data inputs. As a result, we recommend evaluating algorithms across a range of data input. We propose to generate realistic data sets with a wide range of parameters. In our proposed synthetic data generation framework, we recommend generating data based on models derived from the experimental data, and the specific synthetic data generation technique used depends on the application and algorithm under evaluation.

In order to encourage algorithm evaluation with realistic data, we are providing a set of synthetic data generation tools and integrate them with Emstar [6] to facilitate emulation with more realistic data input. In addition, we are going to provide a suite of readily usable test data sets to the community.

9 ACKNOWLEDGEMENTS

We are grateful to all LECS members for many helpful discussions, in particular, Alberto Cerpa, Lewis Girod, Andrew Parker, Hanbiao Wang, Ben Greenstein for providing useful feedback on the draft. We also wish to thank Mark Hansen for consulting on the statistics terminology. This work is made possible by a generous grant from National Science Foundation through CENS.

References

[1] M. A. Batalin, M. Rahimi, Y.Yu, D.Liu, A.Kansal, G.S. Sukhatme, W.J. Kaiser, M.Hansen, G. J. Pottie, M. Srivastava, and D. Estrin. Towards event-aware adaptive sampling using static and mobile nodes. Technical Report 38, UCLA/CENS.

[2] B. Beferull-Lozano, Robert Konsbruck, and Martin Vetterli. Rate-distortion problem for physics based distributed sensing. In *IPSN*, 2004.

[3] Ronique Delouille, Ramesh Neelamani, and Richard Baraniuk. Robust distributed estimation in sensor networks using the embedded polygons algorithm. In *Proceedings of the third international symposium on Information processing in sensor networks*, pages 405–413. ACM Press, 2004.

[4] Sally Floyd and Eddie Kohler. Internet research needs better models. In *1st Workshop on Hot Topics in Networks (HotNets-I)*, Oct 2002.

[5] Deepak Ganesan, Rzvan Cristescu, and Baltasar Beferull-Lozano. Power-efficient sensor placement and transmission structure for data gathering under distortion constraints. In *Proceedings of the third international symposium on Information processing in sensor networks*, pages 142–150. ACM Press, 2004.

[6] L. Girod, J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, and D. Estrin. Emstar: a software environment for developing and deploying wireless sensor networks. In *USENIX*, 2004.

[7] M. B. Greenwald and S. Khanna. Power-conserving computation of order-statistics over sensor networks. In *23rd ACM Symposium on Principles of Database Systems (PODS)*, 2004.

[8] Apoorva Jindal and Konstantinos Psounis. Modeling spatially-correlated sensor network data. Technical Report CENG-2004-10, University of Southern California.

[9] Animesh Kumar, Prakash Ishwar, and Kannan Ramchandran. On distributed sampling of smooth non-bandlimited fields. In *IPSN*, 2004.

[10] Suman Nath and Phillip Gibbons. Synopsis diffusion for robust aggregation in sensor networks. Technical Report IRP-TR-03-08, Intel Research.

[11] Michael Rabbat and Robert Nowak. Distributed optimization in sensor networks. In *Proceedings of the third international symposium on Information processing in sensor networks*, pages 20–27. ACM Press, 2004.

[12] Mohammad Rahimi, Richard Pon, Deborah Estrin, William J. Kaiser, Mani Srivastava, and Gaurav S. Sukhatme. Adaptive sampling for environmental robotics. In *IEEE International Conference on Robotics and Automation*, 2004.

[13] S. D. Servetto. Distributed signal processing algorithms for the sensor broadcast problem. In *Proceedings of the 37th Annual Conference on Information Sciences and Systems, (CISS)*, March 2003.

[14] Joy A. Thomas and Thomas M. Cover. Elements of information theory. John Wiley & Sons, Inc., 1991.

[15] Rebecca Willett, Aline Martin, and Robert Nowak. Backcasting: An adaptive approach to energy conservation in sensor networks. In *IPSN*, 2004.

[16] Yan Yu, Deborah Estrin, Ramesh Govindan, and Mohammad Rahimi. Evaluating data processing algorithm in sensor networks with more realistic data input. Technical report, UCLA/CENS.

[17] Yan Yu, Deepak Ganesan, Lewis Girod, Deborah Estrin, and Ramesh Govindan. Synthetic data generation to support irregular sampling in sensor networks. In *Geo Sensor Networks*. Taylor and Francis Publishers, Oct 2003.

A Appendix: the error statistics of median computation based on uniform sampling

Evaluation metric: Let Z denote the poputation of interest, $|Z| = n$, a sample set S is uniformly drawn from Z, $|S| = l$. The median computated from S, x, is returned as an estimate of median of Z. Suppose the position of *x* in the original population Z is x_p, the estimation error is here defined as: $x_p - n/2$.

Next, we consider the *probability distribution of this error*: $P((X_p - n/2) = \epsilon n)$, where ϵ is a small number.

Let random variable X_p denotes the position of the median returned by the algorithm using l samples

Without loss of generality, assume l is odd, and elements in the original population are distinct. The event that "$(X_p - n/2) = \epsilon n$" is equivalent to *exactly* $l/2$ *of the samples have positions less than* $n/2 + \epsilon n$, i.e.,

$$P((X_p - n/2) = \epsilon n) = P(S_l = l/2) \tag{1}$$

where S_l denotes the total number of samples having positions less than $n/2 + \epsilon n$.

Suppose p_i denotes the position for $i - th$ sample,

$Y_i (i = 1, ..., l)$ denotes a random variable, $Y_i = 1$ if $p_i < (n/2+\epsilon n)$; otherwise, $Y_i = 0$.

From uniform sampling, we have:
$P(Y_i = 1) = 1/2 + \epsilon$
$P(Y_i = 0) = 1/2 - \epsilon$

Independently taking l uniform samples is a n *Bernoulli trials with success probability* $p = 1/2 + \epsilon$.

$$P(S_l = l/2) = \binom{l}{l/2} * (1/2 + \epsilon)^{l/2} * (1/2 - \epsilon)^{l-l/2} \tag{2}$$

From Equation 2, we have:

$$P((X_p - n/2) = \epsilon n) = \binom{l}{l/2} * (1/2 + \epsilon)^{l/2} * (1/2 - \epsilon)^{l-l/2} \tag{3}$$

Note that equation 3 only depends on l and ϵ, not on the original population distribution.

This proves that the error statistics (assuming the evaluation metric is in terms of order difference as defined in the beginning) of median computation based on uniform sampling does not depend on the underlying data distribution, but only on the proportions of samples taken.