Algorithms for Calculating Statistical Properties of Moving Points

Sorelle A. Friedler

Dissertation Proposal

Committee
David Mount (Chair)
William Gasarch
Samir Khuller
Amitabh Varshney

Department of Computer Science
University of Maryland, College Park
January 2009

Abstract

Robust statistics and kinetic data structures are two frequently studied theoretical areas with practical motivations. The former topic is the study of statistical estimators that are robust to data outliers. The latter topic is the study of data structures for calculations on moving point sets. The combination of these two areas has not previously been studied. In studying this intersection, we consider these problems in the context of both an established kinetic framework (called KDS) that relies on advance point motion information and calculates properties continuously and a new sensor-based framework that uses discrete point observations. Using the KDS model, we present an approximation algorithm for the kinetic robust k-center problem, a clustering problem that requires k clusters but allows some outlying points to remain unclustered.

For many practical problems that inspired the exploration into robustness, the KDS model is inapplicable due to the point motion restrictions and the advance flight plans required. Working towards a solution to the kinetic robust k-center problem on a framework that allows unrestricted point motion, we present a new framework for kinetic data that allows calculations on moving points via sensor-recorded observations. This new framework is one of the first within the computational geometry community to allow analysis of moving points without *a priori* knowledge of point motion. Analysis within this framework is based on the entropy of the point set's motion, so efficiency bounds are a function of observed complexity instead of worst-case motion. A compression algorithm within this framework is presented.

The focus of this proposal will be on examining robust statistical problems on kinetic data. These problems will be explored within the sensor-based framework. Specific questions to be answered will lead from the current compression algorithm to more complex statistical analyses.

Contents

1 Introduction

Collecting information about moving points has become increasingly common. This has led to often massive data sets containing point observations for multiple time steps. As sensing technologies become increasingly inexpensive and available, these data sets will only get larger. Currently, these data sets are quite diverse, ranging from the sounds and positions of owls [34], to the movements of people in a building [33], to the locations of cars via GPS navigation devices [45]. The wide scope of applications involving motion has given rise to a diversity of questions about these data sets. For example, these questions may involve migration of animals, the dynamics of social relationships, and navigation in the presence of traffic. Due to the vast amounts of data involved, these problems require automated and efficient solutions. Additionally, these data sets contain real-world data that does not always exactly fit a given model; thus outliers are common. This work proposes answering questions inspired by this real-world data, which lie in the intersection of the study of data structures for moving points and robust statistics, via a sensor-based theoretical framework for motion data.

Many areas within and outside of computer science consider calculation of properties of moving points. The nature of these calculations and the associated computational challenges depend on the particular area of study. For example, when working with wireless sensor networks, the question is often how to capture the data. Problems of sensor limitations due to processing capacity, limited power sources, and communication range are considered along with calculation on the data collected by the entire network [4]. Within the database community, the question is how to store and answer queries involving these points [47]. Physicists, on the other hand, consider statistical calculations on moving points as part of simulations and similar simulations underly animation in computer graphics [35]. Compression of moving points is also considered within computer graphics for video compression needs [12, 20].

Motion is a continuous phenomenon. However, the practical limits on its measurement have resulted in a variety of methods ranging from continuous modeling to discrete modeling involving discrete time steps. In the field of computational geometry, most of the research has focused on piecewise continuous motion of objects, called *kinetic data* [6,8,26,39,40]. The *kinetic data structures* (KDS) model proposed by Basch, Guibas, and Hershberger [8] has become the standard method for dealing with kinetic data. The KDS model requires algebraic expressions representing point motion, but allows these "flight plans" to change. There has not been as much work within computational geometry on discrete modeling of motion [30]. Continuous modeling allows for perfect accuracy in theory, assuming that objects adhere to the given flight plans, while discrete modeling is less precise but more practical, simulating actual data collection.

Robustness is one of the problems that has not yet been considered within the domain of moving points. Robustness is an important statistical characteristic of an estimator; it ensures that the presence of some data outliers (due to measurement error or heterogeneity of the data set) does not alter the result by a large amount. *Clustering* problems make up a natural class of geometric problems with statistical properties that can be considered robustly. These problems group points into optimal subsets (known as *clusters*) based on some criterion.

Practical data collection often involves sensing technology. Sensors collect the data at synchronized time steps and no restrictions are made on the motion of observed objects. Additionally, no advance information is known about the objects' motion. The vast quantities of data collected necessitate data compression. Data compression algorithms perform a process in which the data set's representation is *encoded* so that it uses less storage than in its original form.

My ultimate goal, no matter the framework, is to answer statistical questions about kinetic

data with provable efficiency. My first preliminary work is based on the standard kinetic model, KDS. This work opened the intersection of understanding motion data and robust statistics by looking at a robust clustering problem [18]. I created an efficient approximation algorithm, but it suffered from the constraints of the model.

A more practical approach that aims to allow statistical calculations with greater efficiency is presented in my second preliminary work [19]. This work presents a new framework for kinetic data that is based on a discrete sampling model. Data collection through sampling is commonly done using sensors, and the new model takes this approach. Under this model, point motion is allowed to be unrestricted and unpredicted. A data compression algorithm on this framework is presented and is shown to require no more than a constant times the optimal storage space. Analysis is based on the information content of the kinetic data.

The preliminary research is just the beginning of developing an understanding of how to calculate statistical properties of moving point sets. It proposes an initial model, the sensor-based framework, and a method of analysis based on information content. Future frameworks may vary based on continuity of motion versus discrete sampling, theoretical versus practical emphasis, and method of analysis. With respect to the preliminary work, my proposed work will focus on demonstration of the practicality of the sensor-based framework through further theoretical analysis of specific robust statistical questions.

The rest of this proposal will describe the statistical properties, frameworks, and methods of analysis that have and will be examined in the context of kinetic data. Section 2 gives a review of the most relevant previous literature in related areas. Section 3.1 discusses the preliminary work based within the KDS model, and Section 3.2 presents the new sensor-based framework. Finally, Section 4 describes proposed future work.

2 Literature Review

2.1 Data Structures for Moving Points

One of the first papers about calculations of properties of moving points in a computational geometry context was written in 1985 by Mikhail Atallah [6]. In this beginning work, points are assumed to follow paths, also known as *flight plans*, modeled by polynomials of degree at most k that are functions of time; these motion paths are provided at the beginning of the algorithm and do not change. Atallah examines the properties of these function interactions in a static context with time as an additional dimension. He gives results regarding the descriptive complexity of the lower envelope (that is, the minimum value over all the functions) and the 2-D convex hull over all time steps. He also gives time complexity results for calculating these changing quantities. While points are modeled by polynomials, they are not assumed to be continuous at all points; instead, the number of undefined periods of time and discontinuous jumps are assumed to be bounded ($O(1)$).

Atallah showed that the concept of Davenport-Schinzel sequences are important in this context. A Davenport-Schinzel sequence of order s is a sequence of characters from an alphabet of size n that does not contain any consecutively repeating characters or any alternating subsequences of length $s + 2$ [43]. Let $\lambda_s(n)$ denote the maximum lengths of such a sequence. Atallah showed that, when considering the functions over time, there can be at most $\lambda_s(n)$ function pieces in the lower envelope of this collection, or $\lambda_s(n)$ possible values for the minimum value of the function set over time. Atallah's proof proceeds by showing that $\lambda_s(n)$ directly models a lower envelope by assigning characters to each function piece on the lower envelope. Since consecutive repeating characters would model the same function appearring next to itself, these do not occur. For func-

tions that do not intersect more than s times, it is impossible to have alternating subsequences of length more than $s + 2$. He shows that since $\lambda_s(n)$ is bounded by $O(n \log^* n)$, and thus the number of function pieces on the lower envelope is also $O(n \log^* n)$. (More recent research has tightened these bounds [43]).

The bound on the number of lower envelope changes is also used to provide a bound on the number of convex hulls over time. Atallah shows that a single point changes from being in the convex hull to not in the convex hull $O(\lambda_s(n))$ times by defining membership in the convex hull based on the minimization of a function. This is then applied to all n points to get a bound of $O(n \cdot \lambda_s(n))$ convex hulls.

Other early papers include a 1983 paper by Guibas, Ramshaw, and Stolfi [24], which views point motion as following predetermined curves and polygons. It frames questions in terms of the relationships between these curves and polygons. In 1991, Kahan [30] was the first to introduce a framework for motion in which flight plans are not used, and the only required prior knowledge is an upper bound on each point's velocity. Instead, Kahan's model relies on a function that can be queried to determine current point locations. Schömer and Thiel examined the problem of collision detection between moving polyhedra in a series of papers in 1995 and 1996 [39, 40]. Motion is described in advance by polynomial functions. Similarly, a 1996 paper by Gupta, Janardan, and Smid [26] considered collision detection, minimum separation, and other problems on points, line segments, or hyperrectangles moving on predetermined linear paths. In 1997, Basch, Guibas, and Hershberger [8] introduced the kinetic data structures (KDS) model and this became the standard for moving point calculation. This framework also requires a predetermined functional model for each point's path, but allows these models to change at discrete time instances. The KDS model is discussed in more detail in Section 2.2.

Despite the progress to date in modeling motion, many real-world inspired issues remain to be addressed. In an effort to precisely identify these areas and spur research into a unified motion framework a group of researchers participated in a workshop in 2002, which produced a survey of these issues across computational geometry, mesh generation, physical simulation, biology, computer vision, robotics, spatio-temporal databases, and mobile wireless networks [3]. They also suggested directions for future research. Here, we focus on the problems they describe that relate directly to the preliminary and proposed work.

Within computational geometry, Agarwal *et al.* [3] propose research into *motion-sensitive* algorithms, bounds on the number of combinatorial changes, and decentralization. Motion-sensitive algorithms give complexity bounds that are based not on the worst case bounds given any point motion, but provide efficiency measures based on the predictability of the moving objects and their relation to each other (note that the framework that we will introduce later in Section 3.2 is motion-sensitive). Similarly, they propose analyzing the number of combinatorial changes to a property as a function of the objects' motion complexity instead of a simple worst-case bound. These proposed methods have the advantage of more realistically modeling the efficiency of algorithms on moving point sets. In addition, for practical use in many situations such as ad-hoc networks, they mention that it would be useful to develop a model for motion that allows computation to be distributed over multiple processors.

In the field of computer vision, motion analysis is used for applications including shape identification and tracking, initialization of a tracking sequence, and handling additional complexity in terms of lighting, motion, or shape. Statistical error analysis is used to create models of object motion. Agarwal *et al.* [3] suggest that further broadening of research consider multiple frames for tracking initialization, robustly handle error through learning-based methods, and generally view motion through a high-level hierarchical view in order to take advantage of all the data available.

Challenges in the field of wireless networks echo the need for a hierarchical view of the data. Agarwal *et al.* [3] propose that research focus on modeling predictable user motion at many levels in the data hierarchy. The hope is that these models would allow future motion to be predicted based on the model for a single level without detailed knowledge of motion at other levels, for example without knowing the motion of an individual at the lowest level of the hierarchy.

Overall, Agarwal *et al.* [3] identify eight common themes of issues that remain to be addressed in motion modeling. These include the ability to handle robustness, both in the context of data error and in the context of higher level analysis of motion trends through the use of hierarchical data structures. The struggle between continuous and discrete models remains to be fully explored, with research in continuous models not being confined to the theoretical community and discrete models not only being analyzed by practical researchers. Coupled with this, methods of identifying current motion of objects with an emphasis on unpredicted motion remain to be fully explored. Finally, the decentralization of data processing, a requirement in many practical applications, is especially important. Section 2.3 returns to some of these questions, and in Section 3.2 we present a framework that addresses some of these issues.

One interesting alternate method for handling kinetic data was proposed by Har-Peled in 2004 [28]. He examines the discrete k-center clustering of a set of points moving with motion modeled by polynomials of degree at most μ. The *discrete k-center problem* is given a set of n points and finds the k center points taken from the input that minimize the maximum distance (called the *radius*) from any point to its closest center. For this clustering problem, each center together with the points that are closest to that center constitute a cluster. Instead of maintaining the k clusters as they change over time, Har-Peled finds a single static clustering with $k^{\mu+1}$ clusters that is within a constant factor of the optimal clustering at any time. He begins with a clustering algorithm for a static point set that randomly samples points, partitions those points into k clusters, and then partitions all points that were not covered by the first clustering into an additional k clusters. This two-round clustering algorithm is then extended to an η-round clustering algorithm by repeating the first two steps for uncovered points.

To handle moving point sets, Har-Peled partitions time into intervals when the relative distances between points do not change. For each of these intervals, the Gonzalez [22] greedy clustering algorithm (which bases the partitioning only on inter-point distances) is used as the black-box clustering algorithm needed by Har-Peled's static clustering algorithm. The radius returned is a 2-approximation of the optimal over that interval, where the time for the optimum over the interval is found by calculating the lowest point on an upper-envelope representing radius lengths. The set with the minimum radius over all the intervals gives the answer to a different kind of clustering problem in a dimension one larger than the original (the dimension for time is added). This set is then expanded to create a static clustering with more centers that holds over time. Har-Peled's algorithm runs in $O(nk)$ time for $k = O(n^{1/14})$. Other algorithms that use a similar strategy of considering the points statically in a higher dimension including time are those posed by Atallah [6], Guibas *et al.* [24], and Gupta [26]. The current standard for calculating properties of moving points is the KDS model that handles motion in an online manner. It is described in the next section.

2.2 Kinetic Data Structures Model

In 1997, Basch, Guibas, and Hershberger [8] introduced a model for kinetic data called a *kinetic data structure* (KDS). This model assumes advance knowledge of point flight plans, but allows these plans to change. Algorithms are developed to track specific properties of moving points in

an online manner. This is done through a set of boolean conditions called *certificates* and a corresponding set of update rules. Certificates guarantee geometric relations necessary to a particular problem's solution, and update rules specify how to respond when a certificate fails. Certificate failures are predicted and queued based on the points' planned paths of motion, assumed to be in the form of algebraic expressions. KDSs are evaluated based on properties of the certificate set.

There are four criteria under which the computational cost of a KDS is evaluated: responsiveness, efficiency, compactness, and locality [23]. *Responsiveness* measures the complexity of the cost to repair the solution after a certificate fails. *Efficiency* measures the number of certificate failures as compared to the number of required changes to the solution as the points move. *Compactness* measures the size of the certificate set. *Locality* measures the number of certificates in which each point participates. Guibas provides a more detailed overview of kinetic data structures in [23].

In order to illustrate the KDS model, Basch *et al.* [8] give a KDS that maintains the 2-D convex hull over time. First they consider the static solution where each point (a, b) is dualized to the line $y = ax + b$ and lower part of the convex hull is viewed as the upper envelope of the set of lines. Finding the upper envelope is done through a divide-and-conquer algorithm. The merging of two upper envelopes proceeds by sweeping from left to right and determining, for each line intersection point on either upper envelope, which envelope is higher. Certificates guarantee relative properties of x and y coordinates and line slopes in order to maintain this merge under motion. These certificates are set-up to only record relative properties of those lines surrounding them, and the number of certificates kept for each line intersection or line is a constant, so the KDS is compact, local, and responsive. The argument for efficiency is based on a multi-dimensional upper envelope complexity bound of $O(n^2 + \varepsilon)$ (which is modeled in the kinetic context by a three dimensional set consisting of two spatial dimensions and one temporal dimension) [42], which bounds the number of events that may be caused by certificate failures in this KDS system. The complexity of the moving convex hull is $\Omega(n^2)$, so the KDS is efficient.

Although most KDS solutions are based on a particular computational problem, Gao *et al.* [21] introduced a flexible kinetic data structure that can be used to solve a number of different problems involving kinetic point sets. (Later in Section 3.1, we will make use of this structure.) This structure is hierarchical in nature, and can be used both as a tree-like access structure as well as a geometric spanner (defined below). Gao et al. dubbed it a *deformable spanner*. The hierarchy and spanner both update dynamically as the points move, so this data structure provides an underlying framework on which future problems that rely on hierarchy or spanner properties can be built.

A *γ-spanner* is a graph connecting points in a point set S in $\Re^d$ with the property that for any two points in S, the distance between those points on the graph is at most γ times the distance between those points in the underlying metric. The deformable spanner is a $(1+\varepsilon)$-spanner. The *aspect ratio*, denoted α, is defined as the ratio between the maximum and minimum distances between any two points of S. For moving point sets, the aspect ratio α is actually a function of time. When considering the aspect ratio in the context of time complexity, one simple solution is to consider the maximum ratio over all times. Additionally, $\varepsilon > 0$ refers to a user-given input parameter.

The Gao *et al.* [21] spanner is constructed based on the concept of a hierarchy of discrete centers. Given a point set S, a *hierarchy of discrete centers* is a sequence of subsets $S = S_0 \supseteq S_1 \supseteq ... \supseteq S_{\lceil \log \alpha \rceil}$ such that the following properties hold for $0 \leq i \leq \lceil \log \alpha \rceil$:

- Each center in S_{i-1} is within distance 2^i of some center in S_i, the ith level of the hierarchy.
- Centers in S_i are chosen from S_{i-1}.

A center p at level i is said to *cover* a center q in level $i-1$ if q is within distance 2^i of p. By definition each center q at level $i-1$ is covered by some center in level i. One such center p is

selected (arbitrarily) to be q's parent. The center q is called the *child* of p. Other standard tree relationships are used including ancestors and descendants (both of which are considered in the improper sense, so that a node is an ancestor and descendant of itself) and siblings [14]. *Cousins* are defined as the children of a node's parents' siblings. Some properties about the deformable spanner, which follow immediately from the above properties or are proven in [21], are given below:

- $S_i \subseteq S_{i-1}$
- For any two center $p, q \in S_i$, with associated points $p, q, \in S$, $\|pq\| \geq 2^i$.
- The hierarchy has a height of at most $\lceil \log_2 \alpha \rceil$.
- Any center in S_0 is within distance 2^{i+1} from its ancestor in level S_i.

The deformable spanner maintains four types of certificates: parent-child certificates, edge certificates, separation certificates, and potential neighbor certificates. *Neighbors* of a node p in level i are defined as all nodes in that level within distance $c \cdot 2^i$ of p. The certificates are based on this distance $c \cdot 2^i$, where $c > 4$ [21]. The KDS for this spanner is appropriately efficient, local, compact, and responsive.

Many other problems have been considered using KDS, including clustering, hierarchical data structures, and minimum spanning trees [1, 2, 9, 11, 25]. For example, in 2008 Abam *et al.* presented a KDS that maintains a $(1 + \varepsilon)$-spanner over point motion independent of the point set's aspect ratio. The spanner is based on the union of Delauney triangulations done over the points where distance is defined based on a metric that uses a diamond instead of a unit circle and the diamond is rotated for each triangulation. The Delauney triangulation based on the Euclidean metric is easy to kineticize since there are local properties that guarantee the triangulation and corresponding local update rules. Abam's spanner uses similar properties and update rules modified for the diamond-based metric and is shown to be efficient.

2.3 Sensors and Streams

Let us now move from the continuous model of motion exemplified by KDS to systems that analyze motion based on discrete time samples. One practical way to observe and record motion is through the use of sensors in a sensor network. *Sensors* are small nodes with the ability to sense characteristics of their environment in addition to limited data processing and communication ability. *Sensor networks* contain many sensors as nodes networked with each other that are densely deployed to observe some environment. These networks can be applied for military benefit, environmental protection, health monitoring, household convenience, vehicle detection, and in many other situations [4].

Akyildiz *et al.* [4] identify eight main issues in sensor network design in a 2002 survey of research in wireless sensor networks. Some of these constraints are specific to sensor networks and require new techniques and technologies. Sensors are assumed to be cheap, which allows broad use, but also means that they are prone to failure. Correspondingly, sensor networks are evaluated based on their *fault tolerance*, their ability to continue operating without interruption if some sensors fail. The specific level of fault tolerance needed, as measured by the probability that no nodes will fail within a given time interval, is dependent on the measurement error and added environmental strain on the sensors based on the application. *Scalability* refers to the ability of the network to operate efficiently on millions of nodes and to take advantage of the high density of sensor deployment. Due to the vast number of sensors in the network and their tendency to fail, the sensor network topology must be highly malleable. Production costs of each node must be low in order to make these sensor networks cost effective, and there are many hardware challenges, both related and unrelated to cost. The network must be able to operate

unattended for long periods of time, and may encounter harsh environmental factors depending on the application and deployment location. In order to utilize this network, the sensors must be able to communicate with each other and, possibly through multiple hops, with a central server. The transmission media chosen should be globally available and not require a line of sight between sender and receiver (as is required by infrared communication). Finally, power consumption on individual sensor nodes is one of the most important areas of research, since the sensors have limited battery life and sensors that run out of power must be removed from the network. Limiting power consumption while sensing, communicating, and processing data is crucial [4].

The data collected by the sensors at small time intervals over the entirety of their lifespan (which is theoretically infinite) is reported at each time step and makes up what is known as a stream of data. More formally, *data streams* are a sequence of data items that arrive online, are potentially unbounded in size, arrive in an undetermined order, and are discarded after they have been processed. In a 2002 survey of models and issues in data stream systems, done from a databases perspective, Babcock *et al.* [7] present current query challenges. The main underlying challenge is the problem of unbounded memory requirements due to the potentially unbounded length of the sequence of data. In addition, Arasu *et al.* [5] show that without knowing the length of the input data stream, it is impossible to bound the memory requirements of a single query using a join operation. Due to these memory constraints, approximations to query answers are desirable. Random sampling, histograms, and other synopsis techniques can be used to provide an overview of the entire data stream. Alternatively, sliding windows allow queries to be answered based only on recent data within some limited time frame. This approximation has the advantage of being well-defined, deterministic, and, for many applications, emphasizing the recent data the user cares about. However, the data to be included in the window becomes less clear when defined over multiple streams. Other approximation techniques depend on the relative speed of the operations that update data and compute query answers. Fast updates (relative to slow answer computing) suggest batch processing, in which many updates are made before the answer is computed. Fast answer computing (relative to slow updates) suggest stream sampling followed immediately by updates, although such a scenario does not admit provable approximation guarantees [7].

In addition to the approximation necessitated by the memory challenges inherent in data stream processing, there are also queries that are impossible to answer accurately in the present, blocking query operators and queries based on past data. *Blocking queries* are queries that may not be answered until they have seen the entirety of the data; for example, sorting queries or those involving aggregation (summation, mean, etc.). Approaches to answering these queries include replacing the query with a non-blocking query with an approximately similar result, maintaining an answer given the data seen so far, and reasoning based on an augmented data stream containing assertions about future data. Queries involving past data must rely on data summaries. Creating these summaries is challenging since future queries are unknown and all data may not be stored. Answers to these queries are likely to be approximate. A solution that side-steps this issue is to state to the user that any such queries will consider only the data stream beginning at the time the query is issued [7].

The literature on sensors and streams is too vast to cover completely here, instead we discuss a small number of relevant papers. Cormode *et al.* [15] consider a problem over data streams generated by distributed sites. They are the first to consider the the maintenance of the continuous discrete k-center clustering problem and give accuracy guarantees. They propose four methods and compare them theoretically and experimentally. These four algorithms combine local and global methods with the Gonzalez farthest point algorithm [22] and the parallel guessing algorithm. The local algorithms distribute the clustering decisions and then merge to find a global clustering.

Cormode *et al.* show that merging α-approximate clusters using a β-approximate technique results in an $(\alpha+\beta)$-approximate clustering. The global algorithms distribute the monitoring of the validity of the current clustering, but assume transmission of that data to a central server for a global recalculation of clustering when required. The Gonzalez farthest point algorithm, used at the distributed sites for local clustering or by the central server for global clustering, sequentially chooses the point farthest from any identified cluster. It is known to give a 2-approximation of the optimal k-center clustering [22]. The parallel guessing algorithm guesses some radius r and creates a new center and marks points within distance r of that center as clustered anytime it encounters a point that is not currently in a cluster. This algorithm is run in parallel for multiple guesses of r and the number of guesses is dependent on the aspect ratio of the point set. The guess with minimum r that still clusters all points into at most k clusters is the resulting clustering. This algorithm is shown to be a $(2+\varepsilon)$-approximation of the optimal clustering [15].

Cormode *et al.* show that both local algorithms are $(4+\varepsilon)$-approximations and both global algorithms are $(2+\varepsilon)$-approximation of the optimal clustering. Both implementations of the Gonzalez algorithm require $O(n)$ space while both implementations of the parallel guessing algorithm require $O(\frac{k}{\varepsilon}\log\alpha)$ space, where α is the aspect ratio. Communication required is shown to be $O(\frac{km}{\varepsilon}\log\alpha)$ for both versions of the parallel guessing algorithm, where m is the number of distributed sites. However, no communication bounds are known for the Gonzalez local and global algorithms. Experimentally, the communication bounds for the versions based on the Gonzalez algorithm were shown to be worse than those for the local parallel guessing algorithm. The local parallel guessing algorithm was shown experimentally to require communication costs of less than one percent of the cost to send all information to a central server [15].

Other problems considered over data streams include functional monitoring problems, which keep track of a function output based on the data stream inputs in an online manner. The accuracy of the function value is based on an error parameter ε. Cormode *et al.* [16] considered the problem of monitoring monotone functions over distributed data streams and give worst-case bounds on the number of updates to the function they maintain. Yi and Zhang [49] considered arbitrary d-dimensional functions over a single data stream. They use competitive bounds to show that they update the function they maintain $O(d^2\log(d\varepsilon))$ times for every one time the function is updated by the optimal algorithm.

2.4 Data Compression and Entropy

The handling of large data sets often requires reducing the necessary storage size through a data compression algorithm. Compression algorithms represent the data set as a single string of information and return a smaller *encoded* string of compressed data. If the encoded string can be restored exactly to its original form, the algorithm is known as *lossless*. If the string cannot be restored exactly after being compressed, the algorithm is known as *lossy*. 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 [41]. The *entropy* of a string of characters taken from a random source X is defined to b $-\Sigma_x p_x \log_2(p_x)$ where x is an outcome of the random process. The optimal length of an encoded string is equal to the string's entropy. Compression algorithms that achieve this optimum are known as *entropy encoding* algorithms.

Many entropy encoding lossless compression algorithms have been considered including Huffman coding [29], arithmetic coding [36], the Lempel-Ziv dictionary algorithm [50], and the

Lempel-Ziv sliding-window algorithm. Huffman coding replaces characters with symbols so that the most probable characters are given the shortest symbols, and the least probable are represented by the longest. Arithmetic compression encodes the entire string in a base so that each character corresponds to a different digit in a fractional number. The entire string is then translated to a binary number that is precise enough so that the original number can be retrieved. The Lempel-Ziv dictionary algorithm adds matches to a prefix-tree and stores the tree along with an ordered list of pointers instead of the full string.

We will examine the Lempel-Ziv sliding-window algorithm in more depth. 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 multi-character matches are then recorded, which reduces the space of encoding each character. The window moves forward as time progresses. Note that this algorithm operates on the string in an online fashion, containing a valid encoding at any time. The encoding involves splitting the string into a collection of phrases, each of which is then encoded into a tuple. There are two types of tuples. A tuple of the form $'(0, X)'$ indicates a single plaintext character $'X'$, and a tuple of the form $'(1, i, j)'$ indicates a repeated string of length j starting at the i-th preceding character. For example, consider the encoding found using a window size of three on the string $AABBAB$. The encoded string is $(0, A)(1, 1, 1)(0, B)(1, 1, 1)(1, 3, 2)$. Intuitively, it is clear that larger window sizes yield better results since long matches are more likely to be found. In the example, this corresponds to a preference for the pointer $(1, 3, 2)$ and not for the pointers $(1, 1, 1)$ since $(1, 3, 2)$ saves more space.

In 1994, Wyner and Ziv proved that the optimal encoded length for the Lempel-Ziv sliding-window algorithm is achieved by taking the limit as the window size tends to infinity [48]. The precise optimal length is shown to be equal to the entropy of the string. The proof proceeds by showing that the expected number of bits to encode any character of the string is at most the number of bits to encode the first window plus the normalized expected value of the sum of the match encoding lengths. The latter value is identified using Kac's Lemma, which states that for some character α that has a positive probability $P_0(\alpha)$ of occurring at time 0 (the present), $\Sigma_{i=1}^{\infty} iP_{-i}(\alpha) = 1/P_0(\alpha)$, where $P_{-i}(\alpha)$ is the conditional probability (given that the character at time 0 is α) that the most recent occurrence of α was at time $-i$. Other lemmas are introduced to show that as the length of the string becomes arbitrarily large, the probability that the closest match is farther away than a function dependent on the entropy goes to zero. So the sum of the match encoding lengths, which is equal to the length to encode unmatched phrases plus the length to encode matched phrases, becomes at most the length of the encoding for matched phrases. These are shown to approach the entropy, or information content, of the string.

3 Preliminary Work

3.1 Approximation Algorithm for the Kinetic Robust K-Center Problem

In real-world applications, two algorithmic complications—motion and the presence of outliers—frequently occur. My preliminary work [18] began with the recognition that while the study of moving points sets, especially through the KDS model, and the study of statistical estimators that are not dramatically affected by data outliers (known as *robust statistics*) are both well-studied fields, their intersection has not previously been considered. The notion of robust estimators are made more precise through the definition of a breakdown point. The *breakdown point* of a statistical estimator is defined to be the smallest percentage of arbitrarily large outlying data that can cause an arbitrarily large estimated aberration in the result [27]. The breakdown point cannot

be larger than 50 percent, since otherwise it would be impossible to distinguish between inliers and outliers. As a first examination of robust statistical properties on kinetic data, preliminary work considers a clustering problem.

Criteria for clustering problems are often based on a user-given parameter k that determines the number of clusters or the number of points per cluster. The best clustering is then defined based on an individual cluster distance measure, such as cluster radius, and a merge function over all the clusters, such as summation [10]. When the points are moving, clustering problems can be considered to create clusters that move with the points over time [21] or can create static clusters that provide an approximation of the optimal at any time step [28]. Clustering problems have many applications including analysis of social networks [46], image or video segmentation [13], and analysis of vehicle motion [38]. My preliminary work considers the k-center clustering problem.

Recall from Section 2.1 that the (non-robust, non-kinetic) *k-center problem* is defined as follows: Given a set of n points, find k center points that minimize the maximum distance (called the *radius*) from any point to its closest center. The *discrete k-center problem* is a version in which each center point must be one of the original n points. The *absolute k-center problem* is a version in which each center may be any point in space [31]. The *robust k-center problem* modifies the k-center problem to handle outliers by allowing flexibility in the number of points that satisfy the distance criteria. In the formulation used, the inputs are a set of n points, a constant k, and a threshold parameter t, where $0 < t \leq 1$. The objective is to compute the smallest radius r such that there exist k disks of radius r that cover at least $\lceil tn \rceil$ points. For $t = 1$ this problem is the same as the non-robust formulation. An approximation algorithm is found within the KDS model for kinetic data, since the k-center problem is NP-hard for arbitrary k or exponential in k for fixed k. An algorithm provides a *c-approximation* to the k-center problem if the radius chosen for the k centers is no more than c times the optimal radius.

Given a real parameter $\varepsilon > 0$, my preliminary work obtains a $(3 + \varepsilon)$-approximation for the static and kinetic forms of the robust discrete k-center problem and a $(4+\varepsilon)$-approximation for the absolute version of the robust k-center problems. Note that the first bound improves upon the 8-approximation for the kinetic discrete k-center problem as given by Gao, Guibas, and Nguyen [21] and generalizes it to the robust setting. However, due to complications arising from the need for robustness, the result assumes that k is constant while the Gao *et al.* result holds for arbitrary k. My preliminary work improves the Gao *et al.* result for the non-robust kinetic problem for arbitrary k by showing that its data structure achieves a $(4+\varepsilon)$-approximation while maintaining the same quality bounds as the Gao *et al.* KDS. This kinetic robust algorithm is the first approximation algorithm for the kinetic absolute k-center problem (even ignoring robustness). An example is given to show that the $(3 + \varepsilon)$-approximation for the robust discrete k-center problem is tight.

The KDS used by the kinetic robust k-center algorithm is efficient based on the evaluation criteria described in Section 2.2. The algorithm's KDS achieves bounds of $O(\log \alpha/\varepsilon^d)$ for locality and $O(n/\varepsilon^d)$ for compactness (where α is an upper bound on the ratio between the largest and smallest inter-point distances of the point set under motion) so it does not create too many certificates. The responsiveness bound is $O((\log \alpha \log n)/\varepsilon^d)$, so the data structure is able to update quickly. The efficiency bound of $O(n^2 \log \alpha)$ is reasonable since the combinatorial structure upon which the kinetic algorithm is based requires $\Omega(n^2)$ updates [21] in the worst case, so any approach based on this structure requires $\Omega(n^2)$ updates.

All these results are obtained for metrics with constant doubling dimension — these metrics form a proper superset of the Euclidean metric. Metrics with constant doubling dimension have the packing property that a ball with radius r can be covered by at most a constant number β of balls with radius $r/2$. The dimension d is defined to be $d = \log_2 \beta$ [32]. In addition, access

is needed to functions giving the distance between two points at a given time and the earliest future time at which two points will be within some distance.

For complete details, see [18].

3.2 A Sensor-Based Framework for Kinetic Data

My preliminary work [18] on the kinetic robust k-center problem using the KDS model is limited by the nature of the problem in combination with the model's assumptions and requirements. The maintenance of individual point identities moving with continuous motion requires frequent rerunning of the center calculation algorithm due to certificate failures (which lead to a solution for fixed and not arbitrary k). In addition, when concerned with statistical properties of large moving point sets, the assumptions of the KDS model are especially unsuitable. The requirements of algebraic point motion and advance knowledge of flight plans are either inapplicable or infeasible in many scientific applications. To address these concerns, our preliminary work [19] creates a new framework in which a point set is processed no matter its motion, but a cost is paid in efficiency based on the information content of this motion. In addition, the underlying goal that structures KDS operations is maintenance of information into the future; preliminary work processes sensed data after the fact, e.g., for efficient retrieval. Unlike KDS, it also ignores point identities in favor of statistical properties; KDS focuses on properties in relation to individual points. For these problem types, the new framework serves as an alternative to the KDS model.

Data collection for vast quantities of data is frequently done via sensors in a wireless sensor network [4]. My second preliminary work bases the new framework on information about moving points as observed by such stationary sensors. These sensors aggregate statistical information at synchronized time steps for moving points within their range. The detection range of each sensor is assumed to be represented by a radius which defines a ball around the sensor. The radius is the same for each sensor in our network. The sensors may be placed so that their detection regions overlap or are disjoint. In the preliminary work, it is assumed that each sensor outputs the count of the number of points within its detection radius at the current time step.

In order to measure the efficiency of this model, the preliminary work relies on the information content of the moving point set. 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. Simple worst-case efficiency bounds do not capture this notion while the information content of the set does; it is low for highly predictable point sets and high for random motion. Algorithms with complexity analyses based on the underlying motion are known as *motion-sensitive* [3]. Motion sensitivity is captured by an information theoretic approach that is used to describe efficiency of algorithms within this framework.

In order to theoretically analyze algorithmic efficiency, we begin by assuming that the outputs of a single sensor (the count of points within its detection range) are drawn from a stationary, ergodic, random process. Since the cardinality of the moving point set is finite, the alphabet from which this sequence is drawn is also finite. The subsequence made up of the first T characters of this sequence is referred to as the stream output by a sensor, where T represents the total time over which a sensor reports its observations and T goes to infinity in the limit. Recall from Section 2.4 that by definition the *entropy* of a stream X is $-\Sigma_x p_x \log_2(p_x)$ where x is an outcome of the random process. The *joint entropy* of a set of streams X of size S is $-\Sigma_x p_x \log_2(p_x)$ where x represents a set of outcomes $\{x_1, x_2, ..., x_S\}$, and x_i is the outcome for stream X_i [17]. Note that the definition of entropy is the same as that of joint entropy for a set X of cardinality one. The

normalized joint entropy of a stream gives the entropy required per bit for lossless compression. More precisely, the normalized joint entropy is

$$H(X) \stackrel{\text{def}}{=} \lim_{T\to\infty} -\frac{1}{T}\Sigma_{x,|x|=T} p_x \log_2(p_x)$$

where T is the length of the stream and X and x are as previously defined.

As an initial problem on this framework, preliminary work considers data compression. While some algorithms based on sensors treat the incoming data as a data stream and process the data online in a manner so that it is never stored [7], my preliminary work focuses on reducing the necessary storage size through a lossless data compression algorithm on this framework. This lossless compression algorithm encodes the motion data to within a constant factor of the optimum size. Note that $H(X)$ is the optimal compression bound for this algorithm and is the optimal bound that an entropy-based encoding algorithm achieves. In addition, to justify the appropriateness of this approach, my preliminary work relates its complexity to that of KDS. My preliminary work shows that the number of bits needed to encode a set of points moving with piecewise linear motion, when considered as a function of some properties of the point motion, is comparable under both frameworks.

The main purpose of this work is as an underlying framework on which future work will be built. It is anticipated that other algorithms operating on sensed kinetic data sets, especially those determining global statistical properties, can be effectively implemented and analyzed using this framework.

For complete details, see [19].

4 Proposed Work

All proposed research will focus on expanding results on the new sensor-based framework for kinetic data. As previously explained, this framework is practical while still allowing a theoretical analysis based on entropy bounds. Future work will first focus on retrieval and then lead to creating robust statistical algorithms on this framework. Finally, given sufficient time, an experimental evaluation of these algorithms will be performed.

The retrieval problem is complementary to data compression; it focuses on retrieving only the portion of the compressed data that has been requested by the user. *Precision* measures the fraction of retrieved data that was requested and *recall* measures the fraction of requested data that was retrieved [44]. Work in this area will focus on total recall while limiting the error due to imperfect precision. The retrieval problem is interesting in its own right, but is also necessary before statistical algorithms on the framework can be created.

The k-center problem, which was previously studied in the preliminary work, is a common clustering problem and a simple global statistical property of a point set. As such, it is a logical starting point for developing statistical algorithms on the sensor-based framework. Exact and robust versions of the problem will be considered for kinetic data.

Depending on time constraints, future work may also include creating a least median of squares algorithm within the sensor-based framework. The *least median of squares* estimator fits a line to a set of points by minimizing the median squared distance between any point and the line. This estimator is robust for any data set containing up to 50 percent outliers [37].

The framework discussed in my preliminary work does not allow for tracking of individual points through the system; for example, the framework could not be used to determine which point travelled the farthest distance. This is a side-effect of the choice of discrete monitoring and

the efforts towards practicality and robustness. While keeping these constraints in mind, it would be interesting to consider a tracking model that allows analysis of individual point properties.

In addition, if time permits, it would be interesting to consider an experimental evaluation of one of the algorithms created within the sensor-based framework. One possible data set to examine is the MERL Motion Sensor Dataset which contains over 30 million data points representing people near sensors they placed around their building [33]. Questions to consider would include implementation details and their impact on the practicality of the framework, an efficiency comparison between use of the sensor-based framework and the current heuristic methods, and an efficiency comparison between the sensor-based framework and KDS if possible based on the provided data. I suspect that a lossy compression algorithm instead of the lossless compression algorithm given by the preliminary work would greatly increase the efficiency of algorithms based on this framework, so lossy compression algorithms will also be considered and analyzed experimentally.

References

[1] M. A. Abam, M. de Berg, et al. A simple and efficient kinetic spanner. In *24th Annual Symposium on Computational Geometry*. 2008.

[2] P. K. Agarwal, J. Gao, et al. Kinetic medians and kd-trees. In *Proceedings of the 10th Annual European Symposium on Algorithms*. 2002.

[3] P. K. Agarwal, L. J. Guibas, et al. Algorithmic issues in modeling motion. *ACM Computing Surveys*, 34:550–572, December 2002.

[4] I. Akyildiz, W. Su, et al. Wireless sensor networks: a survey. In *Computer Networks*, pp. 393–422. 2002.

[5] A. Arasu, B. Babcock, et al. Characterizing memory requirements for queries over continuous data streams. In *Proc. of the 2002 ACM Symp. on Principles of Database Systems*. June 2002.

[6] M. J. Atallah. Some dynamic computational geometry poblems. In *Comput. Math. Appl*, vol. 11(12), pp. 1171–1181. 1985.

[7] B. Babcock, S. Babu, et al. Models and issues in data stream systems. In *Symposium on Principles of Database Systems*, pp. 1–16. 2002.

[8] J. Basch, L. J. Guibas, et al. Data structures for mobile data. In *ACM-SIAM Symposium on Discrete Algorithms*. 1997.

[9] —. Proximity problems on moving points. In *Proceedings of the 13th Annual ACM symposium on Computational Geometry*, pp. 344–351. 1997.

[10] M. Bern and D. Eppstein. Approximation algorithms for geometric problems. In D. S. Hochbaum, ed., *Approximation Algorithms for NP-hard Problems*, chap. 8. PWS publishing co., Boston, 1997.

[11] S. Bespamyatnikh, B. Bhattacharya, et al. Mobile facility location. In *Fourth International ACM Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*. 2000.

[12] D. Brunello, G. Calvagno, et al. Lossless compression of video using temporal information. In *IEEE Trans. on Image Processing*, pp. 132–139. Feb 2003.

[13] C. H. Chen, L. F. Pau, et al., eds. *The Handbook of Pattern Recognition & Computer Vision*. World Scientific, 2nd ed., 1999.

[14] T. H. Cormen, C. E. Leiserson, et al. *Introduction to Algorithms*. MIT Press, 2001.

[15] G. Cormode, S. Muthukrishnan, et al. Conquering the divide: Continuous clustering of distributed data streams. In *IEEE 23rd International Conference on Data Engineering*, pp. 1036–1045. 2007.

[16] —. Algorithms for distributed functional monitoring. In *Proc. of the 19th ACM-SIAM Symposium on Discrete Algorithms*, pp. 1076–1085. 2008.

[17] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-IEEE, second ed., 2006.

[18] S. A. Friedler and D. M. Mount. Approximation algorithm for the kinetic robust k-center. Manuscript submitted for publication, 2008.

[19] —. A sensor-based framework for kinetic data. Manuscript submitted for publication, 2008.

[20] D. L. Gall. Mpeg: A video compression standard for multimedia applications. In *Communications of the ACM*, vol. 34, pp. 46–58. 1991.

[21] J. Gao, L. J. Guibas, et al. Deformable spanners and applications. In *Computational Geometry: Theory and Applications*. 2006.

[22] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. In *Theoretical Computer Science*, vol. 38, pp. 293–306. 1985.

[23] L. Guibas. Kinetic data structures. In D. Mehta and S. Sahni, eds., *Handbook of Data Structures and Applications*, pp. 23–1–23–18. Chapman and Hall/CRC, 2004.

[24] L. Guibas, K. Ramshaw, et al. A kinetic framework for computational geometry. In *Proc. of Foundations of Computer Science*, pp. 100–111. 1983.

[25] L. J. Guibas. Kinetic data structures: A state of the art report. In *Proc. 3rd Workshop on the algorithmic foundations of robotics*, pp. 191–209. 1998.

[26] P. Gupta, R. Janardan, et al. Fast algorithms for collision and proximity problems involving moving geometric objects. In *Comput. Geom. Theory Appl*, vol. 6, pp. 371–391. 1996.

[27] F. R. Hampel. A general qualitative definition of robustness. *The Annals of Mathematical Statistics*, 42(6):1887–1896, Dec 1971.

[28] S. Har-Peled. Clustering motion. *Discrete Comput. Geom.*, 31(4):545–565, 2004.

[29] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40, Sept. 1952.

[30] S. Kahan. A model for data in motion. In *STOC '91: Proc. of the 23rd ACM Symp. on Theory of Computing*, pp. 265–277. 1991.

[31] O. Kariv and S. Hakimi. An algorithmic approach to network location problems. i: The p-centers. In *SIAM Journal on Appl. Math.*, vol. 37, pp. 513–538. 1979.

[32] R. Krauthgamer and J. R. Lee. Navigating nets: Simple algorithms for proximity search. In *Symposium on Discrete Algorithms*. 2004.

[33] MERL. Merl motion sensor dataset. `http://www.merl.com/wmd/details.html`.

[34] MIT Media Lab. The owl project. `http://owlproject.media.mit.edu/`.

[35] J. J. Monaghan. Smoothed particle hydrodynamics. In *Reports on Progress in Physics*, vol. 68, pp. 1703–1759. 2005.

[36] J. Rissanen. Generalized kraft inequality and arithmetic coding. *IBM Journal of Research and Development*, 20, 1976.

[37] P. J. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79(388), 1984.

[38] N. Saunier and T. Sayed. Automated analysis of road safety with video data. In *Transportation Research Record*, pp. 57–64. 2007.

[39] E. Schomer and C. Thiel. Efficient collision detection for moving polyhedra. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pp. 51–60. 1995.

[40] —. Subquadratic algorithms for the general collision detection problem. In *Abstracts 12th European Workshop Comput. Geom*, pp. 95–101. 1996.

[41] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948.

[42] M. Sharir. Almost tight upper bounds for lower envelops in higher dimensions. *Discrete and Computational Geometry*, 12(1):327–345, Dec 1994.

[43] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, 1995.

[44] A. Singhal. Modern information retrieval: A brief overview. In *IEEE Data Engineering Bulletin 24*, vol. 4, pp. 35–43. 2001.

[45] TeleAtlas. Dynamic content - real time traffic information. `http://www.teleatlas.com/OurProducts/MapEnhancementProducts/DynamicContent/index.htm`.

[46] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*, chap. 6.2:

Clusterability. Cambridge University Press, 1994.

[47] O. Wolfson. Moving objects information management: The database challenge (vision paper). In *Proceedings of the 5th International Workshop on Next Generation Information Technologies and Systems*. 2002.

[48] A. D. Wyner and J. Ziv. The sliding-window lempel-ziv algorithm is asymptotically optimal. In *Proceedings of the IEEE*, pp. 872–877. Jun 1994.

[49] K. Yi and Q. Zhang. Multi-dimensional online tracking. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*. 2009.

[50] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Trans. on Information Theory*, 24(5):530–536, 1978.