A Space-Optimal Data-Stream Algorithm for Coresets in the Plane[*]

Pankaj K. Agarwal[†] Hai Yu[‡]

Abstract

Given a point set $P \subseteq \mathbb{R}^2$, a subset $Q \subseteq P$ is an ε-kernel of P if for every slab W containing Q, the $(1+\varepsilon)$-expansion of W also contains P. We present a data-stream algorithm for maintaining an ε-kernel of a stream of points in $\mathbb{R}^2$ that uses $O(1/\sqrt{\varepsilon})$ space and takes $O(\log(1/\varepsilon))$ amortized time to process each point. This is the first space-optimal data-stream algorithm for this problem. As a consequence, we obtain improved data-stream approximation algorithms for other extent measures, such as width, robust kernels, as well as ε-kernels in higher dimensions.

1 Introduction

Data stream computation has recently attracted much attention because of many real-world applications; see [5, 19, 21] and the references therein. In the data stream model, the input elements arrive one by one, and one has to maintain the value of a certain function on the elements seen so far. Because the input size is huge and the accessible memory is typically limited, the algorithm can only afford to store a small sketch of the input. For this reason, the answer returned by the algorithm is approximate in general. In geometric data streams, the input usually consists of a set of points in $\mathbb{R}^d$, and the task can be as simple as computing the median or as complex as computing the convex hull or some other geometric structures. The goal is to design a data-stream algorithm that accomplishes the task with a prescribed accuracy, using as little space as possible. Space-efficient data-stream algorithms for numerous geometric problems have been developed in recent years [1, 3, 6, 9, 11, 14, 15, 16, 22].

In this paper we are interested in space-efficient data-stream algorithms for maintaining the so-called *ε-kernels* of a stream of points in $\mathbb{R}^2$. Let P be a set of n points in $\mathbb{R}^2$. We denote by $P[u]$ the extreme point of P along a direction $u \in \mathbb{S}^1$, that is, $P[u] = \arg\max_{p \in P} \langle p, u \rangle$. The *directional width* of P along a direction u is defined by $\omega(P, u) = \langle P[u] - P[-u], u \rangle$. For a parameter $\varepsilon > 0$, a subset $Q \subseteq P$ is an *ε-kernel* of P if for any $u \in \mathbb{S}^1$,

$$\langle P[u] - Q[u], u \rangle \leq \varepsilon \cdot \omega(P, u).$$

[*] Work on this paper has been supported by NSF under grants CCR-00-86013, EIA-01-31905, CCR-02-04118, and DEB-04-25465, by ARO grants W911NF-04-1-0278 and DAAD19-03-1-0352, and by a grant from the U.S.–Israel Binational Science Foundation.

[†] Department of Computer Science, Duke University, Durham, NC 27708, USA; pankaj@cs.duke.edu

[‡] Department of Computer Science, Duke University, Durham, NC 27708, USA; fishhai@cs.duke.edu

Note that this implies $(1-2\varepsilon)\cdot\omega(P,u) \le \omega(Q,u) \le \omega(P,u)$, i.e., $\omega(Q,u)$ is a $(1-2\varepsilon)$-approximation to $\omega(P,u)$. The notion of ε-kernels was introduced by Agarwal *et al.* [1] as a generic method for designing fast approximation algorithms for computing extent measures of a point set. In this paper we present a data-stream algorithm for maintaining an ε-kernel of a stream of points in $\mathbb{R}^2$ that uses optimal space.

Related work. Agarwal *et al.* [1] developed the first data-stream algorithm for computing ε-kernels in $\mathbb{R}^d$, by applying a variant of the logarithmic method of Bentley and Saxe [8] (sometimes also called merge-and-reduce in the streaming setting). Their algorithm requires $O\left((1/\varepsilon^{(d-1)/2})\log^d n\right)$ space and takes $O(1/\varepsilon^{3(d-1)/2})$ time to process each point. Here n denotes the total number of points in the stream. This algorithm leads to fast approximation algorithms for maintaining various extent measures under the streaming model. A natural question suggested by their result is whether the space bound can be made independent of n. Chan [9] answered this question in the affirmative by proposing an algorithm that uses only $O\big((1/\varepsilon^{d-3/2})\log^d(1/\varepsilon)\big)$ space and updates the ε-kernel in $O(1/\sqrt{\varepsilon})$ time when a new point arrives.

Specialized to $\mathbb{R}^2$, Chan's algorithm is able to maintain an ε-kernel of a stream of points with $O\big((1/\sqrt{\varepsilon})\log^2(1/\varepsilon)\big)$ space and $O(1/\sqrt{\varepsilon})$ update time. Independently, Hershberger and Suri [18] discovered a data-stream algorithm that maintains a subset Q of $O(1/\sqrt{\varepsilon})$ points in the stream S such that the Hausdorff distance between $\mathrm{conv}(S)$ and $\mathrm{conv}(Q)$ is at most $\varepsilon \cdot \mathrm{diam}(S)$. The time to process each point is $O(\log^2(1/\varepsilon))$, which can be improved to $O(\log(1/\varepsilon))$ in the RAM model. The notion of approximation provided by the above Hausdorff distance measure is weaker than the approximation provided by ε-kernels, and in general ε-kernels provide much more faithful approximations (see Figure 1). For example, their algorithm can be used to maintain the diameter of the point set, but unlike ε-kernels, it does not lead to efficient data-stream approximation algorithms for many other extent measures, including approximating width, minimum enclosing rectangle, etc.

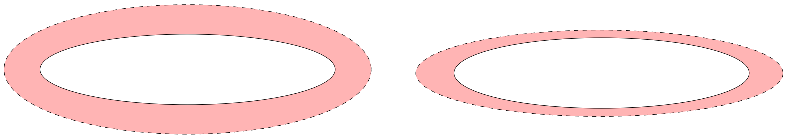

Figure 1. The approximation qualities provided by the Hausdorff distance measure (left) and by ε-kernels (right). The shaded regions correspond to "uncertainty" areas allowed by the approximation error.

Besides the data stream model, there has also been work on geometric computations under the stronger sliding-window model, in which one is only interested in the latest N points in the stream for a prescribed parameter N. We refer the reader to [10, 13] for results on maintaining the diameter and width of streaming points under this model.

Our results. In this paper we obtain the first space-optimal data-stream algorithm for maintaining ε-kernels of streaming points in $\mathbb{R}^2$. It uses $O(1/\sqrt{\varepsilon})$ space and processes each point in $O(\log(1/\varepsilon))$ time. The $O(1/\sqrt{\varepsilon})$ bound on the space complexity is asymptotically optimal because any ε-kernel of sufficiently many points uniformly distributed on a circle must have size $\Omega(1/\sqrt{\varepsilon})$. However, we do not yet know whether our $O(\log(1/\varepsilon))$ update time is optimal if only $O(1/\sqrt{\varepsilon})$ space is allowed. Our result also improves the data-stream algorithms for maintaining ε-kernels in $\mathbb{R}^d$; the space bound is $O(1/\varepsilon^{d-3/2})$ and the update time is $O(\log(1/\varepsilon))$.

Our results immediately lead to data-stream algorithms that maintain the approximate width and several other extent measures in $\mathbb{R}^2$ with $O(1/\sqrt{\varepsilon})$ space. It also leads to a space-optimal data-stream algorithm for maintaining the so-called (k, ε)-kernels [17, 2] (which is useful for dealing with outliers) in $\mathbb{R}^2$; the space bound is $O(k/\sqrt{\varepsilon})$ and the update time is $O(k \cdot \log(1/\varepsilon))$. For $\mathbb{R}^d$, the space bound becomes $O(k/\varepsilon^{d-3/2})$ and the update time becomes $O(k \cdot \log(1/\varepsilon))$. These results in turn lead to fast data-stream approximation algorithms for maintaining robust extent measures in fixed dimensions.

Roadmap. The rest of the paper is organized as follows. The next section reviews some useful definitions and facts, and introduces a tool called minimal additive kernel that will be useful in the design and analysis of our algorithm. Section 3 describes the outline of the overall algorithm, which is similar to that of Chan [9]. Sections 4, 5 and 6 present the details of the space-optimal data-stream algorithm for maintaining ε-kernels in the plane. Section 7 describes a few interesting implications of our result.

2 Kernels and Additive Kernels

Since we primarily describe the algorithm for the planar case, we state all the relevant definitions and facts in their planar form for the sake of simplicity. We represent a direction $u \in \mathbb{S}^1$ by its orientation, a real value in the interval $[0, 2\pi]$: For any $\alpha \in [0, 2\pi]$, we write $\mathsf{u}_\alpha = (\sin\alpha, \cos\alpha) \in \mathbb{S}^1$. For $\alpha, \beta \in [0, 2\pi]$, let $[\alpha, \beta]$ denote the (closed) angular interval from α to β in counterclockwise direction. We also write $P[\mathsf{u}_\alpha]$ as $P[\alpha]$ and $\omega(P, \mathsf{u}_\alpha)$ as $\omega(P, \alpha)$, and define

$$P[\alpha, \beta] = \bigl\{P[\gamma] \mid \gamma \in [\alpha, \beta]\bigr\}.$$

With these notations, we will often not distinguish between a direction in $\mathbb{S}^1$ and its orientation in $[0, 2\pi]$.

Additive δ-kernels. Let P be a set of points in $\mathbb{R}^2$. For a parameter $\delta > 0$, a subset $Q \subseteq P$ is called an *additive δ-kernel* if for all $\gamma \in [0, 2\pi]$,

$$\langle P[\gamma] - Q[\gamma], \mathsf{u}_\gamma \rangle \leq \delta. \tag{1}$$

If (1) is required to hold only for an interval $I \subseteq [0, 2\pi]$, we say that Q is an additive δ-kernel of P *with respect to the range* I. A similar notion extends to ε-kernels. It is not hard to see that a subset $Q \subseteq P$ is an additive δ-kernel of P if and only if $\mathrm{conv}(P) \subseteq \mathrm{conv}(Q) \oplus B_\delta$, where B_δ is the disk of radius δ centered at the origin and $\oplus$ is the Minkowski sum. Therefore one can use Dudley's construction [1, 12] to compute an additive δ-kernel of P of size $O(\sqrt{\mathrm{diam}(P)/\delta})$.

Affine transforms and ε-kernels. A point set $P \subset \mathbb{R}^2$ is *fat* if $\omega(P, \gamma) \geq \Omega(\mathrm{diam}(P))$ for all $\gamma \in [0, 2\pi]$. Clearly, if P is fat, then one can compute an ε-kernel of P of size $O(1/\sqrt{\varepsilon})$ by simply computing an additive $(c\varepsilon \cdot \mathrm{diam}(P))$-kernel of P for some constant $c > 0$. If P is not fat, then as in [1], one can first apply the algorithm of Barequet and Har-Peled [7] to compute an affine transform τ such that $\tau(P)$ is fat, and then compute an ε-kernel K of $\tau(P)$; the set $Q = \tau^{-1}(K) \subseteq P$ will be an ε-kernel of P by the following observation.

Lemma 1 ([1]) *For any point set P and any affine transform τ, a subset $Q \subseteq P$ is an ε-kernel of P if and only if $\tau(Q)$ is an ε-kernel of $\tau(P)$.*

It will be helpful to review a variant of the algorithm of Barequet and Har-Peled [7] for computing an affine transform τ such that $\tau(P)$ is fat. Let o be an arbitrary point in P and assume w.l.o.g. that $o = (0, 0)$. It consists of two steps:

(1) Pick a point $x \in P$ such that $\|op\| \le 2\|ox\|$ for any $p \in P$; w.l.o.g., assume $x = (1, 0)$.

(2) Pick a point $y \in P$ such that $d(p, ox) \le 2 \cdot d(y, ox)$ for any $p \in P$, where $d(p, ox)$ denotes the distance from a point p to the line passing through o and x.

Then the affine transform τ that maps each point (a, b) to the point $\big(a, b/d(y, ox)\big)$ is the desired transform. Note that $\mathrm{diam}(\tau(P)) \ge 1$ (as $\|\tau(o)\tau(x)\| = 1$) and $\tau(P) \subset [-2, 2] \times [-2, 2]$. See Figure 5 (of Section 3) for an illustration.

Minimal additive δ-kernels. We now introduce the notion of minimal additive kernels that will be useful in subsequent sections. An additive δ-kernel Q of a set P of points (with respect to a range $[\alpha, \beta]$) is *minimal* if for any $q \in Q$, the set $Q \setminus \{q\}$ is no longer an additive δ-kernel of P (with respect to a range $[\alpha, \beta]$). Note that points in a minimal additive kernel are necessarily in convex position. Indeed, any point that lies in the interior of the convex hull cannot be the extremal point in any direction and thus can be deleted from the additive kernel. The problem of computing minimal additive kernels is similar to polygon simplification or polytope approximation (e.g., [4, 20]). We sketch a simple $O(n \log n)$-time algorithm for it.

Let P be a set of n points in $\mathbb{R}^2$. Since $\mathrm{conv}(P)$ can be computed in $O(n \log n)$ time, we can assume w.l.o.g. that P is a set of n points in convex position. Initially we set $Q = P$, which is clearly an additive δ-kernel of P. We then remove redundant points from Q while retaining the invariant that Q is an additive δ-kernel, until Q becomes minimal. Initially we mark all the points in Q as "undecided", and denote the subset of undecided points in Q as Q_{u}. The algorithm proceeds in rounds (see Figure 2). In each round, we pick a subset of points in Q_{u} so that no two picked points are consecutive along $\partial\,\mathrm{conv}(Q)$; note that half of the points in Q_{u} are picked. For each picked point p, we check whether $Q \setminus \{p\}$ remains to be an additive δ-kernel of P. If so, we then simply remove p from Q and Q_{u}; otherwise, we mark p as "permanent" (that is, p will appear in the final output Q) and remove p from Q_{u}. The algorithm terminates when Q_{u} becomes empty.

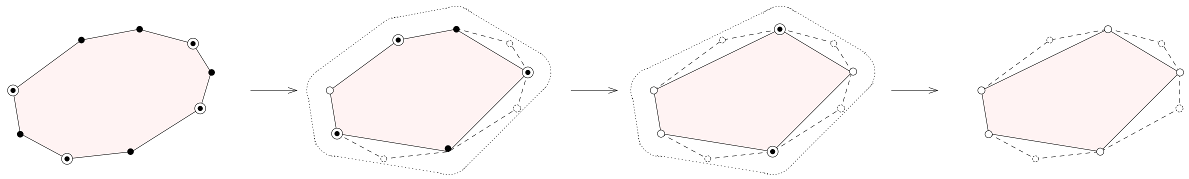

Figure 2. An illustration of the minimal additive δ-kernel algorithm. Empty circles denote "permanent" points, filled (together with double) circles denote "undecided" points (i.e., Q_{u}), double circles denote the subset of Q_{u} selected in each round, and dotted circles denote points removed from Q. The shaded regions are $\mathrm{conv}(Q)$, and the dotted boundaries represent $\mathrm{conv}(Q) \oplus B_\delta$.

Each round of the algorithm can be performed in $O(n)$ time as follows. Recall that in each round we pick a subset of points in Q_{u} so that no two picked points are consecutive along

$\partial\,\mathrm{conv}(Q)$. For each picked point p, we let p_ℓ (resp. p_r) be the point in Q adjacent to p in clockwise (resp. counterclockwise) order along $\partial\,\mathrm{conv}(Q)$. Let α_ℓ be the direction $\overrightarrow{p_\ell p}$ rotated by $\pi/2$ clockwisely, and α_r be the direction $\overrightarrow{pp_r}$ rotated by $\pi/2$ clockwisely. See Figure 3. Note that $[\alpha_\ell, \alpha_r]$ is the set of directions along which p is extreme in Q; this is because points in Q are in convex position and p_ℓ, p, p_r are consecutive in Q. Let A be the intersection of the halfplane bounded by $p_\ell p_r$ and not containing p with the wedge apexed at p and spanned between $\overrightarrow{pp_r}$ and $\overrightarrow{pp_\ell}$. Let $A_\delta = A \oplus B_\delta$. Then by simple geometry, we observe that the set $Q \setminus \{p\}$ remains to be an additive δ-kernel if and only if $P[\alpha_\ell, \alpha_r]$ lies entirely in A_δ. Whether a point lies in A_δ can be determined in constant time because the region has constant description complexity. Hence, excluding the time to locate the first point $P[\alpha_\ell]$, whether $P[\alpha_\ell, \alpha_r]$ lies entirely in A_δ can be determined in $\big|P[\alpha_\ell, \alpha_r]\big|$ time. Observe that in each round the subsequence of points in P examined for each picked point (i.e., $P[\alpha_\ell, \alpha_r]$) do not overlap with each other (except at the endpoints of the subsequences). Hence each round can be performed in $O(n)$ time by transversing along the convex hull of P in one pass. Since in each round $\Omega(|Q_\mathrm{u}|)$ undecided points in Q_u are either removed from or permanently assigned to Q, the algorithm terminates in $O(\log n)$ rounds, and the total running time is therefore $O(n \log n)$.

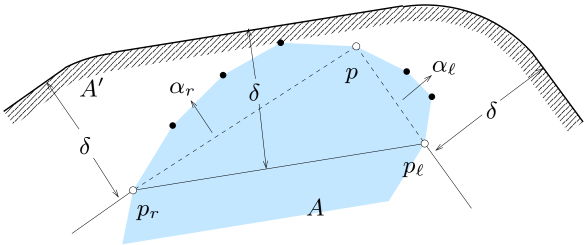

Figure 3. The point p can be removed from the additive δ-kernel if and only if points in $P[\alpha_\ell, \alpha_r]$ fall completely inside the region A_δ (delimited by the thick edges).

It is straightforward to extend the above algorithm to computing a minimal additive δ-kernel of P with respect to a certain range $[\alpha, \beta]$. We conclude:

Lemma 2 *Let P be a set of n points in the plane. Given any range $[\alpha, \beta]$, a minimal additive δ-kernel Q of P with respect to the range $[\alpha, \beta]$ can be computed in $O(n \log n)$ time.*

A useful property of minimal additive kernels is given in the following lemma.

Lemma 3 *Let P be a set of points in convex position, and let Q be a minimal additive δ-kernel of P with respect to the range $[\alpha, \beta]$. Then for any $[\alpha', \beta'] \subseteq [\alpha, \beta]$, $\big|Q[\alpha', \beta']\big| \le 2 \cdot \big|Q_{\mathsf{opt}}\big| + 2$, where Q_{opt} is the smallest additive δ-kernel of P with respect to the range $[\alpha', \beta']$.*

Proof: For two points $p, q \in \partial\,\mathrm{conv}(P)$, let $P[p, q]$ denote the subset of vertices of $\mathrm{conv}(P)$ lying between p and q in counterclockwise order, and let $P(p, q) = P[p, q] \setminus \{p, q\}$. Let $q_1, q_2, \cdots, q_k$ be the points of $Q[\alpha', \beta']$ in counterclockwise order along the boundary of $\mathrm{conv}(Q)$, where $k = \big|Q[\alpha', \beta']\big|$. Consider three consecutive points q_i, q_{i+1}, q_{i+2}. Since Q is minimal, $Q \setminus \{q_{i+1}\}$ is no longer an additive δ-kernel of P with respect to the range $[\alpha, \beta]$. So there is a point $p \in P[q_i, q_{i+2}]$ and a direction $\gamma \in [\alpha, \beta]$ such that (i) $q_{i+1} = Q[\gamma]$, and (ii) $\langle p - q_i, \mathsf{u}_\gamma\rangle > \delta$ and $\langle p - q_{i+2}, \mathsf{u}_\gamma\rangle > \delta$. (In order words, the point p is a "witness" that q_{i+1} has to be present in Q.). It follows from (i) and $q_i, q_{i+1}, q_{i+2} \in Q[\alpha', \beta']$ that $\gamma \in [\alpha', \beta']$. Moreover, since points in P are in convex position, it follows from (ii) that $\langle p - q, \mathsf{u}_\gamma \rangle > \delta$ for every $q \in P[q_{i+2}, q_i]$. Therefore, the set Q_{opt} must contain a point from $P(q_i, q_{i+2}) = P \setminus P[q_{i+2}, q_i]$, that is,

$$|Q_{\mathsf{opt}} \cap P(q_i, q_{i+2})| \geq 1.$$

Applying this fact for $i = 1, 3, \cdots, 2\lfloor (k-1)/2 \rfloor - 1$, we obtain $\left|Q_{\mathsf{opt}}\right| \geq \lfloor (k-1)/2 \rfloor$. Hence, $\left|Q[\alpha', \beta']\right| \leq 2\left|Q_{\mathsf{opt}}\right| + 2$, as claimed. □

Since there always exists an additive δ-kernel of size $O(\sqrt{\mathrm{diam}(P)/\delta})$, it follows from Lemma 3 that the output size of the algorithm in Lemma 2 is $O(\sqrt{\mathrm{diam}(P)/\delta})$. In some cases bounds better than this straightforward bound can be obtained using Lemma 3. This will become useful in the analysis of our algorithm.

3 Overall Algorithm

Chan's algorithm. Since the overall structure of our algorithm is very similar to that of Chan's algorithm [9], we first sketch his algorithm and then outline our approach. Let $o = (0, 0)$ be the first point in the stream. The algorithm divides the stream S into *epochs*. In the i-th epoch started by point x_i, $\|op\| \leq 2\|ox_i\|$ for every point p arriving within this epoch. Once a new point p arrives such that $\|op\| > 2\|ox_i\|$, the $(i+1)$-th epoch is started and we let $x_{i+1} = p$. In an epoch started by the point x, the stream is further divided into *subepochs*. In the j-th subepoch started by point y_j, $d(p, ox) \leq 2 \cdot d(y_j, ox)$ for every point p arriving within this subepoch. Once a new point p arrives such that $d(p, ox) > 2 \cdot d(y_j, ox)$, but $\|op\| \leq 2\|ox\|$ (i.e., p does not start a new epoch), then the $(j+1)$-th subepoch of the current epoch is started and we set $y_{j+1} = p$.

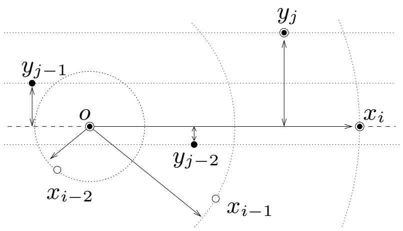

Figure 4. $(i-2)$-th, $(i-1)$-th, and i-th epochs, and $(j-2)$-th, $(j-1)$-th, and j-th subepochs within epoch i.

Note that the definitions of epochs and subepochs closely follow the two steps in the algorithm of Barequet and Har-Peled (see Section 2). Hence, for a fixed subepoch, one can find an affine transform τ such that τ applied to points in this subepoch results in a stream $\tau(S)$ of points such that $\tau(S)$ is fat, $\mathrm{diam}(\tau(S)) \geq 1$, and $\tau(S) \subset [-2, 2] \times [-2, 2]$ at any time, as shown in Figure 5. We call τ the affine transform associated with this subepoch.

Chan observes that it suffices to maintain an ε-kernel of the points in each of the last $\log(1/\varepsilon)$ epochs;[1] points in the older epochs are so close to the point o that they can be rounded to a single line, and the preimages of the two extreme points on that line are sufficient to represent all those points. Similarly, within the current epoch, he maintains an ε-kernel for each of the last $\log(1/\varepsilon)$

[1] Throughout the rest of the paper, all log's are in base 2, and we assume $\log(1/\varepsilon)$ is a positive integer.

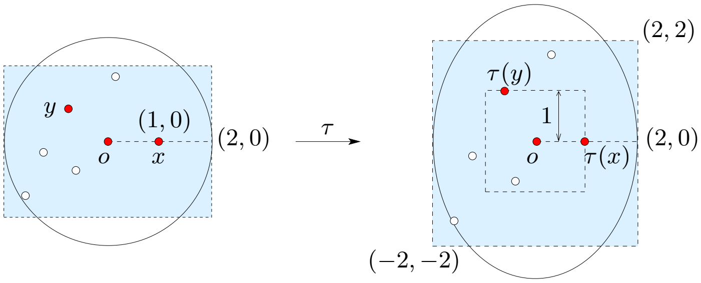

Figure 5. Effect of the affine transform τ.

subepochs. Overall, the algorithm uses $O(1) + N(\varepsilon)\log^2(1/\varepsilon)$ space, where $N(\varepsilon) = O(1/\sqrt{\varepsilon})$ is the space needed for a single subepoch.

Outline of our algorithm. The overall structure of our algorithm is the same as of Chan's algorithm, but we exploit the interactions between epochs and between subepochs to reduce the size of the maintained ε-kernel to $O(N(\varepsilon))$. Let S be the set of points that have arrived so far, and let $S_i \subseteq S$ be the set of points that arrived in the i-th epoch. For each of the last $\log(1/\varepsilon)$ epochs, we maintain a subset $R_i \subseteq S_i$ of points so that

($\star$) for any $\gamma \in [0, 2\pi]$, if $S[\gamma] \in S_i$, then $\langle S_i[\gamma] - R_i[\gamma], \mathsf{u}_\gamma \rangle \leq \varepsilon \cdot \omega(S, \gamma)$.

The set R_i is pruned at the beginning of each new epoch to ensure that the size of the overall set remains $O(N(\varepsilon))$. Similarly, within the current epoch, let $P_j \subseteq S$ be the points in the j-th subpeoch. For each of the last $\log(1/\varepsilon)$ subepochs in the current epoch, we maintain a subset $Q_j \subseteq P_j$ that maintains an invariant similar to ($\star$). The set Q_j is pruned at the beginning of each new subepoch. Finally, we use a more efficient procedure for maintaining an ε-kernel of the points in the current subepoch.

Section 4 describes the procedure Within_Subepoch for maintaining the ε-kernel of the points in the current subepoch. Next, Section 5 describes the procedure Start_Subepoch that prunes the sets Q_j at the beginning of each new subepoch. Finally, Section 6 describes the procedure Start_Epoch that prunes the sets R_j at the beginning of each new epoch. The amortized running time of each of these procedures is $O(\log(1/\varepsilon))$.

4 Algorithm for a Subepoch

We now describe the procedure Within_Subepoch for updating the ε-kernel within a subepoch. By the above discussion, it suffices to deal with a stream S such that S is fat, $\operatorname{diam}(S) \geq 1$, and $\tau(S) \subset [-2, 2] \times [-2, 2]$. Chan [9] and Yu *et al.* [23] observed the following simple construction of an ε-kernel of such S (which is a variant of Dudley's method). Let C be the square $[-4, 4] \times [-4, 4]$ and $\mathcal{N}$ be a set of $O(1/\sqrt{\varepsilon})$ points uniformly distributed on ∂C. Let $\sigma(p) \in S$ denote the nearest neighbor of a point p in S. Then the set $Q = \{\sigma(\xi) \in S \mid \xi \in \mathcal{N}\}$ is an additive $O(\varepsilon)$-kernel of S. Since S is fat and $\operatorname{diam}(S) \geq 1$, Q is in fact an $O(\varepsilon)$-kernel of S. Note that $|Q| \leq |\mathcal{N}| = O(1/\sqrt{\varepsilon})$. We only need to show how to maintain the set Q when new points are inserted into S.

W.l.o.g., we only describe how to maintain the set Q_0 of nearest neighbors of the subset of points $\mathcal{N}_0 \subset \mathcal{N}$ that lie on one side L of ∂C. For any point $p \in Q_0$, let $\sigma^{-1}(p) = \{\xi \in \mathcal{N}_0 \mid$

$\sigma(\xi) = p\}$. Note that $\sigma^{-1}(p) = \mathcal{N}_0 \cap \mathrm{Vor}(S, p)$, where $\mathrm{Vor}(S, p)$ denotes the Voronoi cell of p in the Voronoi diagram of S. Since $\mathrm{Vor}(S, p)$ is convex and $\mathcal{N}_0$ lies on the line segment L, points in $\sigma^{-1}(p)$ must be contiguous along L. We use ℓ_p and r_p to denote the leftmost and rightmost points of $\sigma^{-1}(p)$ along L (see Figure 6). We store the points in Q_0 in a balanced search tree $\mathcal{T}$ based on the ordering of their corresponding sets $\sigma^{-1}(p)$ along L.

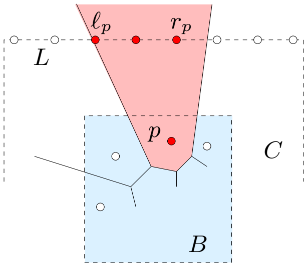

Figure 6. $\sigma^{-1}(p)$ consists of a contiguous sequence of points lying between ℓ_p and r_p.

When a new point q is to be inserted, the leftmost point $\ell_q \in \sigma^{-1}(q)$ can be identified by a binary search on $\mathcal{T}$ as follows. Let p be the point stored at the node v that we are currently visiting. We compute the leftmost point $\ell \in L$ in the intersection $L \cap \mathrm{Vor}(\{p, q\}, q)$. There are three cases:

(1) ℓ lies to the left of ℓ_p. Then ℓ_q also lies to the left of ℓ_p, because otherwise $\ell_p \in \mathrm{Vor}(\{p, q\}, q)$ and therefore $\ell_p \in \sigma^{-1}(q)$, a contradiction with ℓ_q being the leftmost point of $\sigma^{-1}(q)$. So we proceed to the left child of v, or stop if v is already a leaf (which means $\sigma^{-1}(q) = \emptyset$).

(2) ℓ lies to the right of r_p. Then ℓ_q also lies to the right of r_p, because ℓ_q always lies to the right of ℓ. So we proceed to the right child of v, or stop if v is already a leaf (which means $\sigma^{-1}(q) = \emptyset$).

(3) ℓ lies between ℓ_p and r_p. Then ℓ_q must also lie between ℓ_p and r_p. In fact, ℓ_p is exactly the first point in $\mathcal{N}_0$ that is to the right of ℓ or ℓ itself if $\ell \in \mathcal{N}_0$. We can then stop the search.

The rightmost point $r_q \in \sigma^{-1}(q)$ can be identified in a similar manner. The set $\sigma^{-1}(q)$ is then the subset of points of $\mathcal{N}_0$ lying between ℓ_q and r_q. For each point p that is previously stored in $\mathcal{T}$ but the old set $\sigma^{-1}(p)$ is now completely contained in $\sigma^{-1}(q)$, we delete p from the tree $\mathcal{T}$. The cost can be charged to the (already happened) insertion of p. Moreover, if $\sigma^{-1}(q) \neq \emptyset$, we insert q into the tree $\mathcal{T}$. Overall the amortized cost for processing each insertion is $O(\log(1/\varepsilon))$.

Lemma 4 *An ε-kernel of the streaming points within each subepoch can be maintained using $O(1/\sqrt{\varepsilon})$ space and $O(\log(1/\varepsilon))$ amortized time per point.*

5 Starting a New Subepoch

We now describe the procedure for pruning the set of points at the beginning of a new subepoch in the current epoch. This procedure does not touch the points of the previous epochs. We first describe the algorithm and the invariant it maintains, then prove the correctness of the procedure and analyze its space and time complexity.

Algorithm. Recall that $o = (0, 0)$ is the first point in the stream. Let P denote the point o together with the set of points that have arrived in the current epoch so far, and let $P_k \subseteq P$ denote the set of points that arrive in the k-th subepoch of the current epoch. Furthermore, suppose that the current epoch is started by the point x (hence $\|op\| \leq 2\|ox\|$ for every $p \in P$); w.l.o.g., we can assume $x = (1, 0)$. Within this epoch, let $y_k \in P_k$ denote the point that starts the k-th subepoch (hence $d(p, ox) \leq 2 \cdot d(y_k, ox)$ for every $p \in P_k$, and $d(y_k, ox) > 2 \cdot d(y_{k-1}, ox)$). Let τ_k be the affine transform that maps every point (a, b) to $\big(a, b/d(y_k, ox)\big)$; that is, τ_k is the affine transform associated with the k-th subepoch (see Figure 7). For a subset $X \subseteq P_k$, let $\widetilde{X}$ denote the set $\tau_k(X)$; if a set $\widetilde{X} \subseteq \widetilde{P}_k$ is defined without prescribing X, then the set X is uniquely defined as $\tau_k^{-1}(\widetilde{X})$ since τ_k is bijective.

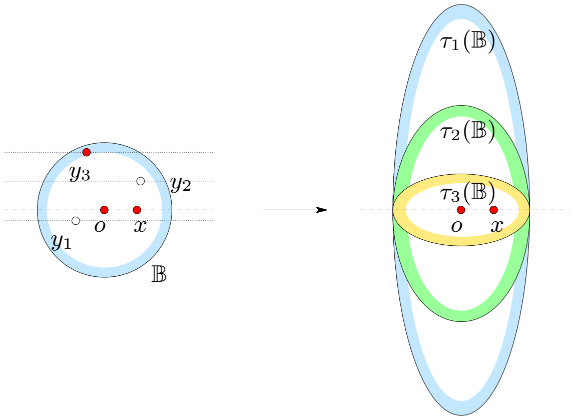

Figure 7. Effects of different affine transforms.

Suppose the current subepoch is the i-th subepoch. Note that the set P_i is changing as the new points are added to the current subepoch, but P_{i-j} is fixed for all $j \geq 1$. We use Within_Subepoch (Lemma 4) to maintain a set Q_i for the current subepoch. For $1 \leq j \leq \log(1/\varepsilon)$, we maintain a set $Q_{i-j} \subseteq P_{i-j}$ that maintains the following invariant:

(I) For any $\gamma \in [0, 2\pi]$, we have $\langle \widetilde{P}_{i-j}[\gamma] - \widetilde{Q}_{i-j}[\gamma], \mathsf{u}_\gamma \rangle \leq O(\varepsilon) \cdot \omega(\tau_{i-j}(P), \gamma)$.

Specifically, we partition $\mathbb{S}^1$ into four quadrants $[i\pi/2, (i+1)\pi/2]$, $0 \leq i \leq 3$, and maintain a set of points for each quadrant whose union will be Q_{i-j}. We describe the construction for the first quadrant $[0, \pi/2]$; the other three can be handled in a similar manner. For a given j, we partition the interval $[0, \pi/2]$ into a family $\mathcal{I}_{i-j}$ of angular intervals

$$I_{i-j}^{j} = [0, \pi/2^j], I_{i-j}^{j-1} = [\pi/2^j, \pi/2^{j-1}], \cdots, I_{i-j}^{1} = [\pi/4, \pi/2]. \tag{2}$$

A crucial property of this partition that we will prove below in Lemma 7 is that

$$\omega(\tau_{i-j}(P), \gamma) = \Omega(2^{j-k})$$

for any $\gamma \in I_{i-j}^k$. For each I_{i-j}^k, we maintain a subset $Q_{i-j}^k \subseteq P_{i-j}$ with the following invariant:

(I') For any $\gamma \in I_{i-j}^k$, we have $\langle \widetilde{P}_{i-j}[\gamma] - \widetilde{Q}_{i-j}^k[\gamma], \mathsf{u}_\gamma \rangle \leq O(\varepsilon) \cdot \omega(\tau_{i-j}(P), \gamma)$.

The set Q_{i-j} is the union of all Q_{i-j}^k $(1 \le k \le j)$ and over all four quadrants. Finally, as in [9], we maintain a single set Q_{ox}, which consists of the two extreme points of $\bigcup_{j > \log(1/\varepsilon)} Q_{i-j}$ in direction $\overrightarrow{ox}$, to take care of the points in older subepochs (those before the $(i - \log(1/\varepsilon))$-th subepoch). Let 1D_KERNEL(X) denote the procedure that computes these two extreme points of the set X. When a new subepoch is started by the point y_{i+1}, we proceed as follows:

```
START_SUBEPOCH($y_{i+1}$)
1:  $Q_{ox} \leftarrow$ 1D_KERNEL($Q_{i-\log(1/\varepsilon)} \cup Q_{ox}$)
2:  $\widetilde{H}_i \leftarrow$ vertices of conv($\widetilde{Q}_i$);
    $\widetilde{Q}_i^1 \leftarrow$ a minimal additive $\varepsilon$-kernel of $\widetilde{H}_i$;
3:  for $j = 1, \cdots, \log(1/\varepsilon) - 1$ do
4:      COMPRESS_SUBEPOCH($Q_{i-j}$)
```

```
COMPRESS_SUBEPOCH($Q_{i-j}$)
5:  $\widetilde{Q}_{i-j}^{j+1} \leftarrow \widetilde{Q}_{i-j}^{j}[0, \pi/2^{j+1}]$
6:  $\widetilde{Q}_{i-j}^{j} \leftarrow \widetilde{Q}_{i-j}^{j}[\pi/2^{j+1}, \pi/2^{j}]$
7:  if $|Q_{i-j}| > j/\sqrt{2^j\varepsilon}$ then
8:      for $k = 1, \cdots, j$ do
9:          $\widetilde{Q}_{i-j}^{k} \leftarrow$ a $(2^{j-1}\varepsilon)$-kernel of $\widetilde{Q}_{i-j}^{k}$
```

In line 2, we use Lemma 2 to compute $\widetilde{Q}_i^1$ (and thus Q_i^1 implicitly) of size $O(1/\sqrt{\varepsilon})$ in $O(|Q_i| \log |Q_i|)$ time. In line 4, we invoke COMPRESS_SUBEPOCH(Q_{i-j}) to reduce the size of Q_{i-j} to $O(j/\sqrt{2^j\varepsilon})$. Note that when we start the new $(i+1)$-th subepoch, then Q_{i-j} becomes $Q_{(i+1)-(j+1)}$. So by (2), the interval I_{i-j}^j is split into two intervals $I_{i-j}^{j+1} = [0, \pi/2^{j+1}]$ and (new) $I_{i-j}^j = [\pi/2^{j+1}, \pi/2^j]$, and the set Q_{i-j}^j is also split (see Figure 8). Lines 5–6 in the procedure COMPRESS_SUBEPOCH perform this split; note that the two subsets $\widetilde{Q}_{i-j}^{j+1}$ and $\widetilde{Q}_{i-j}^{j}$ have exactly one point in common, i.e., $\widetilde{Q}_{i-j}^{j}[\pi/2^{j+1}]$. In line 9, we use an algorithm of [9] to compute an $(2^{j-1}\varepsilon)$-kernel of size $O\big(1/\sqrt{2^j\varepsilon}\big)$ in $O(|Q_{i-j}^k|)$ time.

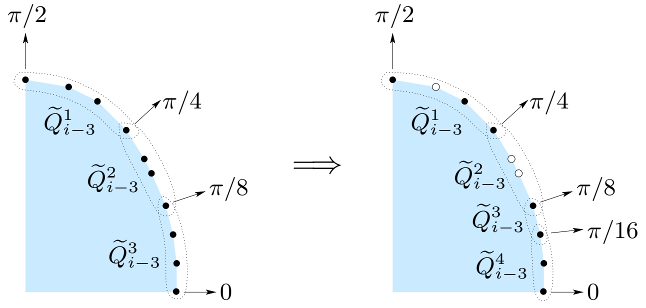

Figure 8. Splitting the interval I_{i-j}^j (here $j = 3$) and pruning the points. Hollow circles are the points pruned by COMPRESS_SUBEPOCH.

After START_SUBEPOCH(y_{i+1}) is executed, we use WITHIN_SUBEPOCH (Lemma 4) to maintain a subset Q_{i+1} for the current $(i+1)$-th subepoch.

Correctness. As mentioned above, the above construction for the quadrant $[0, \pi/2]$ needs to be repeated for the other three quadrants. So we remind the reader that Q_{i-j} is indeed the union of the sets maintained for all quadrants. Let $\mathcal{Q}$ be the union of all sets $Q_{ox}, Q_{i-\log(1/\varepsilon)+1}, \cdots, Q_{i+1}$. We prove that $\mathcal{Q}$ is an $O(\varepsilon)$-kernel of P.

First observe that (I) is equivalent to $\langle P_{i-j}[\gamma] - Q_{i-j}[\gamma], \mathsf{u}_\gamma \rangle \le O(\varepsilon) \cdot \omega(P, \gamma)$ for all $\gamma \in [0, 2\pi]$. Assuming (I) is true, the correctness of the algorithm is then obvious: for any $\gamma \in [0, 2\pi]$, depending on in which subepoch $P[\gamma]$ arrived, we can always find a corresponding point $q \in \mathcal{Q}$ such that $\langle P[\gamma] - q, \mathsf{u}_\gamma \rangle \le O(\varepsilon) \cdot \omega(P, \gamma)$. Specifically, if $P[\gamma]$ arrived in the current $(i+1)$-th subepoch, then we can let $q = Q_{i+1}[\gamma]$ by Lemma 4; if $P[\gamma]$ arrived in the $(i-j)$-th subepoch for some $0 \le j < \log(1/\varepsilon)$, then we can let $q = Q_{i-j}[\gamma]$ by the invariant (I); and if $P[\gamma]$ arrived in one of the older subepochs, we can let $q = Q_{ox}[\gamma]$ by Chan's argument [9].

Since (I) readily follows from (I'), all that remains is to prove (I'), for the beginning of the $(i+1)$-th subepoch. We need the following lemma.

Lemma 5 *Let P be a set of points in $\mathbb{R}^2$, and let $[\alpha, \beta]$ be a given range with $\alpha < \beta$. For any $\xi \in [\alpha, \beta]$,*

$$\omega(P[\alpha, \beta], \xi) \le \operatorname{diam}(P) \cdot (\beta - \alpha).$$

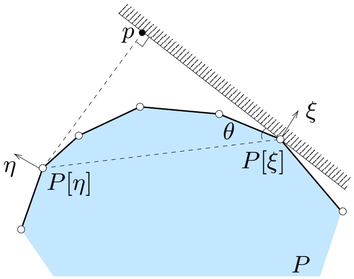

Figure 9. Illustration for Lemma 5.

Proof: Pick an arbitrary angle $\eta \in [\alpha, \beta]$. Let p be the projection of $P[\eta]$ onto the line passing through $P[\xi]$ and orthogonal to ξ. Let θ be the angle $\angle P[\eta]P[\xi]p$. (See Figure 9.) Note that $\theta \le |\eta - \xi| \le \beta - \alpha$. Therefore,

$$\langle P[\xi] - P[\eta], \mathsf{u}_\xi \rangle = \left\| P[\eta]P[\xi] \right\| \cdot \left| \sin\theta \right| \le \operatorname{diam}(P) \cdot (\beta - \alpha).$$

Since the above inequality holds for an arbitrary $\eta \in [\alpha, \beta]$, the lemma then follows. □

Recall that $\widetilde{Q}_{i-j}$ is an additive ε-kernel of $\widetilde{P}_{i-j}$, maintained by the procedure Within_Subepoch described in Section 4, just before the $(i-j+1)$-th subepoch started. By line 2 of Start_Subepoch, $\widetilde{H}_{i-j}$ is the set of vertices of $\operatorname{conv}(\widetilde{Q}_{i-j})$, therefore $\widetilde{H}_{i-j}$ is also an additive ε-kernel of $\widetilde{P}_{i-j}$. Let $\widetilde{H}^1_{i-j}$ be the set $\widetilde{Q}^1_{i-j}$ when $\widetilde{Q}^1_{i-j}$ was created in the subroutine Start_Subepoch(y_{i-j+1}); note that $\widetilde{H}^1_{i-j}$ denotes a fixed set, while $\widetilde{Q}^1_{i-j}$ may change in subsequent subepochs. By line 2 of Start_Subepoch, $\widetilde{H}^1_{i-j}$ is a minimal additive ε-kernel of $\widetilde{H}_{i-j}$.

In the following, we fix an arbitrary integer $1 \le k \le j+1$.

Lemma 6 *For any $\gamma \in I^k_{i-j}$, $\langle \widetilde{P}_{i-j}[\gamma] - \widetilde{Q}_{i-j}[\gamma], \mathsf{u}_\gamma \rangle = O(2^{j-k}\varepsilon)$.*

Proof: If $k = j+1$, then we have $\gamma \in [0, \pi/2^{j+1}]$. Observe that $\widetilde{Q}^{j+1}_{i-j} = \widetilde{H}^1_{i-j}[0, \pi/2^{j+1}]$ and therefore $\widetilde{Q}^{j+1}_{i-j}[\gamma] = \widetilde{H}^1_{i-j}[\gamma]$. Since $\widetilde{H}_{i-j}$ is an additive ε-kernel of $\widetilde{P}_{i-j}$ and $\widetilde{H}^1_{i-j}$ is an additive ε-kernel of $\widetilde{H}_{i-j}$, we can write

$$\langle \widetilde{P}_{i-j}[\gamma] - \widetilde{Q}^{j+1}_{i-j}[\gamma], \mathsf{u}_\gamma \rangle = \langle \widetilde{P}_{i-j}[\gamma] - \widetilde{H}_{i-j}[\gamma], \mathsf{u}_\gamma \rangle + \langle \widetilde{H}_{i-j}[\gamma] - \widetilde{H}^1_{i-j}[\gamma], \mathsf{u}_\gamma \rangle \le 2\varepsilon.$$

Next we consider the case $k \leq j$. By line 9 and a simple induction, one can show that $\widetilde{Q}_{i-j}^{k}$ is a $(2^j\varepsilon)$-kernel of $\widetilde{H}_{i-j}^{1}[\pi/2^{k+1}, \pi/2^k]$. Hence,

$$\begin{aligned}\langle \widetilde{P}_{i-j}[\gamma] - \widetilde{Q}_{i-j}^{k}[\gamma], \mathsf{u}_\gamma\rangle &= \langle \widetilde{P}_{i-j}[\gamma] - \widetilde{H}_{i-j}^{1}[\gamma], \mathsf{u}_\gamma\rangle + \langle \widetilde{H}_{i-j}^{1}[\gamma] - \widetilde{Q}_{i-j}^{k}[\gamma], \mathsf{u}_\gamma\rangle \\ &\leq 2\varepsilon + 2^j\varepsilon \cdot \omega(\widetilde{H}_{i-j}^{1}[\pi/2^{k+1}, \pi/2^k], \gamma) = O(2^{j-k}\varepsilon),\end{aligned}$$

where the last inequality follows from Lemma 5 and the fact that $\mathrm{diam}(\widetilde{P}_{i-j}) \leq 8$. In either case, we have proved the lemma. □

Lemma 7 *For any $\gamma \in I_{i-j}^{k}$, $\omega(\tau_{i-j}(P), \gamma) = \Omega(2^{j-k})$.*

Proof: Let $\tau_{i-j}(y_{i+1}) = (w_1, w_2)$. Recall that τ_{i-j} is the affine transform that maps any point (a, b) to the point $\big(a, b/d(y_{i-j}, ox)\big)$. Since $|oy_{i+1}| \leq 2|ox| = 2$ and $d(y_{i+1}, ox) \geq 2^{j+1}d(y_{i-j}, ox)$, we obtain $|w_1| \leq 2$ and $|w_2| \geq 2^{j+1}$. Moreover, because $o, y_{i+1} \in P$, we can write

$$\begin{aligned}\omega(\tau_{i-j}(P), \gamma) &\geq \big|\langle \tau_{i-j}(y_{i+1}), \mathsf{u}_\gamma\rangle\big| \geq |w_2 \cdot \sin\gamma| - |w_1 \cdot \cos\gamma| \\ &\geq 2^{j+1}(\pi/2^{k+1})/2 - 2 \qquad (\sin\gamma \geq \gamma/2),\end{aligned}$$

which is $\Omega(2^{j-k})$ if $k < j$; for $k = j$ or $j + 1$, we use the fact $\omega(\tau_{i-j}(P), \gamma) = \Omega(1)$ to obtain the lemma. □

The invariant (I') follows directly from Lemmas 6 and 7, which in turn completes the proof of the claim that $\mathcal{Q}$ is an $O(\varepsilon)$-kernel of P.

Space complexity. Because we represent each Q_{i-j} as a collection of subsets, the space needed by the algorithm after the $(i+1)$-th subepoch starts is given by

$$|Q_{i+1}| + \sum_{j=0}^{\log(1/\varepsilon)-1} \sum_{k=1}^{j+1} |Q_{i-j}^{k}| + |Q_{ox}|. \tag{3}$$

We already know that $|Q_{i+1}| = O(1/\sqrt{\varepsilon})$ and $|Q_{ox}| = 2$. For $j < \log(1/\varepsilon)$ and any $1 \leq k \leq j$, $|Q_{i-j}^{k}|$ is bounded by $O(1/\sqrt{2^j\varepsilon})$ by line 9. It remains to bound the term $|Q_{i-j}^{j+1}|$.

Lemma 8 $|Q_{i-j}^{j+1}| = O(1/\sqrt{2^j\varepsilon})$.

Proof: Recall that $\widetilde{H}_{i-j}^{1}$ is a minimal additive ε-kernel of $\widetilde{H}_{i-j}$, and that the points in $\widetilde{H}_{i-j}$ are in convex position. Furthermore, by construction we have $\widetilde{Q}_{i-j}^{j+1} = \widetilde{H}_{i-j}^{1}[0, \pi/2^{j+1}]$. Then by Lemma 3, in order to bound $|\widetilde{Q}_{i-j}^{j+1}|$, it suffices to bound the size of the smallest additive ε-kernel of $\widetilde{H}_{i-j}$ with respect to the range $[0, \pi/2^{j+1}]$.

Consider a $(2^{j-2}\varepsilon/\pi)$-kernel K of $\widetilde{H}_{i-j}[0, \pi/2^{j+1}]$ of size $O(1/\sqrt{2^j\varepsilon})$. We claim that K is an additive ε-kernel of $\widetilde{H}_{i-j}$ with respect to the range $[0, \pi/2^{j+1}]$. In fact, for any $\gamma \in [0, \pi/2^{j+1}]$, we have

$$\begin{aligned}\langle \widetilde{H}_{i-j}[\gamma] - K[\gamma], \mathsf{u}_\gamma\rangle &\leq (2^{j-2}\varepsilon/\pi) \cdot \omega(\widetilde{H}_{i-j}[0, \pi/2^{j+1}], \gamma) \\ &\leq (2^{j-2}\varepsilon/\pi) \cdot 8 \cdot \pi/2^{j+1} = \varepsilon,\end{aligned}$$

where the first inequality follows from the definition of K being a $(2^{j-2}\varepsilon/\pi)$-kernel of $\widetilde{H}_{i-j}[0, \pi/2^{j+1}]$, and the second inequality follows from Lemma 5. Therefore, the size of the smallest additive ε-kernel of $\widetilde{H}_{i-j}$ with respect to the range $[0, \pi/2^{j+1}]$ is no more than $|K| = O(1/\sqrt{2^j\varepsilon})$. This completes the proof. □

Substituting the values in (3), we obtain that the space is bounded by

$$O(1/\sqrt{\varepsilon}) + \sum_{j=0}^{\log(1/\varepsilon)-1} O\big(j/\sqrt{2^j\varepsilon}\big) + O(1) = O(1/\sqrt{\varepsilon}).$$

Update time. Line 1 takes $O(|Q_{i-\log(1/\varepsilon)}|)$ time, which can be amortized by charging $O(1)$ to each point in $Q_{i-\log(1/\varepsilon)}$. Line 2 takes $O(|Q_i|\log|Q_i|) = O(|Q_i|\log(1/\varepsilon))$ time, which can be amortized by charging $O(\log(1/\varepsilon))$ to each point in Q_i. In each execution of line 4, if $|Q_{i-j}| > j/\sqrt{2^j\varepsilon}$, then COMPRESS_SUBEPOCH (Q_{i-j}) takes a total of $O\big(\sum_{k=1}^{j+1}|Q_{i-j}^k|\big) = O(|Q_{i-j}|+j)$ time (note that each Q_{i-j}^k and Q_{i-j}^{k+1} has one common point); otherwise it only takes $O(|Q_{i-j}^j| + |Q_{i-j}^{j+1}|)$ time. In either case, the running time is bounded by $O(|Q_{i-j}|)$, which can be amortized by charging $O(1)$ to each point in Q_{i-j}. It follows that, over the entire history, each point in the $(i-j)$-th subepoch is charged at most $O(\log(1/\varepsilon))$ times, as line 4 will be executed at most $\log(1/\varepsilon) - 1$ times on Q_{i-j}, one in each invocation of START_SUBEPOCH(y_{i-j+k}), for $k = 2, \cdots, \log(1/\varepsilon)$. Hence, the amortized update time of START_SUBEPOCH is $O(\log(1/\varepsilon))$. By combining this with Lemma 4, we obtain:

Lemma 9 *An ε-kernel of the streaming points within each epoch can be maintained using $O(1/\sqrt{\varepsilon})$ space and $O(\log(1/\varepsilon))$ amortized time per point.*

6 Starting a New Epoch

In this section we describe the procedure for pruning the set of points at the beginning of a new epoch. The algorithm is similar to what we did for subepochs, but slightly more involved.

Algorithm. Let S denote the set of points that have arrived so far in the stream, and let S_k denote the set of points that arrived in the k-th epoch. Let $x_k \in S_k$ denote the point that starts the k-th epoch. Hence $\|op\| \le 2\|ox_k\|$ for every $p \in S_k$, and $\|ox_k\| > 2\|ox_{k-1}\|$. Let τ_k be the affine transform associated with the last subepoch of the k-th epoch. As before, we use the notation $\widetilde{X} = \tau_k(X)$ for $X \subseteq S_k$.

Suppose we are currently in the i-th epoch. We use the algorithm of previous section (Lemma 9) to maintain a set Q_i for this epoch. As in [9], since points that arrived before the $(i-\log(1/\varepsilon))$-th epoch are much closer to o than x_{i+1}, they can be mapped to a fixed line passing through o without incurring much error; a single set Q_o consisting of the preimages of the two extreme points along that line is maintained to represent all these points. Although this procedure is somewhat different from the one in Section 5, we also call it 1D_KERNEL.

For $1 \le j \le \log(1/\varepsilon)$, we maintain a set Q_{i-j} implicitly as a collection of $O(j^2)$ subsets. Specifically, for the $(i-j)$-th epoch, we maintain a partition $\mathcal{I}_{i-j}$ of the range $[0, 2\pi]$ into $O(j^2)$ intervals. Each interval $I \in \mathcal{I}_{i-j}$ has a corresponding set Q_I, and the union $\bigcup_{I\in\mathcal{I}_{i-j}} Q_I$ is precisely

Q_{i-j}. Furthermore, each interval $I \in \mathcal{I}_{i-j}$ is associated with a real value $\kappa_I \in \mathbb{R}$. The following three invariants are maintained by the algorithm:

(I1) Each κ_I is of the form $2^\ell \varepsilon$ for an integer $\ell \geq 0$.

(I2) For any $I \in \mathcal{I}_{i-j}$ and $\gamma \in I$, $\omega(\tau_{i-j}(S), \gamma) = \Omega(\kappa_I/\varepsilon)$.

(I3) For any $I \in \mathcal{I}_{i-j}$, $\widetilde{Q}_I$ is an additive κ_I-kernel of $\widetilde{H}_{i-j}$ with respect to the range I.

Note that (I2)–(I3) are extensions for the subepoch case (see Lemmas 6 and 7).

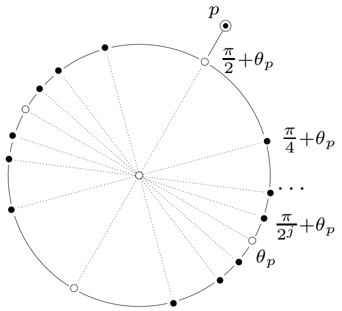

Figure 10. The points $X_j + \theta_p$ and the induced partition $\mathcal{D}_j[p]$.

For a finite set $X \subseteq \mathbb{S}^1$of orientations, let $\mathcal{A}(X)$ denote the partition of $\mathbb{S}^1$ (1-dimensional arrangement) induced by the orientations in X. For two finite sets $X, Y \subseteq \mathbb{S}^1$, $\mathcal{A}(X \cup Y)$ is the overlay of $\mathcal{A}(X)$ and $\mathcal{A}(Y)$. For $j \geq 1$, let

$$X_j = \{0, \pm\pi/2^j, \pm\pi/2^{j-1}, \cdots, \pi/2, \pi, \pi \pm \pi/2^j, \pi \pm \pi/2^{j-1}, \cdots, 3\pi/2\}.$$

For a point $p \in \mathbb{R}^2$, let $\mathcal{D}_j[p] = \mathcal{A}\big(X_j + \theta_p\big)$, where θ_p is the orientation of a vector normal to $\overrightarrow{op}$; see Figure 10. For $\theta_p = 0$, $\mathcal{D}_j[p]$ (restricted to the first quadrant) is the same as $\mathcal{I}_{i-j}$ in Section 5 (see (2)). At the beginning of each epoch, the anchor point p in the above partition will change from $\tau_{i-j}(x_i)$ to $\tau_{i-j}(x_{i+1})$, so $\mathcal{I}_{i-j}$ involves a more significant change than for subepochs as in Section 5. We ensure the invariants (I1)–(I3) by overlaying the current $\mathcal{I}_{i-j}$ with $\mathcal{D}_{j+1}[\tau_{i-j}(x_{i+1})]$, and choosing an appropriate value of κ_I for each interval I in the resulting overlay. The exact algorithm is sketched below.

```
START_EPOCH ($x_{i+1}$)
  $Q_o \leftarrow$ 1D_KERNEL($Q_{i-\log(1/\varepsilon)} \cup Q_o$)
  $\widetilde{H}_i \leftarrow$ vertices of conv($\widetilde{Q}_i$);
  $I \leftarrow [0, 2\pi]$, $\mathcal{I}_i \leftarrow \{I\}$, $\kappa_I \leftarrow \varepsilon$;
  $\widetilde{Q}_I \leftarrow$ a minimal additive $\varepsilon$-kernel of $\widetilde{H}_i$
  for $j = 1, \cdots, \log(1/\varepsilon) - 1$ do
      COMPRESS_EPOCH($Q_{i-j}, x_{i+1}$)
```

```
COMPRESS_EPOCH($Q_{i-j}, x_{i+1}$)
  if $|Q_{i-j}| \leq j^2/\sqrt{2^j\varepsilon}$ then return
  $\mathcal{I} \leftarrow \mathcal{D}_{j+1}[\tau_{i-j}(x_{i+1})]$;
  $\kappa_I \leftarrow \varepsilon \cdot 2^{j+1}|I|/\pi$, for each $I \in \mathcal{I}$
  $\mathcal{I}_{i-j} \leftarrow$ overlay of $\mathcal{I}_{i-j}$ and $\mathcal{I}$;
  for each $I \in \mathcal{I}_{i-j}$ do
      $I_1 \leftarrow$ interval in old $\mathcal{I}_{i-j}$ that contains $I$
      $I_2 \leftarrow$ interval in $\mathcal{I}$ that contains $I$
      $\kappa_I \leftarrow \max\{\kappa_{I_1}, \kappa_{I_2}\}$
      if $\kappa_{I_1} \geq \kappa_{I_2}$ then $\widetilde{Q}_I \leftarrow \widetilde{Q}_{I_1}[I]$
          else $\widetilde{Q}_I \leftarrow$ a minimal additive
                 $\big(\kappa_{I_2}/2\big)$-kernel of $\widetilde{Q}_{I_1}$ w.r.t. $I$
```

In line 2, we use Lemma 2 to compute $\widetilde{Q}_I$ (and thus Q_I implicitly) of size $O(1/\sqrt{\varepsilon})$ in $O(|Q_i|\log|Q_i|)$ time. In line 4, we invoke COMPRESS_EPOCH(Q_{i-j}, x_{i+1}) to reduce the size of Q_{i-j} to $O(j^2/\sqrt{2^j\varepsilon})$. In line 6, the notation $|I|$ denotes the length of an interval I. In line 7, since $\mathcal{I}$ has $O(j)$ intervals, a simple induction shows that the new $\mathcal{I}_{i-j}$ consists of at most $\sum_{k\leq j} O(k) = O(j^2)$ intervals. Lines 8–12 compute κ_I and $\widetilde{Q}_I$ for each interval I in the new partition $\mathcal{I}_{i-j}$. See Figure 11 for an illustration.

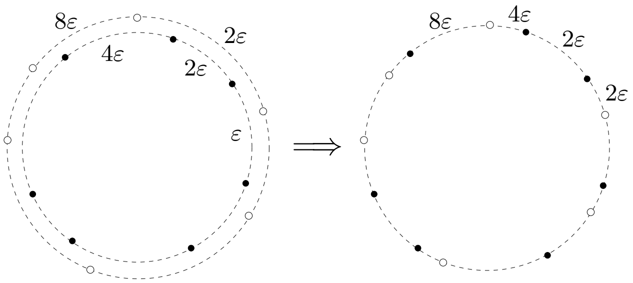

Figure 11. Overlay of two partitions. The numbers are the real values κ_I associated with each interval.

After START_EPOCH(x_{i+1}) is executed, we then use the algorithm of previous section to maintain an ε-kernel Q_{i+1} for the current epoch. This completes the description of the algorithm. Next we prove its correctness and analyze its performance.

Correctness. Let $\mathcal{Q}$ be the union of all sets $Q_o, Q_{i-\log(1/\varepsilon)+1}, \cdots, Q_{i+1}$. We first show that the algorithm indeed maintains an $O(\varepsilon)$-kernel $\mathcal{Q}$ of S. In fact, this follows easily from the invariants (I2) and (I3). For any $\gamma \in [0, 2\pi]$, let $I \in \mathcal{I}_{i-j}$ be the interval that contains γ. Using (I2) and (I3), we can write

$$\begin{aligned}\langle \widetilde{S}_{i-j}[\gamma] - \widetilde{Q}_{i-j}[\gamma], \mathsf{u}_\gamma\rangle &= \langle \widetilde{S}_{i-j}[\gamma] - \widetilde{H}_{i-j}[\gamma], \mathsf{u}_\gamma\rangle + \langle \widetilde{H}_{i-j}[\gamma] - \widetilde{Q}_I[\gamma], \mathsf{u}_\gamma\rangle \\ &\leq \varepsilon + \kappa_I = O(\kappa_I) \leq O(\varepsilon)\cdot\omega(\tau_{i-j}(S), \gamma).\end{aligned}$$

Since this holds for all $\gamma \in [0, 2\pi]$, it is then equivalent to

$$\langle S_{i-j}[\gamma] - Q_{i-j}[\gamma], \mathsf{u}_\gamma\rangle \leq O(\varepsilon)\cdot\omega(S, \gamma)$$

for all $\gamma \in [0, 2\pi]$. As in the previous section, this implies that $\mathcal{Q}$ is indeed an $O(\varepsilon)$-kernel of S.

Lemma 10 *The procedure* COMPRESS_EPOCH *maintains the three invariants* (I1)–(I3) *at all times.*

Proof: (I1) By construction in line 6, the term $|I_2|$ for each $I_2 \in \mathcal{I}$ is of the form $\pi/2^\ell$ for some $\ell \leq j+1$; as such, κ_{I_2} is of the form $2^{j+1-\ell}\varepsilon$. Then by induction and line 10, (I1) is proved.

(I2) If $\kappa_{I_1} \geq \kappa_{I_2}$, then following line 11 and by induction, we have

$$\omega(\tau_{i-j}(S), u) = \Omega(\kappa_{I_1}/\varepsilon) = \Omega(\kappa_I/\varepsilon).$$

Otherwise the argument given for Lemma 7 shows that

$$\omega(\tau_{i-j}(S), u) = \Omega(\kappa_{I_2}/\varepsilon) = \Omega(\kappa_I/\varepsilon).$$

(I3) If $\kappa_{I_1} \geq \kappa_{I_2}$, then (I3) is clearly true by line 11 and the induction hypothesis. Otherwise by (I1), $\kappa_{I_1} < \kappa_{I_2}$ implies $\kappa_{I_1} \leq \kappa_{I_2}/2$; then by induction and line 12, the set $\widetilde{Q}_I$ is an additive $\big(\kappa_{I_1} + \kappa_{I_2}/2\big)$-kernel of $\widetilde{H}_{i-j}$ with respect to the range I, which proves (I3) since $\kappa_{I_1} + \kappa_{I_2}/2 \leq \kappa_{I_2} = \kappa_I$. □

Space complexity. The following lemma can be proved by a simple induction similar to Lemma 10.

Lemma 11 *For any $I \in \mathfrak{I}_{i-j}$, the set $\widetilde{Q}_I$ consists of the extreme points of a certain minimal additive $(\kappa_I/2)$-kernel $K \subseteq \widetilde{H}_{i-j}$ in the range I.*

Proof: Consider lines 11-12 in which $\widetilde{Q}_I$ is computed. If $\kappa_{I_1} < \kappa_{I_2}$, then the lemma is trivially true by line 12. Otherwise, by induction, $\widetilde{Q}_{I_1}$ consists of the extreme points of a certain minimal additive $(\kappa_{I_1}/2)$-kernel $K \subseteq \widetilde{H}_{i-j}$ in the range I_1. If $\kappa_{I_1} \geq \kappa_{I_2}$, then by line 11 and noting that $\kappa_{I_1} = \kappa_I$ and $I \subseteq I_1$, $\widetilde{Q}_I$ therefore consists of the extreme points of K in the range I, as desired. □

By the above lemma, and following the same line of argument as in Lemma 8, we can now obtain $|Q_I| = O\big(1/\sqrt{\kappa_I/2|I|}\big)$. Since

$$\kappa_I \geq \kappa_{I_2} = \varepsilon \cdot 2^j |I_2|/\pi \geq \varepsilon \cdot 2^j |I|/\pi$$

by lines 6 and 10, it follows that $|Q_I| = O\big(1/\sqrt{2^j\varepsilon}\big)$. As $|\mathfrak{I}_{i-j}| = O(j^2)$, we conclude that at any time the total space used by the algorithm is

$$|Q_{i+1}| + \sum_{j=0}^{\log(1/\varepsilon)-1} \sum_{I \in \mathfrak{I}_{i-j}} |Q_I| + |Q_o| = O(1/\sqrt{\varepsilon}) + \sum_{j=0}^{\log(1/\varepsilon)-1} O(j^2/\sqrt{2^j\varepsilon}) + O(1) = O(1/\sqrt{\varepsilon}).$$

Note that $|Q_{i-j}| \leq \sum_{I \in \mathfrak{I}_{i-j}} |Q_I| = O(j^2/\sqrt{2^j\varepsilon})$. This claim can be extended to past epochs, i.e., the size of Q_{i-j} after the execution of START_EPOCH(x_{i-j+k}), for each $1 \leq k \leq \log(1/\varepsilon)$, is given by $|Q_{i-j}| = O(k^2/\sqrt{2^k\varepsilon})$. We need this fact below to bound the update time.

Update time. The bottleneck of the algorithm is in lines 3–4. It is easy to see that in the subroutine START_EPOCH(x_{i-j+k}), for $2 \leq k \leq \log(1/\varepsilon)$, the execution time of lines 6–12 in COMPRESS_EPOCH(Q_{i-j}) is bounded by

$$O\Big(\sum_{I \in \mathfrak{I}_{i-j}} |Q_I| \log |Q_I|\Big) = O\big((|Q_{i-j}| + k^2)\log(1/\varepsilon)\big) = O\big(|Q_{i-j}|\log(1/\varepsilon)\big); \tag{4}$$

here the first equality holds because $|\mathfrak{I}_{i-j}| = O(k^2)$ and $|Q_{I_1} \cap Q_{I_2}| \leq 1$ for any $I_1, I_2 \in \mathfrak{I}_{i-j}$, and the second equality holds because by line 5, lines 6–12 in COMPRESS_EPOCH are executed only if $|Q_{i-j}| > k^2/\sqrt{2^k\varepsilon} \geq k^2$ at that time. Let $\ell \geq 2$ be the integer so that lines 6–12 in COMPRESS_EPOCH are executed on Q_{i-j} for the first time in START_EPOCH$(x_{i-j+\ell})$. This implies $|K_{i-j}| \geq \ell^2/\sqrt{2^\ell\varepsilon}$, where K_{i-j} denotes the set Q_{i-j} right after the execution of

Start_Epoch(x_{i-j+1}). Then by (4), the total time for all executions of Compress_Epoch on Q_{i-j} is at most

$$O\big(|K_{i-j}|\log(1/\varepsilon)\big) + \sum_{k\ge\ell} O\big((k^2/\sqrt{2^k\varepsilon})\log(1/\varepsilon)\big) = O\big(|K_{i-j}|\log(1/\varepsilon)\big).$$

This cost can be amortized by charging $O(\log(1/\varepsilon))$ to each point in K_{i-j}. Together with Lemma 9, it then follows that the algorithm processes each point in the stream in $O(\log(1/\varepsilon))$ amortized time.

Putting things together, we obtain the main result of this paper.

Theorem 1 *Given a stream S of points in $\mathbb{R}^2$ and a fixed parameter $\varepsilon > 0$, we can maintain an ε-kernel of S using $O(1/\sqrt{\varepsilon})$ space and $O(\log(1/\varepsilon))$ amortized time per point.*

7 Extensions

In this section we prove a few implications of Theorem 1, including its application to maintaining extent measures in the streaming model, and generalizations to higher dimensions and robust kernels.

Maintaining extent measures. The application of Theorem 1 to maintaining $(1+\varepsilon)$-approximation of the various extent measures of a stream of points in $\mathbb{R}^2$ is straightforward. For example, in order to maintain a $(1+\varepsilon)$-approximation to the width, one simply maintains an ε-kernel $\mathcal{Q}$ of the stream and then maintains the width of $\mathcal{Q}$. The maintenance of other extent measures (e.g., minimum-volume bounding box, smallest enclosing ball, etc.) is similar.

Theorem 2 *Given a stream S of points in $\mathbb{R}^2$ and a fixed parameter $\varepsilon > 0$, we can maintain a $(1+\varepsilon)$-approximation of the width (minimum-volume bounding box, smallest enclosing ball, etc.) of S using $O(1/\sqrt{\varepsilon})$ space.*

Higher dimensions. Chan [9] also presented a data-stream algorithm for maintaining ε-kernels of streaming points in $\mathbb{R}^d$ that uses $O\big(1/\varepsilon^{d-3/2}\log^d(1/\varepsilon)\big)$ space and $O(1/\sqrt{\varepsilon})$ update time. We can slightly improve this result as follows.

Our algorithm closely follows Chan's algorithm [9] (except a simple twist), so we only give a sketch of the algorithm and refer the reader to the original paper. The stream of points in $\mathbb{R}^d$ are divided into epochs in a manner similar to the two-dimensional case. Only the last $\log(1/\varepsilon)$ epochs are active, and an ε-kernel for each of these epochs is stored; points in the older epochs are mapped to a fixed $(d-1)$-dimensional space in which an ε-kernel is maintained using a data-stream algorithm in $\mathbb{R}^{d-1}$ by induction. In the current epoch, an ε-kernel is maintained by rounding each point in this epoch onto one of $O(1/\varepsilon)$ parallel hyperplanes and maintaining ε-kernels within each of these hyperplanes using a data-streaming algorithm in $\mathbb{R}^{d-1}$ by induction. The twist is, whenever an epoch ends (and subsequently a new epoch starts), if the size of the maintained ε-kernel Q for this epoch is $\Omega(1/\varepsilon^{d-2})$, we then reduce it to $O(1/\varepsilon^{(d-1)/2})$ $(= O(1/\varepsilon^{d-2})$ for $d \ge 3)$ by computing an ε-kernel of Q using an algorithm of Chan [9], which runs in $O\big((|Q| + 1/\varepsilon^{d-2})\log(1/\varepsilon)\big) = O(|Q|\log(1/\varepsilon))$ time. This cost can be amortized by charging $O(\log(1/\varepsilon))$ to each point in Q.

When the recursion in dimension reaches $\mathbb{R}^2$, we simply apply our new algorithm as described in previous sections. The above algorithm maintains an $O(\varepsilon)$-kernel of S. The amortized time to process each point remains $O(\log(1/\varepsilon))$. Furthermore, the space $F(\varepsilon, d)$ used by the algorithm satisfies the following recurrence:

$$F(\varepsilon, d) = \begin{cases} O(1/\varepsilon) \cdot F(\varepsilon, d-1) + O\big((1/\varepsilon^{d-2}) \log(1/\varepsilon)\big) & d \geq 3, \\ O(1/\sqrt{\varepsilon}) & d = 2. \end{cases}$$

The recurrence solves to $F(\varepsilon, d) = O(1/\varepsilon^{d-3/2})$. We conclude:

Theorem 3 *Given a stream S of points in $\mathbb{R}^d$ and a fixed parameter $\varepsilon > 0$, we can maintain an ε-kernel of S using $O(1/\varepsilon^{d-3/2})$ space and $O(\log(1/\varepsilon))$ amortized time per point.*

Robust kernels. Har-Peled and Wang [17] introduced a generalized notion of ε-kernels, the so-called (k, ε)-*kernels*, as a tool for efficiently approximating extent measures of points in the presence of outliers. Formally, let P be a set of points in $\mathbb{R}^d$. The level of a point $p \in P$ in direction $u \in \mathbb{S}^{d-1}$ is the number of points $q \in P$ with $\langle q, u\rangle > \langle p, u\rangle$. We use $P_\ell[u]$ to denote the point in P at level ℓ along direction u. A subset $Q \subseteq P$ is called a (k, ε)-*kernel* if for every direction $u \in \mathbb{S}^{d-1}$, and every two nonnegative integers h, ℓ with $h + \ell = k$, we have

$$\langle Q_h[u] - Q_\ell[-u], u\rangle \geq (1 - \varepsilon) \cdot \langle P_h[u] - P_\ell[-u], u\rangle.$$

In the worst case a (k, ε)-kernel of a point set P in $\mathbb{R}^2$ has size $\Omega(k/\sqrt{\varepsilon})$.

As shown in [2], a data-stream algorithm for computing an ε-kernel can be used to compute a (k, ε)-kernel in the streaming model. If the former algorithm uses $N(\varepsilon)$ space and $T(\varepsilon)$ time, the latter uses $O(k \cdot N(\varepsilon/4))$ space and $O(k \cdot T(\varepsilon/4))$ time. Hence, by Theorem 3, we obtain the following.

Theorem 4 *Given a stream S of points in $\mathbb{R}^d$ and a fixed parameter $\varepsilon > 0$, we can maintain a (k, ε)-kernel of S using $O(k/\varepsilon^{d-3/2})$ space and $O(k \cdot \log(1/\varepsilon))$ amortized time per point, which is space-optimal for $d = 2$.*

8 Conclusions

In this paper we presented a data-stream algorithm for maintaining ε-kernels in $\mathbb{R}^2$ that uses $O(1/\sqrt{\varepsilon})$ space and processes each point in $O(\log(1/\varepsilon))$ amortized time. While it is known that the space complexity of the algorithm is optimal, it is an open problem whether the update time is also optimal if only $O(1/\sqrt{\varepsilon})$ space is allowed. Another interesting question is whether our technique can be extended to the sliding-window model [10].

References

[1] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *J. ACM*, 51:606–635, 2004.

[2] P. K. Agarwal, S. Har-Peled, and H. Yu. Robust shape fitting via peeling and grating coresets. *Discrete Comput. Geom.*, to appear.

[3] P. K. Agarwal, S. Krishnan, N. Mustafa, and S. Venkatasubramanian. Streaming geometric optimization using graphics hardware. In *Proc. 11th European Sympos. Algorithms*, pages 544–555, 2003.

[4] P. K. Agarwal, N. Mustafa, S. Har-Peled, and Y. Wang. Near-linear time approximation algorithms for curve simplification. *Algorithmica*, pages 203–219, 2005.

[5] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proc. 21st Annu. ACM Sympos. Principles of Database Systems*, pages 1–16, 2002.

[6] A. Bagchi, A. Chaudhary, D. Eppstein, and M. Goodrich. Deterministic sampling and range counting in geometric data streams. In *Proc. 20th Annu. Sympos. Comput. Geom.*, pages 144–151, 2004.

[7] G. Barequet and S. Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *J. Algorithms*, 38:91–109, 2001.

[8] J. Bentley and J. Saxe. Decomposable searching problems I: Static-to-dynamic transformations. *J. Algorithms*, 1:301–358, 1980.

[9] T. M. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Comput. Geom. Theory Appl.*, 35:20–35, 2006.

[10] T. M. Chan and B. Sadjad. Geometric optimization problems over sliding windows. *Internat. J. Comput. Geom. Appl.*, 16:145–157, 2006.

[11] G. Cormode and S. Muthukrishnan. Radial histograms for spatial streams. Technical report, DIMACS, 2003.

[12] R. Dudley. Metric entropy of some classes of sets with differentiable boundaries. *J. Approx. Theory*, 10:227–236, 1974.

[13] J. Feigenbaum, S. Kannan, and J. Zhang. Computing diameter in the streaming and sliding-window models. *Algorithmica*, 41:25–41, 2004.

[14] G. Frahling, P. Indyk, and C. Sohler. Sampling in dynamic data streams and applications. In *Proc. 21st Annu. Sympos. Comput. Geom.*, pages 142–149, 2005.

[15] G. Frahling and C. Sohler. Coresets in dynamic geometric data streams. In *Proc. 37th Annu. ACM Sympos. Theory Comput.*, pages 209–217, 2005.

[16] S. Har-Peled and S. Mazumdar. Coresets for k-means and k-medians and their applications. In *Proc. 36th Annu. ACM Sympos. Theory Comput.*, pages 291–300, 2004.

[17] S. Har-Peled and Y. Wang. Shape fitting with outliers. *SIAM J. Comput.*, 33:269–285, 2004.

[18] J. Hershberger and S. Suri. Adaptive sampling for geometric problems over data streams. In *Proc. 23rd ACM Sympos. Principles of Database Systems*, pages 252–262, 2004.

[19] P. Indyk. Streaming algorithms for geometric problems. In *Proc. 24th Intl. Conf. Foundat. Software Tech. Theoret. Comput. Sci.*, pages 32–34, 2004.

[20] J. Mitchell and S. Suri. Separation and approximation of polyhedral objects. *Comput. Geom. Theory Appl.*, pages 95–114, 1995.

[21] S. Muthukrishnan. *Data Streams: Algorithms and Applications.* Now Publishers Inc., 2005.

[22] S. Suri, C. Toth, and Y. Zhou. Range counting over multidimensional data streams. *Discrete Comput. Geom.*, pages 633–655, 2006.

[23] H. Yu, P. K. Agarwal, R. Poreddy, and K. R. Varadarajan. Practical methods for shape fitting and kinetic data structures using coresets. *Algorithmica*, to appear.