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. In order to justify our locality assumptions, we provide a theoretical comparison with a variant of the kinetic data structures framework. We prove that the storage size required by an optimal system operating under our locality assumptions is on the order of the size required by our variant. Additionally, we provide experimental justification for our locality assumptions.

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 [29], environment surveillance [22], and wildlife tracking [23, 34]. 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 [29] that videotape congestion produce many hours of video 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 how to compress 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 loss through approximation). Because 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 (such as Huffman coding [18], arithmetic coding [27], Lempel-Ziv [38]) rely on the statistical redundancy present in the data stream in order to achieve high compression rates. In the context of sensor networks, this redundancy arises naturally due to correlations in the

* A shorter version of this work appeared in the *Proceedings of the 5th International Workshop on Algorithmic Aspects of Wireless Sensor Networks (AlgoSensors)*, 2009.

** 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

outputs of sensors that are spatially close to each other. As with existing methods for lossy compression [9,13], 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 [6,7,33]. We focus on doing lossless compression on 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 [19]. Compressing the data before transmitting additionally improves the communication efficiency of the network.

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. The data sets generated by sensor networks have a number of spatial, temporal, and statistical properties that render them interesting for study. We assume that we do not get to choose the sensor deployment based on object motion (as done in [26]), 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. For example, many of the vehicles driving by one traffic camera are likely to be observed by nearby cameras, perhaps a short time later or earlier. If we assume that the data can be modeled by a random process, it is reasonable to expect that a high degree of statistical dependence exists between the data streams generated by nearby sensors. If so, the information content of the assembled data will be significantly smaller than the size of the raw data. In other words, the raw sensor streams, when considered in aggregate, will contain a great deal of redundancy. 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 [9, 10, 15, 16, 28]. 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 [3, 5, 17, 20, 30, 31]. A survey of practical and theoretical aspects of modeling motion can be found in [2]. 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 [5]. The basic entities in this framework are points in motion, where the motion is expressed as piecewise algebraic flight plans. Geometric structures are maintained through a set of boolean conditions, called certificates, and a set of associated update rules. The efficiency of algorithms in this model is a function of the number of certificates involved and the efficiency of processing them. In a sensor context, moving data has been considered in relation to sensor placement based on possible object trajectories modeled by a set of 3D curves over space and time [26].

As valuable as KDS has been for developing theoretical analyses of point motion (see [14] for a survey), it is unsuitable for many real-world contexts and for theoretical problems that do not have locally determined properties. The requirements of algebraic point motion and advance knowledge of flight plans are either inapplicable or infeasible in many scientific applications. Agarwal *et al.* [2] identify fundamental directions that future research should pursue. Our work addresses four of these issues; unpredicted motion, motion-sensitivity, robustness, and theoretical discrete models of motion. In our framework we will process a point set without predicted knowledge and no matter its motion. *Motion-sensitive algorithms* admit complexity analyses based on the underlying motion. Imagine a set of points following a straight line or moving continuously in a circle; any algorithm calculating statistical information about such a point set should be more efficient than the same algorithm operating on a set of randomly moving points. Our motion-sensitive framework will pay a cost in efficiency based on the information content of the point motion. *Robustness* is a quality of statistical estimators that allow outliers. Unlike KDS, we will ignore point identities in favor of statistical properties;

KDS focuses on properties in relation to individual points. In the KDS model, a rearrangement of points which maintained a global statistical property could trigger many certificate failures despite the maintenance of the statistical property being calculated. For example, two points which exactly switch position do not change the diameter of the point set, but may cause multiple certificate failures. Through anonymization of the points and discrete time sampling, our framework reduces the overhead in these instances. Finally, Agarwal *et al.* note that most theoretical work relies on continuous frameworks while applied work experimentally evaluates methods based on discrete models. Our framework uses a discrete sampling model, but is still theoretically sound. In addition, the underlying goal which structures KDS operations is maintenance of information into the future; we will process sensed data after the fact, e.g., for efficient retrieval. For these problem types, our framework serves as an alternative to the KDS model.

There has also been study of algorithms that involve the distributed online processing of sensor-network data. One example is the *continuous distributed model* described by Cormode *et al.* [7]. This model contains a set of sensors, which each observe a stream of data describing the recent changes in local observations. Each sensor may communicate with any other sensor or with a designated central coordinator. Efficiency is typically expressed as a trade-off between communication complexity and accuracy. This framework has been successfully applied to the maintenance of a number of statistics online [4, 6, 7]. Another recent example is the competitive online tracking algorithm of Yi and Zhang [37], in which a tracker-observer pair coordinate to monitor the motion of a moving point. Again, complexity is measured by the amount of communication between the tracker and the observer. The idea of the tracker and observer is reminiscent of an earlier model for incremental motion by Mount *et al.* [25]. 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.* [12] and do not rely on a sensor's ability to accurately record distance, angle, etc.

As mentioned above, our objective is to compress this data in a lossless manner by exploiting redundancy in the sensor streams. In order to establish formal bounds on the quality of this compression, we assume (as is common in entropy encoding) that the output of each sensor can be modeled as a stationary, ergodic random process. We allow for statistical dependencies between the sensor streams. Shannon's source coding theorem implies that, in the limit, the minimum number of bits needed to encode the data is bounded from below by the normalized joint entropy of the resulting system of random processes. There are known lossless compression algorithms, such as Lempel-Ziv [38], that achieve this lower bound asymptotically. It would be utterly infeasible, however, 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. The problem in our context is how to bound 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.

The full contributions of this paper are described in the following sections. In Section 2, we introduce a new framework for the compression and analysis of kinetic sensor data. In Section 3, we prove that any k-local system that resides in a space of fixed dimension can be nicely partitioned in the manner described above, 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 justify our k-local model theoretically as compared to a variant of the KDS model. We prove that the compressed data from our model takes space on the order of the space used by the KDS variant. In Section 5, we give experimental justification showing that the assumptions of the k-local model are borne out by real-world data.

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 [21]. 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. In general, our framework makes no assumptions about the size, shape, or density of these regions, but additional assumptions may be imposed later in special cases. 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., [8]). 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 can 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 over time.

We can also generalize the concept of 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 [8]. The generalization to *normalized joint entropy* is straightforward. Normalized joint entropy 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 [32]. 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. One reason is that it adapts to the local density of sensors. Another reason arises by observing that, in the absolute model, all the sensors might lie within distance r of each other. This means that all the sensors could be mutually statistically dependent, which would render optimal compression intractable. On the other hand, if we deal with this by imposing the density restriction that no sensor has more than some number, say k, sensors within distance r, then the absolute approach reduces to a special case of the relative approach.

This restriction allows reasoning about sensor outputs in subsets. Previous restrictions of this form include the Lovász Local Lemma [11] which also assumes dependence on at most k events. Particle simulations (often used to simulate physical objects for animation) based on smoothed particle hydrodynamics have also used similar locality restrictions to determine which neighboring particles impact each other. These calculations are made over densely sampled particles and are based on a kernel function which determines the impact of one particle on another. This frequently amounts to a cut-off distance after which we assume that the particles are too far away to impact each other [1]. For a survey on smoothed particle hydrodynamics see [24].

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 [8], we have

$$H(\boldsymbol{X}) \;=\; H(X_1) + H(X_2 \mid X_1) + \ldots + H(X_i \mid X_1, \ldots, X_{i-1}) + \ldots + H(X_S \mid X_1, \ldots, X_{S-1}).$$

Letting

$$D_i(k) \;=\; \{X_j : 1 \leq j < i \text{ and sensors } i \text{ and } j \text{ are mutually } k\text{-close}\}$$

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 sensor system with set of observations $\boldsymbol{X}$, $H(\boldsymbol{X}) = H_k(\boldsymbol{X})$.*

The assumption of statistical independence is rather strong, since two distant sensor streams may be dependent simply because they exhibit a dependence with a common external event, such as the weather or time of day. Presumably, such dependencies would be shared by all sensors, and certainly by the k nearest neighbors. The important aspect of independence is encapsulated in the above lemma, since it indicates that, from the perspective of joint entropy, the k nearest neighbors explain essentially all the dependence with the rest of the system. Although we assume perfect statistical independence beyond the range of the kth nearest neighbor, the practical impact of this assumption is that any dependencies that may exist beyond this range have a negligible impact on the joint entropy of the system, and hence a negligible impact on the degree of compressibility in the system.

One advantage of our 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. Note, however, that our model can be applied in contexts where absolute distances are meaningful. For example, consider a setting in which each sensor monitors a region of radius r. Given two positive parameters α and β, we assume that the number of sensors whose centers lie within any ball of radius r is at most α, and 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.

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 [21]. Euclidean spaces are of constant doubling dimension.

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 which 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. Each partition is k-clusterable since (by the marking process) for any cluster with diameter $2r$ there are no points of this partition which are not members of that cluster which are within distance $2r$ of a member of the cluster. So each cluster consists of at most $k+1$ points, each of whose k nearest neighbors are within the cluster, i.e., are mutually k-close.

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. Consider some point p which is marked by the first round. Let p' be the point that marked p in round one, and let $r = r_k(p')$ be the radius of $NN_k(p')$. (Note that for any $p \in P$, $r_k(p)$ is defined based on the original point set.) The marked point p is within distance $3r$ of p'. Since there are k points within distance r of p', there are at least k points within distance $4r$ of p, so $NN_k(p)$ cannot have a radius larger than $4r$. Let M be the set of points that mark p in any round i. Since nearest neighbor balls are chosen in increasing order of radius, no point q with $r_k(q)$ greater than $4r$ can be in M, since p would have been chosen first. So all points in M are within distance $3 \cdot 4r = 12r$ of p.

We now show that points in M are r-sparse, i.e., are separated by at least distance r. Since radii are chosen in increasing order and $r = r_k(p')$ has already been chosen, any point $q' \in M$ must have $r_k(q') > r$. In addition, since all points in $NN_k(q')$ are removed, for all $q'' \in M$, $\|q'q''\| > r$. Given a circle of radius R in a doubling

partition(point set P, k)

```
for all p ∈ P                                  // Determine the k nearest neighbors and the radius
    determine NN_k(p) and r_k(p)               // of the k nearest neighbors ball based on the original
i = 1                                          // point set. These values do not change.
while P ≠ ∅                                    // While unpartitioned points remain
    unmarked(P) = P                            //     unmark all remaining points.
    P_i = ∅                                    //     Create a new, empty partition.
    while unmarked(P) ≠ ∅                      //     While unmarked points remain
        r = min_{p∈unmarked(P)} r_k(p)         //         find the point p with the minimum radius (r)
        p' = p ∈ P : r = r_k(p)                //         nearest neighbor ball and add that point and
        P_i = P_i ∪ {p ∈ P : ||pp'|| ≤ r}      //         all points within r to the new partition.
        P = P \ {p ∈ P : ||pp'|| ≤ r}          //         Remove these points from P and mark
        unmarked(P) = unmarked(P) \ {p ∈ unmarked(P) : ||pp'|| ≤ 3r}
    increment i                                //         points within 3r of p.
return {P_1, P_2, ..., P_c}                    // Return the resulting partitions.
```

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

metric, it follows from a standard packing argument that any δ-sparse set that lies entirely within a ball of radius R has cardinality $O(1+(R/\delta)^{O(1)})$. Taking $R = 12r$ and $\delta = r$, we have that $|M| \leq O(1+12^{O(1)}) = O(1)$. For points in $\mathbb{R}^d$ this constant is $1 + 12^d$. Letting c denote this quantity, we see that no point can be marked more than c times, and hence the process terminates after at most c rounds, producing at most 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. A cluster without $k+1$ points 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 3, 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 [38]. We reason about the size of the resulting stream set encoding.

First, we introduce some notation. Let X be the set of streams containing the information recorded by the sensors of set P where $|X| = |P|$. 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 set of streams which represent 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

Figure accompanying the proof of Lemma 2

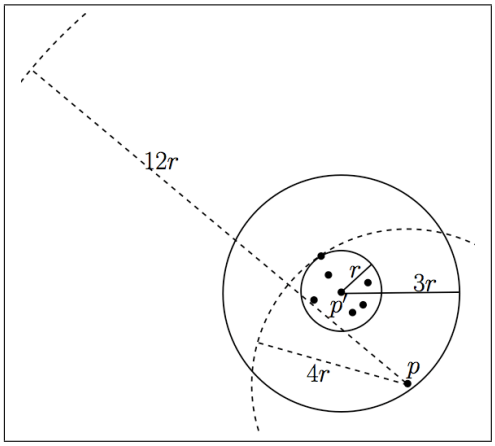

Fig. 2. Proof illustration for Lemma 2 for $k = 6$. Solid circles are centered at point p'. Solid lines show the radii of these circles. Dashed arcs are partial circles centered at point p. Dashed lines show the radii of these circles.

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. 3. The compression algorithm which takes a set X of streams of length T and the associated set P of sensors which recorded them and returns a compressed encoding of length $c \cdot H$, where H is the joint entropy of the streams. The partitioning algorithm shown in Figure 1 is called and determines the constant c and represents the concatenation of characters to create a larger character. *entropy_compress* is an entropy-based compression algorithm which achieves the optimal encoding length in the limit and returns an encoded stream.

the length of the stream's entropy and that compression is optimal [36], 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 [8] 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 $\sum_{i=1}^{c} H(X_i) \leq c \cdot \max_i H(X_i) \leq c \cdot H(X)$, where $H(X)$ is the joint entropy, which is a lower bound on the optimal encoding size, and the last inequality follows since $|X| \geq |X_i|$ 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 [38]. 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 [36]. 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 Efficiency with Respect to Short-Haul KDS

We believe that k-local entropy is a reasonable measure of the complexity of geometric motion. It might seem at first that any system that is based on monitoring the motion of a large number of moving objects by the incremental counts of a large number of sensors would produce such a huge volume of data that it would be utterly impractical as a basis for computation. Indeed, this is why compression is such an important ingredient in our framework. But, is it reasonable to assume that lossless compression can achieve the desired degree of data reduction needed to make this scheme competitive with purely prescriptive methods such as KDS? In this section, we consider a simple comparison, which suggests that lossless compression can achieve nearly the same bit rates as KDS would need to describe the motion of moving objects.

This may seem like comparing "apples and oranges," since KDS assumes precise knowledge of the future motion of objects through the use of flight plans. In contrast, our framework has no precise knowledge of individual point motions (only the occupancy counts of sensor regions) and must have the flexibility to cope with whatever motion is presented to it. Our analysis will exploit the fact that, if the motion of each point can be prescribed, then the resulting system must have relatively low entropy. To make the comparison fair, we will need to impose some constraints on the nature of the point motion and the sensor layout. First, to model limited statistical dependence we assume that points change their motion plans after traveling some local distance threshold ℓ. Second, we assume that sensor regions are modeled as disks of constant radius, and (again to limit statistical dependence) not too many disks overlap the same region of space. These assumptions are not part of our framework. They are just useful for this comparison.

Here we will assume that flight plans are linear and that motion is in the plane, but generalizations are not difficult. Let Q denote a collection of n moving objects over some long time period $0 \le t \le T$. We assume that the location of the ith object is broken into some number of linear *segments*, each represented by a sequence of tuples $(\boldsymbol{u}_{i,j}, \boldsymbol{v}_{i,j}, t_{i,j}) \in (\mathbb{Z}^2, \mathbb{Z}^2, \mathbb{Z}^+)$, which indicates that in the time interval $t \in (t_{i,j-1}, t_{i,j}]$, the ith object is located at the point $\boldsymbol{u}_{i,j} + t \cdot \boldsymbol{v}_{i,j}$. (Let $t_{i,0} = 0$.) We assume that all these quantities are integers and that the coordinates of $\boldsymbol{u}_{i,j}$, $\boldsymbol{v}_{i,j}$ are each representable with at most b bits. Let $\Delta_{i,j} = t_{i,j} - t_{i,j-1}$ denote the length of the j time interval for the ith point.

In most real motion systems objects change velocities periodically. To model this, we assume we are given a locality parameter ℓ for the system, and we assume that the maximum length of any segment (that is, $\max_{i,j} \Delta_{i,j} \cdot \|\boldsymbol{v}_{i,j}\|$) is at most ℓ. Let m be the minimum number of segments that need to be encoded for any single object. Assuming a fix-length encoding of the numeric values, each segment requires at least $4b$ bits to encode, which implies that the number of bits needed to encode the entire system of n objects for a single time step is at least

$$B_{\text{KDS}}(n, \ell) \ge \frac{4n \cdot m \cdot b}{T}.$$

We call this the *short-haul KDS bit rate* for this system.

In order to model such a scenario within our framework, let P denote a collection of S sensors in the plane. Let us assume that each sensor region is a disk of radius λ. We may assume that the flight plans have been drawn according to some stable random process, so that the sensor output streams satisfy the assumptions of stationarity and ergodicity. We will need to add the reasonable assumption that the sensors are not too densely clustered (since our notion of locality is based on k-nearest neighbors and not on an arbitrary distance threshold.) More formally, we assume that, for some constant $\gamma \ge 1$, any disk of radius $r > 0$ intersects at most $\gamma \lceil r/\lambda \rceil^2$ sensor regions. Let $\boldsymbol{X} = (X_1, X_2, \ldots, X_S)$ denote the resulting collection of sensor output streams, and let $H_k(n, \ell) \stackrel{\text{def}}{=} H_k(\boldsymbol{X})$ denote the normalized k-local entropy of the resulting system. Our main result shows that the k-local entropy is within a constant factor of the short-haul KDS bit rate, and thus is a reasonably efficient measure of motion complexity even when compared to an approach based on prescribing the motions.

Theorem 2. *Consider a short-haul KDS and the sensor-based systems defined above. Then for all sufficiently large k*

$$H_k(n, \ell) \;\le\; \left(\frac{4\,\ell}{\lambda}\sqrt{\frac{\gamma}{k}} + 1\right) B_{\text{KDS}}(n, \ell).$$

Before giving the proof, observe that this implies that if the locality parameter k grows proportionally to $(\ell/\lambda)^2$, then we can encode the observed continuous motion as efficiently as its raw representation. That

is, k should be proportional to the square of the number of sensors needed to cover each segment of linear motion. Note that this is independent of the number of sensors and the number of moving objects. It is also important to note that this is independent of the sensor sampling rate. Doubling the sampling frequency will double the size of the raw data set, but it does not increase the information content, and hence does not increase the system entropy.

Corollary 1. *By selecting $k = \Omega((\ell/\lambda)^2)$, we have $H_k(n, \ell) = O(B_{\text{KDS}}(n, \ell))$.*

Proof. Consider an arbitrary moving object j of the system, and let $X_{i,j}$ denote the 0–1 observation counts for sensor i considering just this one object. Let $\boldsymbol{X}_{(j)} = (X_{1,j}, X_{2,j}, \ldots, X_{S,j})$ denote the resulting single-object sensor system. Clearly, $H_k(\boldsymbol{X}) \leq \sum_{j=1}^{n} H_k(\boldsymbol{X}_{(j)})$, since the latter is an upper bound on the joint k-local entropy of the entire system, and the sum of observations cannot have greater entropy than the joint system since the sum generally contains less information than the individual observations.

Let m_j be the number of segments representing the motion of object j. Each segment is of length $\leq \ell$. Consider the per-object KDS bit-rate for object j, denoted $B_{\text{KDS}}(j)$. Note that KDS considers the motion of each object individually, so $B_{\text{KDS}} = \sum_{j=1}^{n} B_{\text{KDS}}(j)$. KDS requires $4b$ bits per segment, so $B_{\text{KDS}}(j) \geq \frac{4 \cdot b \cdot m_j}{T}$. Let $\ell' = (\lambda/4)\sqrt{k/\gamma}$. Observe that $\ell' > 0$. Subdivide each of the m_j segments into at most $\lfloor \ell/\ell' \rfloor$ subsegments of length ℓ' and at most one of length less than ℓ' . Then there are a total of at most $m_j(\ell/\ell'+1)$ subsegments of length ℓ'.

We claim that the joint entropy of the sensors whose regions intersect each subsegment is at most $4b$. To see this, observe that there are 2^{4b} possible linear paths upon which the object may be moving, and each choice completely determines the output of all these sensors (in this single-object system). The entropy is maximized when all paths have equal probability, which implies that the joint entropy is $\log_2 2^{4b} = 4b$. Recall that at most $\gamma\lceil r/\lambda \rceil^2$ sensor regions intersect any disk of radius r. Let $r = \ell'$ be the radius of a disk that covers a subsegment. Then at most $\gamma\lceil (1/4)\sqrt{k/\gamma} \rceil^2$ sensor regions can intersect some subsegment. We assert that all sensors intersecting this subsegment are mutually k-close. To see this, consider some sensors s_1 and s_2, with sensing region centers c_1 and c_2 respectively, that intersect such a subsegment. Observe that, by the triangle inequality, $||c_1 c_2|| \leq 2\lambda + \ell'$. Recall that $\ell' = (\lambda/4)\sqrt{k/\gamma}$. Choosing $k \geq 16\gamma$, this means that $\lambda \leq \ell'$, so $2\lambda + \ell' \leq 3\ell'$. Thus, for each sensor s_1 whose region overlaps this subsegment, the centers of the other overlapping sensor regions lie within a disk of radius $3\ell'$ centered at c_1. In order to establish our assertion, it suffices to show that the number of sensor centers lying within such a disk is at most k. Again recall that at most $\gamma\lceil r/\lambda \rceil^2$ sensor regions intersect any disk of radius r, so at most $\gamma\lceil (3/4)\sqrt{k/\gamma} \rceil^2$ sensor regions intersect the disk of radius $3\ell'$. Under our assumption that $k \geq 16\gamma$, it is easy to verify that $\gamma\lceil (3/4)\sqrt{k/\gamma} \rceil^2 \leq k$, as desired. Since the overlapping sensors are all mutually k-close, their k-local entropy is equal to their joint entropy, and so the the k-local entropy is also at most $4b$. Thus, to establish an upper bound on $H_k(\boldsymbol{X}_{(j)})$, it suffices to multiply the total number of subsegments by $4b$ and normalize by dividing by T. So we have

$$H_k(\boldsymbol{X}_{(j)}) \leq \left(\frac{\ell}{\ell'} + 1\right) \frac{4bm_j}{T} \leq \left(\frac{\ell}{\ell'} + 1\right) B_{\text{KDS}}(j) = \left(\frac{4\ell}{\lambda}\sqrt{\frac{\gamma}{k}} + 1\right) B_{\text{KDS}}(j).$$

Considering the normalized k-local entropy of the entire system, we have

$$H_k(\boldsymbol{X}) \leq \sum_{j=1}^{n} \left(\frac{4\ell}{\lambda}\sqrt{\frac{\gamma}{k}} + 1\right) B_{\text{KDS}}(j) = \left(\frac{4\ell}{\lambda}\sqrt{\frac{\gamma}{k}} + 1\right) B_{\text{KDS}},$$

which completes the proof.

5 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 [35]. It consists of sensor activation times for over 200 sensors in a network covering the hallways of the building. Each sensor records times of nearby motion in coordinated universal time in milliseconds. For our analysis, we group activations into *time steps* consisting of the count of all activations for a single sensor over 0.1 second.

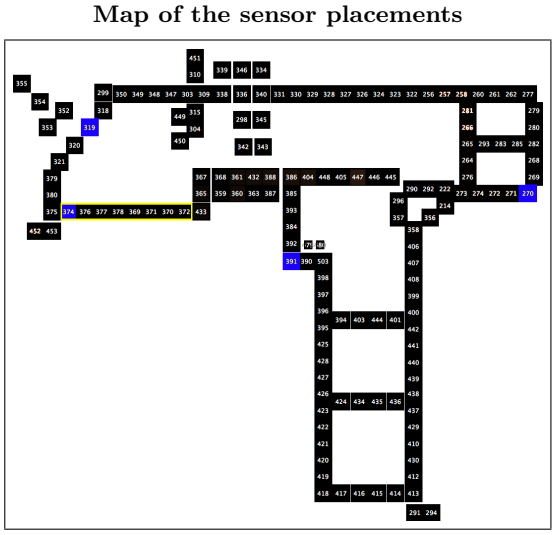

Fig. 4. Highlighted individual sensor regions are those selected for joint entropy analysis. The highlighted hallway shows the eight sensors that were used for the compression analysis. Unconnected regions are lobbies or conference rooms, which were not completely covered by sensor ranges.

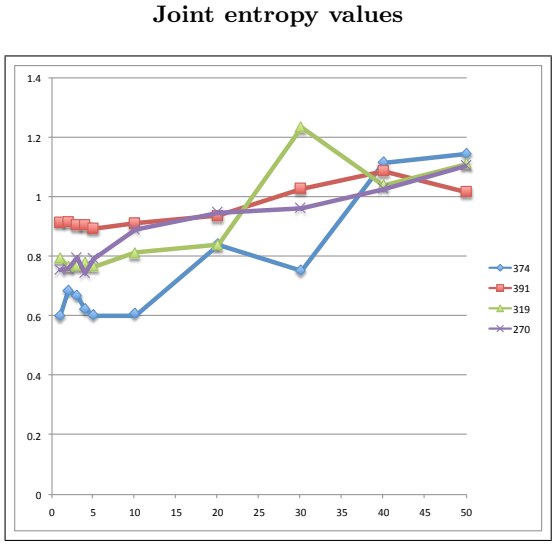

Fig. 5. Plotted joint entropy values for values of k. These are shown for $k = 1$ to $k = 5$ at increments of 1 and $k = 10$ to $k = 50$ at increments of 10.

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.

From the definitions given in Section 2, it follows that the normalized joint entropy of two sequences generated by a common process is defined to be

$$H(X,Y) \stackrel{\text{def}}{=} \lim_{T\to\infty} -\frac{1}{T} \sum_{(x,y),|x|=|y|=T} p_{(x,y)} \log_2(p_{(x,y)}),$$

where $p(x,y)$ is the probability that two events x and y both occur. 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 5 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 (see Figure 5). The activation status was considered for these sensors for 70,000 intervals, each of length 0.1 of a second (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 3), used 7.1 MB, or an additional 45.7% of the separately compressed space.

6 Conclusions

We introduced a sensor-based framework for kinetic data which can handle unrestricted point motion and only relies on past information. We analyzed our framework's encoding size and gave a c-approximation algorithm for compressing point motion as recorded by our framework for a constant c. Open questions include solving global statistical questions on kinetic data using this framework, e.g., answering clustering questions, finding the centerpoint, or finding the point set diameter. In order to do this, it would first be necessary to allow retrieval, in the form of range queries, over the compressed data. Ideally, this retrieval would operate without decompressing the data. In addition, the algorithms given on this framework were stated assuming a central processing node with global knowledge. It would be interesting to modify these algorithms to operate in a more distributed manner.

7 Acknowledgements

The authors thank anonymous reviewers for their helpful and detailed comments.

References

1. B. Adams, M. Pauly, R. Keiser, and L. J. Guibas. Adaptively sampled particle fluids. In *ACM SIGGRAPH*, 2007.
2. P. K. Agarwal, L. J. Guibas, H. Edelsbrunner, J. Erickson, M. Isard, S. Har-Peled, J. Hershberger, C. Jensen, L. Kavraki, P. Koehl, M. Lin, D. Manocha, D. Metaxas, B. Mirtich, D. M. Mount, S. Muthukrishnan, D. Pai, E. Sacks, J. Snoeyink, S. Suri, and O. Wolefson. Algorithmic issues in modeling motion. *ACM Computing Surveys*, 34:550–572, December 2002.
3. M. J. Atallah. Some dynamic computational geometry poblems. In *Comput. Math. Appl*, volume 11(12), pages 1171–1181, 1985.
4. B. Babcock and C. Olston. Distributed top-k monitoring. In *SIGMOD*, pages 28–39, 2003.
5. J. Basch, L. J. Guibas, and J. Hershberger. Data structures for mobile data. In *SODA*, 1997.
6. G. Cormode, S. Muthukrishnan, and K. Yi. Algorithms for distributed functional monitoring. In *SODA*, pages 1076–1085, 2008.
7. G. Cormode, S. Muthukrishnan, and W. Zhuang. Conquering the divide: Continuous clustering of distributed data streams. In *IEEE 23rd International Conference on Data Engineering*, pages 1036–1045, 2007.
8. T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-IEEE, second edition, 2006.
9. A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Processing approximate aggregate queries in wireless sensor networks. *Inf. Syst.*, 31(8):770–792, 2006.
10. A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Dissemination of compressed historical information in sensor networks. *The VLDB Journal*, 16(4):439–461, 2007.
11. P. Erdös and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In A. Hajnal, L. Lovász, and V. Sos, editors, *Infinite and Finite Sets*, volume 10, pages 609–627, 1975.
12. S. Gandhi, R. Kumar, and S. Suri. Target counting under minimal sensing: Complexity and approximations. *Workshop on Algorithmic Aspects of Wireless Sensor Networks (AlgoSensors)*, pages 30–42, 2008.
13. S. Gandhi, S. Nath, S. Suri, and J. Liu. Gamps: Compressing multi sensor data by grouping and amplitude scaling. In *ACM SIGMOD*, 2009.
14. L. Guibas. Kinetic data structures. In D. Mehta and S. Sahni, editors, *Handbook of Data Structures and Applications*, pages 23–1–23–18. Chapman and Hall/CRC, 2004.
15. L. J. Guibas. Sensing, tracking and reasoning with relations. *IEEE Signal Processing Mag.*, 19(2), Mar 2002.
16. A. Guitton, N. Trigoni, and S. Helmer. Fault-tolerant compression algorithms for sensor networks with unreliable links. Technical Report BBKCS-08-01, Birkbeck, University of London, 2008.
17. P. Gupta, R. Janardan, and M. Smid. Fast algorithms for collision and proximity problems involving moving geometric objects. In *Comput. Geom. Theory Appl*, volume 6, pages 371–391, 1996.
18. D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proc. of the IRE*, 40, Sept. 1952.
19. C. Johnen and L. H. Nguyen. Self-stabilizing weight-based clustering algorithm for ad hoc sensor networks. *Workshop on Algorithmic Aspects of Wireless Sensor Networks (AlgoSensors)*, pages 83–94, 2006.
20. S. Kahan. A model for data in motion. In *STOC '91: Proc. of the 23rd ACM Symp. on Theory of Computing*, pages 265–277, 1991.
21. R. Krauthgamer and J. R. Lee. Navigating nets: Simple algorithms for proximity search. In *SODA*, 2004.
22. A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *ACM international workshop on wireless sensor networks and applications*, pages 88–97, 2002.

23. MIT Media Lab. The owl project. `http://owlproject.media.mit.edu/`.
24. J. J. Monaghan. Smoothed particle hydrodynamics. In *Reports on Progress in Physics*, volume 68, pages 1703–1759, 2005.
25. D. M. Mount, N. S. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu. A computational framework for incremental motion. In *Proc. 20th Annu. ACM Sympos. Comput. Geom.*, pages 200–209, 2004.
26. S. Nikoletseas and P. G. Spirakis. Efficient sensor network design for continuous monitoring of moving objects. *Theoretical Computer Science*, 402(1):56–66, 2008.
27. J. Rissanen. Generalized Kraft inequality and arithmetic coding. *IBM Jour. of Research and Dev.*, 20, 1976.
28. C. M. Sadler and M. Martonosi. Data compression algorithms for energy-constrained devices in delay tolerant networks. In *SENSYS*, November 2006.
29. N. Saunier and T. Sayed. Automated analysis of road safety with video data. In *Transportation Research Record*, pages 57–64, 2007.
30. E. Schomer and C. Theil. Efficient collision detection for moving polyhedra. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 51–60, 1995.
31. E. Schomer and C. Theil. Subquadratic algorithms for the general collision detection problem. In *Abstracts 12th European Workshop Comput. Geom*, pages 95–101, 1996.
32. C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948.
33. E. Soroush, K. Wu, and J. Pei. Fast and quality-guaranteed data streaming in resource-constrained sensor networks. In *ACM Symp. on Mobile ad hoc networking and computing*, pages 391–400, 2008.
34. B. J. M. Stutchbury, S. A. Tarof, T. Done, E. Gow, P. M. Kramer, J. Tautin, J. W. Fox, and V. Afanasyev. Tracking long-distance songbird migration by using geolocators. *Science*, page 896, February 2009.
35. C. R. Wren, Y. A. Ivanov, D. Leigh, and J. Westbues. The MERL motion detector dataset: 2007 workshop on massive datasets. Technical Report TR2007-069, Mitsubishi Electric Research Laboratories, Cambridge, MA, USA, August 2007.
36. A. D. Wyner and J. Ziv. The sliding-window lempel-ziv algorithm is asymptotically optimal. In *Proceedings of the IEEE*, pages 872–877, Jun 1994.
37. K. Yi and Q. Zhang. Multi-dimensional online tracking. In *SODA*, 2009.
38. J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, IT-23(3), May 1977.