DIMACS Technical Report 2003-11

Radial Histograms for Spatial Streams

by

Graham Cormode [1,2]   S. Muthukrishnan [3,4]

[1]DIMACS Postdoctoral Fellow. Supported in part by DIMACS through grant no. NSF EIA 02-05116
[2]Center for Discrete Mathematics and Computer Science (DIMACS), Rutgers University, Piscataway NJ, graham@dimacs.rutgers.edu, Tel: 732-445-4580, Fax: 732-445-5932
[3]DIMACS Permananent Member
[4]Division of Computer Science, Rutgers University, Piscataway NJ, muthu@cs.rutgers.edu. Also, AT&T Research, Florham Park, NJ 07932.

ABSTRACT

Many data streams relate to geographic or spatial information such as the tracking of moving objects, or location-specific measurements and queries. While several techniques have been developed recently for processing numerical, text or XML streams, new techniques are needed to process spatial queries on streaming geometric data.

We propose a novel data structure, the *Radial Histogram*, to process streams including spatial data points. It allows a number of geometric aggregates involving the spread and extent of the points in the data streams—diameter, furthest neighbors, convex hulls—to be accurately estimated to arbitrary precision. By using multiple Radial Histograms for a set of given facilities, we can process data streams consisting of massive numbers of client points. We can then accurately estimate number of spatial aggregates involving relative placement of clients with respect to the facilities, including reverse nearest neighbors, spatial joins and more.

Radial histograms use very small space and are exceedingly simple to implement. Nevertheless, we prove that they guarantee accurate estimation for spatial aggregate queries mentioned above. An experimental evaluation shows them to perform very well in practice.

1 Introduction

There has been great interest recently in the *data stream* phenomenon, where the quantities of data that are generated are truly massive, and beyond our capabilities to store, or compute on as we would with traditional methods. One sign of the significance of this phenomenon is the number of tutorials and invited talks on the matter in recent conferences: see for example [2, 12, 26, 28]. The Stanford STREAM group maintain a webpage of resources with many useful pointers for this emerging area [36].

The sources of data streams are many and various. The data source that has drawn a great deal of attention is network flows. IP networks generate logs of packet, flow and other traffic in the network at high rates and network monitoring involves managing and mining these data streams. There has been considerable study of this application including identification of suitable queries, study of the fundamental aspects, and demos [10, 13, 31, 7, 3].

In all of these applications, the input stream comprises a series of numerical values. The data can be represented as a vector on the underyling domain with each data stream item updating this underlying vector; data stream analyses become computation on that vector [14]. Other data streams that have been studied include text streams. A simple example is one of web traffic logs where web addrresses are text fields. More elaborately, email, fax and other documents collected at distributed sites is an instance of maasive text streams. [25]. There has also been study of streams which arrive in the form of XML documents, and bring associated challenges (e.g., [34] and many others). In many of these applications, the data cannot be represented simply as a vector; instead, the data is drawn from a wide range of domains: words and phrases, XML tags and paths, and so on.

We pursue our premise that many streams of interest will include spatial data in one form or another, and that systems to support spatial data streams are a vital part of the ongoing data stream research. We describe in brief some examples where spatial data streams arise, and must be processed in a way that takes into account the spatial nature of the data.

1. **Biosensor Measurements.** Consider a large scale distributed system of sensors around a city, monitoring air quality or measuring bioagents. Then each reading has not only a value, but also a location. Any system to process the readings must take into account the location where they were obtained (and any additional spatial data available, such as direction or distance associated with the reading). Such sensor networks for biosensor readings are currently of great interest for homeland security reasons: multiple biosensors per home (much as fire alarms), mobile biosensors (which can be small to large, single purpose and cheap to multipurpose and expensive). Analyses of data gathered from such measurements involves tracking the evolution of extent of points that indicate an attack or adverse event. Also, given a number of possible positions to locate high resolution sensors, where best to place them to ensure coverage of specified targets, based on profile of low resolution sensor readings? [6, 23]

2. **Traffic and Herd Monitoring.** Other sensor monitoring applications arise in the form of traffic monitoring [24]. This can be traffic in the sense of road or other vehicles, but also in the location monitoring of products and personnel [20]. For example, consider data mining

wireless calling information. Each cell tower handling a call also receives location information about the phone user: signal strength and direction allow the appoximate location of the user to be found. Knowing and understanding the usage distribution of users helps determine where additional capacity is needed, to detect where problems may be occurring in the network, provision of location based services, and so on. Similar streams arise from tracking herds (of animals or users), or even groups of soldiers. We typically want to track the distribution and density of the herd, as well as more sophisticated queries relating to events of interest: finding if any herd member has strayed from the group, if there is a build up (eg of military personnel), if there is impending danger of a "friendly fire" incident [36].

3. **Scientific Data Streams.** Recently, the National Academy of Sciences held a meeting on "Statistical analysis of massive data streams" [30] where scientists spoke extensively about data streams from satellite and telescope-derived observations. There are geograpically dispersed observations of earth and atmospheric phenomena over time continuously monitored by various space-filling trajectories of satellites and telescopes to a very fine resolution [19]. The goal of data analyses is meteorological and climatological prediction, detecting variations and trends, tracking emissions and pollution. These then lead to other questions: what is the spatial extent affected by these events, how they evolve, which areas to engage additional detectors in order to refine the predictions, etc.

Although there may be a very large number of data points, we can usually afford to make some sacrifices in terms of delivering exact answers. Instead of knowing that the distance between a point and a facility is exactly 97.3 miles away, we might be content to know that the distance is between 97 and 98 miles. Indeed, in many cases distances are not known to that degree of accuracy anyway. If there is already a small error in the location, then storing this value exactly is unnecessary[1]. So we advocate using approximation to reduce the requirements of data storage, but in a principled way so that a relative error guarantee can be made.

Our contributions are in processing large scale spatial data streams. Specifically:

1. We propose a novel, simple indexing method for handling spatial data streams, called *Radial Histograms*. It allows a number of geometric aggregates involving the spread and extent of the points in the data streams—diameter, furthest neighbors, convex hulls—to be accurately estimated to arbitrary precision. Radial histograms use very small space and are exceedingly simple to implement. Nevertheless, we prove that they guarantee accurate estimation for spatial aggregate queries mentioned above.

2. We propose using multiple Radial Histograms for a set of given facilities to allow processing data streams consisting of massive numbers of client points. We can then accurately estimate number of spatial aggregates involving relative placement of clients with respect to the facilities, including reverse nearest neighbors, spatial joins and more.

[1]Even GPS systems are only accurate to within a few meters, which can be significant at the granularity of streets. Location based on wireless phone signals is even less accurate.

3. Using real and synthetic dataset we experimentally evaluate single and multiple radial histograms for providing variety of spatial aggregates, and show them to perform very well in practice.

This is the first work to address computing of spatial aggregates in a unified manner. We expect that radial histograms will find many other uses as further investigation of spatial data is made. An important feature of our methods are that they do not depend on the quantity of data, but rather on a single user definable parameter, ϵ, which defines how well the input data is approximated.

1.1 Outline

The rest of the paper is organized as follows. We present an overview of spatial aggregate problems in Section 2: they come in two flavors that we shall define, the monochromatic or bichromatic versions. Radial Histograms are defined in Section 3, and applications to approximating monochromatic spatial aggregates on extent and proximity — diameter, convex hull, furthest neighhbor — are given in Section 4. Section 5 shows how keeping multiple radial histograms — one for each facility — can provide bichromatic spatial aggregates on distribution of clients with respect to the facilities such as Reverse Nearest Neighbor Aggregates, and Approximate Spatial Join Sizes. Section 5 reports on a set of experiments to verify the quality of single and multiple radial histograms, and Section 6 discusses some possible extensions. Conclusions are given in Section 7.

2 Preliminaries

We will introduce the spatial aggregates that we are able to approximate efficiently by using radial histograms. We then go on to discuss related work on these aggregates and on streams. Throughout, we shall write $|pq|$ to indicate the distance between two points p and q. We take this to be the Euclidean (shortest line) distance, in later sections will shall discuss variations based on other distances.

2.1 Monochromatic Spatial Aggregates

We first study *monochromatic spatial aggregates*. These are monochromatic in the sense that all the data points are treated uniformly; we shall later describe bichromatic situations where certain points are special. We are given a large number of points in the form of a stream — a sequence of points to process, to which we have linear access (we cannot backtrack on the stream but must process each point as it is encountered). The scenarios we consider maybe *inserts only* (we see only arrivals of new points), *insert and delete* (new points arrive, old points depart), or *mobile* (points arrive and move, which we may model as as deletion followed by an insertion at its new location). In the scenarios we consider, the number of points is too great to store all of them explicity, instead we must process them to store only a small amount of information about the point set. Spatial aggregates are functions which give important information about the extent and location of the points. We consider the follow examples of spatial aggregate queries.

- The DIAMETER of a point set P is the maximum distance between any pair of points. Formally the diameter, d is the maximum $|pq|$ over all points p and q from the set S. This gives a very natural measure of spread of the points. It also lets us track the distance spanned by mobile clients.

- The CONVEX HULL problem is to find the smallest convex polygon that contains the whole point set. It is the area analog of the diameter. Again, it gives an indication of the spread of all points, and tells us what area needs to be convered in order to reach every point. For biosensor and other detection applications, the convex hull of a set of alerted sensors gives an idea of the spread of the outbreak, the size of an area to be quarantined, and so on.

- The scenario for FURTHEST NEIGHBOR queries is that, interleaved with the sequence of points, are queries in the form of a new point: the goal is to return a point from the data set whose distance from the query point is maximized. That is, for a query point q, find the point p from P that maximizes $|pq|$. Again, this relates to the spread of the data points, but in a much more general way. In "herd" tracking applications, the furthest neighbor of a point is the herd member at greatest distance, important to know if we must reach every member of the herd. If we are locating a base station or tracking center, then the furthest neighbor of a point gives the greatest distance that it is necessary to handle, as well as a sample client that achieves that distance.

2.2 Bichromatic Spatial Aggregates

Bichromatic spatial aggregates are those that involve two classes of points, that we refer to here as clients and facilities. They are bichromatic in the sense that we imagine client points to be colored with one color, facilities with another, and we are interested predominantly in measurements involving pairs of points of differing colors.

We consider the following class of problems: we have a small number, n, of important points which we call facilities, F (such as hospitals, cell towers, biosensors or coffee shops), and a much greater number of m points which are clients for these facilities. The location of the facilities will be provided to us at the start, but the clients will arrive and depart in large numbers, and these changes will be provided in the form of an update stream. While managing information about the facilities will be practical, the number of clients is likely to be too great to deal with individually in memory or even in external memory. Even if the number of clients is managable, our methods use small space and are fast, so using these approximation may be preferable to lengthy exact computations.

We will describe a number of fundamental aggregate query types on bichromatic spatial data.

- CLOSEST FACILITY(d, d') queries are queries of the form "How many clients are within distance d of their closest (active) facility" and "How many clients are further than distance d' of their closest (active) facility", or more generally, "how many clients are between d and d' of their closest facility".[2] A CLOSE FACILITY(d, f) query generalizes closest facility queries, and asks, "how many clients have (more than / less than) f facilities within distance d".

[2] d may be zero, and d' may be ∞.

- The Reverse Nearest Neighbors of a facility are all those clients for which it is their nearest facility. This is a very natural and general concept, which has applications to numerous areas, in terms of traffic monitoring (which cell tower will clients use), and herd monitoring (which is the nearest base camp for military personnel). We will typically be interested in such questions as: how many reverse nearest neighbors does a particular facility have; what is the greatest distance between a facility and its reverse nearest neighbors for a particular facility or over all facilities; what is the effect if some of the facilities go offline or are closed; and so on. We will use Radial Histograms in order to compute REVERSE NEAREST NEIGHBOR AGGREGATES.

  A particularly important question in this dynamic scenario is to know, at any time, what is the greatest distance between any client and its nearest facility. This determines, for example, emergency vehicle response times, quality of cellphone or biosensor coverage, or how far customers have to travel to their nearest coffee shop.

- We consider spatial joins with another set of g facilities, G in order to compute APPROXIMATE SPATIAL JOIN SIZE. The aim of the join is to find clients which satisfy predicates such as "client is (closer than / further than) d from their closest facility in F and (closer than / further than) d' from their closest facility in G".[3] We may also compute joins to find clients which are closer to their closest facility in F than their closest in G, or more generally, they are c times closer to their closest facility in S than their closest in G, for any constant $c > 1$. We are interested in computing the size of these spatial join sizes, that is, the number of clients which satisfy this predicate.

In general, it is possible to generalize further and discuss trichromatic, quadchromatic and other polychromatic query times involving multiple kinds of facility type. These are more application specific, and are not as intuitive as the monochromatic and bichromatic aggregates we discuss here. It is possible to use radial histograms in these scenarios, but we do not discuss this extension.

2.3 Related Work

Traditionally, many data structures have been employed to answer the kinds of queries we address. These include k-d trees [4], Voronoi diagrams [33] and R-trees [15]. These all have the common property that they store each data point explicitly, and so for n client points, they require $\Omega(n)$ space. With the massive size of data streams, we are presented with the situation that this space is too great to deal with. In order to be manageable, we must look for solutions whose dependence on n is *sublinear*, preferably close to logarithmic or constant.

For high dimensions, many recent papers have broken the "curse of dimensionality", and produced solutions which are sub-exponential in the dimension [22, 17]. Here, we are addressing a different problem: the number of dimensions is low (usually two), but the number of data points is large. These theoretical solutions typically still use an amount of space that depends at least linearly on the number of data points. Even in cases where the space usage is small [16], our solutions

[3] There is a technichal requirement of using Radial Histograms that $d < d'$

are simpler and more efficient, by virtue of working in low-dimensional space. One of the ways in which progress has been made on geometric problems in high dimensions is by incorporating approximation: finding points whose distance approximates the desired quantity [17, 22, 16]. We apply this paradigm to our problems of interest to yield simpler and more efficient solutions in low dimensions.

Data stream results on spatial data are sparse. The work in [20] proposes exponential histograms in one dimension for maintaining statistical on the distribution of points on a straight line. Likewise, the work on windowed streams [8] proposes exponential histograms on one dimensions, be it value, count or sketch on an attribute. Exponential histograms have bucket boundaries exponentially separated from each other along one dimension. Radial histograms may be thought of as their multidimensional generalizations, but exploiting them for the variety of spatial aggregates we do is more involved and has to make use of geometry more intricately.

There is a recent result on estimating diameter of point sets in two dimensions [11], which uses a method that is similar in spirit to ours. However, their focus is on the windowed streaming model, where this problem presents nontrivial difficulties. Radial histograms can be used to provide diameter estimates quite easily, but we are able to use them for providing a large variety of sophisticated spatial aggregates. The data structure underlying Radial Histograms is simple, and has an intuitive feel to it. As such, it is reminiscent of the log-polar co-ordinates made use of in Computer Vision [39], and structures used in Computational Geometry [9]. However, we believe the application to spatial data streams and the kinds of queries we analyze is novel.

There is large literature on histograms in general, and multidimensional histograms in particular, where the emphasis is on partitioning or covering a multidimensional area with rectangles, either regular spaced or spaced carefully to optimize various error metrics [37, 29, 27, 32]. While these are suitable for approximating multidimensional data distributions and hence for selectivity estimation, they are not optimized for spatial aggregate queries and as such do not provide any guarantees for important spatial aggregates we consider here. Multidimensional histograms for spatial data was proposed in [1], but they focused on selectivity estimation involving geometric objects such as lines and rectangles, and do not provide any provable guarantees on spatial aggregates of our interest. There is a large literature on space decompositions; radial histograms are a particular type of space decomposition but designed to be prove bounds on spatial aggregates we study.

Regarding the spatial aggregates we compute, the concept of Reverse Nearest Neighbors was introduced by Korn and Muthukrishnan [21]. The one-dimensional version of the REVERSE NEAREST NEIGHBORS AGGREGATE computation problem was considered recently in [20]. Here, we also consider more general spatial queries, and importantly,

solve the problem in two dimensions which was left open there.

3 Radial Histograms

We begin by defining the data structure we introduce here, of *radial histograms*.

Definition 1 *A* Radial Histogram *is a partition of the plane defined by a point, the center of the histogram, and a parameter* $\epsilon \leq 1$*. The plane is divided by lines extending outwards from the*

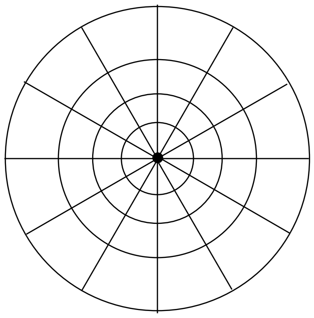

Figure 1: The spatial structure of a radial histogram

center ("spokes"), and by concentric rings around the center with exponentially increasing radii. The radius of the i*'th ring is* $r_i = (1 + \epsilon/2)^i$ *(for integer* i*). The angle of the* j*'th spoke is* $\theta_j = j\epsilon$*.*

This structure divides space up into a number of 'cells', so that any point in the plane falls into one of these cells. The spatial structure of a radial histogram is shown in Figure 1. We can give some facts about the partitioning of space induced by this data structure.

Fact 1 *There are* $2\pi/\epsilon$ *spokes. Each cell of the histogram is defined by successive two spokes,* θ_j *and* θ_{j+1}*, and two successive rings,* r_i *and* r_{i+1}*.*

Fact 2 *We can find which cell any point* $p = (x, y)$ *lies in using geometry: If we translate points so that location of the center is the origin, then*

$$j = \lfloor 2 \tan^{-1}(x/y)/\epsilon \rfloor$$
$$i = \lfloor \ln(x^2 + y^2)/2 \ln(1 + \epsilon/2) \rfloor$$

Fact 3 *The greatest distance between two points which lie in the same cell defined by* r_i *and* θ_j *is* $\frac{3}{2}\epsilon r_i$*. This is seen by observing that the greatest distance within a cell is that between opposing vertices, which is less than* $r_{i+1} - r_i + r_i(\theta_{j+1} - \theta_j) = (1 + \epsilon/2)r_i - r_i + r_i((j + 1)\epsilon - j\epsilon) = \frac{3}{2}\epsilon r_i$*.*

This last fact is crucial to our use of radial hisograms: it means that although we may approximate the location of points by placing them within cells, the quality of the approximation relative to their distance from the center will be guaranteed.

4 Radial Histograms for Monochromatic Aggregates

We can use radial histograms to let us estimate spatial aggregates on the extent and proximity of massive point sets, while storing only a small amount of data. In fact, the storage space used is essentially independent of the number of points, and depends only on the parameter ϵ. We focus on the inserts only scenario. In Section 6 we discuss how Radial Histograms can be extended to handle insertions and deletions, and mobile points. Throughout, we use a single update procedure to insert new points into the data structure.

Update Procedure

- Initialize $max[1 \dots 2\pi/\epsilon] = \mathbf{0}$.
- Locate the center of the radial histogram at the first point that arrives.
- For each new point p encountered, compute which spokes of the radial histogram the point belongs between (ie compute θ_j).
- If the new point is further from the center than $max[j]$, then update $max[j] = p$.

4.1 Convex Hull

We can use the point set stored by the radial histogram to find an approximate solution to CONVEX HULL for the point set. We simply compute the convex hull of the maximal points stored. Since these points all occurred, then this is clearly a lower bound on the convex hull. It is approximate in the following sense: any point which should have been included on the convex hull is a bounded distance away from it.

Theorem 1 *Using a radial histogram, we can compute an approximation of* CONVEX HULL *for a stream of points with space $O(1/\epsilon)$. No point in the input stream is more than ϵd outside the approximate convex hull, where d is the diameter of the point set. Processing each input point takes time $O(1)$, and constructing the approximate convex hull takes time $O(1/\epsilon)$.*

Proof: Consider some point, p, which should have been on or within the convex hull, but is not contained within it. Then, this point is as close as or closer to the center than some other point in the same sector (else, it is maximal, and should be included on the hull). The radial line segement which links the center to the maximal point in that sector is entirely contained within the convex hull, and has length r. The distance from p to the convex hull is at most the distance from p to this line, which is at most $r\epsilon$. r is a lower bound on the diameter of the point set, so any point outside the approximate convex hull is a distance of at most ϵ times the diameter of the point set away from the approximate convex hull. ∎

The time to compute the convex hull is $O(1/\epsilon)$: starting from the point furthest from the center, walk around the sectors in order of increasing angle. Using the standard Graham scan algorithm will return the desired hull, since the points are encountered in the order in which they should occur on the hull. This result is similar to that of [5], in that it bounds the distance of any point that falls outside the approximate hull; it differs by dividing space radially instead of linearly, and requires a single pass over the data instead of two.

4.2 Diameter

We use the same update procedure to Radial Histograms in order to approximate DIAMETER of a point set. We use the algorithm of Shamos [35] for computing the diameter of a convex point set by first computing the convex hull of the points using the above method, in time $O(1/\epsilon)$. Suppose the algorithm computes an approximation to the diameter, $\hat{d}$, based on the convex hull. Then we know the true diameter, d, is a lower bound on $\hat{d}$, since the approximation is the distance between two points that we observed. Suppose that the two points which achieve the diameter were not stored by the algorithm. The maximum distance between them is $\epsilon d + \hat{d} + \epsilon d \leq (1+2\epsilon)d$. Hence $d \leq \hat{d} \leq (1+2\epsilon)d$. Formally, our result is as follows.

Theorem 2 *We can compute an approximation of* DIAMETER *of a stream of points using $O(1/\epsilon)$ space. The result $\hat{d}$ is bounded by the diameter d as $d \leq \hat{d} \leq (1+\epsilon)d$. The time to compute the diameter is $O(1/\epsilon)$.*

4.3 Furthest Neighbors

We again update the Radial Histogram using the above update procedure on the arrival of new points. For queries, we use the following query procedure.

Query Procedure

- Given a query point q, return the point from $max[]$ that maximises the distance from q to itself.

We claim that this procedure returns a point which is within a factor of $(1+\epsilon)$ of the furthest neighbor.

Theorem 3 *Radial Histograms can be used to answer* FURTHEST NEIGHBOR *queries approximately, with $O(1)$ update time and $O(1/\epsilon)$ query time. If the set of all points is P, then we guarantee to return a point p whose distance to the query point q guarantees that for all other points p' from P,*

$$(1+\epsilon)|pq| \geq |p'q|$$

Proof: We imagine all points that fall in any cell to be shifted in the angular direction so that they fall on one of the spokes. This does not affect the distance of any p from the center $|cp|$, and affects their position by less than $\epsilon|cp|$. We split the data points into two parts by drawing a line from q to the center, and then drawing the tangent to this line through the center, c. This divides the plane into two parts, which we shall refer to as the near side, and the far side. Consider any point p in the far side: it is clear that points which maximize their distance from c also maximize their distance from q. Also for these points $|qp| \geq |cp|$. By shifting them onto spokes, we increased their distance by at most $\epsilon|cp| \leq \epsilon|qp|$. Suppose p and p' are points on the far side which both fall on the same line from the center. Then only the one which maximises its distance from c need be considered as a possible furthest neighbor of q. Analyzing the points on the far side gives us a candidate furthest neighbor for q (note that the center itself is always a candidate furthest neighbor if there are no points on the far side).

We next consider the near side. Since we have a candidate point whose distance is at least $|cq|$, we know any furthest neighbor must be at least this far away. We now claim that any furthest neighbor p on the near side must maximize its distance from c. Consider a point p' such that moving it

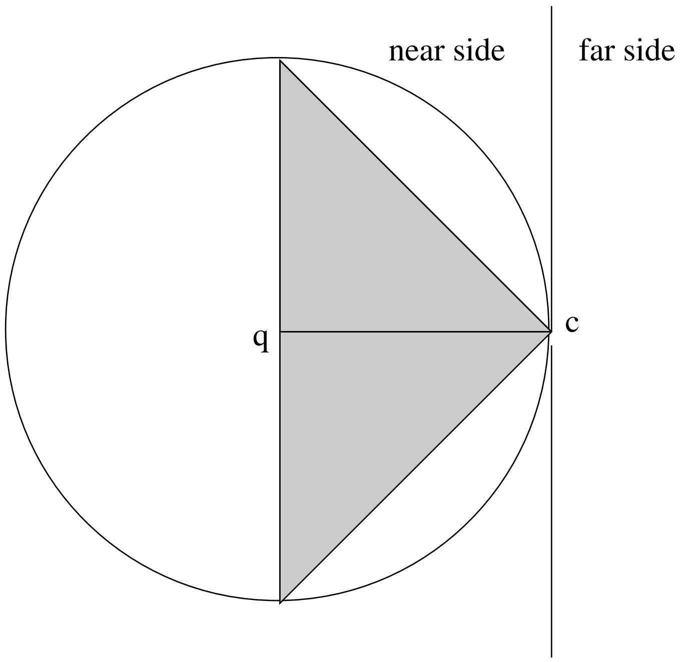

Figure 2: All furthest neighbors will maximise their distance from the center: only points in the grey triangle get closer to q as they move on straight line from c. But these cannot be furthest neighbors of q, since c is as far or further.

radially further from c decreases its distance to q. Then the angle qcp' must be less than $\pi/4$, and additionally, the projection of p onto the line qc must be between q and c. This defines a triangle of possible points, defined by c, and the two points on the line perpendicular to cq passing through q at distance $|cq|$ from q. However, all points in this triangle are at distance at most $|cq|$ from q, and we know that the furthest neighbor is at a greater distance. This is illustrated in Figure 2. So again, we only need to consider points which maximize their distance from the center, and find which of these is furthest from q. Overall, then, we only need to consider the point in each sector which maximizes its distance from c, and compute its distance from q. This gives a point which is guaranteed to be at approximately the maximum distance. We just need to investigate each sector of the radial histogram, and there are $2\pi/\epsilon = O(1/\epsilon)$ of these. To insert new points in the data structure, then determining which sector it belongs in, and whether it exceeds the current max of that sector can clearly be done in constant time. ∎

The space used by this structure is very small, and is independent of the number of data points or their distribution. We need to store one point for each sector, of which there are $O(1/\epsilon)$, as well as the center itself.

4.4 Choosing the Center

An important issue is where to locate the center of a radial histogram. From a theoretical perspective, any point from the set of points is a good choice for the location of the center: the (worst case) bounds that we showed hold whenever the center is contained within the convex hull of the point set, which is met by any data point. In practice, we often have a strong idea of the domain (for example, a city), and choosing the center in any sensible way will suffice.

In our experiments, we considered several alternate strategies for choosing the center: picking at random from the pointset, the center of gravity of a small sample, and contrasted these with locating the center at an extreme location (at the smallest or greatest value of a co-ordinate). We observed only small variation between these settings: in all cases, the answer found was much more accurate than that guaranteed by the worst case bounds. Experimental results are shown in Section 6. In next section, we will focus on principled choice of multiple centers, which is more suitable for bichromatic data sets.

5 Multiple Radial Histograms for Bichromatic Spatial Aggregates

In the bichromatic case, described in Section 2.2, we are given a number of facilities, and a stream of client arrivals and departures to process. We shall use a single algorithm for inserting or deleting clients as they arrive and depart to answer all of these problems.

5.1 Insertion and Deletion of Clients

Client arrivals and departures can be dealt with in near constant time to update the data structure.

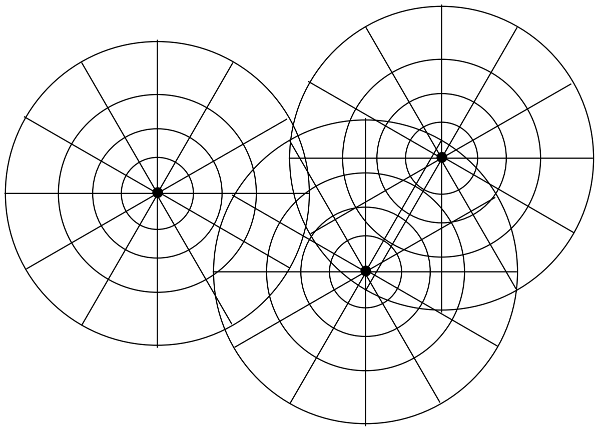

Figure 3: Multiple Radial Histograms

- Initialize a radial histogram for each facility in the set of facilities, F.
- Compute a data structure that allows nearest-neighbor queries to be answered for the facilities (eg Voronoi Diagram or k-d tree)
- For each client arrival or departure:
  - Find which facility it is closest to.
  - Find which cell of that facility's radial histogram it belongs in
  - Increment or decrement the count of that cell.

Fact 4 *Inserting or deleting a client point takes time O(1 + t), where t is the time required to determine which of the n facilities the client is closest to. Typically, t is $O(\log n)$, for example, using Voronoi diagrams [33].*

An example arrangement of three facilities and their associated Radial Histograms is shown in Figure 3. Mobile clients can also be handled by simple computations: when a client leaves one cell and enters another then decrement the count of the old cell and increment the count of the new one.

5.2 Reverse Nearest Neighbor Aggregates

Now, we represent all points in the same cell (i, j) with a single point $a = ((1 + \epsilon/2)^i, \epsilon(j + \frac{1}{2}))$ in polar coordinates, and a count of the number of points falling in that cell. We call a the *representative point* of the cell, and we shall use it as a surrogate for all the points that fall in that cell. This is illustrated in Figure 4. The following fact assures us that this does not alter the position by more than a bounded quantity:

Fact 5 *For any point p in a cell, $|ap| \leq \epsilon |as|$*

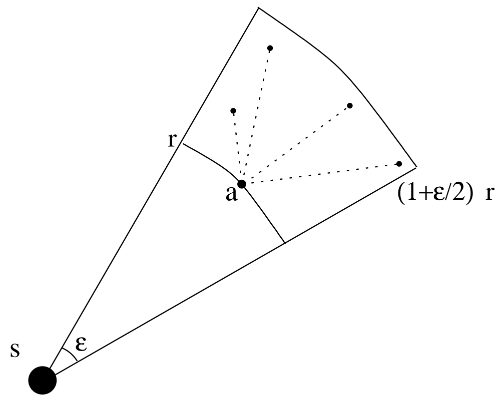

Figure 4: Contents of a cell

Query procedure

The query is a set of active facilities, AF, which is a subset of the set of facilities, F. The goal is to return the maximum distance over all clients of the distance between a client and its closest active facility. We describe a procedure to return a $1 \pm \epsilon$ approximation to this distance. The procedure in full is:

- Build a Voronoi diagram for the Active Facilities AF
- For each non-empty cell, compute its representive point a.
- Find f_a, the closest facility in AF to a
- Return $\hat{r} = \max_a |af_a|$.

In general, for all queries in this section, the procedure is similar: every point a non-empty cell is treated as if it was located at a, and is then processed accordingly.

Theorem 4 *Let $r = \max_{p \in P} \min_{f \in AF} |ps|$. The above procedure returns $\hat{r}$ such that $(1-\epsilon)r \leq \hat{r} \leq (1+\epsilon)r$*

Proof: Let p be a point in a cell belonging to facility f, and let f_p be its closest active facility. Let a be the representative point in that cell, and let its closest facility be f_a. Then:

$$|pf_p| \leq |pf_a| \leq |pa| + |af_a| \tag{1}$$
$$|af_a| \leq |af_p| \leq |ap| + |pf_p| \tag{2}$$

Using Fact 5 allows us to conclude that $(1-\epsilon)|af_a| \leq |pf_p| \leq (1+\epsilon)|af_a|$.

Note that even if f is active, it is not necessarily the case that $f_a = f$. However, if f is active, then $f_p = f$, and $|af| \leq |pf|$. So $|af| \leq |pf_p| = |pf| \leq (1+\epsilon)|fa|$.

The consequence of the above is that $|af_a|$ is a good approximation for $\min_{f \in AF} |pf|$ for all the points in the cell. By taking the maximum of $|af_a|$ over all a, then we find a good approximation for r. ∎

Space and Time Cost

The amount of space required depends on the maximum radius of a cell. Let d_{min} denote the distance between the closest pair of facilities, and let d_{max} denote the greatest achieved between any client and any facility. We will denote the ratio of these two distances as D. Note that for any two facilities, f, f' with a being a representative point of a cell of f, that $|ff'| \leq |fa| + |af'| \leq 2|af'|$. So $\epsilon|af_a| \geq \epsilon|f, f'|/2 \geq \epsilon d_{min}/2$. Therefore, we only need to keep track of points whose distance from their closest facility exceeds $\epsilon d_{min}/2$. The number of cells we need to be sure of catching all points is given by $\epsilon/2d_{min}(1+\epsilon/2)^j \geq d_{max}$, resulting in $j = O(1/\epsilon \ln(D/\epsilon))$. Therefore, the total space required is $2\pi j/\epsilon = O(n/\epsilon^2 \ln D/\epsilon)$. In the worst case, we have to examine the contents of every occupied cell, and do a constant amount of work. The time cost to find the maximum distance for a set of active facilities is $O(n/\epsilon^2 \ln D/\epsilon)$. Additionally, we have to build a nearest neighbor structure (ie Voronoi Diagram) for the facilities, which takes time $O(n \log n)$. Putting this together gives the following theorem.

Theorem 5 *We can use multiple radial histograms to process a stream of client points given a set of n facility locations. Maximal* REVERSE NEAREST NEIGHBOR AGGREGATE *queries of a subset of active facilities can be answered approximately in time $O(n(\log n + \frac{1}{\epsilon^2} \ln D/\epsilon))$. The space used is $O(n/\epsilon^2 \ln D/\epsilon)$, and the time to process each client arrival or departure is $O(1)$. The answer is between 1 and $1+\epsilon$ times the true answer.*

5.3 Spatial Queries and Approximate Spatial Join Sizes

We will consider the bichromatic spatial queries, and spatial joins based on spatial predicates, described in Section 2.2. We have the same situation as before, that we have a number n of facility locations, and a much larger number of clients. We use the procedure given in Section 5.1 to process arrivals and departures of clients. We will discuss a number of query and join types, and show that they can be answered approximately. We now state the results that we are able to obtain using radial histograms. The proofs are mostly similar to those above, and are omitted here for brevity.

Theorem 6 CLOSEST FACILITY(d, d') *queries can be answered approximately in time $O(n/\epsilon^2 \ln d'/d)$ and space $O(n/\epsilon^2 \ln D/\epsilon)$. These are answered approximately, meaning that we will be sure of including every client between $d(1+\epsilon)$ and d', and we shall exclude every client closer than d or further than $d'(1+\epsilon)$.*

Theorem 7 CLOSE FACILITY(d, s) *queries can be answered approximately in time $O(n^2/\epsilon^2 \ln D/\epsilon)$ and space $O(n/\epsilon^2 \ln D/\epsilon)$. Answers are approximate in the sense that facilities within d are included, facilities further than $d(1+\epsilon)$ are excluded, but between d and $d(1+\epsilon)$ facilities may or may not be included.*

Theorem 8 APPROXIMATE SPATIAL JOIN *sizes can be computed in time $O(t \log t + n^2/\epsilon^2 \ln D/\epsilon)$ and space $O(n/\epsilon^2 \ln D/\epsilon)$. We approximate by relaxing distances to a $1+\epsilon$ factor.*

These are just a few examples of the applications of radial histograms. We expect that they can find application for solving many other problems in addition to the ones that we have sketched here.

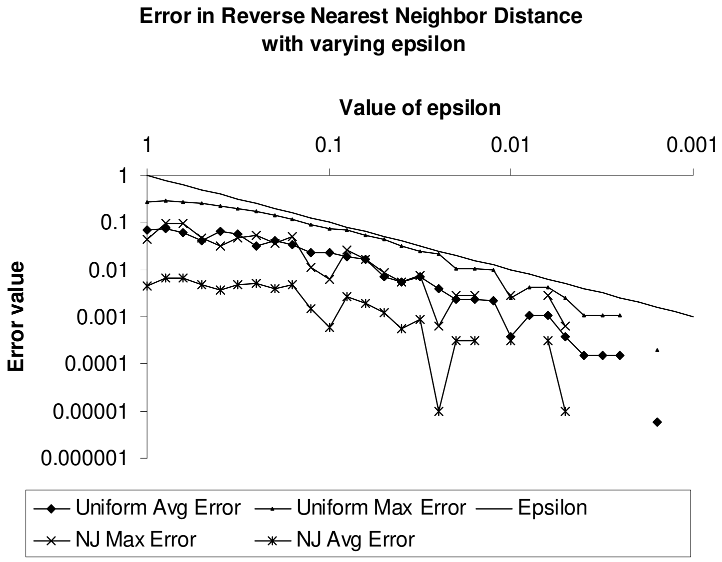

Figure 5: Testing the effect of varying epsilon

6 Experiments

We performed a variety of experiments to evaluate the efficiency and accuracy of radial histograms. We implemented the core methods in C, and examined the effect of the algorithms on two data sets. The first was generated randomly, with each co-ordinate selected uniformly. This allowed large numbers of points to be generated as needed, and turned out to be a good challenge of the methods. We also obtained data points from the publicly available US Census location database, TIGER [38]. This data set is a good representative of how typical human-influenced data sets vary density depending on how developed the area is. We do not include any timing results, since we did not attempt to optimize our code, however we observed that our methods appeared significantly faster than the corresponding exact methods used to evaluate the errors arising from approximation.

6.1 Monochromatic Spatial Aggregates

We tested the methods of Section 4, and in particular, the methods for answering FURTHEST NEIGHBOR queries. Firstly, we examined the effect of varying epsilon. We took a million points generated randomly, and also the points from the New Jersey data set from TIGER (these totalled about six hundred thousand unique points). We processed the data set, and then asked several hundred FURTHEST NEIGHBOR queries from points within the point sets. The results are plotted in Figure 5. We have plotted, for each data set, the average error in the distance (as a fraction of the true distance), and also the maximum error seen. Additionally, we have indicated the bound given by the value of the parameter ϵ.

The results are very encouraging. The *a priori* bound given by ϵ is respected by the results, as

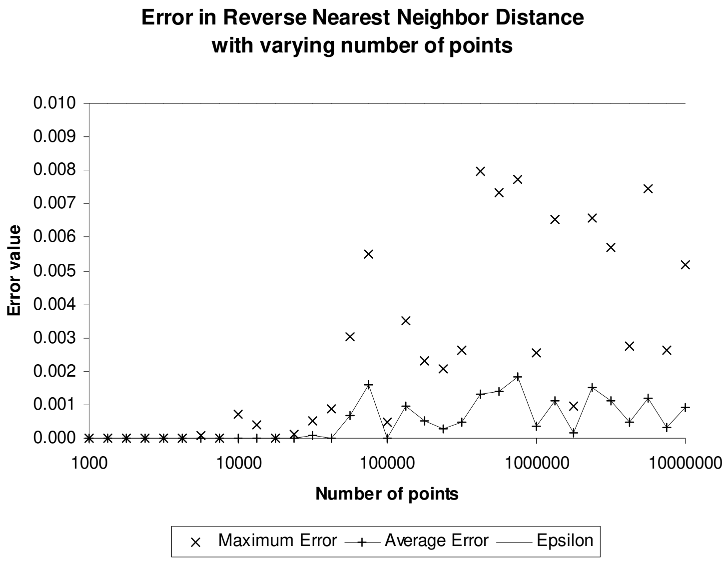

Figure 6: Testing the effect of varying the number of points

we expect from our analysis. But in most cases it is bettered, by a significant amount. The worst case obtained by the uniformly generated random data. Even here, the maximum error was typically around half that guaranteed by our analysis. The average error observed was typically between five and ten times smaller than the guaranteed worst case error, a whole order of magnitude. On the New Jersey data set, which we believe is more likely to represent typical data sets, the results observed were stronger. The highest error seen was still three times less than the guaranteed error, and the average error was as much as twenty times smaller.

There are some gaps in the values plotted: this is indicative of the case when the value of ϵ was sufficiently small that no error was observed. That is, for each of the hundreds of queries that were posed, the correct furthest neighbor was returned. This occurs on the uniform data set with values of epsilon around 0.2%, and on the New Jersey data set for values around 0.5%.

For our second experiment, we fixed ϵ at 0.01, and looked at the effect of varying the number of points. We took only the uniform distribution, since this appears to challenge the accuracy of radial hisograms more, and generated point sets with up to 10 million points. The results are shown in Figure 6. For the point sets up to 5000, almost no error is observed. This is because, for this many points, the chances of the true furthest neighbor of a point being stored in the radial histogram are high (since, as we observed in our analysis of radial histograms, furthest neighbors typically lie on the edge of the point set, and hence are likely to get picked up by the radial histogram as maximal points). Above this point, we see that the average error and the worst observed error both stay within the same regions: the average error is around 20% of ϵ; the maximum error fluctuates but never exceeds 80% of guaranteed error ϵ. There appear to be no trends of worsening error as the number of points increases.

The space required for these radial histograms is exactly $2\pi/\epsilon$ points, meaning that for $\epsilon = 0.01$,

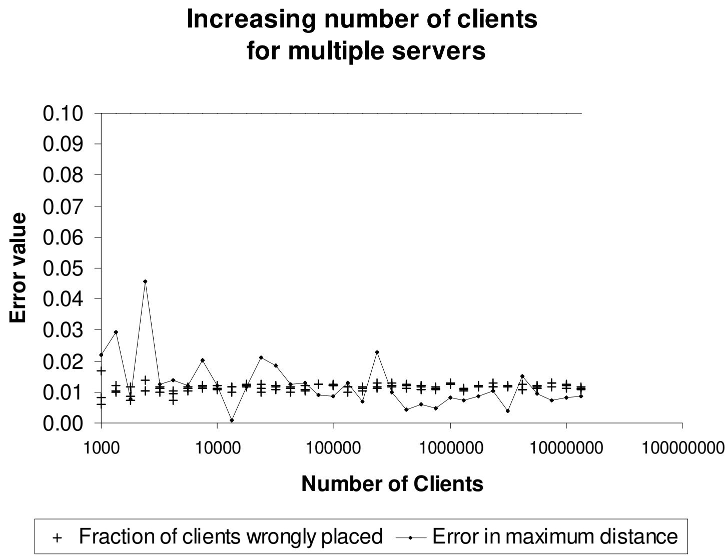

Figure 7: Multiple facilities with increasing numbers of clients

then around 600 points, or less than 5 kilobytes of storage, are needed. By comparison, the data sets we experimented with were up to 100Mb in size. This shows that for large data streams, Radial Histograms are a space efficient way of approximating distances of interest.

6.2 Bichromatic Spatial Aggregates

We implemented support for computing the bichromatic spatial aggregates using multiple radial histograms, where each facility has a corresponding histogram which contains many client points. For our experiments, we chose a number of facility locations, and then inserted a large number of client points. We then chose a subset of the facilities to be "active", and used the radial histograms to allocate the clients to active facilities. We made two computations of the error that using radial histograms induced. Firstly, we examined the error between the largest observed distance between a client and the active facility it is associated with, compared to the exact maximum distance between a client and its correct active facility, giving an approximation of maximal REVERSE NEAREST NEIGHBOR AGGREGATE. We also counted how many clients were allocated to an active facility that was not their closest. Note that we have no bounds on this second quantity from our theorems; however, it turns out that in practice very few clients are allocated to a facility that is not the correct one.

Our first experiment was to test how well radial histograms performed with increasing numbers of client points. We fixed the number of facilities to be 200, from which we selected 100 to be the active facilities at random for each experiment. We set ϵ to be 0.1: recall that the space required in this case is proportional to $1/\epsilon^2$, so setting ϵ too small can result in a blow up of space usage. Here, with $\epsilon = 0.1$, then the space usage of each radial histogram is a few kilobytes, so the overall space

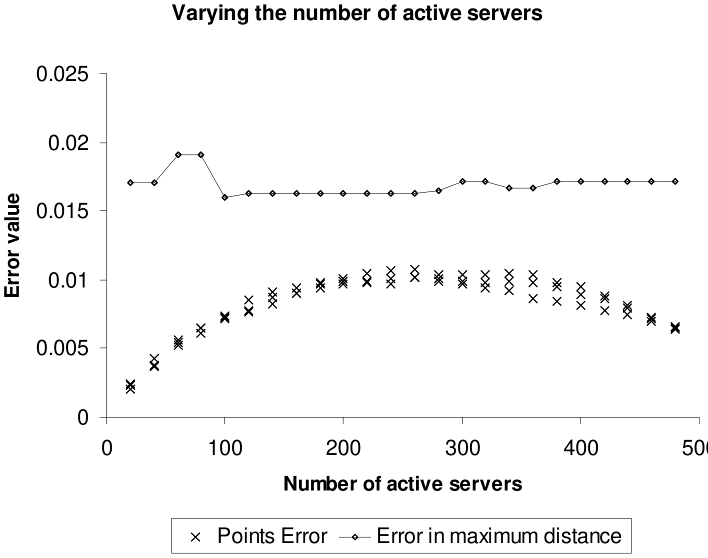

Figure 8: Multiple facilities with varying numbers of active facilities

required is still much less than representing all the points exactly (this also means that processing can be much faster, since we deal with points grouped together at a single point instead of every point individually).

We varied the number of client points, and the results are shown in Figure 7. The error in the distance behaves broadly as with a single histogram: it is typically bounded by 10% of the error guarantee ϵ, and appears to grow more reliable as the number of client points increases. The fraction of errors in allocating points to active facilities is very small: typically around 1% for this setting of ϵ. We repeated each experiment three times, and it can be seen that the repetitions cluster very close together.

It was also important to see how the approach fared with different numbers of active facilities. We created a larger data set, based on points drawn from New Jersey, Delaware, Connecticut, Washington DC, Maryland and Hawaii. This gave a diverse data set, with varying density, and some outlier points. In total, there were several million points. We picked 500 points to be facilities, then from these selected some at random to be the active facilities, and ran the procedure. We again measured how many client points were allocated to an active facility that was not their closest (although these were only a relatively small distance away), and what the error in approximating the maximum distance was. The results are displayed in Figure 8.

The error guarantee was set to 0.1, but as before, the observed results were much better than this guarantee. We observe that the highest error is observed when the number of active facilities is around half the total number of facilities. This makes sense: when the number of active facilities are high, then most points have their default facility active, and so do not contribute to the error; when the number of active facilities is low, then the facilities are more spread out, and so most points are allocated to their correct facility. The error in approximating the maximum distance is also small, as before.

7 Extensions of Radial Histograms

There are many possible extensions of Radial Histograms for other purposes: in dynamic situations, where clients arrive and depart; in higher dimensions and in settings where non-euclidean distance measurements are used. We describe how the data structure can be extended to these scenarios in outline.

7.1 Deletions for Monochromatic Aggregates

We can extend the above methods to allow points to be deleted. We need to extend our use of radial histograms in order to achieve this. Now, whenever we encounter a new point, we find the cell in which it belongs, and increment a counter for that cell. Additionally, we can track the point within the cell that maximizes the distance from the center. A deletion of a point can be accomplished by finding the cell it belongs in, and decrementing the counter. We can maintain the cell containing the point at maximum distance from the center efficiently, using an appropriate heap structure.

This procedure has certain disadvantages in theory: in order to guarantee the above algorithms, we sometimes require that the center be a member of the point set. If the center is deleted, then the approximations may no longer be valid. This can be avoided in a number of ways: if we are sure that some point will always be in the point set, then use that as the center; or we can keep several copies of the data structure, based on different center points, in the hope that at least one of these will survive. On most practical data sets this will not be an issue, since we do not strictly require that the center remains in the data set. For the guarantees to hold, we merely require that the center of the radial histogram be some point contained within the convex hull. It is likely that most points picked will be within the convex hull, even if that point itself is later deleted. With these restrictions, we can now state the result with deletions. These depend on D, the ratio of the diameter of the pointset to the closest pair of points.

Theorem 9 *Providing the center remains within the convex hull of the surviving point set, then using space* $O(1/\epsilon^2 \log D)$ *we can approximate the* DIAMETER, CONVEX HULL, *and* FURTHEST NEIGHBOR *queries all accurate to a factor of* $(1+\epsilon)$ *in time* $O(1/\epsilon)$. *Point insertions and deletions can be processed in constant time.*

7.2 Other dimensional data

We can extend radial histograms to higher dimensions in a straightforward manner. For the three dimensional case, we imagine concentric spherical shells around the center, again at exponentially increasing diameters. These are divided by planes through the center rotating at constant angles through two orthogonal axes. This again ensures that the maximum distance within any cell defined by two successive shells and two successive planes from each axis, will be bounded in terms of the distance from the center and ϵ. All the above algorithms should trivially extend to this three dimensional case, incurring a factor of $1/\epsilon$ in time and space requirements.

Higher dimensions still can be handled by applying the same ideas, but these very rapidly suffer the "curse of dimensionality", as time and space requirements depend on ϵ^{-d}, where d is the dimension. We claim that this is not a significant worry here, since for what we term "spatial data", the data is usually based in two or three dimensions. For significantly highly dimensional data sets, then the standard techniques of dimensionality reduction can be applied [18].

7.3 Other Distance Measurements

So far we have considered only Euclidean (L_2) distance. This has a number of advantages: the Euclidean distance between any two points remains unaffected by rotations, whereas other distance measures typically rely on some imposed axes. Nevertheless, the concept of radial histograms extend to other distance measures, with some alterations.

The rectilinear (L_1) distance between two points is given by the sum of the absolute differences between their co-ordinates. In two dimensions, this is the sum of the distance in the x-direction and the y-direction. [4] Adapting the idea of radial histograms to this distance produces division of space with squares of increasing size around a central square of width ϵ. Each successive row and column of squares has width a factor of $1 + \epsilon$ of the previous one. This guarantees that the distance within any cell is at most a fraction ϵ of its distance from the center, and so the same algorithms as before will work with this style of radial histogram.

Other distances may also be of interest. The L_∞ distance takes the maximum of the distances in the x and y directions. Using the same structure as for L_1 will allow this distance to be approximated upto a factor of $(1+\epsilon)$. In some cases, the distance of interest may not depend on the position of the points in space, but some other metric space. A similar idea to radial histograms can be applied to give a factor 3-approximation for arbitrary metric spaces: for each center, storing a single point which is closer to that center than to any other, and which maximizes the distance this center guarantees that the distance from any client to any server can be approximated up to a factor of three. In general, the distance of interest may have specific features which allow it to be better approximated.

8 Conclusions

Spatial data streams arise in many applications including location-specific sensor measurements and tracking location of objects. A number of spatial aggregates—monochromatic and bichromatic versions of estimating extent and proximity of points—is of great interest in these applications. We propose single and multiple Radial Histograms for estimating these aggregates, to arbitrary precision with provable guarantees. Radial histograms are very small, and extremely easy to maintain.

[4] Hence this distance would be of interest for locating coffee shops in Manhattan, or other U.S. cities based on a grid system.

Our experiments with real and synthetic datasets show them to be far more effective than our theoretical bounds would indicate. We believe that systems to support spatial data streams are a vital part of the ongoing data stream research. Radial histograms will likely find many applications in such systems.

References

[1] S. Acharya, V. Poosala, and S. Ramaswamy. Selectivity estimation in spatial databases. In *Proceedings of ACM SIGMOD*, 1999.

[2] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proceedings of Symposium on Principles of Database Systems (PODS)*, 2002.

[3] S. Babu, L. Subramanian, and J. Widom. A data stream management system for network traffic management. In *Proceedings of Workshop on Network-Related Data Management*, 2001.

[4] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[5] J. L. Bentley, M. G. Faust, and F. P. Preparata. Approximation algorithms for convex hulls. *Communications of the ACM*, 25(1):64–68, 1982.

[6] DIMACS special focus on computational and mathematical epidemiology, 2003.

[7] C. Cranor, L. Gao, T. Johnson, and O. Spatscheck. Gigascope: High performance network monitoring with an SQL interface. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, page 262, 2002.

[8] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. In *Proceedings of 13th Annual Symposium on Discrete Algorithms*, 2002.

[9] W. Fernandez de la Vega and C. Kenyon. A randomized approximation scheme for metric MAX-CUT. *JCSS: Journal of Computer and System Sciences*, 63, 2001.

[10] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. An approximate L_1-difference algorithm for massive data streams. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 501–511, 1999.

[11] J. Feigenbaum, S. Khannan, and J. Zhang. Computing diameter in the streaming and sliding-window models. Technical Report YALEU/DCS/TR-1245, Yale University, `http://www.cs.yale.edu/homes/jf/FKZ.pdf`, 2002.

[12] M. Garofalakis, J. Gehrke, and R. Rastogi. Querying and mining data streams: You only get one look. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2002.

[13] A. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. QuickSAND: Quick summary and analysis of network data. Technical Report 2001-43, DIMACS, 2001.

[14] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. How to summarize the universe: Dynamic maintenance of quantiles. In *Proceedings of 28th International Conference on Very Large Data Bases*, 2002.

[15] A. Guttman. R-trees: a dynamic index structure for spatial searching. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 14(2):47–57, 1984.

[16] P. Indyk. Better algorithms for high-dimensional proximity problems via asymmetric embeddings. In *Proceedings of the 14th Annual Symposium on Discrete Algorithms*, 2003.

[17] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the 30th Symposium on Theory of Computing*, pages 604–613, 1998.

[18] W.B. Johnson and J. Lindenstrauss. Extensions of Lipshitz mapping into Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[19] NASA jet propulsion laboratory. `http://www7. nationalacademies. org/bms/ BravermannasPDF.pdf`.

[20] F. Korn, S. Muthukrishnan, and D. Srivastava. Reverse nearest neighbors aggregates over data streams. In *Proceedings of 28th International Conference on Very Large Data Bases*, 2002.

[21] Flip Korn and S. Muthukrishnan. Influence sets based on reverse nearest neighbor queries. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 29(2):201–212, 2000.

[22] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. In *Proceedings of the 30th Symposium on Theory of Computing*, pages 614–623, 1998.

[23] P. Lioy. Approaches to conducting air sampling in the community environment, air sampling instruments. In *American Conference of Governmental Industrial Hygienists*, pages 51–76, 2001.

[24] S. Madden and M. J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *Proceedings of 18th International Conference on Data Engineering*, pages 555–566, 2002.

[25] D. Madigan. DIMACS working group on monitoring message streams. `http://stat.rutgers.edu/ ∼madigan/mms/`, 2003.

[26] R. Motwani. Models and issues in data stream systems. In *PODS Plenary Talk at the ACM SIGMOD/PODS 2002 Conference*, 2002.

[27] M. Muralikrishna and D. DeWitt. Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. In *Proceedings of ACM SIGMOD*, 1988.

[28] S. Muthukrishnan. Data streams: Algorithms and applications. In *ACM-SIAM Symposium on Discrete Algorithms*, `http://athos.rutgers.edu/∼muthu/`, 2003.

[29] S. Muthukrishnan, V. Poosala, and T. Suel. On rectangular partitionings in two dimensions: Algorithms, complexity, and applications. In *Proceedings of ICDT*, pages 236–256, 1999.

[30] Statistical analysis of massive data streams. http://www7.nationalacademies.org/bms/Massive_Data_Workshop.html, 2002. The National Academies Committee on Applied and Theoretical Statistics.

[31] Cisco NetFlow. More details at http://www.cisco.com/warp/public/732/Tech/netflow/.

[32] V. Poosala. *Histogram-Based Estimation Techniques in Database Systems*. PhD thesis, University of Wisconsin, Madison, 1997.

[33] F. P. Preparata and M.I. Shamos. *Computational Geometry : An Introduction*. Springer-Verlag, 2nd edition, 1985.

[34] L. Segoufin and V. Vianu. Validating streaming xml documents. In *Proceedings of Principles of Database Systems*, pages 53–64, 2002.

[35] M. I. Shamos. *Computational Geometry*. PhD thesis, Department of Computer Science, Yale University, 1978.

[36] Stanford stream data manager. http://www-db.stanford.edu/stream/sqr, 2003. J. Widom and *et al*.

[37] N. Thaper, S. Guha, P. Indyk, and N. Koudas. Multidimensional dynamic histograms. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2002.

[38] Topologically Integrated Geographic Encoding and Referencing system, 2003. U. S. Census Bureau.

[39] C. F. R. Weiman and G. Chaikin. Logarithmic spiral grids for image processing and display. *Computer Graphics and Image Processing*, 11(3):197–226, 1979.