Scalable Spatial Scan Statistics for Trajectories

MICHAEL MATHENY, DONG XIE, and JEFF M. PHILLIPS, University of Utah

We define several new models for how to define anomalous regions among enormous sets of trajectories. These are based on spatial scan statistics, and identify a geometric region which captures a subset of trajectories which are significantly different in a measured characteristic from the background population. The model definition depends on how much a geometric region is contributed to by some overlapping trajectory. This contribution can be the full trajectory, proportional to the length within the spatial region, or dependent on the flux across the boundary of that spatial region. Our methods are based on and significantly extend a recent two-level sampling approach which provides high accuracy at enormous scales of data. We support these new models and algorithms with extensive experiments on millions of trajectories and also theoretical guarantees.

CCS Concepts: • **Theory of computation** → **Computational geometry**; • **Computing methodologies** → Anomaly detection; • **Mathematics of computing** → *Maximum likelihood estimation.*

ACM Reference Format:
Michael Matheny, Dong Xie, and Jeff M. Phillips. 2020. Scalable Spatial Scan Statistics for Trajectories. *ACM Trans. Knowl. Discov. Data.* 1, 1, Article 1 (January 2020), 23 pages. https://doi.org/10.1145/3394046

1 INTRODUCTION

Large data sets of trajectories have driven much recent research with the goal of understanding the intentions and causes of various correlations hidden within the data. These data objects are structured to capture movements, interactions, and possibly the intentions of humans and other objects. Yet, when one considers these trajectories in unison, they usually appear as just a tangled mess. And as the data grows (e.g., millions of objects with billions of data points), this task does not seem to aggregate and statistically simplify, rather it just becomes more unwieldy and unmanageable.

We consider new methods that use the underlying geometry of the trajectories to identify statistically significant spatial anomalies. Critically, these models deviate from density-based models (like DBScan [11]) which would only identify populous regions (of course a lot of traffic exists in New York or Beijing!). Rather, our new models and algorithms identify geometric regions where some labeled aspect (e.g., trajectories of sick people) significantly deviates from what is expected or is in contrast to the trajectories of the background population. And our approaches work at enormous scale – required for modern large data sets, and for the statistics to be meaningful.

Developing such models for trajectories, comparing to a background population, is highly motivated: identifying significant population or demographic shifts, pinpointing the likely location responsible for disease due to prolonged exposure among a dynamic population, or geolocating a nefarious wifi access point affecting cell phones which transiently pass by. There are no existing mechanisms for addressing some of these goals specific to trajectory data, and certainly not for massive data sets.

Authors' address: Michael Matheny, mmath@cs.utah.edu; Dong Xie, dongx@cs.utah.edu; Jeff M. Phillips, jeffp@cs.utah.edu, University of Utah, Salt Lake City, Utah, 84112.

1556-4681/2020/1-ART1 $15.00
https://doi.org/10.1145/3394046

When base objects are points (not trajectories), such comparative anomaly tasks are typically resolved with a *Spatial Scan Statistic* (SSS) [16]. This is one of the most common tasks within Geographic Information Science, with applications to detecting hotspots with elevated levels of disease, crime, or demographic traits [4, 12, 16, 24, 30]. These identify a region with maximum log-likelihood score Φ, out of a large prescribed family of regions $\mathcal{C}$, and have been shown empirically and theoretically to have high statistical power [4, 16]. But this search of *all* regions $C \in \mathcal{C}$ (usually associated with a geometric family of shapes) is computationally onerous, and the most common and widely-used software SatScan [17] is only able to scale by restricting the class of regions to ones specifically chosen by the user. Moreover, there are only a few limited extensions towards trajectory data [19, 25] and these rely heavily on heuristic aggregation.

Our Contributions. We introduce three new models to allow for geometric analysis of trajectory data. These models (derived in Section 3) identify geometric regions where many trajectories of interest have passed through (full model), spent time in (partial model), or began or ended in (flux model). These models are new and well-motivated, but they have not before been a computationally feasible objective to consider at scale.

We design sampling and scanning algorithms (in Section 6) that allow for extremely scalable methods for identifying anomalous patterns captured in the above geometric models. Our methods are not limited by the data size, instead they depend on the error—spatial and combinatorial—that researchers are willing to approximate the final statistical quantity to. The scalability as well as statistical power of these models are demonstrated on enormous data sets containing up to several million trajectories with over 1 billion waypoints (in Section 7).

Of the three models, we develop a reduction for the *flux* and *partial* model to a scalable point-based scanning framework (see Section 6). However for the third model, *full*, such reductions are not possible, and considerable new scanning and sampling mechanisms are developed: Section 4 designs compact coresets for each trajectory that preserves spatial guarantees and converts trajectories to labeled point sets. Section 5 provides trajectory specific sampling theorems (VC-dimension bounds for trajectory-based range spaces) necessary to apply the underlying sample approach. Section 6.2 develops algorithms that use new data structures to scan the data resulting from trajectory coresets and their samples in a scalable way. The overall accuracy and runtime bounds for of our sampling, coreset, and scanning algorithms are ultimately summarized in Table 2.

2 PRELIMINARIES, NOTATION, AND OVERVIEW

We begin with an overview of the mathematical modeling, geometric, statistical, and algorithmic preliminaries and their notation to frame our new contributions.

Trajectory models. We model a trajectory t via waypoints $P_t = \langle p_1, \ldots, p_m \rangle$ as the tracing out of the ordered sequence of connected segments $s_1, \ldots, s_{m-1}$, with $s_j = \overline{p_j p_{j+1}}$. Trajectory t has total arclength $L(t)$ (or when t is clear, just L). Computationally and also as a way of defining regions, we often consider trajectories via just some set of ordered waypoints $p_1, \ldots, p_m$ constructed via one of the methods discussed in Section 4 instead of as an ordered sequence of connected segments. For the partial model the parameterization of the trajectories is important, and we use arclength by default. We could alternatively use a time-based parametrization, but we otherwise do not explicitly model timing information in this paper.

Range spaces. To study spatial anomalies applied to trajectories we need a way to model how or when a trajectory interacts or intersects with a potential region of interest. To do this we review the definition of a *range space*. A range space is a pair $(X, \mathcal{A})$ consisting of a set of objects X (the *ground set*) and a set $\mathcal{A} \subset 2^X$ of subsets of X (the *ranges*), that is a subset of all possible subsets 2^X. Range spaces are essential objects in both data structures (e.g., for range searching [2]) and machine learning (for sample complexity of learning [13, 31]). For example, classically let X be points in $\mathbb{R}^2$, and $\mathcal{A}_C$ be the subsets induced by intersection with a disk $C \in \mathcal{C}$.

For this paper, we will define new families $\mathcal{A}_C$ induced by a set of shapes $\mathcal{C}$, focusing on those induced by halfspaces, disks, or axis-aligned rectangles. In particular, we are interested when the ground set is a set of

trajectories T (or derived appropriately from T). The new models (we will define in Section 3) will specify the definition of intersection $T \cap C$, for prescribed spatial regions $C \in \mathcal{C}$, to induce a set $A_C \in \mathcal{A}_C$. That is each induced subset of trajectories $A_C \in \mathcal{A}_C$ corresponds with a potentially anomalous region of interest.

Statistical discrepancy. In all of the forthcoming models, each trajectory $t \in T$ has a recorded value $r(t)$ and baseline value $b(t)$. It is typically sufficient to consider *all* trajectories $t \in T$ have $b(t) = 1$ (that is they are all part of an observed background population) and that $r(t) \in \{0, 1\}$ where the set $T_r = \{t \in T \mid r(t) = 1\}$ are the trajectories of interest – although these can change in accordance with objective function ϕ in a variety of statistical settings [1, 17]. Even when we study parts of trajectories and segments of trajectories, these traits (especially the recorded value $r(t)$) is held fixed for the entire trajectory.

Then let $r(C) = \frac{|T_r \cap C|}{|T_r|}$ (resp. $b(C) = \frac{|T \cap C|}{|T|}$) be the fraction of all recorded trajectories (resp. all trajectories) within the range C. Modeling r as Poisson, the log-likelihood of the recorded rate being different than the baseline rate is $\Phi(C) = \phi(r(C), b(C))$, where $\phi(r, b) = r \log \frac{r}{b} + (1 - r) \log \frac{1-r}{1-b}$. However, in general, we can computationally reduce to a simpler subproblem [1, 21] with a "linear" model e.g., $\phi(r, b) = |r - b|$. Ultimately, the spatial scan statistic is $\max_{C \in \mathcal{C}} \phi(C)$. It uses the magnitude of $\Phi(C^*)$ to determine if the most anomalous region C^* is statistically significant; specifically, the typical way is via permutation testing where about 1000 data sets with same spatial patterns but different r, b values are generated at random, and the distribution of found $\Phi(C^*)$ values are compared to the one with the input r, b values. We will specifically be interested in an approximate variant:

Definition 2.1 (ε-Approx. Spatial Scan Statistic). *Consider a range space $(X, \mathcal{A}_C)$ defined by a family of shapes $\mathcal{C}$, and a discrepancy function $\Phi : \mathcal{C} \to \mathbb{R}$. An* ε-approximate spatial scan anomaly *is a shape $\hat{C} \in \mathcal{C}$ so that*

$$\Phi(\hat{C}) + \varepsilon > \Phi(C^*)$$

where $C^* = \arg\max_{C \in \mathcal{C}} \Phi(C)$. Then the corresponding ε-approximate spatial scan statistic *is $\Phi(\hat{C})$.*

Two-level sampling. The algorithmic goal in this paper is to find an ε-approximate spatial scan anomaly for trajectory range spaces (with forthcoming definitions in Section 3). Our scanning algorithms are based on recent work for calculating ε-approximate spatial scan statistics on simple geometric shapes over point sets at scale [20, 22]. The main ideas are to construct a *two-level sampling* of a large data set into sets S and N; see Figure 1. The larger "sample" set $S = S_r \cup S_b$ (where S_r is the recorded set and S_b is the baseline set, and S is of size s) is used as proxy for the density of the data in any range, and the smaller "net" data sets N (of size n) is used to define the combinatorial set of ranges we will consider. That is, N *defines a subset of shapes* $C \in \mathcal{C}_N$; those which include combinatorial distinct subset of points in N, and then S is used to *estimate the* $\Phi(C)$ *value* via $C \cap S$. To achieve ε error in $\Phi(C)$, we (roughly) need to set $n = \nu/\varepsilon$ and $s = \nu/(2\varepsilon)^2$, where ν is the VC-dimension of the

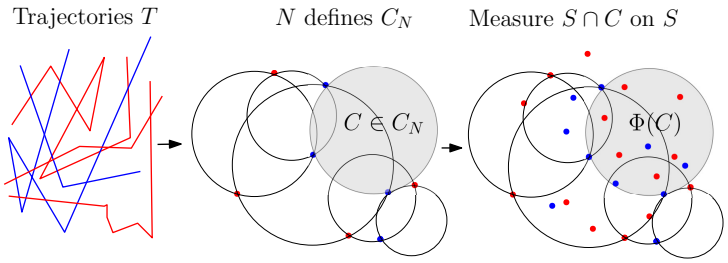


Fig. 1. A sparse sample of the data defines a set of regions and these regions can be efficiently measured on a larger sample that maintains the density of the original data set.

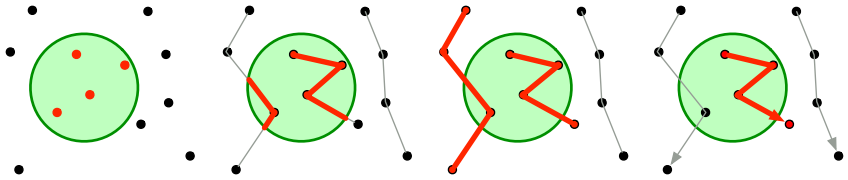


Fig. 2. Models of how 3 trajectories intersect (in red) with circular shape (in green). From left to right: point-based model; partial range model; full range model; flux model.

range space [20]. Then there exist shape specific methods to quickly scan and evaluate the ranges (induced by N) and values (from S) [20].

3 NEW TRAJECTORY RANGE MODELS

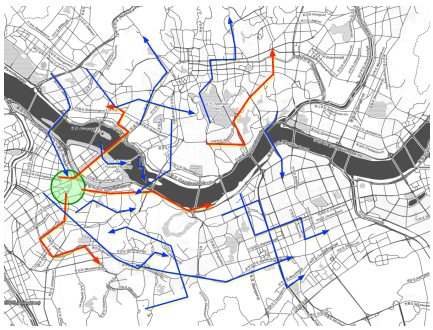
We introduce three new models of how to define range spaces when applied to trajectory-derived ground sets and geometric ranges which define the region of interest. These models capture: (i) regions with a high percentage of measured trajectories passing from inside to outside (the *flux model*), (ii) regions with a high percentage of the total arclength of measured trajectory data (the *partial range model*), and (iii) regions with a high percentage of measured trajectories pass through them (the *full range model*).

3.1 Flux Model

The simplest version of this problem is the flux model. We search for a shape $C \in \mathcal{C}$ where a proportionally high number of trajectories start inside the shape C, and end outside of the shape C, or vice versa.

Mathematical Definition. In this setting, we can reduce each trajectory to two waypoints: the first and the last. To satisfy a range intersection ($t \cap C$ for range C), the first must be inside and the last must be outside (or vice-versa with opposite effect). That is we define two sets $X^b = \{p_1 \mid t \in T, t = \langle p_1, \ldots, p_m \rangle\}$ (the beginning set) and $X^e = \{p_m \mid t \in T, t = \langle p_1, \ldots, p_m \rangle\}$ (the end set). Then we attempt to find the range $\arg\max_{C \in \mathcal{C}} |C \cap X^e| - |C \cap X^b|$ or vice-versa.

Motivating scenarios. This model arises when finding a region that differentiates two types of movements. Consider the GPS tracking of an animal like deer, and these paths are either slow (normal walking) or fast (running, perhaps escaping a predator). Then a region corresponding to the start of many fast trajectories may correspond with the region occupied by a predator. Alternatively, if trajectories document addresses of people over time (for instance, this information is maintained for residents of the state of Utah in the Utah Population Database [29], which has tracked addresses for 50 years), this can be used to identify significant migratory patterns of parts of the population. Regions of unusually high flux in any fixed window of time may also be useful for managing crowd control in packed sporting or music events.



Flux Model Example:
In the image on the left, let the trajectories represent fare paths of a Kakao Taxi driver in Seoul over a day. The red trajectories represent significant tips, and the blue ones all other fares paths. Identifying a region (in green) where red (high tip) routes started, and left from, without too many blue routes of the same form, will indicate a good place to try to find profitable customers. This would correspond with the disk maximizing Φ under the flux model.[1]

3.2 Partial Range Model

In the partial range model, we want to find a shape $C \in \mathcal{C}$ where the weight a trajectory contributes to C is proportional to the normalized length of the trajectories inside. A trajectory's intersection with a shape $C \in \mathcal{C}$ is fractional; specifically $\mu(t \cap C)$ is the fraction of the total arclength of t within C. That is if $1/3$ of a trajectory t intersects C, then $\mu(t \cap C) = 1/3$ or if $1/50$th of the total length of all trajectories intersect C then $\mu(T \cap C) = 1/50$. The contribution $\mu(t \cap C)$ depends on the parametrization of t. This could be by arclength, by time, by fuel used, or any other quantity. In this paper we simply use arclength in our experiments.

Mathematical Definition. More formally, the set $C \cap T$ corresponds to the set of points comprising partial trajectories which are inside C. The true ground set X is then $X = \{x \in \mathbb{R}^2 \mid x \in s_j \in t \in T\}$. That is X is a subset of $\mathbb{R}^2$ so each point $x \in X$ lies on some segment $s_j \in t$ on some trajectory $t \in T$. This corresponds to an infinite (not combinatorially defined) range space [31] as $(X, \mathcal{A}_C)$ where $\mathcal{A}_C = \{A_C = X \cap C \mid C \in \mathcal{C}\}$.

Motivating scenarios. The partial range model is important to help automatically identify regions which statistically lead to some measured characteristic, proportional to how long the object generating the trajectory spends in that region. For instance, consider a mysterious sickness that health officials suspect is tied to prolonged exposure to some chemical event. By finding a compact region where many of the inflicted people spend a considerable amount of time, compared to all in that region, this will provide a candidate location for the epicenter of that exposure. Alternatively, consider a measured set of unprofitable (or highly profitable) taxi/Uber drivers; can we identify regions of a city where they spend a proportionally higher percentage of their time. These and similar scenarios are directly modeled by the partial range scan statistic, and hence demand scalable solutions.

Partial Range Model Example:
Let the left image model the daily commutes of patients that have checked into Paris hospitals. The red curves correspond with patients who reported a serious lung and stomach symptoms when they arrived to work, consistent with exposure to a dangerous chemical. The blue trajectories are other patients with more standard and mainly benign symptoms. A region where more red trajectories spent significant time, and blue trajectories did not spend much time, would help pinpoint the region with largest potential to be the source of these symptoms. And finding such region corresponds with the disk that maximizes Φ under the partial range model. [1]

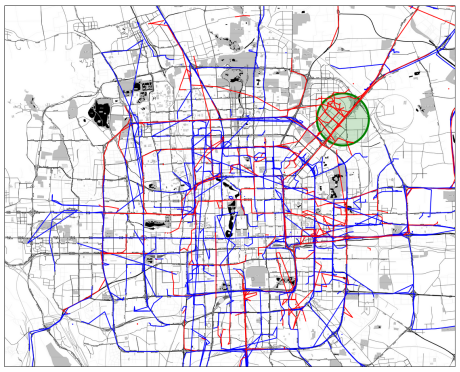
3.3 Full Range Model

In the full range model, we seek to find the shape $C \in \mathcal{C}$ which intersects the most trajectories of interest, compared to some baseline set of trajectories. A trajectory's contribution to C is *not* proportional to intersection size, but instead all or nothing.

Mathematical Definition The contribution of a trajectory $t \in T$ in a shape $C \in \mathcal{C}$ is binary. The intersection $t \cap X$ is 1 if there is any point where the trajectory t enters the shape C, and is only 0 if the trajectory is never inside the shape. And as such ranges $A_C \in \mathcal{A}_\mathcal{C}$ are defined $A_C = \{t \in T \mid t \text{ intersects } C\}$, for some shape $C \in \mathcal{C}$. There is no need to parameterize the trajectories in this scenario.

We will consider simplified trajectories by creating a small set of labeled points P'_t to represent them. In this scenario, we say t intersects shape C if *any* $p'_j \in P'_t$ intersects C.

Motivating scenarios. This model arises when just the fleeting intersection with a spatial region is enough to trigger a measured event for that entire trajectory. Consider a set of cars with slow leaks from nails in their tires (found at the end of the day); a region many of them passed through would more likely be someplace with nails on the road. Or consider a set of people's cell phones which a virus, suspected to be infected when they pinged some wifi access point; then just passing near that access point may be enough to trigger the event, and finding a region with high-density of cases of the virus would provide a probable location of the offending access point. Or consider tracking a set of animals (e.g., cows, migrating birds) where a subset become sick; then a full range spatial anomaly may indicate a contaminated watering hole.



Full Range Model Example:
Consider the set of limousine trajectories from a day in Beijing in the image. The red trajectories may represent taxis which reported navigation systems malfunctioning, while the blue ones did not. You suspect a jamming devise, with a fixed radius, was placed by a competitor. Detecting the circular shape (in green) which maximizes the fraction of red trajectories which pass through, while minimizing the fraction of blue, is the most likely model for where a jamming devise could have been located. This corresponds precisely to finding the disk with largest Φ value under the full range model.[1]

3.4 Remaining Challenges

Several significant tasks remain to adapt the two-level sampling framework for spatial scan statistics to these newly defined trajectory scanning models. First, we require to spatially approximate the trajectories – the raw trajectories are either unnecessary or too difficult to work with; we describe our methods in Section 4. Second, we need to formalize the definitions of regions $\mathcal{C}_N$ from the set N, and bound the VC-dimension ν with respect to the resulting range spaces; we do so in Section 5. Given these mechanisms, we next show that for the partial and flux models, we can devise direct reductions to point set variants in Section 6.1. Given these reductions, we can invoke existing fast scanning algorithms [20]. However, for the full model such reductions are not possible, and we will need to develop new methods to quickly iterate over all these ranges on S; that is how to quickly "scan" over the net. This varies with the geometric properties of the shapes; we focus on halfspace, disk, and rectangle ranges, and devise new scanning methods for each of them in Section 6.2.

4 SPATIAL APPROXIMATION

As described, for a shape $C \in \mathcal{C}$ and trajectory t, the critical operation in the full model is determining $t \cap C = \emptyset$. This is far more efficient if we can approximate t by a set of k approximate waypoints $P'_t = \{p'_1, p'_2, \ldots, p'_k\}$, and then use $P'_i \cap C$ as a proxy for $t \cap C$. The critical aspects of such an approximation is to keep the size of the approximation k small, even for long trajectories, and to ensure that the answer to the intersection between a shape C and the point set P'_t approximates $C \cap t$.

Specifically, we desire an *α-spatial approximation* (or just *α-approximation* for short). For a trajectory t and any range $C \in \mathcal{C}$, we say P'_t is an α-spatial approximation of t under two conditions (see Figure 3(Left)):

(1) (no false positives) If t does not intersect C, then P'_t does not intersect C.
(2) (limited false negatives) If for *all* unit vectors v, $t + v\alpha$ intersects C, then P'_t intersects C; here $t + \alpha v$ is a shift of the entire trajectory in a direction v by α.

For the full model, the α-approximation corresponds well to no false positives and reasonable false negatives. For the partial model, we want the addition property that $|C \cap P'_t|$ is proportional to $\mu(C \cap t)$; and for this in some cases we sometimes add weights or make other small adjustments to the points in P'_t. As we will see in Section 6 for the flux model we can always find a set P'_t with $k = 2$ and achieve no error.

[1] Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL.

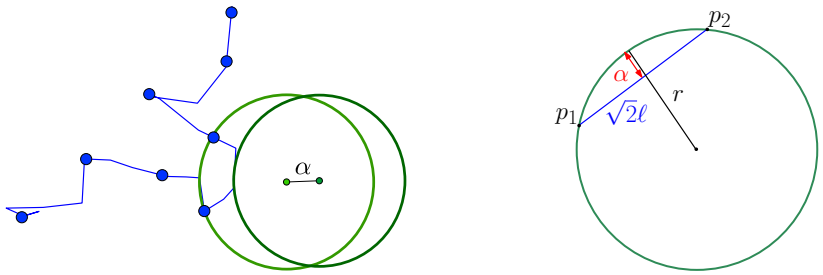


Fig. 3. Left: Converting a trajectory into a set of points which approximate its shape preserves an α-spatial guarantee, since we can find a nearby disk that still intersects the trajectory. Right: The relation between α, γ, and r in proof of Lemma 4.3 for some line segment $\overline{p_1p_2}$ on the boundary of one of our kernels.

We next list a series of trajectory approximations we study (some approaches are well-known and standard), and describe how to achieve and α-approximation under each method if possible. All bounds assume all trajectories lie in a $[0,1] \times [0,1]$ domain, otherwise α is scaled accordingly.

All Waypoints. This baseline simply sets $P'_t = P_t$ and retains all waypoints. This does not deal with long segments well, and does not achieve an α-approximation except for halfspaces, but may be appropriate for data collected at regular and short intervals.

Random Sampling. This baseline randomly samples k points from segments proportional to arclength. Let L be the total arclength of a trajectory t. Based on VC-dimensional [31] and ε-net [13] arguments, if $(\mathbb{R}^2, C)$ has constant VC-dimension (as with disks, halfspaces, and rectangles) then for $k = O((L/\alpha)\log(L/\alpha))$, with constant probability it is an α-approximation.

Even. In this sketch, we select $k = L/\alpha$ points evenly spaced according to arclength, where again L is the total arclength of a trajectory. This deterministically creates an α-approximation. To preserve the proportionality property for the partial model, we treat the trajectories as if they are chained together to adjust the first selected point from each trajectory – so the first points on trajectory t_j is a distance α from last point on trajectory t_{j-1}.

DP algorithm. The Douglas-Peucker (DP) algorithm [14] is frequently used in practice as a compression step for trajectory simplification. This method iteratively removes waypoints from the original trajectory in a greedy fashion until removing another one would cause the Hausdorff distance between the original and the simplified one to exceed a chosen parameter α. This ensures, for instance, that no query shape $C \in \mathcal{C}$ can intersect the original trajectory t at a depth α into C without also intersecting the simplified trajectory. For halfspaces this provides an α-approximation (see Lemma 4.1). However for rectangles and disks, this guarantee is only over the trajectory's segments, but not the waypoints P'_t, so it does not provide an α-approximation as desired.

Convex Hull. This sketch puts all vertices on the convex hull of P_t in P'_t. This is a 0-approximation (has no error) for halfspaces (see the next lemma), but does not have guarantees for other shapes.

Lemma 4.1. *A halfspace $h \in \mathcal{C}$ intersects a trajectory if and only if it intersects at least one of its waypoints.*

Proof. A halfspace h intersects part (but not all) of a trajectory if and only if its boundary intersects one (or more) of its segments. If it intersects all of the trajectory, it must contain all waypoints. For a boundary plane to intersect a segment s_j, it must be that one of its waypoints p_j, p_{j+1} is inside the halfspace and the other is not since the segment is a convex object with these points as the only extreme points. Hence, we can check intersection of a trajectory t with h by checking if any of its waypoints are in h; otherwise all of the waypoints and the entire trajectory must be outside of h. □

Approx Hull. In this sketch, we create P'_t as the α-kernel coreset which approximates the convex hull of P_t [3]. This provides an α-approximation for halfspaces with only $O(1/\sqrt{\alpha})$ points, independent of arclength, but with restriction that all trajectories are in $[0,1] \times [0,1]$.

Lifting and Convex Hull for Disks. For disks, there is a reduction via a data transformation (the Veronese Map v) that provides similar approximations as the convex hull approach for halfspaces. Given a point set $P \in \mathbb{R}^2$, the intersection of that point set with a disk is preserved under a map to $\mathbb{R}^3$ where disks are mapped to halfspaces. For $p = (p_x, p_y) \in \mathbb{R}^2$ we replace it with $v(p) = (p_x, p_y, p_x^2 + p_y^2) \in \mathbb{R}^3$. Every disk becomes a halfspace in $\mathbb{R}^3$ and contains the same subset of points as the disk did in $\mathbb{R}^2$.

After this transformation, set P'_t as the points on the convex hull (or in the α-kernel coreset [3]) in $\mathbb{R}^3$. However, because Lemma 4.1 does not apply to disks, this does not have any approximation guarantee. That is, a disk C in $\mathbb{R}^2$ which intersects a segment but no waypoints, transforms to a halfspace $h_C \in \mathbb{R}^3$ which contains part of a segment (these segments are now quadratic curves, and are not straight), but still no waypoints.

Gridding. This sketch considers an (implicit) regular, orthogonal grid in $\mathbb{R}^2$ for each trajectory with cells of size $\alpha \times \alpha$. We retain one point on the trajectory t in each cell that t intersects. For a trajectory t of arclength L, this retains at most $k = O(L/\alpha)$ points in P'_t.

Lemma 4.2. *The number of cells of a regular grid with grid cells of size $\ell \times \ell$ that a polyline can enter is $O(L/\ell)$.*

Proof. We will group cells into 9 groups and analyze each separately. Each cell is in the same group as other cells two hops over in one of the 8 directions (left, right, 45 deg, etc...). Each cell touches 8 other cells which are not in its group, and each one is in a distinct group. Now within each group, to intersect a cell and enter another one the trajectory must travel a distance of $\Omega(\ell)$, since it will have to pass the complete vertical or diagonal distance of a cell. Thus, within a group a trajectory of length L can touch at most $O(L/\ell)$ cells. And in 9 groups, the total number of cells is at most $9O(L/\ell) = O(L/\ell)$. □

Grid Kernel. For disks it makes sense to adjust the approximation for different radii as smaller values of k are needed for very large radii disks which can potentially intersect many trajectories. By constructing multiple approximations and scanning each with different radii significant speedups in practice can be realized.

We adjust the gridding technique for disks, and specifically for a family $\mathcal{C}_r$ which only considers disks of radius at least r. We consider grid cells of size $\gamma \times \gamma$ with $\gamma = \sqrt{2\alpha r - \alpha^2/2}$. Within each grid cell we retain multiple points in P'_t, specifically those on a $(\alpha/(2\sqrt{2}\gamma))$-kernel coreset of the points in that cell.

Lemma 4.3. *For a trajectory with arclength L, at most $O(L/(r^{1/4}\alpha^{3/4}))$ points are put in P'_t for Grid Kernel, and it is an α-approximation for $\mathcal{C}_r$.*

Proof. The maximum distance between two points in a grid cell is $\sqrt{2}\gamma$. Thus the $(\alpha/(2\sqrt{2}\gamma))$-kernel incurs at most $\sqrt{2}\gamma \cdot (\alpha/2\sqrt{2}\gamma) = \alpha/2$ error between the convex hull of all points in that cell, and the hull of the approximate ones.

However, a disk may intersect part of a trajectory without intersecting any of the waypoints on the convex hull. But, if the longest possible edge in convex hull is $\sqrt{2}\gamma$ then a disk of radius r not containing a waypoint can protrude into the hull at most a distance (see Figure 3(Right))

$$\begin{aligned} r - \sqrt{r^2 + (\sqrt{2}\gamma/2)^2} &= r - \sqrt{r^2 + (\sqrt{2\alpha r - \alpha^2/2}/\sqrt{2})^2} \\ &= r - \sqrt{r^2 + (\alpha r - \alpha^2/4)} \\ &= r - \sqrt{(r + \alpha/2)^2} = r - r + \alpha/2 = \alpha/2. \end{aligned}$$

The sum of these two errors is at most $\alpha/2 + \alpha/2 = \alpha$, as desired.

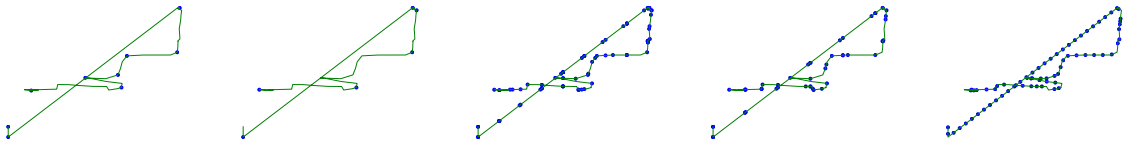


Fig. 4. Full scanning coresets from left to right: DP for halfplanes, Approx Hull for halfspaces, Grid Kernel for disks (radius r), Grid Kernel for disks (radius $2r$), and Even.

The total number of points associated with a trajectory of length L is: the number of cells it intersects $O(L/\gamma)$ times the number of points in each cell $O(1/\sqrt{(\alpha/(2\sqrt{2}\gamma)} = O(\sqrt{\gamma/\alpha})$. In total this is

$$O\Big(\frac{L}{\gamma} \cdot \sqrt{\gamma/\alpha}\Big) = O\Big(\frac{L}{\sqrt{\alpha\gamma}}\Big) = O\Big(\frac{L}{\sqrt{\alpha\sqrt{\alpha r}}}\Big) = O\Big(\frac{L}{\alpha^{3/4} r^{1/4}}\Big). \qquad \square$$

5 TRAJECTORY SAMPLING

To enable efficient scanning for the full model we require that the number of regions grows polynomially with the number of trajectories as otherwise random sampling cannot be used to attain an additive approximation bound. For the trajectory range spaces we consider it was not previously known if this was true. A bound on the VC-dimension of these range spaces would ensure this. It turns out the VC-dimension bounds are tied in some manner to k, the number of points representing each trajectory; the number of possible subsets is then a function of the range complexity (ν, e.g., dimension d) and the trajectory complexity (k).

We now consider that each trajectory t is represented by exactly k labeled points P'_t (if it is less than k, we can duplicate the last point to increase to k). That is $P'_t \subset \mathbb{R}^{d \times k}$. Next consider an alternative range space $(\mathbb{R}^{d\times k}, \mathcal{A}_k)$ where the ground set is the approximate waypoints. The number of ranges induced on a set of labeled points (two ranges are the same if they contain the same set of labels) is upper bounded by the number of unique subsets on the mk unlabeled points. Therefore the growth function on m sets is upper bounded by $O((mk)^\nu) = m^{O(\nu \log k)}$, where the base range space $(\mathbb{R}^d, \mathcal{A})$ has VC-dimension ν.

Lemma 5.1. *The growth constant for $(\mathbb{R}^{d\times k}, \mathcal{A}_k)$ is $O(\nu \log k)$ where ν is the growth constant of the range space $(\mathbb{R}^d, \mathcal{A})$. Hence the VC-dimension is $O(\nu \log k)$.*

In our context, this means after trajectories in $\mathbb{R}^2$ are represented by at most k points, then the VC-dimension for ranges defined by disks, halfspaces, or rectangles have VC-dimension $O(\log k)$. Similar logarithmic dependence on k has recently been shown for range spaces defined between pairs of trajectories [8].

So as long as the number of trajectory waypoints is bounded by k, the sample sizes for S and N (needed in the two-level framework [20, 22]) are increased by a rather benign near-logarithmic in k. We will invoke this in the context of scanning algorithms in Section 6.2.

Bounded k is needed. It seems hopeful that a better bound independent of k may be possible, but we can show that for halfspaces, disks, and rectangles it is possible to construct cases where the complexity of the trajectories, and hence the VC dimension is unbounded. To see this we will first restrict to the set of ranges $\mathcal{H}_d$ induced by halfspaces in $\mathbb{R}^d$. Indeed, we can replace each trajectory with a convex set by Lemma 4.1, so we only need to work with a ground set of all convex sets $\mathbb{C}$. However, if the complexity of the trajectories, and hence the convex sets, is unbounded, then so is the VC-dimension even in $\mathbb{R}^2$, as the next lemma shows.

Because disks in $\mathbb{R}^d$ are special cases of halfspaces in $\mathbb{R}^{d+1}$ (by the Veronese map), this bound holds for disks as well.

Lemma 5.2. *The VC-dimension of $(\mathbb{C}, \mathcal{H}_2)$ is unbounded, and hence so is the VC-dimension of $(T, \mathcal{H}_2)$ with no restriction on k.*

Proof. For any integer z, we can design a set of z convex sets $c_1, c_2, \ldots, c_z \in \mathbb{C}$ in $\mathbb{R}^2$ so that we can *shatter* the set; that is, every subset of C can be induced by a range. Recall the VC-dimension of a range space is the size of the largest subset of the ground set which can be shattered. Let each c_j be a nearly identical convex polygon with 2^z vertices. Each of the 2^z vertices, which are in nearly the same location for each polygon, corresponds with one of the 2^z different subsets we seek to define. If the vertex is in a polygon that should be in the subset, shift it slightly counter-clockwise; if it is not in the subset, shift it slightly clockwise. Then a halfspace can include the vertex with all points shifted counter-clockwise (those intended for the subset) and nothing else in any polygon. Since we can do this independently for all 2^z subsets for any z, we can shatter a subset of any size z.

Now because a halfspace intersecting a trajectory t is equivalent to intersecting any of its waypoints (see Lemma 4.1), then it is equivalent to intersecting the convex hull of those waypoints. Thus, we can generate the same construction with a trajectories with $k = 2^z$ waypoints for any value z, and thus if we have no bound on k, we have no bound on the VC-dimension of $(T, \mathcal{H}_2)$. □

For rectangles we can construct a similiar proof.

Lemma 5.3. *The VC-dimension of $(T, \mathcal{R}_2)$ is unbounded if there is no restriction on k.*

Proof. Again for any integer z, we can design a set of z trajectories $t_1, t_2, \ldots, t_z \in T$ in $\mathbb{R}^2$ so that we can shatter the set. Each t_i will have $z!$ vertices such that for each trajectory t_i it has $t!/2$ vertices where each vertex takes part in a permutation of $t_1, t_2, \ldots, t_z$ on the line $y = x$ and similarly $t!/2$ vertices on the line $y = x + 1$ partaking in their own set of permutations. Between the two lines every permutation can be constructed. The corner of a rectangle can then cut off a subset of each permutation independendently of intersecting another subset and since all possible 2^z subsets are contained in the permutations, every subset of trajectories can be induced. Thus if we have no bound on k, we have no bound on the VC-dimension of $(T, \mathcal{R}_2)$. □

6 SCALABLE ALGORITHMS FOR FINDING TRAJECTORY ANOMALIES

We next describe how to efficiently scan over the trajectory range spaces to efficiently find ε-approximate spatial scan anomalies on the various range spaces defined for trajectories, and statistical discrepancy functions Φ. In the case of the flux and partial models, we provide new direct reductions to the point-set based scanning algorithms. For the full model these reductions are not possible, and we require the development of several new insights – different ones for each scanning shape. In particular, for scanning under the full model with disks (perhaps the most intuitive definition) we develop new ways (the MultiScale Disk approach) to represent and approximate the range space which becomes much more efficient that what was even previously known about point-set based scanning.

6.1 Reductions for Partial and Flux Models

We first describe two reductions for the flux model and partial range model to algorithms for scanning over points instead of trajectories.

Flux model reduction. For the flux model, the reduction starts again by sampling trajectory subsets $S, N \subset T$. Now we convert each trajectory $t \in S$ (or in N) to a point set in S_P (or N_P) as follows. We first convert every trajectory t into only its two endpoints p_1 and p_k and place both of these points in S_P (or in N_P). In S we require r (recorded) and b (baseline) values, and we only focused on the simpler variant which considers the linear model

$\phi(r(C), b(C)) = |r(C) - b(C)|$. Then we set $b(p_k) = b(t)$, $r(p_k) = -r(t)$ and $b(p_1) = -b(t)$, $r(p_1) = r(t)$. Note now that if both $p_1, p_k \in C$, then the total contribution $r(C)$ and $b(C)$ is 0; the points cancel each other out. When only $p_1 \in C$, then the contribution is $r(p_1) - b(p_1) = r(t) - b(t)$ as desired, and if only $p_k \in C$, then the contribution is $r(p_k) - b(p_k) = -r(r) + b(t) = -(r(t) - b(t))$, also as desired.

Theorem 6.1. *A flux model scan statistic for the linear statistical discrepancy function Φ on trajectories can be reduced to a point-based scan statistic on the endpoints.*

Partial model reduction. For the partial range model, the key quantities for $\Phi(C)$ are $r(C)$ and $b(C)$, the fraction of all possible contributions from trajectories from the recorded and baseline sets. Since in the partial range model we restrict to parts of trajectories, independently of which trajectory they are part of, we can convert to a point set input as follows. Given the full sets of trajectories T, we denote the continuous set of points in these trajectories as X_T. We then take uniform (or weighted) samples of X_T to construct S_T and N_T.

Since the contribution of a point in S_T towards $\Phi(C)$ is independent of the contribution of other points on the trajectory (unlike the full model), running a point-based scanning model on S_T, N_T will return the same $\Phi(C)$ value for any C as the trajectory-based partial range model.

Theorem 6.2. *A partial range model scan statistic on trajectories can be reduced to a point-based scan statistic on a uniform sample over the trajectory by arclength.*

6.2 Scanning under the Full Model

There are three major challenges in extending scanning algorithms to the full model – even after first converting each trajectory t into a point set P'_t of size k. The resulting approaches are multi-faceted, and different for each scanning shape, and summarized in Table 2.

First, in order to obtain the runtime bounds of point-based two-level sampling algorithms, the sets N and S, were of size roughly $n = \nu/\varepsilon$ and $s = \nu/(2\varepsilon^2)$ in the point-based model, now need to be of size $n_k = n \cdot k$ and $s_k = s \cdot k$, respectively. Each object placed into the "net" or "sample" set now is required to be a set of k points (on average) from each of the trajectories sampled. The scanning algorithms have linear time in s_k and moderate polynomial (degree 1 to 3) in n_k, so this increases the runtimes beyond the point-based setting by a moderate polynomial factor in k. The results for the various spatial approximation techniques are summarized in Table 1. We will demonstrate that for halfspaces and rectangular ranges, this increase is tolerable if the right α-spatial approximation is used. But for disks the resulting algorithms are not yet tractable, and so we design a new multi-level approximate scanning approach which works in tandem with the α-spatial approximation.

Second, extra bookkeeping is required to maintain which point sets P'_t are already intersecting a shape C during the scanning process–so that a trajectory is not overcounted when multiple points intersect C. For this we use a global integer counter array S.counter, where a non-zero counter serves as an indicator that the trajectory $t \in S$ is in the shape C in question. We maintain and only update Φ when a counter toggles between zero and non-zero. This extra bookkeeping may seem a trivial change, but it prevents the use of some approaches, in particular for rectangle scanning.

Third, to extend the approximation guarantee [21, 22] based on two-level sampling to this setting, we need to bound the VC-dimension ν of the range space $(\mathcal{T}, \mathcal{A}_C)$. In Section 5 we showed that for halfspaces, disks, and rectangles, if there is no bound on k, then the VC-dimension is unbounded. However, we also showed for all of these shapes that when each trajectory is represented by a point sets of size k (the ground set is $\mathbb{R}^{d \times k}$), then the VC-dimension is only $O(d \log k)$ in d dimensions. That is when the objects are point sets of size at most k, they only increase the VC-dimension by a benign $\log k$ factor, and hence the coreset sizes for n_k and s_k. These sample bounds are incorporated into Table 3.

Runtime Overview in the Full Model

SSS runtimes for the full model vary by shape, error parameters, and methods of spatial approximation. From the scanning perspective they depend on $n_k = nk$ and $s_k = sk$, shown in the middle column of Table 2. Parameters n is the small net size, and s is the large sample size are described in Table 3 for allowing ε-error in Φ. The spatial approximation size k is described in Table 1, and depends on the method used to achieve an α-approximation.

The best runtime bounds for ε-error on Φ and α-spatial error are shown in the right column of Table 2.

Method	Size = k	Spatial Error	Shapes
Convex Hull	m	0	$\mathcal{H}_2$ (halfspaces)
DP Algorithm	m	α	$\mathcal{H}_2$ (halfspaces)
Approx Hull	$O(1/\sqrt{\alpha})$	α	$\mathcal{H}_2$ (halfspaces)
Grid Kernel	$O(L/(r^{1/4}\alpha^{3/4}))$	α	$\mathcal{C}_r$ (disks)
Random Sample	$O\left((L/\alpha)\log\frac{L}{\alpha}\right)$	α	All
Even	$O\left(L/\alpha\right)$	α	All
Gridding	$O(L/\alpha)$	α	All

Table 1. The α-spatial approximation bounds for trajectory coreset algorithms with output of size k. Trajectories are of length L and with m waypoints, and all in a domain $[0,1]\times[0,1]$. Derivations of bounds in Section 4.

Shape	Runtime	Best Bounds in $1/\alpha$ & $1/\varepsilon$
Rectangles	$O(n_k^3 s_k)$	$\tilde{O}\left(\frac{L}{\alpha^4}\frac{1}{\varepsilon^2}\log(\frac{L}{\varepsilon\alpha})\right)$
Halfspaces	$O(n_k s_k \log n_k)$	$\tilde{O}\left(\frac{1}{\alpha}\frac{1}{\varepsilon^3}\log^3(\frac{1}{\varepsilon\alpha})\right)$
Disks	$O(n_k^2 s_k \log n_k)$	$\tilde{O}\left(\left(\frac{L}{\alpha}\right)^3\frac{1}{\varepsilon^4}\log^4(\frac{L}{\varepsilon\alpha})\right)$

Table 2. Overall algorithm runtimes in terms of sample size ($n_k = nk$ and $s_k = sk$) or error parameters (statistical error ε, spatial error α, and arclength L). $\tilde{O}(\cdot)$ hides log log factors and is for constant probability of success. Derivations of bounds are in Section 6.

	Sample Size
Sparse Net N	$n = O\left(\frac{\log k}{\varepsilon}\log\frac{\log k}{\varepsilon\delta}\right)$
Dense Sample S	$s = O\left(\frac{1}{\varepsilon^2}(\log k + \log\frac{1}{\delta}\right)$

Table 3. Sample size bounds to obtain $|\Phi(C^*) - \Phi(\hat{C})| \le \varepsilon$ with probability $1-\delta$ where C^* is the true maximum and $\hat{C}$ is the found approximation or trajectories of size k. Derivations of bounds in Sections 5 and Section 6.2.

Rectangles. We extend a $O(n^4 + s)$ time algorithm for scanning rectangles [20], which in our case becomes $O(\frac{1}{\alpha^3}s_k)$. The faster algorithms from that paper [20] (taking $O(n^3)$ and $O(n^2 \log\log n)$ time) do not extend because they require a special decomposition for implicit processing of the ranges which cannot accommodate the maintenance of the counter.

This algorithm defines a non-uniform grid by using a scan line in the x and y direction to ensure each row and column has at most εs trajectories or has at least α width whichever is larger. We recommend the Gridding approach for an α-approximation since then many of these coordinates are duplicated reducing the effective grid size. Then all of the points in S (of size s_k) are mapped into the appropriate grid cells, and duplicates for the

same trajectory can be removed. Then we consider each of the at most n_k^4 rectangles defined on this grid. We can scan them efficiently by fixing every possible upper, bottom, and left side of the rectangle (there are $O(n_k^3)$ combinations), and then sliding the right side of the rectangle from the left edge until it hits the end of the grid. The entire scanning of the right edge updates the counters at most s_k times. So the total runtime is $O(n_k^3 s_k)$.

Spatially-approximated Rectangles. An important optimization for rectangles takes advantage of the α-spatial approximation. Instead of considering all n_k^3 rectangles, we only need to consider endpoints differing by at least $1/\alpha$, there are no more than $O(1/\alpha^3)$ of these. Thus the total runtime becomes $O(s_k/\alpha^3)$. This variant is used in experiments.

We can also add a restriction where we short circuit the algorithm if the resulting subgrid will have height or width greater than some set value. This restriction can significantly decrease the runtime.

Halfspaces. To create approximate sets for each trajectory we use the Convex Hull and the Approx Hull methods to reduce to size k point sets. The former induces no spatial error, and the later provides and absolute bound on the size k (at $k = O(1/\sqrt{\alpha})$).

Let n_k represent the total number of net points required, with n trajectories sampled into N and then on average requiring k points P_t' to approximate. Using a combinatorial arrangement view of halfspaces and point sets, these can be scanned in $O(ns)$ time [7], using advanced techniques from computational geometry. This can be converted into a $O(n_k s_k)$ bound in our setting. We review a simpler model here that takes $O(ns \log n)$ time on point sets, and $O(n_k s_k \log n_k)$ in our setting. Let N_k be the n_k points approximating the n trajectories in N, and similarly let S_k be the s_k points from S. For each point $q \in N_k$ sort all s_k points in S_k radially around q. Then consider halfspaces with q on the boundary, and scan through them radially around q updating the counters and Φ as necessary.

Higher dimensional halfspaces can be reduced to lower dimensional halfplane problems by doing an affine projection down into the 2-dimensional space. Using the halfplane algorithm to solve these problems gives a runtime of $O(n_k^{d-1} s_k)$. An optimization we call the "Hull Trick" does an additional pruning step after each projection to 2 dimensions; it creates the convex hull in 2-dimensions, and only retains the points on the hull before scanning.

Disks. After approximating trajectories by point sets, disk-scanning can be implemented as halfspace scanning for $d = 3$ using the Veronese map. This gives runtime bounds of $O(n_k^2 s_k)$.

However, these shapes are significantly more sensitive to the α-spatial approximation technique, especially if there are long edges. For instance in our experimental data we found disks might need 10 or even 50 times more points to generate an α-approximation for a trajectory (see Table 4). As such we design a new scanning method that works in concert with the Grid Kernel approach.

MultiScale Disks. The previous disk scanning algorithm is not tractable or scalable, but we can combine a large number of tricks to handle these issues.

We consider scanning over disks with radius in a range $[r_{\min}, r_{\max}]$, specifically where $r_{\max} = r_{\min} 2^z$ for a small integer z. We decompose this into z subranges, so in each the radius is in a range $[r, 2r]$ and handle these subranges independently. In experiments, we set $z = 4$ (or in some cases $z = 5$) which covers a natural and intuitive set of radii.

Then we scan in concert with the α-spatial approximation method Grid Kernel using parameter r. This reduces the size of each trajectory approximation k, especially for large r; see Table 4. Assuming the data has been mapped to a $[0, 1] \times [0, 1]$ range, we can create a $1/r \times 1/r$ grid, with cell edge length r. We now consider the $O(1/r^2)$ 5×5 subsets of grid cells where the center cell has at least one point in the N_k set of points. Then for each such center cell point q (a "pivot" point), we consider only disks with q on the boundary, and other boundary points among those in this 5×5 subset of N_k. Especially for small r, the restriction to this local subset of the data greatly

reduces the number of points which are considered when constructing disks with respect to each pivot q: from all n_k net points to only those points from N_k inside this 5×5 subset. Furthermore this also significantly reduces the number of sample points in S_k that must be tested for inclusion in each disk.

Together the effects of lower bounding the radius (to reduce k), and upper bounding the radius (to effectively reduce the number of considered regions and the number of points to scan s_k for each pivot), makes every range $[r, 2r]$ of this MultiScale Disk approach quite tractable.

This method can be further improved with the Hull Trick, especially for complex trajectories. Focusing on the disk scanning algorithm for all disks passing through the pivot q; this maps to the problem of scanning all 2d halfspaces in a different lifted parameter space, as discussed in Section 4. This reduction allows us to apply the Hull Trick from the Halfspace algorithm section or another halfspace coreset method from Table 1 to significantly reduce the number of points that must be considered.

7 EXPERIMENTS

In this section we show scalability of all of the proposed algorithms, and their effects on the statistical power of the scan statistics. We show for the partial and flux models, through our new direct reductions to recent work, we can compute scan statistics in a scalable yet statistically powerful way. For the full model, the most naive reductions to existing methods are not viable, but our proposed geometric and algorithmic observations for each scanning shape lead to significant speed ups.

We demonstrate these improvements with two types of measurements. The first is directly showing the runtime of the algorithms as a function of either the statistical error parameter ε or the spatial error parameter α. We also measure the *discrepancy error*, where we attempt to find a large $\Phi(C)$ value, and we show at increasing parameter settings how the largest $\Phi(C)$ region found approaches $\arg\max_{C \in \mathcal{C}} \Phi(C)$ as a function of the runtime. This demonstrates that these algorithms are not just fast, but they become statistically powerful in tractable runtimes.

7.1 Setup and Data

All of our code and the scripts used to generate experiments are publicly available on github and our project website [2]. All of our code is written in C++ with an easy to use python wrapper that we hope will allow for researchers to apply these algorithms to many other data sets. Current algorithm implementations are serial, but the algorithms are embarrassingly parallel, so converting to a parallel implementation would be trivial.

Experiments were conducted using the python wrapper on computers with an Intel Core i7-3820 and 64GB of memory. It ran Ubuntu 14.04 with kernel version 3.13.0-147. Code was compiled with GCC version 8.1.0, Boost version: 1.69.0, and Python version 3.4.3. Experiments were run in successive fashion on a per node basis. No experiments were run together on the same node at the same time to minimize the effect of other processes on run time.

We perform experiments on two large trajectory data sets, described next, which have very different conditions. One is diverse, and has trajectories of significantly different sizes and lengths, but the overlap is clustered. The other has more uniform trajectory sizes and lengths, but they are all intermingled.

Open Street Map Trace Data. This data set is our default, and consists of a subset of close to 6 million traces from Open Street maps with 1.282 billion total waypoints. We restrict the set to ones contained completely inside of a rectangular region in Europe with latitude and longitude of $[35, 60] \times [-10, 25]$. The large extent of these traces means that there are comparatively sparse regions corresponding to rural areas and also densely packed regions such as cities. Many trajectories are restricted to small regions compared to the full domain size.

[2] https://mmath.dev/pyscan

Name	Beijing #Pts	OSM #Pts
Grid Kernel r = .1km	489	117
Grid Kernel r = .2km	286	69.6
Grid Kernel r = .4km	191	47.3
Grid Kernel r = .8km	134	34.1
Grid Kernel r = 1.6km	96.9	25.4
Grid Kernel r = 3.2km	71.8	19.5
DP	24.6	8.74
Approx Hull	7.27	3.35
Convex Hull	10.0	9.50
Gridding	389	106
Even	395	110.9

Table 4. Setting $\alpha = 0.1km$ shows the average value of k (size of approximation) for α-approximation algorithms.

Beijing Taxi Data. This is a densely packed and highly overlapping set of roughly 3 million trajectories with 129 million total waypoints collected from taxi drivers in Beijing [18]. This data set has a very high sampling frequency per trajectory with 75% of the points being collected less than 1 minute apart. Many roads have been driven over by hundreds to even thousands of separate traces. Since small regions can be densely packed with trajectories, local scanning approaches that restrict the region of interest to be of small size work comparatively worse. There are comparatively few sparse regions. A set of only 25 of these trajectories are shown in the extended Full Range Model Example. We have restricted the trajectories to be confined to a region with latitude and longitude of $[39.788, 40.094] \times [116.15, 116.612]$.

7.2 Spatial Approximation Size

We first empirically evaluate how well the various α-spatial approximation algorithms work on average. We simplify all trajectories in each data set such that α is set to $0.1km$ – about one city block (when normalized to $[0, 1] \times [0, 1]$, for Beijing this is $\alpha = 1/500$ and for OSM it is $1/\alpha = 30{,}000$). We show the average value of k for the various algorithms in Table 4. We observe that the OSM data is much easier to approximate than the Beijing data – and in fact we use a larger α as default later which would lead to even smaller k values on average. Next we observe that the methods designed for halfspaces (Convex Hull, Approx Hull, and DP) generate very small average values of k of 3 to 10 (or 25 for DP on Beijing). For large values of r, Grid Kernel can start to approach k of around 20 or 30 for OSM, but otherwise the methods for rectangles and disks (Grid Kernel large r, Gridding, and Even) have larger average k of around 100 for OSM, and over 300 for Beijing. This factor of 10 difference in k can cause dramatic slow-downs in the experiments.

7.3 Scalability Experiments

We next demonstrate how various parameters affect the runtime, and how various algorithms compare in scalability. The runtime as a function of $1/\varepsilon$ for flux model and partial model algorithms are shown in Figure 5; setting $s = 1/(2\varepsilon)^2$ and $n = 1/\varepsilon$ as suggested [22]. The partial data is sampled using the Even mechanism. Some runtime curves become linear as S becomes the entire data set (12 million), and only n increases.

We observe that the rectangle-based algorithms are quite scalable, and are able to set $1/\varepsilon \approx 1{,}500$ and still complete in about 1 minute. The generic two-level sampling algorithm for halfspace scanning is almost as scalable, and performs better than rectangle when the “Ham" coreset [21] is used. However, the scanning algorithms for disks require several minutes to deal with even $1/\varepsilon \approx 100$, and becoming intractable for anything larger. In

summary, using our new reductions to the point-based algorithms, under the partial and flux models, rectangles and halfplanes scanning for trajectory anomalies is already scalable.

Full Scanning on OSM. The algorithms for the full intersection model are significantly more nuanced. These runtimes are shown as a function of $1/\varepsilon$ and $1/\alpha$ in Figure 6. While the other is varied we set $1/\alpha = 6000$ and $1/\varepsilon = 100$ (for Disk and for Best in Class) and $1/\varepsilon = 20$ (for Baseline and for Halfplane). Rectangles are restricted to having max side length less than $1/150$ to make them have roughly the same size as the restricted disk variants in all OSM plots. Note that all runtimes are on a logarithmic scale.

The Baseline rows shows the scalability of the basic algorithms using Even α-spatial approximation. Again, for halfspace and rectangle scanning, the runtimes are tolerable, but the disk scanning becomes already intractable for moderate parameter values. Note that now the runtimes are quite noisy due to the high variance in the trajectories length in the OSM data – even if k is small on average, for some trajectories sampled it might be quite large.

The Halfplane row shows the difference in runtimes for the All Waypoints, Convex Hull, and Approx Hull methods for spatial approximation. The All Waypoints and Convex Hull have 0 spatial error (by Lemma 4.1), so horizontal lines with error bars are shown. Convex Hull provides a dramatic improvement over All Waypoints, of 2 to 3 orders of magnitude (i.e., from several minutes to less than a second). The Approx Hull approach shows smaller but tangible improvement over using all hull points, but adds some spatial error.

The Disk row shows another dramatic improvement in scalability as we restrict the radii considered to $r \in [1/6000, 1/300]$ and apply other improvements. With no bound on the radius, the (Disk + Even) algorithms are intractable. Still invoking Even, but using a $1/r \times 1/r$ grid to prune the scanning to a radius range bound (Small Disk + Even) allows for moderate values of $1/\varepsilon$ and $1/\alpha$ to complete in minutes. But combining the MultiScale Disk approach with the adaptive Grid Kernel spatial approximation allows the same moderate error parameter runs to complete in 10s of seconds, and in about 1 or 2 minutes this approach can scale to $1/\varepsilon \approx 400$. Adding the Hull Trick does not induce significant gains here.

Finally in the Best in Class, row we show the best algorithms runtime for each scanning shape: MultiScale Disks for disks, Approx Hull spatial approximation for halfplanes, and Even spatial approximation for rectangles. The improvement over the baseline for halfspaces and disks is dramatic. With these algorithms it is now possible to set $1/\alpha \approx 100{,}000$ and complete in about 10 seconds for any shape. Halfspaces and radius restricted disks can set $1/\varepsilon \approx 500$ or 1000 and complete in about a minute or two; they are now very scalable in $1/\varepsilon$. For Rectangles it

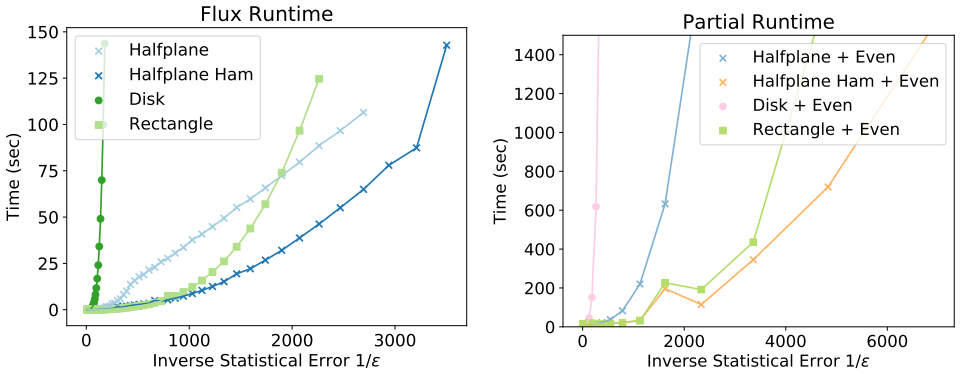


Fig. 5. Runtime for flux model algorithms (left) and partial model algorithms (right) in terms of input net size $1/\varepsilon$.

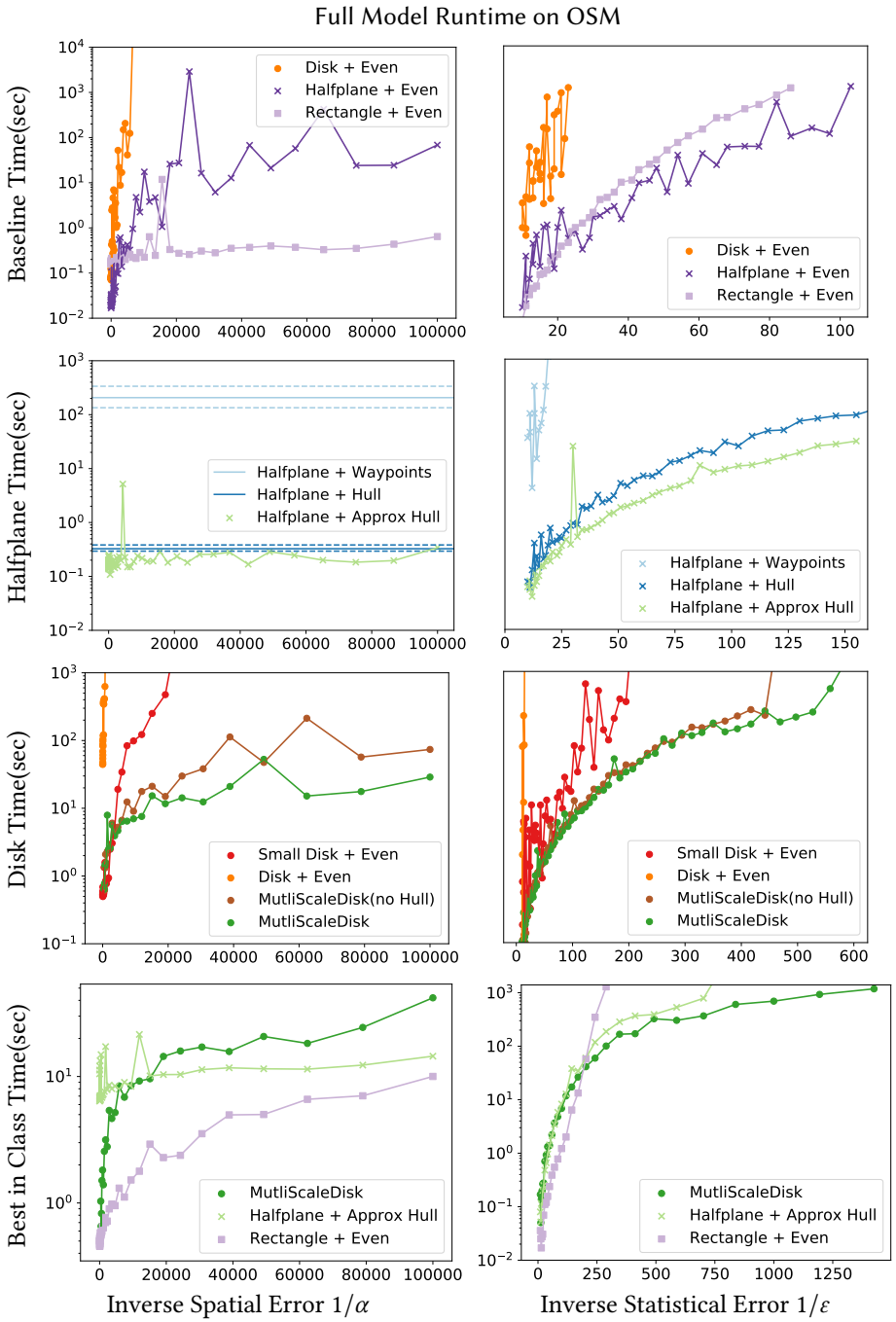


Fig. 6. Runtime for full model scanning as function of inverse spatial error, $1/\alpha$, or inverse statistical error, $1/\varepsilon$, on OSM data. The inverse spatial error and inverse statistical error act as a size parameter for our algorithm since our guarantees and runtime do not depend on the initial data size – only on the statistical or spatial resolution.

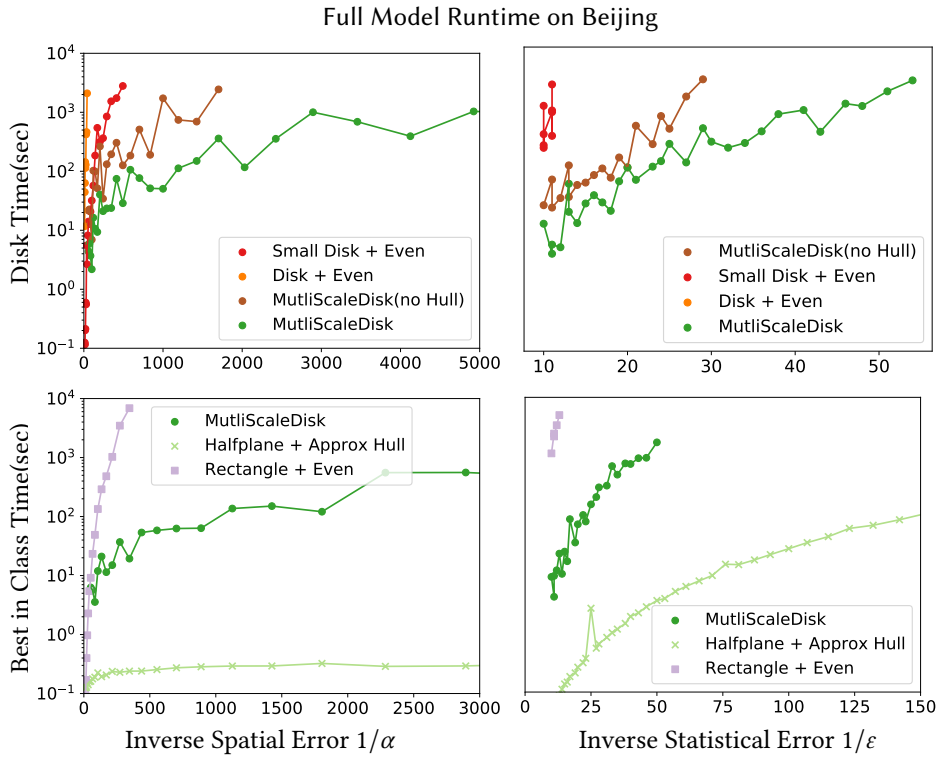


Fig. 7. Runtime for full model scanning as function of inverse spatial error, $1/\alpha$, or inverse statistical error, $1/\varepsilon$, on Beijing data.

can set $1/\varepsilon \approx 250$ and still complete in a minute or two. On the other hand, the algorithm for rectangles is more tolerant to very small values of α.

Full Scanning on Beijing. The runtimes for the algorithms under the full model on the Beijing data set are shown in Figure 7. We restrict $r \in [1/50, 1/500]$ for radius-restricted disks, and for rectangles side length is restricted to be less than $1/25$. The first (Disk) row shows the improvement when scanning with radius-restricted disks. Fixing $1/\varepsilon = 20$, only the MultiScale Disk scanning with Approx Hull and the Hull Trick can complete $1/\alpha > 2000$ in under an hour, and indeed can scale to $1/\alpha = 5000$ in that time. Here the consistent long trajectories greatly benefit from the extra Hull Trick pruning. Fixing $1/\alpha = 500$ only this disk scanning variant can reach a moderate $1/\varepsilon = 50$ in under an hour.

The second (Best in Class) row shows that now the rectangle scanning times are strictly worse than the radius restricted disk times. The halfspace algorithms have similar runtimes as with the OSM data, and are now much more scalable than the MultiDisk approach. Thus the tangled nature of the Beijing data affects the rectangle scanning the most and the halfspace the least.

7.4 Statistical Power Tests

In this section we measure the statistical power of these scanning algorithms. We fix the spatial layout of trajectories from the OSM or Beijing data sets, and introduce a spatially anomalous region under the various models by adjusting the r and b values of trajectories. Such regions (either a halfplane, disk, or rectangle to match the scanning region) are "planted" by choosing a shape $C \in \mathcal{C}$ and setting a desired rates for inside C as

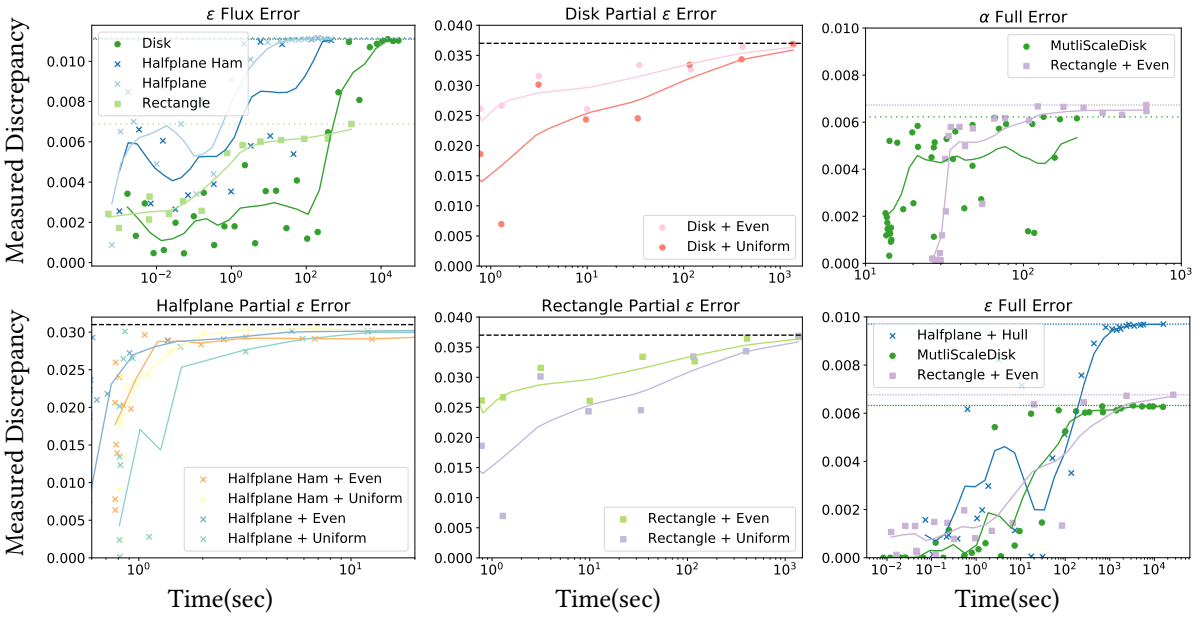


Fig. 8. Statistical power vs time for partial, flux, and full model on OSM data.

$q = r(C)/b(C)$, for all data $p = r(T \setminus C)/b(T \setminus C)$, and the anomaly size $f = b(C)/b(T)$. The optimal shape may be different (usually slightly shifted), and so we evaluate these tests by tracking the value $\max_{C \in \mathcal{C}} \Phi(C)$ (the "Measured Discrepancy") found for various parameter settings. When this maximum value plateaus, it indicates those parameters settings and runtime have high power and are sufficient to recover the anomalous shapes.

This process is much noisier than just measuring runtime as a function of parameter settings. So in plots we show many data points and fit a trend line using a local average. In the same plots, different scanning shapes naturally converge to distinct Measured Discrepancy values.

For all of these experiments we use a rate of $p = .5$ for the data outside of the planted region and a rate of $q = .8$ for the data inside of the region.

Flux Scanning Power. The time required to achieve statistical power for the flux scanning model is shown in Figure 8(Upper Left). It shows the Measured Discrepancy values for disk, rectangle, and halfspace algorithms under the flux model on the Beijing data. We fix the planted shape to contain 5% of the data ($f = 0.05$). As observed earlier, the disk regions are very slow to scan for under this model, and do not achieve high power until 2 hours of runtime. However, the rectangle and halfspace scanning algorithms converge to a high power setting in about 10 seconds to 1 minute – hence for the flux model, we recommend these shapes.

Partial Scanning Power. Figure 8(Lower Left, and Middle Column) also shows the time require to achieve high statistical power under the partial model on OSM data. We plant small ranges of size $f = 0.005$. Each scanning shape (rectangle, disk, halfplane) is in a separate chart. They all eventually achieve high statistical power, but the halfplane scanning algorithms only take about $1 - 2$ seconds, while the disk and rectangle algorithms require about 30 minutes. We can also observe that Even spatial approximation consistently converges faster than Random Sampling.

Full Scanning Power. Finally, Figure 8(Right Column) shows how long it takes to achieve high statistical power under the full model. We plant regions in the OSM data with size $f = 0.005$. We show the results of scaling runtime by varying $1/\varepsilon$ (setting $1/\alpha = 6000$) and varying $1/\alpha$ (setting $1/\varepsilon = 200$). For halfplanes, we use Convex Hull which has no spatial error, so it only has its runtime vary as a function of $1/\varepsilon$; and it achieves high power after about 1 hour. For rectangles, it achieves high power through larger $1/\varepsilon$ values in about 10 minutes in both $1/\varepsilon$ and $1/\alpha$ scaling. For radius-restricted disks, we show the MultiDisk scanning with Grid Kernel (and Hull Trick); it appears less tied to the choice of α as this setting is tied to the resolution of the grid approximation. For a fixed $1/\varepsilon = 200$, it sometimes, but does not consistently find a high score region indicating that the setting $1/\varepsilon = 200$ might be too small, but as $1/\varepsilon$ increases after about 10 minutes it achieves high power.

8 PREVIOUS AND RELATED WORK

Our algorithms build upon the recent two-level sampling framework for spatial scan statistics by Matheny *et al.* [20, 22]. This focused on making scalable ε-approximate spatial scan statistics over point sets. This line of work provides improvements over non-approximate variants [16, 17], in terms of the number samples, and the runtime of the scanning algorithms for the same shapes we study: halfspaces, disks, and rectangles. The introduction of two-level sample makes these approaches tractable, and better coresets or faster scanning makes them extremely efficient. However, these fast SSS methods only apply for point sets.

There exist other mechanisms for finding anomalous behavior among trajectories. These involve clustering by density and then identifying outliers, training models on pre-labeled data, or identifying sets trajectories with similar characteristics on predefined region families [5, 6, 15, 23, 27, 28, 32, 33]. These approaches do not specifically identify spatial regions from characteristics of the trajectories or compare against a background population.

A few papers have attempted to port scan statistics to trajectories, with goal of finding spatial anomalous regions. Pang *et al.* [25] discretized cities into grids and recorded traffic conditions in each cell. They then computed a likelihood ratio test over all sub-grids to detect the most anomalous region. And Liu *et al.* [19] partition a city into regions defined by the road network, and their adjacency defines a graph. Then spatial anomalous links in this graph are scanned over various times to find a time×region of high traffic. But neither of these approaches directly operate on the trajectories, and fix regions ahead of time which restricts the set and nature of possible anomalies.

An alternative approach (for point sets) removes the notion of shapes, and focuses on clustering the measured objects to find potential anomalies [9, 10, 12, 26, 30]. However, this approach already does not have guarantees about statistical accuracy for point sets, and the task of clustering trajectories appropriately (and in this case one should be concerned about not over-fitting) has its own set of challenges, which are beyond the scope of this paper.

9 CONCLUSION

We introduce three new models for quantifying spatial anomalies among large sets of trajectories using spatial scan statistics and defined by geometric shapes. These identify regions which exhibit high flux, a large percentage of the total arclength of a measured quantity, or a large percentage of a set of full trajectories of interest pass through that region. These models have numerous applications in traffic analysis, disease outbreak monitoring, epidemiology, and demography.

Through either combinatorial, geometric reductions, or new scanning algorithms, we are able to identify these anomalous regions efficiently on data sets containing millions of trajectories with billions of way points. This efficiency requires various insights tuned to the families of shapes we considered which include halfplanes, disks, and rectangles. This includes careful ways to approximate trajectories by point sets, and fast enumeration

methods. These approximations are backed by a theoretical analysis, which shows guaranteed bounded error in the spatial trajectory approximation (α) as well as in the measurement of the statistical quantities (ε). The runtime depends only on these parameters. In the first two models the best techniques are reductions to algorithms for point sets, while the model accounting for full trajectories requires significant new ideas. For the most natural anomaly shape of disks, our MultiscaleDisk technique provides the best efficiency and accuracy trade-off.

And most importantly, we also measure the statistical power of the scanning algorithms. That is, if we plant an anomalous region under each of the models, our scanning algorithms can with high probability, recover that region (or a similarly anomalous one) in tractable amounts of time. Even on the millions of trajectories, high statistical power is often achieved within minutes, or for more challenging variants, in hours. There are no previous approaches which have demonstrated any statistical power on these tasks.

Acknowledgments. We thank funding from NSF spanning awards CCF-1350888, ACI-1443046, CNS- 1514520, CNS-1564287, and IIS-1816149. Besides, Dong Xie is also supported by Microsoft Research Ph.D Fellowship.

REFERENCES

[1] Agarwal, D., Phillips, J. M., and Venkatasubramanian, S. The hunting of the bump: On maximizing statistical discrepancy. *SODA* (2006).
[2] Agarwal, P. K., and Erickson, J. Geometric range searching and its relatives, 1999.
[3] Agarwal, P. K., Har-Peled, S., and Varadarajan, K. R. Approximating extent measure of points. *J. ACM 51*, 4 (2004), 2004.
[4] Arias-Castro, E., Castro, R. M., Tánczos, E., and Wang, M. Distribution-free detection of structured anomalies: Permutation and rank-based scans. *JASA 113* (2018), 789–801.
[5] Baryannis, G., Tachmazidis, I., Batsakis, S., Antoniou, G., Alviano, M., Sellis, T., and Tsai, P.-W. A trajectory calculus for qualitative spatial reasoning using answer set programming. *Theory and Practice of Logic Programming 18* (2018), 355–371.
[6] Dee, H. M., and Velastin, S. A. How close are we to solving the problem of automated visual surveillance? *Machine Vision and Applications 19*, 5 (Oct 2008), 329–343.
[7] Dobkin, D. P., Eppstein, D., and Mitchell, D. P. Computing the discrepancy with applications to supersampling patterns. *ACM Trans. Graph. 15*, 4 (Oct. 1996), 354–376.
[8] Driemel, A., Phillips, J. M., and Psarros, I. On the VC dimension of metric balls under frechet and hausdorff distances. In *International Symposium on Computational Geometry* (2019).
[9] Duczmal, L., and Assunção, R. A simulated annealing strategy for the detection of arbitrarily shaped spatial clusters. *Computational Statistics & Data Analysis 45*, 2 (2004), 269 – 286.
[10] Duczmal, L., Cançado, A. L., Takahashi, R. H., and Bessegato, L. F. A genetic algorithm for irregularly shaped spatial scan statistics. *Computational Statistics & Data Analysis 52*, 1 (2007), 43 – 52.
[11] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (1996), KDD'96, AAAI Press, pp. 226–231.
[12] Fitzpatrick, D., Ni, Y., and Neill, D. Support vector subset scan for spatial outbreak detection. *Online Journal of Public Health Informatics 9*, 1 (2017).
[13] Haussler, D., and Welzl, E. epsilon-nets and simplex range queries. *Discrete and Computational Geometry 2* (1987), 127–151.
[14] Hershberger, J., and Snoeyink, J. Speeding up the Douglas-Peucker line-simplification algorithm. Tech. rep., UBC, 1992.
[15] Hu, W., Xiao, X., Fu, Z., Xie, D., Tan, T., and Maybank, S. A system for learning statistical motion patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence 28*, 9 (Sept 2006), 1450–1464.
[16] Kulldorff, M. A spatial scan statistic. *Communications in Statistics: Theory and Methods 26* (1997), 1481–1496.
[17] Kulldorff, M. *SatScan User Guide*, 9.6 ed. http://www.satscan.org/, 2018.
[18] Lian, J., and Zhang, L. One-month beijing taxi gps trajectory dataset with taxi ids and vehicle status. In *Proceedings of the First Workshop on Data Acquisition To Analysis* (New York, NY, USA, 2018), DATA '18, ACM, pp. 3–4.
[19] Liu, W., Zheng, Y., Chawla, S., Yuan, J., and Xing, X. Discovering spatio-temporal causal interactions in traffic data streams. In *SIGSPATIAL* (New York, NY, USA, 2011), KDD '11, ACM.
[20] Matheny, M., and Phillips, J. M. Computing approximate statistical discrepancy. *ISAAC* (2018).
[21] Matheny, M., and Phillips, J. M. Practical low-dimensional halfspace range space sampling. *ESA* (2018).
[22] Matheny, M., Singh, R., Zhang, L., Wang, K., and Phillips, J. M. Scalable spatial scan statistics through sampling. In *SIGSPATIAL* (2016).

[23] Morris, B. T., and Trivedi, M. M. A survey of vision-based trajectory learning and analysis for surveillance. *IEEE Trans. Circuits and Sys for Video Tech. 18* (Aug 2008), 1114–1127.
[24] Neill, D. B., and Moore, A. W. Rapid detection of significant spatial clusters. In *KDD* (2004).
[25] Pang, L. X., Chawla, S., Liu, W., and Zheng, Y. On mining anomalous patterns in road traffic streams. In *International Conference on Advanced Data Mining and Applications* (2011).
[26] Patil, G. P., and Taillie, C. Upper level set scan statistic for detecting arbitrarily shaped hotspots. *Environmental and Ecological Statistics 11*, 2 (Jun 2004), 183–197.
[27] Piciarelli, C., Micheloni, C., and Foresti, G. L. Trajectory-based anomalous event detection. *IEEE Trans. Circuits and Sys for Video Tech. 18* (Nov 2008), 1544–1554.
[28] Shang, S., Chen, L., Wei, Z., Jensen, C. S., Zheng, K., and Kalnis, P. Parallel trajectory similarity joins in spatial networks. *The VLDB Journal* (2018).
[29] Smith, K. Utah population database. https://uofuhealth.utah.edu/huntsman/utah-population-database/.
[30] Tango, T., and Takahashi, K. A flexibly shaped spatial scan statistic for detecting clusters. *Inter. J. of Health Geographics 4* (2005), 11.
[31] Vapnik, V., and Chervonenkis, A. On the uniform convergence of relative frequencies of events to their probabilities. *Theo. of Prob and App 16* (1971), 264–280.
[32] Vlachos, M., Kollios, G., and Gunopulos, D. Discovering similar multidimensional trajectories. In *Proceedings of IEEE International Conference on Data Engineering* (2002).
[33] Yankov, D., Keogh, E., and Rebbapragada, U. Disk aware discord discovery: Finding unusual time series in terabyte sized datasets. *Knowl. Inf. Syst. 17*, 2 (Nov. 2008), 241–262.