An Almost Space-Optimal Streaming Algorithm for Coresets in Fixed Dimensions

Hamid Zarrabi-Zadeh*

Abstract

We present a new streaming algorithm for maintaining an ε-kernel of a point set in $\mathbb{R}^d$ using $O((1/\varepsilon^{(d-1)/2})\log(1/\varepsilon))$ space. The space used by our algorithm is optimal up to a small logarithmic factor. This significantly improves (for any fixed dimension $d \geqslant 3$) the best previous algorithm for this problem that uses $O(1/\varepsilon^{d-(3/2)})$ space, presented by Agarwal and Yu. Our algorithm immediately improves the space complexity of the previous streaming algorithms for a number of fundamental geometric optimization problems in fixed dimensions, including width, minimum-volume bounding box, minimum-radius enclosing cylinder, minimum-width enclosing annulus, etc.

1 Introduction

The coreset framework has recently attracted considerable attention as a powerful tool for approximating various measures of a geometric data set. In this framework, a small subset of the input point set, called a *coreset*, is extracted in such a way that solving the optimization problem on the coreset yields an approximate solution to the entire set.

Agarwal *et al.* [2] developed a generic method for computing coresets for various optimization problems by introducing the notion of ε-kernel. Roughly speaking, a subset $Q \subseteq P$ is called an *ε-kernel* of P if for every slab S containing Q, the $(1+\varepsilon)$-expansion of S contains P. The technique of Agarwal *et al.* yields approximation algorithms for a wide range of shape-fitting problems.

The coreset framework also plays an essential role in designing approximation algorithms operating under the *data stream* model. In this model, the input is given to the algorithm as a stream over time, and the algorithm has to process the input elements as they arrive in only one pass. Furthermore, the algorithm has only a limited amount of working storage and cannot store the whole input in its memory. This one-pass streaming model is attractive both in theory and in practice due to emerging applications which involve massive data sets. The coreset framework is useful here as it allows streaming algorithms to maintain only a "sketch" of the input, which is typically small compared to the whole data set. For example, see [1, 9, 10, 11] on the growing literature of streaming algorithms developed over the recent few years for various geometric problems using the notion of coresets.

In this paper, we are interested in space-efficient streaming algorithms for maintaining ε-kernels in $\mathbb{R}^d$. Using the general dynamization technique of Bentley and Saxe [8], Agarwal *et al.* [2] gave a streaming algorithm for maintaining an ε-kernel of a stream of points in $\mathbb{R}^d$ using $O((1/\varepsilon^{(d-1)/2})\log^d n)$ space and $O(1/\varepsilon^{d-1})$ time per update, where n is the number of points in the stream. Chan [9] succeeded to remove the dependency of the space bound to n and gave the first constant-space streaming

*School of Computer Science, University of Waterloo, Waterloo, Ont. N2L 3G1, Canada; hzarrabi@uwaterloo.ca

Table 1. Space complexity of various streaming algorithms for maintaining ε-kernels in $\mathbb{R}^d$.

Algorithm	Space Bound	Ref
Agarwal, Har-Peled, and Varadarajan '04	$O((1/\varepsilon^{\frac{d-1}{2}})\log^d n)$	[2]
Chan '06	$O((1/\varepsilon^{d-(3/2)})\log^d(1/\varepsilon))$	[9]
Agarwal and Yu '07	$O(1/\varepsilon^{d-(3/2)})$	[6]
This work	$O((1/\varepsilon^{\frac{d-1}{2}})\log(1/\varepsilon))$	Here

algorithm that uses only $O([(1/\varepsilon)\log(1/\varepsilon)]^{d-1})$ space and requires $O(1)$ amortized time for processing each new point. He also showed how the space bound can be improved to $O((1/\varepsilon^{d-(3/2)})\log^d(1/\varepsilon))$ at the expense of increasing the update time to $O(1/\sqrt{\varepsilon})$. Later on, Agarwal and Yu [6] removed the extra logarithmic factors and slightly improved the space complexity to $O(1/\varepsilon^{d-(3/2)})$, with $O(\log(1/\varepsilon))$ update time per input point. Agarwal and Yu's algorithm is indeed space-optimal in two dimensions, but is still far from optimal in dimensions higher than two.

Our Results. Chan [9] left this question open whether the space bound for the problem of maintaining ε-kernels in $\mathbb{R}^d$ can be brought down to near $O(1/\varepsilon^{(d-1)/2})$. In this paper, we answer Chan's question in the affirmative by providing a streaming algorithm that uses a near optimal space. More precisely, our algorithm maintains an ε-kernel in $\mathbb{R}^d$ using only $O((1/\varepsilon^{(d-1)/2})\log(1/\varepsilon))$ space and $(1/\varepsilon)^{O(d)}$ update time per insertion. The space bound of our algorithm is optimal up to a logarithmic factor, as one can easily verify that any ε-kernel for sufficiently many points uniformly distributed on the surface of a d-dimensional hypersphere has size $\Omega(1/\varepsilon^{(d-1)/2})$ [3]. Our algorithm differs from its predecessors [6, 9] in that the high level structure of our algorithm is not dimensionality-reduction (which ultimately exploits efficient techniques in two dimensions), but rather a high-dimensional partition of the input stream into d-dimensional "fat substreams" for which ε-kernels can be maintained efficiently by a kind of bucketing scheme. Our algorithm can be viewed as a generalization of the algorithm proposed by Chan [9] in two dimensions, which also employs a version of the compression technique used in [6]. See Table 1 for a comparison between the space complexity of our algorithm and the previous ones.

Our algorithm immediately improves the space complexity of the best previous streaming algorithms for a wide range of geometric problems in fixed dimensions, including width, minimum-volume bounding box, minimum-radius enclosing cylinder, minimum-width enclosing cylindrical shell, etc. The improvement obtained by our algorithm is substantial when the problem's dimension is large. However, the algorithm improves several previous results in lower dimensions as well. For example, for the two-dimensional minimum-width enclosing annulus problem, combined with the lifting technique of Agarwal *et al.* [2], our algorithm requires only $O((1/\varepsilon)\log(1/\varepsilon))$ space, while the best previous space bound for this problem was $O(1/\varepsilon^{3/2})$. As a byproduct of our algorithm, we also show how to maintain an ε-kernel of a stream of points in $\mathbb{R}^d$ in an optimal time of $O(1)$ using $O((1/\varepsilon^{d-(3/2)})\log(1/\varepsilon))$ space, which improves the best previously known time-optimal algorithm by Chan [9] that requires $O([(1/\varepsilon)\log(1/\varepsilon)]^{d-1})$ space.

2 Preliminaries

We first introduce the notation used in this paper. For a point set $P \subset \mathbb{R}^d$ and a direction $u \in \mathbb{S}^{d-1}$, the *directional width* of P along u is defined by $w(P,u) = \max_{p,q \in P} \langle p-q, u\rangle$, where $\langle \cdot, \cdot \rangle$ denotes the inner product function. A subset $Q \subseteq P$ is called an *ε-kernel* of P, if for all $u \in \mathbb{S}^{d-1}$,

$$w(Q,u) \geqslant (1-\varepsilon) w(P,u).$$

We will use the following result from Chan throughout this paper:

Theorem 1 (Chan [9]) *Given a set P of n points in $\mathbb{R}^d$, an ε-kernel of P of size $O(1/\varepsilon^{(d-1)/2})$ can be computed in $O(n + 1/\varepsilon^{d-(3/2)})$ time for $d \geqslant 2$, or in $O((n + 1/\varepsilon^{d-2}) \log(1/\varepsilon))$ time for $d \geqslant 3$.*

Let $S = \{p_1, \ldots, p_k\}$ be a set of k points in $\mathbb{R}^d$ ($k \leqslant d+1$). We denote by $\mathcal{F}(S)$ the *flat* spanned by S, i.e., $\mathcal{F}(S) = \left\{ \sum_{i=1}^k a_i p_i \mid a_1, \ldots, a_k \in \mathbb{R} \text{ and } \sum_{i=1}^k a_i = 1 \right\}$. Given a point $p \in \mathbb{R}^d$ and a flat $\mathcal{F} \subset \mathbb{R}^d$, the *orthogonal projection* of p onto $\mathcal{F}$ is defined as $\mathrm{proj}(\mathcal{F}, p) = \arg\min_{p' \in \mathcal{F}} \|pp'\|$. The Euclidean distance between p and its projection onto $\mathcal{F}$ is denoted by $d_{\mathcal{F}}(p)$. Throughout this paper, we simply use $d_S(p)$ instead of $d_{\mathcal{F}(S)}(p)$ to refer to the distance of p to a flat defined by the point set S. If $S = \emptyset$, then $d_S(p) = 0$ by definition.

Let $X = \langle x_0, x_1, \ldots, x_k \rangle$ be a sequence of $k+1$ points in $\mathbb{R}^d$ ($k \leqslant d$). We say that a point $p \in \mathbb{R}^d$ is *X-respecting*, if for every $0 \leqslant i < k$,

$$d_{X_i}(p) \leqslant 2 \cdot d_{X_i}(x_{i+1}),$$

where $X_i = \{x_0, x_1, \ldots, x_i\}$. If $|X| \leqslant 1$, then every point is X-respecting by definition. The sequence X is called *self-respecting*, if for every $1 \leqslant j \leqslant k$, x_j is $\langle x_0, \ldots, x_{j-1}\rangle$-respecting. Moreover, a point set P is called X-respecting, if for every point $p \in P$, p is X-respecting.

In this paper, we assume a real-RAM model of computation [13] in which arithmetic operations on real numbers take constant time.

3 Fat Substreams

In this section, we present a simple efficient algorithm for maintaining ε-kernels of fat substreams to be used as a subroutine in Section 4. Let $\mathbb{B}$ be a hyperbox in $\mathbb{R}^d$, and $\alpha \leqslant 1$ be a positive constant. A point set $P \subset \mathbb{R}^d$ is called *α-fat with respect to $\mathbb{B}$*, if there exist two points v and v' so that $v + \alpha\mathbb{B} \subseteq \mathrm{conv}(P) \subseteq v' + \mathbb{B}$. If P is fat with respect to a hyperbox $\mathbb{B}$, then an ε-kernel of P of size $O(1/\varepsilon^{(d-1)/2})$ can be computed efficiently using a simple grid-rounding method [9, 14] to be described later in this section. This grid-rounding method actually works for the case where $\mathbb{B}$ is a hypercube. However, we can easily transform $\mathbb{B}$ to a hypercube by an affine transform τ, and then compute an ε-kernel Q of the set $\tau(P)$. The set $\tau^{-1}(Q)$ is then an ε-kernel of P, as proved in [2]. In the following, we show how this idea can be used for X-respecting substreams.

Let $X = \langle x_0, x_1, \ldots, x_d \rangle$ be a sequence of $d+1$ points in $\mathbb{R}^d$. For each $1 \leqslant i \leqslant d$, we denote by $\hat{x}_i$ the projection of x_i onto $\mathcal{F}(X_{i-1})$, where $X_i = \{x_0, x_1, \ldots, x_i\}$. Let $w_i = \|\hat{x}_i x_i\|$ and $u_i = (1/w_i)\overrightarrow{\hat{x}_i x_i}$. We denote by B_X the d-dimensional box centered at x_0, whose i-th side has length $2w_i$ in direction u_i ($1 \leqslant i \leqslant d$), and define $\mathcal{B}_X = 2B_X$. The following lemma (which is analogous to what is proved in [7] for three dimensions) provides a connection between X-respecting and fat sets.

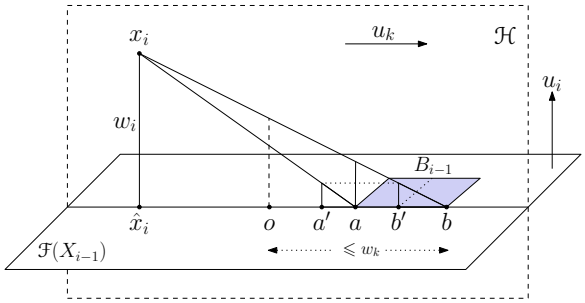

Figure 1. Proof of Lemma 1.

Lemma 1 *Let X be a self-respecting sequence of $d+1$ points in $\mathbb{R}^d$. Given a point set $P \subset \mathbb{R}^d$, if P is X-respecting, then $P \cup X$ is $(1/4)^d$-fat with respect to $\mathcal{B}_X$.*

Proof: Obviously, $\mathcal{B}_X$ contains all the points of $P \cup X$. We show that $\mathrm{conv}(X)$ (and therefore, $\mathrm{conv}(P \cup X)$) contains a translated copy of $(1/4)^d \mathcal{B}_X$. Suppose by induction that $\mathrm{conv}(X_{i-1})$ contains an $(i-1)$-dimensional box $B_{i-1} \subset B_X$, whose j-th side has length $w_j/4^{i-1}$ in direction u_j $(1 \leqslant j < i)$. Now, consider the pyramid $\mathcal{P}_i = \mathrm{conv}(B_{i-1} \cup \{x_i\})$. Clearly, $\mathcal{P}_i \subseteq \mathrm{conv}(X_i)$. We only need to show that $\mathcal{P}_i$ contains an i-dimensional box $B_i \subset B_X$, whose j-th side has length at least $w_j/4^i$ in direction u_j $(1 \leqslant j \leqslant i)$.

Fix a k $(1 \leqslant k < i)$, and consider the plane $\mathcal{H}$ through $x_i\hat{x}_i$ parallel to u_k (see Fig. 1). The projection of B_{i-1} onto $\mathcal{H}$ is a line segment ab of length $w_k/4^{i-1}$. Let o be the orthogonal projection of x_0 onto $\mathcal{H}$. Then we have $\|oa\|, \|ob\| \leqslant w_k$ (because $B_{i-1} \subset B_X$), and $\|o\hat{x}_i\| \leqslant \|x_0x_i\| \leqslant 2w_k$ (because X is self-respecting).

Let a' (respectively, b') be a point on $\overleftrightarrow{ab}$ whose vertical distance (in direction u_i) from x_ia (respectively, from x_ib) is equal to $w_i/4^i$. We have $\|aa'\|, \|bb'\| \leqslant 3w_k/4^i$, due to similarity of triangles, and because $\|a\hat{x}_i\|, \|b\hat{x}_i\| \leqslant 3w_k$. Let $s_k = \overline{ab} - (\overline{aa'} \cup \overline{bb'})$. If one of the two angles $\angle abx_i$ and $\angle bax_i$ is obtuse, then one of the segments aa' and bb' falls completely outside ab, and therefore $\|s_k\| \geqslant w_k/4^i$. If both $\angle abx_i$ and $\angle bax_i$ are at most $\pi/2$, then $\|aa'\| + \|bb'\| = \|ab\|/4 = w_k/4^i$, and therefore, $\|s_k\| = 3w_k/4^i \geqslant w_k/4^i$. Now, we cut that portion of B_{i-1} whose projection onto $\mathcal{H}$ lies inside s_k, and repeat this procedure for every $1 \leqslant k < i$. The remaining box, B'_{i-1}, has length at least $w_k/4^i$ in each direction u_k. If we expand B'_{i-1} by $w_i/4^i$ units in direction u_i, we obtain the desired i-dimensional box $B_i \subset B_X$ which completely remains inside $\mathcal{P}_i$. □

Note that the fatness parameter in Lemma 1 is exponential in d, but is a constant in any fixed dimension.

Algorithm for Fat Streams. Let P be a point set in $\mathbb{R}^d$ which is fat with respect to a d-dimensional hyperbox B. We may assume that B is centered at origin by a simple translation. Moreover, we may assume that $B = [-1, 1]^d$ by an affine transform[1]. Now, we can easily compute an ε-kernel of P using the following grid-rounding method proposed in [9, 14]: Let $\mathcal{R}$ be the set of points of a $\sqrt{\varepsilon}$-grid over the boundary of the cube $[-2, 2]^d$, and let $\xi_S(r)$ denote the nearest neighbor of a point

[1]In order for this affine transform to exist, B must have a positive width along each of its d axes. If this is not the case, we simply ignore all those axes along which B has zero width and consider B to be a hyperbox in a lower dimension.

$r \in \mathcal{R}$ in the set S. Then the set $Q = \{\xi_P(r) \mid r \in \mathcal{R}\}$ is an ε-kernel of P (see Fig. 2). Obviously, $|Q| \leqslant |\mathcal{R}| = O(1/\varepsilon^{(d-1)/2})$. It just remains to show how we can efficiently maintain Q when new points are inserted into P, while P remains fat with respect to B.

Let $\textsc{Kernel}(S) = \{\xi_S(r) \mid r \in \mathcal{R}\}$. The function $\textsc{Insert-Box}$ described below inserts a point p into the fat stream P (enclosed by B) and returns an ε-kernel of P. The algorithm maintains two subsets Q_0 and Q_1 at each time, which are initially empty.

$B.\textsc{Insert-Box}(p)$:

1: $Q_1 \leftarrow Q_1 \cup \{p\}$
2: **if** $|Q_1| > 1/\varepsilon^{(d-1)/2}$ **then**
3: $\quad Q_0 \leftarrow \textsc{Kernel}(Q_0 \cup Q_1)$
4: $\quad Q_1 \leftarrow \emptyset$
5: return $Q_0 \cup Q_1$

The algorithm divides the stream P into substreams of size $\lfloor 1/\varepsilon^{(d-1)/2} \rfloor$. Whenever a substream is completely received, it is merged to the kernel maintained for the previous substreams in order to obtain a single kernel for the whole stream received so far. The correctness of the algorithm immediately follows from the following two facts: (i) $\textsc{Kernel}(P \cup Q) \subseteq \textsc{Kernel}(P) \cup \textsc{Kernel}(Q)$, and (ii) $\textsc{Kernel}(\textsc{Kernel}(P)) = \textsc{Kernel}(P)$. The kernel in line 3 can be computed using Theorem 1 in $O(n+1/\varepsilon^{d-(3/2)})$ or $O((n+1/\varepsilon^{d-2})\log(1/\varepsilon))$ time, where $n = |Q_0 \cup Q_1| = \Theta(1/\varepsilon^{(d-1)/2})$. Therefore, the amortized update time charged to each input point is $O(1 + (1/\varepsilon^{(d-3)/2})\log(1/\varepsilon))$. We conclude:

Theorem 2 *Given a stream of points P in $\mathbb{R}^d$ which is fat with respect to a fixed hyperbox, an ε-kernel of P can be maintained using $O(1/\varepsilon^{(d-1)/2})$ space and $O(1 + (1/\varepsilon^{(d-3)/2})\log(1/\varepsilon))$ amortized time per input point.*

Remark. In two dimensions, Agarwal and Yu [6] used a balanced binary search tree to maintain an ε-kernel of a fat stream in $O(\log(1/\varepsilon))$ time. Theorem 2 immediately improves their method by providing an algorithm that requires only $O(1)$ amortized time, without using any extra data structure.

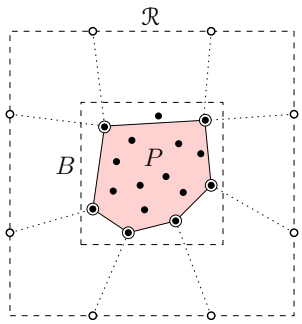

Figure 2. Constructing an ε-kernel of a fat point set.

Corollary 1 *Let X be a self-respecting sequence of $d+1$ points in $\mathbb{R}^d$. Given an X-respecting stream P in $\mathbb{R}^d$, an ε-kernel of $P \cup X$ can be maintained using $O(1/\varepsilon^{(d-1)/2})$ space and $O(1+(1/\varepsilon^{(d-3)/2})\log(1/\varepsilon))$ amortized update time.*

Proof: By Lemma 1, $P \cup X$ is fat with respect to $\mathcal{B}_X$. Therefore, we can use the INSERT-BOX function on $\mathcal{B}_X$ with the only exception that Q_0 is initially set to X. □

4 The Main Algorithm

In this section, we describe our main algorithm for maintaining an ε-kernel of a data stream $P \subset \mathbb{R}^d$. The INSERT function presented in Fig. 3 inserts a point p into an X-respecting stream P and returns an ε-kernel Q of P. Each new point p is inserted into the stream by calling P.INSERT$(p, \langle\rangle)$, where $\langle\rangle$ denotes the empty sequence. In this algorithm, $b = \lceil \log(1/\varepsilon) \rceil$, and ϕ_i (used in line 15) is a function to be defined shortly.

P.INSERT(p, X):

```
1:  if p is the first point of P then
2:      i ← 1, v_1 ← p
3:  if |X| = d + 1 then
4:      return Q = B_X.INSERT-BOX(p)
5:  if d_X(p) ⩽ 2 · d_X(v_i) then
6:      Q_i ← P_i.INSERT(p, X + ⟨v_i⟩)
7:  else
8:      if |Q_i| > 1/ε^{(d−1)/2} then                    ┐
9:          Q_i ← ε-KERNEL(Q_i)                          │ compression step
10:     P_i.FREE()                                       ┘
11:     i ← i + 1, v_i ← p
12:     Q_i ← P_i.INSERT(p, X + ⟨v_i⟩)
13:     if i − b > 0 then                                ┐
14:         for each q ∈ Q_{i−b} do                      │
15:             q' ← φ_1 ∘ ··· ∘ φ_i(q)                  │ merging step
16:             Q' ← P'.INSERT(q', ⟨⟩)                   │
17:         Q_0 ← {q | q' ∈ Q'}                          │
18:         Q_{i−b} ← ∅                                  ┘
19: return Q = ∪_{j=0}^{i} Q_j
```

Figure 3. The main algorithm.

The overall structure of the algorithm is simple. For an easier understanding, one can temporarily ignore the two code blocks in the algorithm labeled by "compression" and "merging" steps (we need these two steps only to guarantee the space bound). The algorithm works recursively, and the level

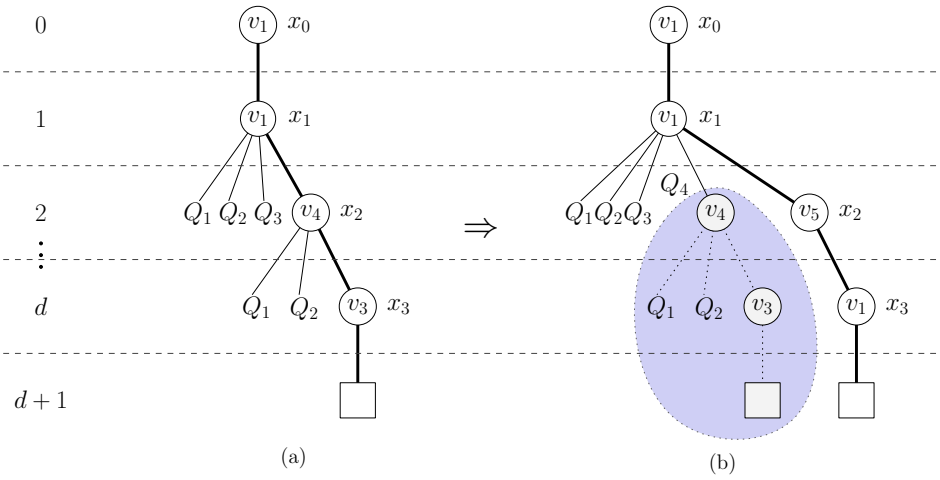

Figure 4. An example of the execution of the algorithm for $d = 3$. (a) All $\langle x_0, x_1, x_2, x_3\rangle$-respecting points go down through the levels until they are inserted into a hyperbox $\mathcal{B}_{\mathcal{X}}$ at level $d+1$. (b) A new point p has arrived which is $\langle x_0, x_1\rangle$-respecting, but not $\langle x_0, x_1, v_4\rangle$-respecting. As a result, the kernel previously maintained for the substream started by v_4 is compressed and stored in Q_4, the subtree rooted at v_4 is discarded, and a new substream is started with $v_5 = p$ as its first point.

of recursion is controlled by an input sequence X. We start with an empty sequence $X = \langle\rangle$ at the topmost level (level 0), and add one point to X at each subsequent level. After $d+1$ recursive calls (i.e., at level $d+1$), we reach a set X of $d+1$ points that defines a d-dimensional hyperbox $\mathcal{B}_X$. The idea is to recursively partition the input stream into substreams each of which is fat with respect to a hyperbox so that an ε-kernel for each substream can be maintained efficiently using the method described in Section 3.

At any time, we have $d+2$ instances of the algorithm corresponding to the levels 0 to $d+1$ of the recursion. Each instance receives as input a stream of points P (which is a subset of its parent's stream), and a sequence of points X such that P is X-respecting. It then divides its input X-respecting stream P into substreams $P_1, \ldots, P_i$ started by the points $v_1, \ldots, v_i$, where v_1 is the first point of P, and each subsequent v_i is chosen by the algorithm as the first point of P for which $d_X(v_i) > 2 \cdot d_X(v_{i-1})$. Each substream P_i is thus $(X + \langle v_i\rangle)$-respecting.

Let x_k denote the latest value of v_i at level k ($0 \leqslant k \leqslant d$), and let $\mathcal{X} = \langle x_0, \ldots, x_d\rangle$ (see Fig. 4). When a new input point p arrives, it is recursively examined against $\mathcal{X}$ by the algorithm. If p is $\mathcal{X}$-respecting, then it finds its way down to level d by recursively calling the INSERT function in line 6. When the recursion in the size of X reaches $|X| = d+1$ (line 3), the point p is inserted into the hyperbox $\mathcal{B}_{\mathcal{X}}$ using the INSERT-BOX function described in Section 3. If the input point p is not $\mathcal{X}$-respecting, then at some level k ($0 \leqslant k \leqslant d$) the condition at line 5 fails, and therefore, we need to start a new substream P_i. This is done in lines 11–12 by incrementing i, updating $v_i (= x_k)$, and inserting p into the newly started substream P_i via a recursive call to the INSERT function (see Fig. 4(b)).

To guarantee the space bound, two steps have been added to the algorithm. The compression step in lines 8-10 is invoked before starting any new substream P_{i+1} to ensure that the size of the ε-kernel Q_i maintained for the current substream P_i is $O(1/\varepsilon^{(d-1)/2})$; if not, it is reduced to $O(1/\varepsilon^{(d-1)/2})$ in line 9 using Theorem 1. All kernels previously maintained for the substreams of P_i are also discarded by calling function FREE in line 10.

The merging step in lines 13–18 ensures that at most $b = \lceil \log(1/\varepsilon)\rceil$ kernels $Q_{i-b+1}, \ldots, Q_i$ are

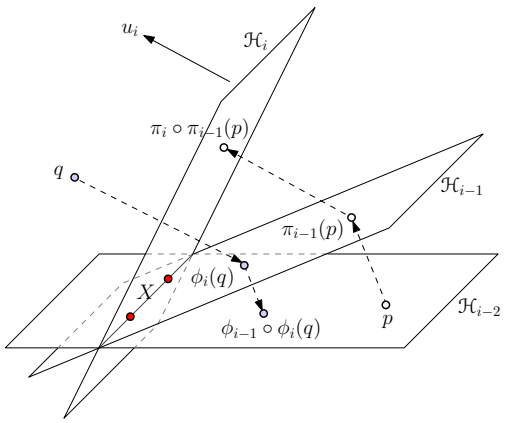

Figure 5. Mapping functions π_i and ϕ_i.

active at any level; earlier ones are merged together to form a $(d-1)$-dimensional substream P' whose kernel is maintained in a set Q'. We are justified to do this because of the following observation due to Chan [9]: points in earlier subsets are so close to the flat $\mathcal{F}(X)$ that rounding them to a single $(d-1)$-dimensional hyperplane passing through X is sufficient for representing the extent of the points in the earlier subsets. The mapping $\phi_1 \circ \cdots \circ \phi_i$ used in line 15 is defined as follows: let $\hat{v}_i = \text{proj}(\mathcal{F}(X), v_i)$, and let $u_i = \overrightarrow{\hat{v}_i v_i}$. We denote by $\mathcal{H}_0$ an arbitrary hyperplane through X, and for $1 \leqslant i \leqslant d$, denote by $\mathcal{H}_i$ the hyperplane through X perpendicular to u_i. The function ϕ_i denotes the projection to $\mathcal{H}_{i-1}$ parallel to the direction u_i (see Fig. 5)[2]. We keep the mapping function $\phi_1 \circ \cdots \circ \phi_i$ in a single matrix and update it only once whenever i is increased.

5 Analysis

In this section we prove the correctness of our algorithm, and analyze its space and time complexities. The notation used here closely follows the one used in [9]. Let π_i denote the orthogonal projection onto $\mathcal{H}_i$. Note that π_i is a weak inverse of ϕ_i in the sense that $\pi_i \circ \cdots \circ \pi_1 \circ \phi_1 \circ \cdots \circ \phi_i = \pi_i$ (see Fig. 5). In the following analysis, we fix an instance of the algorithm (corresponding to a level k of the recursion) that receives as input a point stream P and a sequence X of k points such that P is X-respecting. Let f denote the latest value of i, and $\psi = \pi_f \circ \cdots \circ \pi_1$. We first prove two technical lemmas.

Lemma 2 *For every $i \leqslant f$ and every direction $u \in \mathbb{S}^{d-1}$, $\langle v_i - \pi_i(v_i), u \rangle \leqslant 4^d w(P \cup X, u)$.*

Proof: Let $\mathcal{X}$ be a sequence of $d+1$ points obtained as follows: starting from $\mathcal{X} = X$, we repeatedly add to $\mathcal{X}$ a point from P which is farthest from $\mathcal{F}(\mathcal{X})$, until $\mathcal{X}$ has $d+1$ points. Obviously, P is $\mathcal{X}$-respecting and $P \cup \mathcal{X} = P \cup X$. Thus, by Lemma 1, $\text{conv}(P \cup X)$ is sandwiched between $\mathcal{B}_\mathcal{X}$ and a translated copy of $(1/4)^d \mathcal{B}_\mathcal{X}$. Both v_i and $\pi_i(v_i)$ lie inside $\mathcal{B}_\mathcal{X}$. Therefore, for each direction $u \in \mathbb{S}^{d-1}$,

$$\langle v_i - \pi_i(v_i), u \rangle \leqslant w(\mathcal{B}_\mathcal{X}, u) = 4^d w((1/4)^d \mathcal{B}_\mathcal{X}, u) \leqslant 4^d w(P \cup X, u).$$

□

[2] We assume that $\mathcal{H}_i$ is not orthogonal to $\mathcal{H}_{i-1}$. Degeneracies can be avoided by general perturbation techniques.

Lemma 3 *Let $q \in Q_{j-b}$ for some $b < j \leqslant f$, and let $q' = \phi_1 \circ \cdots \circ \phi_j(q)$. Then for every direction $u \in \mathbb{S}^{d-1}$, $\langle q - \psi(q'), u\rangle \leqslant 4^{d+1}\varepsilon w(P \cup X, u)$.*

Proof: By the doubling property we have $d_X(q) \leqslant 2d_X(v_{j-b}) \leqslant 2^{1-b}d_X(v_j)$. Moreover, $\langle q - \pi_i(q), u\rangle / d(q, \mathcal{H}_i) = \langle v_i - \pi_i(v_i), u\rangle / d(v_i, \mathcal{H}_i)$. Since $d_X(v_i) = d(v_i, \mathcal{H}_i)$ and $d_X(q) \geqslant d(q, \mathcal{H}_i)$, we have

$$\langle q - \pi_i(q), u\rangle \leqslant \frac{d_X(q)}{d_X(v_i)} \langle v_i - \pi_i(v_i), u\rangle \leqslant \frac{d_X(q)}{d_X(v_i)} \cdot 4^d w(P \cup X, u), \tag{1}$$

where the last inequality holds by Lemma 2. Now, define $q_j = q$, and $q_t = \pi_{t-1} \circ \cdots \circ \pi_j(q)$ for all $t > j$. It is clear that $\pi_i(q_i) = q_{i+1}$. Therefore,

$$\sum_{i=j}^{f} \langle q_i - \pi_i(q_i), u\rangle = \sum_{i=j}^{f} \langle q_i - q_{i+1}, u\rangle \geqslant \langle q_j - q_{f+1}, u\rangle . \tag{2}$$

Furthermore,

$$\sum_{i=j}^{f} \frac{d_X(q_i)}{d_X(v_i)} \leqslant \sum_{i=j}^{f} \frac{1}{2^{(j-i)}} \cdot \frac{d_X(q_i)}{d_X(v_j)} \leqslant 2 \cdot \frac{d_X(q)}{d_X(v_j)} \leqslant 2(2^{1-b}) \leqslant 4\varepsilon.$$

Therefore, if we replace q by q_i in (1) and sum up the inequality over i from j to f, we get

$$\sum_{i=j}^{f} \langle q_i - \pi_i(q_i), u\rangle \ \leqslant \ \sum_{i=j}^{f} \frac{d_X(q_i)}{d_X(v_i)} \cdot 4^d w(P \cup X, u) \ \leqslant \ 4^{d+1}\varepsilon w(P \cup X, u). \tag{3}$$

The lemma statement follows by (2) and (3) and the fact that $q_{f+1} = \pi_f \circ \cdots \circ \pi_j(q) = \psi(q')$, due to the weak-inverse relationship between π_i's and ϕ_i's. □

Theorem 3 *Given a stream of points P in $\mathbb{R}^d$, an ε-kernel of P can be maintained using $O(1/\varepsilon^{(d-1)/2} \log(1/\varepsilon))$ space and $O(1 + (1/\varepsilon^{(d-3)/2}) \log(1/\varepsilon))$ update time.*

Proof: We show that for every X-respecting stream P, the set Q returned by our algorithm is an ε-kernel of $P \cup X$. The proof is by induction on the size of X. If $|X| = d + 1$ (the base case), then the set Q computed in line 4 is an ε-kernel of $P \cup X$ by Corollary 1. Otherwise, $|X| \leqslant d$. Consider an arbitrary point $p \in P$. The algorithm inserts p into a substream P_i $(1 \leqslant i \leqslant f)$ upon its arrival. If $i = f$, then the set Q_i is an ε-kernel of $P_i \cup X$ by induction. If $f - b < i < f$, then Q_i has passed the compression step in lines 8–9, but it is still active, i.e., is not merged into Q_0. Therefore, Q_i is a (2ε)-kernel of $P_i \cup X$ due to the fact that an ε-kernel of a ε'-kernel of a set, is an $(\varepsilon + \varepsilon')$-kernel of that set. The only remaining case is when $i \leqslant f - b$. In this case, Q_i is merged into Q_0 in lines 13–18. Since Q_i is a (2ε)-kernel of $P_i \cup X$ before merging, there exists a point $q \in Q_i$ such that for every direction $u \in \mathbb{S}^{d-1}$, $\langle q, u\rangle \geqslant \langle p, u\rangle - 2\varepsilon w(P_i \cup X, u)$. The mapped point of q, q', is inserted into P' in line 16. Since Q' is an ε-kernel of P', $\psi(Q')$ is an ε-kernel of $\psi(P')$. Moreover, there exists a point $r \in Q_0$ with $r' \in Q'$, such that $\langle r', u\rangle \geqslant \langle q', u\rangle - \varepsilon w(P', u)$, and hence

$$\big\langle \psi(r'), u\big\rangle \geqslant \big\langle \psi(q'), u\big\rangle - \varepsilon w(\psi(P'), u). \tag{4}$$

Let $\rho = 4^{d+1}$. By Lemma 3, for every $q \in Q_1 \cup \cdots \cup Q_{f-b}$ and its corresponding $q' \in P'$,

$$\langle q, u\rangle - \rho\varepsilon w(P \cup X, u) \leqslant \big\langle \psi(q'), u\big\rangle \leqslant \langle q, u\rangle + \rho\varepsilon w(P \cup X, u),$$

which implies that $w(\psi(P'), u) \leqslant w(Q_1 \cup \cdots \cup Q_{f-b}, u) + 2\rho\varepsilon w(P \cup X, u) \leqslant (1 + 2\rho\varepsilon)w(P \cup X, u)$. Furthermore, by Lemma 3 we have $\langle \psi(r'), u\rangle \leqslant \langle r, u\rangle + \rho\varepsilon w(P \cup X, u)$ and $\langle \psi(q'), u\rangle \geqslant \langle q, u\rangle - \rho\varepsilon w(P \cup X, u)$. Replacing in (4), we get

$$\langle r, u\rangle + \rho\varepsilon w(P \cup X, u) \geqslant \langle q, u\rangle - \rho\varepsilon w(P \cup X, u) - \varepsilon[(1 + 2\rho\varepsilon)w(P \cup X, u)],$$

and hence, $\langle r, u\rangle \geqslant \langle q, u\rangle - O(\varepsilon)w(P \cup X, u)$. Since $\langle q, u\rangle \geqslant \langle p, u\rangle - 2\varepsilon w(P \cup X, u)$, we have $\langle r, u\rangle \geqslant \langle p, u\rangle - O(\varepsilon)w(P \cup X, u)$. Therefore, in any of the above cases, there exists a point in $\cup_{i=0}^{f} Q_i$ whose projected length along direction u differs from that of p by at most $O(\varepsilon)w(P \cup X, u)$, and hence, $Q = \cup_{i=0}^{f} Q_i$ is an $O(\varepsilon)$-kernel of $P \cup X$. (Note that in our proof, the algorithm returns an $O(\varepsilon)$-kernel rather than an ε-kernel; but this is not a problem as the depth of the recursion tree of our algorithm is $d + 1$, and therefore, we can adjust ε at the beginning by a constant, depending only on d.)

Space Complexity. Let $S(d, k)$ denote the space used by the algorithm to compute an ε-kernel of the d-dimensional X-respecting stream P, where $|X| = k$. Then, $|Q_0| = |Q'| = S(d - 1, 0)$, $|Q_1|, \ldots, |Q_{f-b}| = 0$ by the merging step, $|Q_{f-b+1}|, \ldots, |Q_{f-1}| = O(1/\varepsilon^{(d-1)/2})$ by the compression step, and $|Q_f| = S(d, k + 1)$. Therefore, $S(d, k)$ is upper-bounded by the following recurrence:

$$S(d, k) = \begin{cases} S(d, k + 1) + S(d - 1, 0) + \log(1/\varepsilon)O(1/\varepsilon^{(d-1)/2}) & 0 \leqslant k \leqslant d, \\ O(1/\varepsilon^{(d-1)/2}) & k = d + 1, \\ O(1) & d = 1, \end{cases}$$

which solves to $S(d, k) = O((1/\varepsilon^{(d-1)/2}) \log(1/\varepsilon))$, for every $0 \leqslant k \leqslant d$.

Update Time. The compression step in line 9 can be done using Theorem 1 in $O(|Q_i| + 1/\varepsilon^{d-(3/2)})$ or $O((|Q_i| + 1/\varepsilon^{d-2}) \log(1/\varepsilon))$ time. Since $|Q_i| = \Theta(1/\varepsilon^{(d-1)/2}))$, we can charge an amortized time of $O(1 + (1/\varepsilon^{(d-3)/2}) \log(1/\varepsilon))$ to each point of Q_i. In lines 14–16, each point is inserted into P' at most once. Therefore, the cost of insertion into the $(d - 1)$-dimensional stream P' can be charged to each point upon its insertion to a P_i, which at most doubles its total insertion cost. The main cost incurred by each point is therefore the time needed to insert the point into a fat subset, which is $O(1 + (1/\varepsilon^{(d-3)/2}) \log(1/\varepsilon))$ by Corollary 1. □

6 Reducing Update Time

While the main focus in designing streaming algorithms is to optimize the working storage, the time needed to process each element is also of particular interest, especially in applications where a huge amount of date arrives in a short period of time. For the problem of maintaining ε-kernels in $\mathbb{R}^d$, Chan [9] proposed a streaming algorithm that processes each input point in $O(1)$ time using a data structure of size $O([(1/\varepsilon) \log(1/\varepsilon)]^{d-1})$. Here, we show how to improve the space complexity of Chan's algorithm for all fixed dimensions, while the optimal update time, $O(1)$, is preserved.

We provide a general framework to trade-off between time and space complexity in our algorithm as follows: Let $\lambda(\varepsilon) = \Omega(1/\varepsilon^{(d-1)/2})$ be a function of ε. We replace $1/\varepsilon^{(d-1)/2}$ by $\lambda(\varepsilon)$ in line 2 of the INSERT-BOX function and in line 8 of the main INSERT function. It is easy to verify that the amortized update time of the new algorithm is $O([1/\varepsilon^{d-(3/2)}]/\lambda(\varepsilon))$ for $d \geqslant 2$, and $O(\log(1/\varepsilon) + [(1/\varepsilon^{d-2}) \log(1/\varepsilon)]/\lambda(\varepsilon))$ for $d \geqslant 3$. Furthermore, the space complexity of the algorithm is upper-bounded by the recurrence $S(d, k) = S(d, k + 1) + S(d - 1, 0) + \log(1/\varepsilon)O(\lambda(\varepsilon))$, with the base cases

$S(d, d+1) = O(\lambda(\varepsilon))$ and $S(1, k) = O(1)$. The recurrence solves to $S(d, k) = O(\lambda(\varepsilon)\log(1/\varepsilon))$. Setting $\lambda(\varepsilon) = 1/\varepsilon^{d-(3/2)}$, we immediately get the following result:

Theorem 4 *Given a stream of points P in $\mathbb{R}^d$, an ε-kernel of P can be maintained using $O((1/\varepsilon^{d-(3/2)})\log(1/\varepsilon))$ space and $O(1)$ amortized update time.*

Note that the above theorem improves the best previous time-optimal streaming algorithm of Chan without using the common dimension-reduction approach used in [9] and [6]. Moreover, by setting $\lambda(\varepsilon) = 1/\varepsilon^{d-2}$, we obtain a streaming algorithm with space complexity $O((1/\varepsilon^{d-2})\log(1/\varepsilon))$ and amortized update time $O(\log(1/\varepsilon))$, which for all $d \geqslant 3$, improves over the algorithm of Agarwal and Yu [6] in terms of space without increasing their update time.

7 Applications

In this section, we briefly review some of the implications of our result. Consider a measure μ so that for any point set $P \subset \mathbb{R}^d$, an ε-kernel of P is an $O(\varepsilon)$-coreset for P with respect to μ. Examples of such a measure include diameter, width, radius of the smallest enclosing cylinder, and volume of the minimum bounding box. Theorem 3 provides space-efficient streaming algorithms to maintain ε-approximations to all these measures using near $O(1/\varepsilon^{(d-1)/2})$ space. Note that $O(1/\varepsilon^{(d-1)/2})$-space streaming algorithms were previously known only for diameter [5], while the best space bound for other measures was $O(1/\varepsilon^{d-(3/2)})$ [6]. Using the general technique described in [2], our result implies improved streaming algorithms for various other shape-fitting problems like minimum-width spherical shell/annulus and minimum-width cylindrical shell. Improved results for kinetic versions of the above problems (where input is a stream of moving points) are implied as well. In these kinetic versions, the trajectory of each moving point is assumed to be a fixed algebraic function determined upon arriving the point, and the objective is to maintain an approximation at any time for the points received up to that time.

Our streaming algorithm can be also used in noisy environments, using the "robust kernel" paradigm proposed in [12, 4]. Roughly speaking, a subset $Q \subseteq P$ is called a (k, ε)-kernel of P, if Q ε-approximates the directional width of P, for any direction, when k outliers can be ignored in that direction. According to [4], one can simultaneously run $2k + 1$ instances of our streaming algorithm to obtain the following result:

Corollary 2 *Given a stream P of points in $\mathbb{R}^d$ and a parameter $k \geqslant 0$, a (k, ε)-kernel of P can be maintained using $O((k/\varepsilon^{(d-1)/2})\log(1/\varepsilon))$ space.*

The kernel size obtained by Corollary 2 substantially improves over the previous known upper bound $O(k/\varepsilon^{d-(3/2)})$ [6], and is very close to the lower bound which is proved to be $O(k/\varepsilon^{(d-1)/2})$ in the worst case [12].

8 Conclusions

In this paper, we have presented a streaming algorithm for maintaining an ε-kernel of a stream of points in $\mathbb{R}^d$ using $O((1/\varepsilon^{(d-1)/2})\log(1/\varepsilon))$ space. The space complexity of our algorithm is optimal

up to a $\log(1/\varepsilon)$ factor. In the special case of two dimensions, Agarwal and Yu [6] proposed a rather involved technique to remove this extra log factor at the expense of increasing update time from $O(1)$ to $O(\log(1/\varepsilon))$. It remains open whether this small log factor can be removed in any fixed dimension.

Acknowledgements

The author would like to thank Timothy M. Chan for his valuable comments and helpful discussions.

References

[1] P. K. Agarwal and S. Har-Peled. Maintaining approximate extent measures of moving points. In *Proc. 12th ACM-SIAM Sympos. Discrete Algorithms*, pages 148–157, 2001.

[2] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *J. ACM*, 51(4):606–635, 2004.

[3] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Geometric approximation via coresets. *Combinatorial and Computational Geometry* (J. E. Goodman, J. Pach, and E. Welzl, eds.), Math. Sci. Research Inst. Pub., Cambridge, 2005.

[4] P. K. Agarwal, S. Har-Peled, and H. Yu. Robust shape fitting via peeling and grating coresets. *Discrete Comput. Geom.*, 39(1-3):38–58, 2008.

[5] P. K. Agarwal, J. Matoušek, and S. Suri. Farthest neighbors, maximum spanning trees and related problems in higher dimensions. *Comput. Geom. Theory Appl.*, 1(4):189–201, 1992.

[6] P. K. Agarwal and H. Yu. A space-optimal data-stream algorithm for coresets in the plane. In *Proc. 23rd Annu. ACM Sympos. Comput. Geom.*, pages 1–10, 2007.

[7] G. Barequet and S. Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *J. Algorithms*, 38(1):91–109, 2001.

[8] J. L. Bentley and J. B. 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(1–2):20–35, 2006.

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

[11] S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *Proc. 36th Annu. ACM Sympos. Theory Comput.*, pages 291–300, 2004.

[12] S. Har-Peled and Y. Wang. Shape fitting with outliers. *SIAM J. Comput.*, 33(2):269–285, 2004.

[13] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, NY, 1985.

[14] H. Yu, P. K. Agarwal, R. Poreddy, and K. R. Varadarajan. Practical methods for shape fitting and kinetic data structures using core sets. In *Proc. 20th Annu. ACM Sympos. Comput. Geom.*, pages 263–272, 2004.