Compressing Kinetic Data From Sensor Networks

Sorelle A. Friedler* and David M. Mount**

Dept. of Computer Science, University of Maryland, College Park, MD 20742, USA
sorelle@cs.umd.edu mount@cs.umd.edu
http://www.cs.umd.edu/~sorelle http://www.cs.umd.edu/~mount

Abstract. We introduce a framework for storing and processing kinetic data observed by sensor networks. These sensor networks generate vast quantities of data, which motivates a significant need for data compression. We are given a set of sensors, each of which continuously monitors some region of space. We are interested in the kinetic data generated by a finite set of objects moving through space, as observed by these sensors. Our model relies purely on sensor observations; it allows points to move freely and requires no advance notification of motion plans. Sensor outputs are represented as random processes, where nearby sensors may be statistically dependent. We model the local nature of sensor networks by assuming that two sensor outputs are statistically dependent only if the two sensors are among the k nearest neighbors of each other. We present an algorithm for the lossless compression of the data produced by the network. We show that, under the statistical dependence and locality assumptions of our framework, asymptotically this compression algorithm encodes the data to within a constant factor of the information-theoretic lower bound optimum dictated by the joint entropy of the system.

1 Introduction

There is a growing appreciation of the importance of algorithms and data structures for processing large data sets arising from the use of sensor networks, particularly for the statistical analysis of objects in motion. Large wireless sensor networks are used in areas such as road-traffic monitoring [1], environment surveillance [2], and wildlife tracking [3, 4]. With the development of sensors of lower cost and higher reliability, the prevalence of applications and the need for efficient processing will increase.

Wireless sensor networks record vast amounts of data. For example, road-traffic camera systems [1] that videotape congestion produce many hours of video

* The work of Sorelle Friedler has been supported in part by the AT&T Labs Fellowship Program.

** The work of David Mount has been supported in part by the National Science Foundation under grant CCR-0635099 and the Office of Naval Research under grant N00014-08-1-1015

or gigabytes of data for analysis even if the video itself is never stored and is instead represented by its numeric content. In order to analyze trends in the data, perhaps representing the daily rush hour or weekend change in traffic patterns, many weeks or months of data from many cities may need to be stored. As the observation time or number of sensors increases, so does the total data that needs to be stored in order to perform later queries, which may not be known in advance.

In this paper we consider the problem of compressing the massive quantities of data that are streamed from large sensor networks. Compression methods can be broadly categorized as being either *lossless* (the original data is fully recoverable), or *lossy* (information may be lost through approximation). Since lossy compression provides much higher compression rates, it is by far the more commonly studied approach in sensor networks. Our ultimate interest is in scientific applications involving the monitoring of the motion of objects in space, where the loss of any data may be harmful to the subsequent analysis. For this reason, we focus on the less studied problem of lossless compression of sensor network data. Virtually all lossless compression techniques that operate on a single stream rely on the statistical redundancy present in the stream in order to achieve high compression rates [5–7]. In the context of sensor networks, this redundancy arises naturally due to correlations in the outputs of sensors that are spatially close to each other. As with existing methods for lossy compression [8,9], our approach is based on aggregating correlated streams and compressing these aggregated streams.

A significant amount of research to date has focused on the efficient collection and processing of sensor network data within the network itself, for example, through the minimization of power consumption or communication costs [10–12]. We focus on losslessly compressing the data locally and then downloading it to traditional computer systems for analysis. Clustering the stationary sensors is a strategy that has been previously used to improve the scalability as well as the energy and communication efficiency of the sensor network [13]. Compressing the data before transmission additionally improves the communication efficiency.

We are particularly interested in *kinetic data*, by which we mean data arising from the observation of a discrete set of objects moving in time (as opposed to continuous phenomena such as temperature). We explore how best to store and process these assembled data sets for the purposes of efficient retrieval, visualization, and statistical analysis of the information contained within them. We assume that we do not get to choose the sensor deployment based on object motion (as done in [14]), but instead use sensors at given locations to observe the motion of a discrete set of objects over some domain of interest. Thus, it is to be expected that the entities observed by one sensor will also likely be observed by nearby sensors, albeit at a slightly different time. Well-designed storage and processing systems should capitalize on this redundancy to optimize space and processing times. In this paper we propose a statistical model of kinetic data as observed by a collection of fixed sensors. We will present a method for the lossless compression of the resulting data sets and will show that this method is within a constant factor of the asymptotically optimal bit rate, subject to the assumptions of our model.

Although we address the problem of compression here, we are more generally interested in the storage and processing of large data sets arising from sensor networks [8, 15–18]. This will involve the retrieval and statistical analysis of the information contained within them. Thus, we will discuss compression within the broader context of a framework for processing large kinetic data sets arising from a collection of fixed sensors. We feel that this framework may provide a useful context within which to design and analyze efficient data structures and algorithms for kinetic sensor data.

The problem of processing kinetic data has been well studied in the field of computational geometry in a standard computational setting [19–24]. A survey of practical and theoretical aspects of modeling motion can be found in [25]. Many of these apply in an online context and rely on *a priori* information about point motion. The most successful of these frameworks is the *kinetic data structures* (KDS) model proposed by Basch, Guibas, and Hershberger [23], which models objects as points in motion, where the motion is expressed as piecewise algebraic flight plans. Although KDS has been valuable for developing theoretical analyses of points in motion (see [26] for a survey), it is unsuitable in many real-world contexts due to these strong assumptions. Similarly, a framework for sensor placement by Nikoleteas and Spirakis assumes that possible object trajectories are modeled by a set of 3D curves over space and time [14]. Our framework makes no *a priori* assumptions about the motion of the objects.

Algorithms that involve the distributed online processing of sensor-network data have also been studied and successfully applied to the maintenance of a number of statistics online [10, 11, 27, 28]. Efficiency is typically expressed as a trade-off between communication complexity and accuracy or by the amount of communication between a tracker and an observer. The idea of the tracker and observer is reminiscent of an earlier model for incremental motion by Mount *et al.* [29]. Unlike these models, our framework applies in a traditional (non-distributed) computational setting.

Here is a high-level overview of our framework, which will be described in greater detail in Section 2. We assume we are given a fixed set of sensors, which are modeled as points in some metric space. (An approach based on metric spaces, in contrast to standard Euclidean space, offers greater flexibility in how distances are defined between objects. This is useful in wireless settings, where transmission distance may be a function of non-Euclidean considerations, such as topography and the presence of buildings and other structures.) Each sensor is associated with a region of space, which it monitors. The moving entities are modeled as points that move over time. At regular time intervals, each sensor computes statistical information about the points within its region, which are streamed as output. For the purposes of this paper, we assume that this information is simply an *occupancy count* of the number of points that lie within the sensor's region at the given time instant. In other words, we follow the minimal assumptions made by Gandhi *et al.* [30] and do not rely on a sensor's ability to accurately record distance, angle, etc.

Again, our objective is to compress this data in a lossless manner by exploiting statistical dependencies between the sensor streams. There are known lossless compression algorithms, such as Lempel-Ziv [7], that achieve the optimal lower bound encoding bit rate (as established by Shannon [31]) asymptotically. It would be infeasible to apply this observation *en masse* to the entire joint system of all the sensor streams. Instead, we would like to partition the streams into small subsets, and compress each subset independently. In our context, the problem is bounding the loss of efficiency due to the partitioning process.

In order to overcome this problem we need to impose limits on the degree of statistical dependence among the sensors. Our approach is based on a locality assumption. Given a parameter k, we say that a sensor system is *k-local* if each sensor's output is statistically dependent on only its k-nearest sensors. In Section 3, we prove that any k-local system that resides in a space of fixed dimension can be partitioned so that joint compressions involve groups of at most $k+1$ sensors. We show that the final compression is within a factor c of the information-theoretic lower bound, where c is independent of k, and depends only on the dimension of the space. In Section 4, we give experimental justification for our k-local model.

2 Data Framework

In this section we present a formal model of the essential features of the sensor networks to which our results will apply. Our main goal is that it realistically model the data sets arising in typical wireless sensor-networks when observing kinetic data while also allowing for a clean theoretical analysis. We assume a fixed set of S sensors operating over a total time period of length T. The sensors are modeled as points in some metric space. We may think of the space as $\mathbb{R}^d$ for some fixed d, but our results apply in any metric space of bounded doubling dimension [32]. We model the objects of our system as points moving continuously in this space, and we make no assumptions *a priori* about the nature of this motion. Each sensor observes some *region* surrounding it. Our framework makes no assumptions about the size, shape, or density of these regions. The sensor regions need not be disjoint, nor do they need to cover all the moving points at any given time.

Each sensor continually collects statistical information about the points lying within its region, and it outputs this information at synchronized time steps. As mentioned above, we assume throughout that this information is simply an *occupancy count* of the number of points that lie within the region. (The assumption of synchronization is mostly for the sake of convenience of notation. As we shall see, our compression algorithm operates jointly on local groups of a fixed size, and hence it is required only that the sensors of each group behave synchronously.)

As mentioned in the introduction, our framework is based on an information-theoretic approach. Let us begin with a few basic definitions (see, e.g., [33]). We assume that the sensor outputs can be modeled by a stationary, ergodic random process. Since the streams are synchronized and the cardinality of the moving point set is finite, we can think of the S sensor streams as a collection of S

strings, each of length T, over a finite alphabet. Letting lg denote the logarithm base-2, the *entropy* of a discrete random variable X, denoted $H(X)$, is defined to be $-\sum_x p_x \lg p_x$, where the sum is over the possible values x of X, and p_x is the probability of x.

We generalize entropy to random processes as follows. Given a stationary, ergodic random process X, consider the limit of the entropy of arbitrarily long sequences of X, normalized by the sequence length. This leads to the notion of *normalized entropy*, which is defined to be $H(X) = \lim_{T\to\infty} -\frac{1}{T}\sum_{x,|x|=T} p_x \lg p_x$, where the sum is over sequences x of length T, and p_x denotes the probability of this sequence. Normalized entropy considers not only the distribution of individual characters, but the tendencies for certain patterns of characters to repeat.

We also generalize the entropy to collections of random variables. Given a sequence $\boldsymbol{X} = \langle X_1, X_2, \ldots, X_S\rangle$ of (possibly statistically correlated) random variables, the *joint entropy* is defined to be $H(\boldsymbol{X}) = -\sum_{\boldsymbol{x}} p_{\boldsymbol{x}} \lg p_{\boldsymbol{x}}$, where the sum is taken over all S-tuples $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_S\rangle$ of possible values, and $p_{\boldsymbol{x}}$ is the probability of this joint outcome [33]. The generalization to *normalized joint entropy* is straightforward and further strengthens normalized entropy by considering correlations and statistical dependencies between the various streams.

In this paper we are interested in the lossless compression of the joint sensor stream. Shannon's source coding theorem states that in the limit, as the length of a stream of independent, identically distributed (i.i.d.) random variables goes to infinity, the minimum number of required bits to allow lossless compression of each character of the stream is equal to the entropy of the stream [31]. In our case, Shannon's theorem implies that the optimum bit rate of a lossless encoding of the joint sensor system cannot be less than the normalized joint entropy of the system. Thus, the normalized joint entropy is the gold standard for the asymptotic efficiency of any compression method. Henceforth, all references to "joint entropy" and "entropy" should be understood to mean the normalized versions of each.

As mentioned above, joint compression of all the sensor streams is not feasible. Our approach will be to assume a limit on statistical dependencies among the observed sensor outputs based on geometric locality. It is reasonable to expect that the outputs of nearby sensors will exhibit a higher degree of statistical dependence with each other than more distant ones. Although statistical dependence would be expected to decrease gradually with increasing distance, in order to keep our model as simple and clean as possible, we will assume that beyond some threshold, the statistical dependence between sensors is so small that it may be treated as zero. There are a number of natural ways to define such a threshold distance. One is an *absolute approach*, which is given a threshold distance parameter r, and in which it is assumed that any two sensors that lie at distance greater than r from each other have statistically independent output streams. The second is a *relative approach* in which an integer k is provided, and it is assumed that two sensor output streams are statistically dependent only if each is among the k nearest sensors of the other. In this paper we will take the latter approach, which we will justify after introducing some definitions.

Formally, let $P = \{p_1, p_2, \ldots, p_S\}$ denote the sensor positions. Given some integer parameter k, we assume that each sensor's output can be statistically dependent on only its k nearest sensors. Since statistical dependence is a symmetric relation, two sensors can exhibit dependence only if each is among the k nearest neighbors of the other. More precisely, let $NN_k(i)$ denote the set of k closest sensors to p_i (not including sensor i itself). We say that two sensors i and j are *mutually k-close* if $p_i \in NN_k(j)$ and $p_j \in NN_k(i)$. A system of sensors is said to be *k-local* if for any two sensors that are not mutually k-close, their observations are statistically independent. (Thus, 0-locality means that the sensor observations are mutually independent.) Let $\boldsymbol{X} = \langle X_1, X_2, \ldots, X_S \rangle$ be a system of random streams associated with by S sensors, and let $H(\boldsymbol{X})$ denote its joint entropy. Given two random processes X and Y, define the *conditional entropy* of X given Y to be $H(X \mid Y) = -\sum_{x \in X, y \in Y} p(x,y) \log p(y \mid x)$. Note that $H(X \mid Y) \leq H(X)$, and if X and Y are statistically independent, then $H(X \mid Y) = H(X)$. By the chain rule for conditional entropy [33], we have $H(\boldsymbol{X}) = H(X_1) + H(X_2 \mid X_1) + \ldots + H(X_S \mid X_1, \ldots, X_{S-1})$. Letting $D_i(k) = \{j : 1 \leq j < i$ and x_i and x_j are mutually k-close$\}$ we define the *k-local entropy*, denoted $H_k(\boldsymbol{X})$, to be $\sum_{i=1}^{S} H(X_i \mid D_i(k))$. Note that $H(\boldsymbol{X}) \leq H_k(\boldsymbol{X})$ and equality holds when $k = S$. By definition of k-locality, $H(X_i \mid X_1, X_2, \ldots, X_{i-1}) = H(X_i \mid D_k(i))$. By applying the chain rule for joint entropy, we have the following easy consequence, which states that, under our locality assumption, k-local entropy is the same as the joint entropy of the entire system.

Lemma 1. *Given a k-local system with set of observations $\boldsymbol{X}$, $H(\boldsymbol{X}) = H_k(\boldsymbol{X})$.*

We show in the full version of this paper that if KDS is used to observe a system in which the sensor regions are modeled as a sparse collection of unit disks and objects change their trajectories relatively frequently, KDS requires on the order of $H_k(\boldsymbol{X})$ bits of storage [34]. Thus, since KDS has full knowledge of the system, $H_k(\boldsymbol{X})$ is a reasonable measure of optimality.

One advantage of our relative characterization of mutually dependent sensor outputs is that it naturally adapts to the distribution of sensors. It is not dependent on messy metric quantities, such as the absolute distances between sensors or the degree of overlap between sensed regions. Another reason arises by observing that, in an absolute model, all the sensors might lie within distance r of each other. This would imply that all the sensors could be mutually statistically dependent on each other, which would render optimal compression based on joint entropy intractable. Nonetheless, by imposing a relatively weak density assumption, our model can be applied in such contexts. For example, consider a setting in which each sensor monitors a region of radius r. Given two positive parameters α and β, suppose that we were to assume that the number of sensors whose centers lie within any ball of radius r is at most α, and (instead of our k-local assumption) we were to assume that the outputs of any two sensors can be statistically dependent only if they are within distance βr of each other. Then, by a simple packing argument, it follows that such a system is k-local for $k = O(\alpha\, \beta^{O(1)})$ in any space of constant doubling dimension. Thus, our model would be applicable in this context.

3 Compression Results

Before presenting the main result of this section, we present a lemma which is combinatorially interesting in its own right. This partitioning lemma combined with a compression algorithm allows us to compress the motion of points as recorded by sensors to an encoding size which is c times the optimal, where c is an integral constant to be specified in the proof of Lemma 2.

3.1 Partitioning Lemma

First, we present some definitions about properties of the static point set representing sensor locations. Let $r_k(p)$ be the distance from some sensor at location p to its k^{th} nearest neighbor. Recall that points are mutually k-close if they are in each other's k nearest neighbors. We say that a point set $P \in \mathbb{R}^d$ is *k-clusterable* if it can be partitioned into subsets $C_{i1}, C_{i2}, \ldots$ such that $|C_{ij}| \leq k+1$ and if p and q are mutually k-close then p and q are in the same subset of the partition. Intuitively, this means that naturally defined clusters in the set are separated enough so that points within the same cluster are closer to each other than they are to points outside of the cluster. The following lemma holds for all metrics with constant *doubling dimension*, where these metrics are defined to limit to a constant the number of balls that cover a ball with twice their radius [32]. Euclidean spaces are of constant doubling dimension.

partition(point set P, k)

```
for all p ∈ P
    determine NN_k(p) and r_k(p)
i = 1
while P ≠ ∅
    unmarked(P) = P
    P_i = ∅
    while unmarked(P) ≠ ∅
        r = min_{p∈unmarked(P)} r_k(p)
        p' = p ∈ P : r = r_k(p)
        P_i = P_i ∪ {p ∈ P : ‖pp'‖ ≤ r}
        P = P \ {p ∈ P : ‖pp'‖ ≤ r}
        unmarked(P) = unmarked(P) \
            {p ∈ unmarked(P) : ‖pp'‖ ≤ 3r}
    increment i
return {P_1, P_2, ..., P_c}
```

Fig. 1. The partitioning algorithm that implements Lemma 2.

Lemma 2. *In any doubling space there exists an integral constant c such that for all integral $k > 0$ given any set P in the doubling space, P can be partitioned into $P_1, P_2, \ldots, P_c$ such that for $1 \leq i \leq c$, P_i is* k-clusterable.

The partitioning algorithm that implements Lemma 2 is shown in Figure 1. It proceeds by iteratively finding the unmarked point p with minimum $r = r_k(p)$, moving all points within r, henceforth called a *cluster*, to the current partition, and marking all points within $3r$ of p. A new partition is created whenever all remaining points have been marked. The marked points are used to create a buffer zone which separates clusters so that all points are closer to points within their cluster than they are to any other points in the partition. The algorithm's inner loop creates these clusters, and the outer loop creates the c partitions.

Proof (Sketch). (See the full version of this paper for a detailed proof [34].)

By the construction of the marking process, each partition is k-clusterable. We will show that at most c partitions P_i are created by the partitioning algorithm of Figure 1. We refer to each iteration of the outer while loop as a *round*. First note that at the end of the first round all points are either marked or removed from P. Each point that remains after the first round was marked by some point during the first round. By a packing argument based on the minimum nearest neighbor radius and the radius of the marked region around a point we show that points can be marked by at most $c = O(1 + 12^{O(1)}) = O(1)$ rounds, creating c partitions.

Note that a cluster centered at p' with less than $k+1$ points does not violate the k-clusterable property since this cluster would have been created by clustering $NN_k(p')$ together as originally identified before any points were partitioned. Such a cluster is formed because some of the original points in $NN_k(p')$ were previously added to a different partition. Since being mutual k-close is based on the entire set, smaller clusters are still mutually k-close within that partition.

3.2 Compression Theorem

We now present the main compression algorithm and analysis. The algorithm, presented in Figure 2, compresses each cluster formed by the partitioning algorithm (Figure 1) separately and returns the union of these. Each cluster is compressed by creating a new stream in which the t^{th} character is a new character which is the concatenation of the t^{th} character of every stream in that cluster. This new stream is then compressed using an entropy-based compression algorithm which achieves the optimal encoding length in the limit. For example, the Lempel-Ziv sliding-window compression algorithm could be used [7]. We reason about the size of the resulting stream set encoding.

compress (stream set X, sensor set P, k)

$\{P_1, P_2, \ldots, P_c\}$ = partition (P, k)
for $i = 1$ to c
  for all clusters j in P_i
    containing streams X_{ij1} through $X_{ijh_{ij}}$
    $\widehat{X}_{ij} = \bigcup_{t=1}^{T} X_{ij1t} \& X_{ij2t} \& \ldots \& X_{ijh_{ij}t}$
    where X_{ijht} is the t^{th} character of X_{ijh}
return $\bigcup_{ij}$ *entropy_compress*$(\widehat{X}_{ij})$

Fig. 2. The compression algorithm, which takes a set $\mathbf{X}$ of streams of length T and the associated set P of sensors which recorded them and returns a compressed encoding of length $c \cdot H(\mathbf{X})$. The partitioning algorithm of Figure 1 is called and determines the constant c. *entropy_compress* is an entropy-based compression algorithm that returns an encoded stream.

First, we introduce some notation. Let $\mathbf{X}$ be the set of streams containing the information recorded by the sensors of set P where $|\mathbf{X}| = |P|$. Given the set of partitions $\{P_i\}$ resulting from the partitioning lemma in Section 3.1, $\{X_i\}$ is the set of associated streams. Let $\{C_{ij}\}$ be the set of clusters that are created by the partitioning algorithm, we call $\{X_{ij}\}$ the set of streams in cluster C_{ij} and X_{ijh} is the h^{th} stream in cluster C_{ij} with cardinality h_{ij}.

Theorem 1. *A stream set which represents observations from a k-local sensor system can be compressed to an encoded string which has length at most c times the optimal, where c is a constant depending on the doubling dimension of the underlying point set.*

Proof. First, we show that each cluster C_{ij} is compressed to a string whose length is equal to the joint entropy of the component streams of that cluster. Each cluster consists of streams $\{X_{ij}\}$ which are merged into one new stream by concatenating the t^{th} character of all the streams to create the t^{th} character of the new stream. This new stream, $\widehat{X}_{ij}$, is then compressed using an optimal compression algorithm. By construction of the streams $\widehat{X}_{ij}$, the entropy $H(\widehat{X}_{ij})$ of a single stream is equal to the joint entropy of its component streams $H(X_{ij1}, X_{ij2}, \ldots, X_{ijh_{ij}})$. The entropy-based encoding algorithm compresses each $\widehat{X}_{ij}$ to an encoded string the length of the stream's entropy and that compression is optimal [35], so $H(X_{ij1}, X_{ij2}, \ldots, X_{ijh_{ij}})$ is the optimal encoding length for cluster C_{ij}.

Our local dependence assumptions, explained in Section 2, say that the stream of data from a sensor is only dependent on the streams of its k nearest neighbors. Additionally, recall that in Section 2 we defined being mutually k-close to require that streams are only dependent if they come from sensors who are in each other's k nearest neighbor sets. By the partitioning lemma from Section 3.1, we know that each cluster C_{ij} is independent of all other clusters in partition P_i. From standard information theoretic results [33] we know that for a collection of streams $Y_1, \ldots, Y_S$, $H(Y_1, Y_2, \ldots, Y_S) = \sum_{i=1}^{S} H(Y_i)$ if and only if the Y_i are independent. Since the elements of $\{\{X_{i1}\}, \{X_{i2}\}, \ldots, \{X_{i|\{C_{ij}\}|}\}\}$ are independent, $H(X_i) = \sum_j H(\{X_{ij}\})$. Combining this with the fact that $H(\widehat{X}_{ij})$ is equal to the joint entropy of its component streams, we have that $H(X_i) = \sum_j H(\widehat{X}_{ij})$. $H(X_i)$ is the optimal compression bound for partition P_i, so we achieve the optimal compression for each partition.

Finally, we show that our compression algorithm is a c-approximation of the optimal. We say that a compression algorithm provides a γ-*approximation* if the length of the compressed streams is no more than γ times the optimal length. Recall that c partitions are generated by the partitioning algorithm from Section 3.1. Each of these partitions is encoded by a string of length $H(X_i)$ in the limit, so the total encoding size is $\sum_{i=1}^{c} H(X_i) \leq c \cdot \max_i H(X_i) \leq c \cdot H(\mathbf{X})$, where $H(\mathbf{X})$ is the joint entropy, which is a lower bound on the optimal encoding size, and the last inequality follows since $|\mathbf{X}| \geq |X_i|$ for all i. So our algorithm provides a c-approximation of the optimal compression.

Note that using the same method we used to compress the members of individual clusters, we could have combined the characters of all streams and compressed these together. This method would have optimal compression to the joint entropy of the streams. For demonstration of the problem with this method, consider the Lempel-Ziv sliding-window algorithm [7]. The algorithm proceeds by looking for matches between the current time position and some previous time

within a given window into the past. The length and position of these matches are then recorded, which saves the space of encoding each character. The window moves forward as time progresses. Larger window sizes yield better results since matches are more likely to be found. The optimal encoded length is achieved by taking the limit as the window size tends to infinity [35]. If all streams are compressed at once, the optimal compression rate is only achieved in the limit as the window size becomes large and in practice compressing all streams at once requires a much larger window before the compression benefits begin. By only compressing k streams together we limit the effect of this problem.

4 Locality Results

In order to justify our claim that sensor outputs exhibit higher statistical dependence on their nearest neighbors, we analyze experimental data recorded by sensors operating under assumptions similar to our framework. The data we analyze was collected at the Mitsubishi Electric Research Laboratory [36]. It consists of sensor activation times for over 200 sensors observing the hallways of a building. Each sensor records times of nearby motion in coordinated universal time. For our analysis, we group activations into *time steps* consisting of the count of all activations for a single sensor over 0.1 second. These serve as the sensor counts over which we find the normalized joint entropy of data for sensor pairs, and we consider these counts only in terms of the presence or absence of motion during a given time step. We consider one minute of this data, or 600 data points.

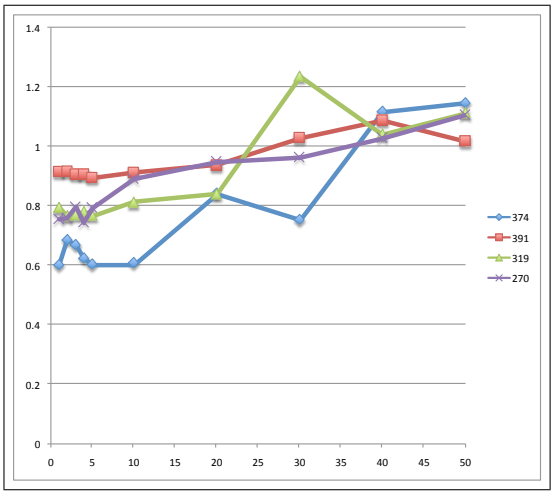

Fig. 3. Plotted joint entropy values for values of k. These are shown for $k = 1$ to $k = 5$ at increments of 1 and $k = 10$ to $k = 50$ at increments of 10.

Recall that the normalized joint entropy of two sequences generated by a common process is defined in Section 2. For our experiment, we consider the value $T = 3$. Probabilities are determined based on the observed outputs of the two sensors whose pairwise joint entropy is being calculated over the sensor streams containing 600 activation status values. The results shown in Figure 3 plot the combinatorial neighbor distances for four sensors against the normalized joint entropy values found. These neighbor distances are calculated based on the sensor locations and do not take walls into account, so some seemingly close sensors turn out not to be statistically dependent on each other. While

each sensor's plot starts at a different initial value, there are few low entropy values (relative to the start value) after $k = 10$, showing that as sensors become farther apart they are less likely to be statistically dependent on each other.

In order to justify our claim on the value of compressing sensor outputs, and further, jointly compressing neighboring sensor outputs, we consider eight sensor outputs from a single hallway. The activation status was considered for these sensors for 70,000 0.1 second intervals (or approximately 2 hours). The raw data used 286.7 MB. These eight streams compressed separately with `gzip` (which uses the sliding-window Lempel-Ziv algorithm) used a total of 15.5 MB or 5.4% of the original space. Compressing the eight streams merged together character by character (as described in the compression algorithm in Figure 2), used 7.1 MB, or an additional 45.7% of the separately compressed space.

References

1. Saunier, N., Sayed, T.: Automated analysis of road safety with video data. In: Transportation Research Record. (2007) 57–64
2. Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., Anderson, J.: Wireless sensor networks for habitat monitoring. In: ACM international workshop on wireless sensor networks and applications. (2002) 88–97
3. MIT Media Lab: The owl project. `http://owlproject.media.mit.edu/`
4. Stutchbury, B.J.M., Tarof, S.A., Done, T., Gow, E., Kramer, P.M., Tautin, J., Fox, J.W., Afanasyev, V.: Tracking long-distance songbird migration by using geolocators. Science (February 2009) 896
5. Huffman, D.A.: A method for the construction of minimum-redundancy codes. Proc. of the IRE **40** (Sept. 1952)
6. Rissanen, J.: Generalized Kraft inequality and arithmetic coding. IBM Jour. of Research and Dev. **20** (1976)
7. Ziv, J., Lempel, A.: A universal algorithm for sequential data compression. IEEE Transactions on Information Theory **IT-23**(3) (May 1977)
8. Deligiannakis, A., Kotidis, Y., Roussopoulos, N.: Processing approximate aggregate queries in wireless sensor networks. Inf. Syst. **31**(8) (2006) 770–792
9. Gandhi, S., Nath, S., Suri, S., Liu, J.: Gamps: Compressing multi sensor data by grouping and amplitude scaling. In: ACM SIGMOD. (2009)
10. Cormode, G., Muthukrishnan, S., Zhuang, W.: Conquering the divide: Continuous clustering of distributed data streams. In: IEEE 23rd International Conference on Data Engineering. (2007) 1036–1045
11. Cormode, G., Muthukrishnan, S., Yi, K.: Algorithms for distributed functional monitoring. In: SODA. (2008) 1076–1085
12. Soroush, E., Wu, K., Pei, J.: Fast and quality-guaranteed data streaming in resource-constrained sensor networks. In: ACM Symp. on Mobile ad hoc networking and computing. (2008) 391–400
13. Johnen, C., Nguyen, L.H.: Self-stabilizing weight-based clustering algorithm for ad hoc sensor networks. Workshop on Algorithmic Aspects of Wireless Sensor Networks (AlgoSensors) (2006) 83–94
14. Nikoletseas, S., Spirakis, P.G.: Efficient sensor network design for continuous monitoring of moving objects. Theoretical Computer Science **402**(1) (2008) 56–66
15. Deligiannakis, A., Kotidis, Y., Roussopoulos, N.: Dissemination of compressed historical information in sensor networks. VLDB Journal **16**(4) (2007) 439–461

16. Sadler, C.M., Martonosi, M.: Data compression algorithms for energy-constrained devices in delay tolerant networks. In: SENSYS. (November 2006)
17. Guibas, L.J.: Sensing, tracking and reasoning with relations. IEEE Signal Processing Mag. **19**(2) (Mar 2002)
18. Guitton, A., Trigoni, N., Helmer, S.: Fault-tolerant compression algorithms for sensor networks with unreliable links. Technical Report BBKCS-08-01, Birkbeck, University of London (2008)
19. Gupta, P., Janardan, R., Smid, M.: Fast algorithms for collision and proximity problems involving moving geometric objects. In: Comput. Geom. Theory Appl. Volume 6. (1996) 371–391
20. Atallah, M.J.: Some dynamic computational geometry poblems. In: Comput. Math. Appl. Volume 11(12). (1985) 1171–1181
21. Schomer, E., Theil, C.: Efficient collision detection for moving polyhedra. In: Proc. 11th Annu. ACM Sympos. Comput. Geom. (1995) 51–60
22. Schomer, E., Theil, C.: Subquadratic algorithms for the general collision detection problem. In: European Workshop Comput. Geom. (1996) 95–101
23. Basch, J., Guibas, L.J., Hershberger, J.: Data structures for mobile data. In: SODA. (1997)
24. Kahan, S.: A model for data in motion. In: STOC '91: Proc. of the 23rd ACM Symp. on Theory of Computing. (1991) 265–277
25. Agarwal, P.K., Guibas, L.J., Edelsbrunner, H., Erickson, J., Isard, M., Har-Peled, S., Hershberger, J., Jensen, C., Kavraki, L., Koehl, P., Lin, M., Manocha, D., Metaxas, D., Mirtich, B., Mount, D.M., Muthukrishnan, S., Pai, D., Sacks, E., Snoeyink, J., Suri, S., Wolefson, O.: Algorithmic issues in modeling motion. ACM Computing Surveys **34** (December 2002) 550–572
26. Guibas, L.: Kinetic data structures. In Mehta, D., Sahni, S., eds.: Handbook of Data Structures and App. Chapman and Hall/CRC (2004) 23–1–23–18
27. Babcock, B., Olston, C.: Distributed top-k monitoring. In: SIGMOD. (2003) 28–39
28. Yi, K., Zhang, Q.: Multi-dimensional online tracking. In: SODA. (2009)
29. Mount, D.M., Netanyahu, N.S., Piatko, C., Silverman, R., Wu, A.Y.: A computational framework for incremental motion. In: Proc. 20th Annu. ACM Sympos. Comput. Geom. (2004) 200–209
30. Gandhi, S., Kumar, R., Suri, S.: Target counting under minimal sensing: Complexity and approximations. Workshop on Algorithmic Aspects of Wireless Sensor Networks (AlgoSensors) (2008) 30–42
31. Shannon, C.E.: A mathematical theory of communication. The Bell System Technical Journal **27** (July, October 1948) 379–423, 623–656
32. Krauthgamer, R., Lee, J.R.: Navigating nets: Simple algorithms for proximity search. In: SODA. (2004)
33. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Second edn. Wiley-IEEE (2006)
34. Friedler, S.A., Mount, D.M.: Compressing kinetic data from sensor networks. Technical Report CS-TR-4941, UMIACS-TR-2009-10, University of Maryland, College Park (2009)
35. Wyner, A.D., Ziv, J.: The sliding-window lempel-ziv algorithm is asymptotically optimal. In: Proceedings of the IEEE. (Jun 1994) 872–877
36. Wren, C.R., Ivanov, Y.A., Leigh, D., Westbues, J.: The MERL motion detector dataset: 2007 workshop on massive datasets. Technical Report TR2007-069, Mitsubishi Electric Research Laboratories, Cambridge, MA, USA (August 2007)