Journal of Machine Learning Research 1 (2000) 1-48 Submitted 4/00; Published 10/00

Coresets for k-Segmentation of Streaming Data

Guy Rosman ROSMAN@CSAIL.MIT.EDU
CSAIL
MIT
32 Vassar St. 02139, MA, USA

Mikhail Volkov VOLKOV@CSAIL.MIT.EDU
CSAIL
MIT
32 Vassar St. 02139, MA, USA

Dan Feldman FELDMAN@CSAIL.MIT.EDU
CSAIL
MIT
32 Vassar St. 02139, MA, USA

John W. Fisher III JWFISHER@CSAIL.MIT.EDU
CSAIL
MIT
32 Vassar St. 02139, MA, USA

Daniela Rus RUS@CSAIL.MIT.EDU
CSAIL
MIT
32 Vassar St. 02139, MA, USA

Editor: Leslie Pack Kaelbling

Abstract

Life-logging video streams, financial time series, and Twitter tweets are a few examples of high-dimensional signals over practically unbounded time. We consider the problem of computing optimal segmentation of such signals by a k-piecewise linear function, using only one pass over the data by maintaining a *coreset* for the signal. The coreset enables fast further analysis such as automatic summarization and analysis of such signals.

A coreset (core-set) is a compact representation of the data seen so far, which approximates the data well for a specific task – in our case, segmentation of the stream. We show that, perhaps surprisingly, the segmentation problem admits coresets of cardinality only linear in the number of segments k and independent of both the dimension d of the signal, and its number n of points. More precisely, we construct a representation of size $O(k/\varepsilon^2)$ that provides a $(1+\varepsilon)$-approximation for the sum of squared distances to any given k-piecewise linear function. Moreover, such coresets can be constructed in a parallel streaming approach. Our results rely on a novel reduction of statistical estimations to problems in computational geometry. We empirically evaluate our algorithms on very large synthetic and real data sets from GPS, video and financial domains, using 255 machines in Amazon cloud.

Keywords:

1. Introduction

There is an increasing demand for systems that learn long-term, high-dimensional data streams. Examples include video streams from wearable cameras, mobile sensors, GPS, financial data and biological signals. In each, a time instance is represented as a high-dimensional feature, for example location vectors, stock prices, or image content feature histograms. In many of these, the stream can be partitioned into time intervals that are governed by different temporal models. Detecting these segments and the models describing each of them is known as temporal segmentation.

We develop real-time algorithms for summarization and segmentation of large streams, by compressing the signals into a compact meaningful representation. This representation can then be used to enable fast analyses such as summarization, state estimation, prediction. The proposed algorithms support data streams that are too large to store in memory, afford easy parallelization, and generic in that they apply to different data types and analyses. For example, the summarization of wearable video data can be used to efficiently detect different scenes and important events, while collecting GPS data for citywide drivers can be used to learn weekly transportation patterns and characterize driver behavior.

In this paper we use a data reduction technique called *coresets* Agarwal et al. (2005); Feldman and Langberg (2010) to enable rapid content-based segmentation of data streams. Informally, a coreset D is *problem dependent* compression of the original data P, such that running algorithm A on the coreset D yields a result $A(D)$ that *provably* approximates the result $A(P)$ of running the algorithm on the original data. If the coreset D is small and its construction is fast, then computing $A(D)$ is fast even if computing the result $A(P)$ on the original data is intractable. See definition 2 for the specific coreset which we develop in this paper.

1.1 Main Contribution

The main contributions of the paper are: (i) A new coreset for the k-segmentation problem (as given in Subsection 1.2) that can be computed at one pass over streaming data (with $O(\log n)$ insertion time/space) and supports distributed computation. Unlike previous results, the insertion time per new observation and required memory is only linear in both the dimension of the data, and the number k of segments. This result is summarized in Theorem 8. Our algorithm is scalable, parallelizable, and provides a provable approximation of the cost function. (ii) Using this novel coreset we demonstrate a new system for segmentation and compression of streaming data. Our approach allows realtime summarization of large-scale video streams in a way that preserves the semantic content of the aggregated video sequences, and is easily extendable. (iii) Experiments to demonstrate our approach on various data types: video, GPS, and financial data. We evaluate performance with respect to output size, running time and quality and compare our coresets to uniform and random sample compression. We demonstrate the scalability of our algorithm by running our system on an Amazon cluster with 255 machines with near-perfect parallelism as demonstrated on $256,000$ frames. We also demonstrate the effectiveness of our algorithm by running several analysis algorithms on the computed coreset instead of the full data. Our implementation summarizes the video in less than 20 minutes, and allows real-time segmentation of video streams at 30 frames per second on a single machine. A preliminary

version of this paper has been presented in the NIPS. We expand upon it by detailing the full proofs for our method, and demonstrating it on additional types of feature vectors.

Streaming and Parallel computations. Maybe the most important property of coresets is that even an efficient off-line construction implies a fast construction that can be computed (a) Embarrassingly in parallel (e.g. cloud and GPUs), (b) in the streaming model where the algorithm passes only once over the (possibly unbounded) streaming data. Only small amount of memory and update time ($\sim \log n$) per new point insertion is allowed, where n is the number of observations so far.

1.2 Problem Statement

The k-segment mean problem optimally fits a given discrete time signal of n points by a set of k linear segments over time, where $k \geq 1$ is a given integer. That is, we wish to partition the signal into k consecutive time intervals such that the points in each time interval are lying on a single line; see Fig. 1(left) and the following formal definition.

We make the following assumptions with respect to the data: (a) We assume the data is represented by a feature space that suitably represents its underlying structure; (b) The content of the data includes at most k segments that we wish to detect automatically; An example for this are scenes in a video, phases in the market as seen by stock behaviour, etc. and (c) The dimensionality of the feature space is often quite large (from tens to thousands of features), with the specific choice of the features being application dependent – several examples are given in Section 9. This motivates the following problem definition.

Definition 1 (k-segment mean) *A set P in $\mathbb{R}^{d+1}$ is a* signal *if $P = \{(1, p_1), (2, p_2), \cdots, (n, p_n)\}$ where $p_i \in \mathbb{R}^d$ is the point at time index i for every $i = [n] = \{1, \cdots, n\}$. For an integer $k \geq 1$, a* k-segment *is a k-piecewise linear function $f : \mathbb{R} \to \mathbb{R}^d$ that maps every time $i \in \mathbb{R}$ to a point $f(i)$ in $\mathbb{R}^d$. The* fitting error *at time t is the squared distance between p_i and its corresponding projected point $f(i)$ on the k-segments. The* fitting cost *of f to P is the sum of these squared distances,*

$$\text{cost}(P, f) = \sum_{i=1}^{n} \|p_i - f(i)\|_2^2, \tag{1}$$

where $\|\cdot\|$ denotes the Euclidean distance. The function f is a k-segment mean *of P if it minimizes* $\text{cost}(P, f)$.

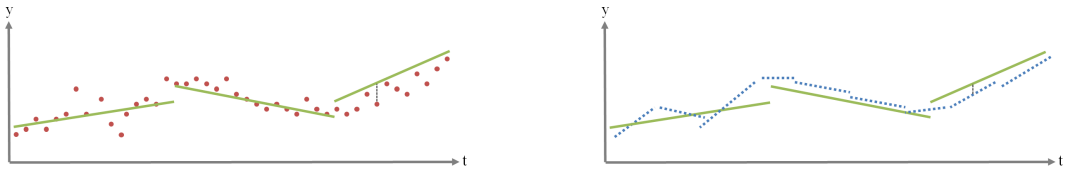


Figure 1: For every k-segment f, the cost of input points (red) is approximated by the cost of the coreset (dashed blue lines). Left: An input signal and a 3-segment f (green), along with the regression distance to one point (dashed black vertical lines). The cost of f is the sum of these squared distances from all the input points. Right: The coreset consists of the projection of the input onto few segments, with approximate per-segment representation of the data. **GR:**redo

For the case $k = 1$ the 1-segment mean is the solution to the linear regression problem. If we restrict each of the k-segments to be a horizontal segment, then each segment will be the mean height of the corresponding input points. The resulting problem is similar to the k-mean problem, except each of the voronoi cells is forced to be a single region in time, instead of nearest center assignment, i.e. the regions are contiguous.

In this paper we are interested in seeking a compact representation D that approximates $\text{cost}(P, f)$ for every k-segment f using the above definition of $\text{cost}'(D, f)$. We denote a set D as a (k, ε)-coreset according to the following definition,

Definition 2 ((k, ε)-coreset) *Let $P \subseteq \mathbb{R}^{d+1}$, $k \geq 1$ be an integer, for some small $\varepsilon > 0$. A set D, with a cost function $\text{cost}'(\cdot)$ is a* (k, ε)-coreset *for P if for every k-segment f we have*

$$(1 - \varepsilon)\text{cost}(P, f) \leq \text{cost}'(D, f) \leq (1 + \varepsilon)\text{cost}(P, f).$$

We present a new coreset construction with provable approximations for a family of natural k-segmentation optimization problems. This is the first such construction whose running time is linear in both the number of data points n, their dimensionality d, and the number k of desired segments. The resulting coreset consists of $O(dk/\varepsilon^2)$ points that approximates the sum of square distances for *any* k-piecewise linear function (k segments over time). In particular, we can use this coreset to compute the k-piecewise linear function that minimize the sum of squared distances to the input points, given arbitrary constraints or weights (priors) on the desired segmentation. Such a generalization is useful, for example, when we are already given a set of candidate segments (e.g. maps or distribution of images) and wish to choose the right k segments that approximate the input signal.

Previous results on coresets for k-segmentation achieved running time or coreset size that are at least quadratic in d and cubic in k Feldman et al. (2012b,a). As such, they can be used with very large data, for example to long streaming video data which is usually high-dimensional and contains large number of scenes. This prior work is based on some non-uniform sampling of the input data. In order to achieve our results, we had to replace the sampling approach by a new set of deterministic algorithms that carefully select the coreset segments and their internal representation.

1.3 Related Work

Our work builds on several important contributions in coresets, k-segmentations, and video summarization.

Approximation Algorithms. One of the main challenges in providing provable guarantees for segmentation w.r.t segmentation size and quality is global optimization. Current provable algorithms for data segmentation are cubic-time in the number of desired segments, quadratic in the dimension of the signal, and cannot handle both parallel and streaming computation as desired for big data. The closest work that provides provable approximations is that of Feldman et al. (2012b).

Several works attempt to summarize high-dimensional data streams in various application domains. For example, Paul et al. (2014) describe the video stream as a high-dimensional stream and run approximated clustering algorithms such as k-center on the points of the stream; see Girdhar and Dudek (2012) for surveys on stream summarization

in robotics. The resulting k-centers of the clusters comprise the video summarization. The main disadvantages of these techniques are (i) They partition the data stream into k clusters that do not provide k-segmentation over time. (ii) Computing the k-center takes time exponential in both d and k Hochbaum (1996). In Paul et al. (2014) heuristics were used for dimension reduction, and in Girdhar and Dudek (2012) a 2-approximation was suggested for the off-line case, which was replaced by a heuristic forstreaming. (iii) In the context of analysis of video streams, they use a feature space that is often simplistic and does not utilize the large data available efficiently. In our work the feature space can be updated on-line using a coreset for k-means clustering of the features seen so far.

k-segment Mean. The k-segment mean problem can be solved exactly using dynamic programming Bellman (1961). However, this takes $O(dn^2k)$ time and $O(dn^2)$ memory, which is impractical for streaming data. In (Guha et al., 2006, Theorem 8) a $(1+\varepsilon)$-approximation was suggested using $O(n(dk)^4 \log n/\varepsilon)$ time. While the algorithm in Guha et al. (2006) support efficient streaming, it is not parallel. Since it returns a k-segmentation and not a coreset, it cannot be used to solve other optimization problems with additional priors or constraints. In Feldman et al. (2012b) an improved algorithm that takes $O(nd^2k + ndk^3)$ time was suggested. The algorithm is based on a coreset of size $O(dk^3/\varepsilon^3)$. Unlike the coreset in this paper, the running time of Feldman et al. (2012b) is cubic in both d and k.

The result in Feldman et al. (2012b) is the last in a line of research for the k-segment mean problem and its variations; see survey in Feldman et al. (2012a); Guha et al. (2006); Gilbert et al. (2002). The application was segmentation of 3-dimensional GPS signal (time, latitude, longitude). The coreset construction in Feldman et al. (2012b) and previous papers takes time and memory that is quadratic in the dimension d and cubic in the number of segments k. Conversely, our coreset construction takes time only linear in both k and d.

While the recent results suggest running time that is linear in n, and space that is near-logarithmic in n, the computation time is still cubic in k, the number of segments, and quadratic in d, the dimension. Since the number k represents the number of scenes, and d is the total number of possible features, such a running time is prohibitive.

Video Summarization One motivating application for us is online video summarization, where input video stream can be represented by a set of points over time in an appropriate feature space. Every point in the feature space represents the frame, and we aim to produce a compact approximation of the video in terms of this space and its Euclidean norm. Application-aware summarization and analysis of ad-hoc video streams is a difficult task with many attempts aimed at tackling it from various perspectives Churchill and Newman (2012); Lu and Grauman (2013); Bandla and Grauman (2013). The problem is highly related to video action classification, scene classification, and object segmentation Lu and Grauman (2013). Applications where life-long video stream analysis is crucial include mapping and navigation medical / assistive interaction, and augmented-reality applications, among others. Our goal differs from video compression in that compression is geared towards preserving image quality for all frames, and therefore stores semantically redundant content. Instead, we seek a summarization approach that allows us to represent the video content by a set of key segments, for a given feature space.

This paper is organized as follows. We begin by describing the k-segmentation problem and the proposed coresets, and describe their construction, and their properties in Section 2. We perform several experiments in order to validate the proposed approach on data collected

from GPS and werable web-cameras, and demonstrate the aggregation and analysis of multiple long sequences of wearable user video in Section 9. Section 10 concludes the paper and discusses future directions.

2. A Novel Coreset for k-segment Mean

We now describe the construction of the coreset. We detail the main theorem and their proofs, after giving the intuition behind the construction. The key insights for constructing the k-segment coreset are: i) We observe that for the case $k = 1$, a 1-segment coreset can be easily obtained using SVD. ii) For the general case, $k \geq 2$ we can partition the signal into a suitable number of intervals, and compute a 1-segment coreset for each such interval. If the number of intervals and their lengths are carefully chosen, most of them will be well approximated by every k-segmentation, and the remaining intervals will not incur a large error contribution.

Based on these observations, we propose the following construction. 1) Estimate the signal's complexity, i.e., the approximated fitting cost to its k-segment mean. We denote this step as a call to the algorithm Bicriteria. 2) Given an complexity measure for the data, approximate the data by a set of segments with auxiliary information, which is the proposed coreset, denoted as the output of algorithm BalancedPartition.

We then prove that the resulting coreset allows us to approximate with guarantees the fitting cost for any k-segmentation over the data, as well as compute an optimal k-segmentation. We state the main result in Theorem 8, and describe the proposed algorithms as Algorithms 3 and 1.

2.1 Computing a k-Segment Coreset - Overview

We would like to compute a (k, ε)-coreset for our data. A (k, ε)-coreset D for a set P approximates the fitting cost of any query k-segment to P up to a small multiplicative error of $1 \pm \varepsilon$. We note that a $(1, 0)$-coreset can be computed using SVD; See Appendix A for details and proof. However, for $k > 2$, we cannot approximate the data by a representative point set (as shown in Appendix A). Instead, we define a data structure D as our proposed coreset, and define a new cost function $\text{cost}'(D, f)$ that approximates the cost of P to any k-segment f.

The set D consists of tuples of the type (C, g, b, e). Each tuple corresponds to a different time interval $[b, e]$ in $\mathbb{R}$ and represents the set $P(b, e)$ of points of P in this interval. The set C is a $(1, \varepsilon)$-coreset for $P(b, e)$.

We note the following: 1) If all the points of the k-segment f are on the same segment in this time interval, i.e, $\{f(t) \mid b \leq t \leq e\}$ is a linear segment, then the cost from $P(b, e)$ to f can be approximated well by C, up to $(1 + \varepsilon)$ multiplicative error. 2) If we project the points of $P(b, e)$ on their 1-segment mean g, then the projected set L of points will approximate well the cost of $P(b, e)$ to f, even if f corresponds to more than one segment in the time interval $[b, e]$. Unlike the previous case, the error here is additive. 3) Since f is a k-segment there will be at most $k - 1$ time intervals that will intersect more than two segments of f, so the overall additive error is small . This motivates the following definition of D and cost'.

Algorithm 1: BalancedPartition(P, ε, σ)

Input: A set $P = \{(1, p_1), \cdots, (n, p_n)\}$ in $\mathbb{R}^{d+1}$
an error parameters $\varepsilon \in (0, 1/10)$ and $\sigma > 0$.
Output: A set D that satisfies Theorem 8.

$Q := \emptyset$; $D = \emptyset$; p_{n+1}:= an arbitrary point in $\mathbb{R}^d$;
for $i := 1$ *to* $n + 1$ **do**
  $Q := Q \cup \{(i, p_i)\}$; Add new point to tuple
  $f^* :=$ a linear approximation of Q; $\lambda := \text{cost}(Q, f^*)$
  if $\lambda > \sigma$ **or** $i = n + 1$ **then**
    $T := Q \setminus \{(i, p_i)\}$; take all the new points into tuple
    $C :=$ a $(1, \varepsilon/4)$-coreset for T; Approximate points by a local representation
    $g :=$ a linear approximation of T, $b := i - |T|$, $e := i - 1$; save endpoints
    $D := D \cup \{(C, g, b, e)\}$; save a tuple
    $Q := \{(i, p_i)\}$; proceed to new point
return D

Definition 3 ($\text{cost}'(D, f)$) *Let $D = \{(C_i, g_i, b_i, e_i)\}_{i=1}^m$ where for every $i \in [m]$ we have $C_i \subseteq \mathbb{R}^{d+1}$, $g_i : \mathbb{R} \to \mathbb{R}^d$ and $b_i \leq e_i \in \mathbb{R}$. For a k-segment $f : \mathbb{R} \to \mathbb{R}^d$ and $i \in [m]$ we say that C_i is* served by one segment *of f if $\{f(t) \mid b_i \leq t \leq e_i\}$ is a linear segment. We denote by* $\text{Good}(D, f) \subseteq [m]$ *the union of indexes i such that C_i is served by one segment of f. We also define $L_i = \{g_i(t) \mid b_i \leq t \leq e_i\}$, the projection of C_i on g_i. We define* $\text{cost}'(D, f)$ *as* $\sum_{i \in \text{Good}(D,f)} \text{cost}(C_i, f) + \sum_{i \in [m] \setminus \text{Good}(D,f)} \text{cost}(L_i, f)$.

Our coreset construction for general $k > 1$ is based on an input parameter $\sigma > 0$ such that for an appropriate σ the output is a (k, ε)-coreset. For the purpose of constructing our coreset we will require the definition of an (α, β)-approximation,

Definition 4 (**(α, β)-approximation**) *For $\alpha, \beta > 0$, an (α, β)-approximation for the k-segment mean of P is a $(k \cdot \beta)$-segment g such that* $\text{cost}(P, g) \leq \alpha \cdot \text{cost}(P, f^*)$.

Specifically, we show that Algorithm 3 constructs an (α, β)-approximation, for α, β small enough so as to estimate the complexity of the data.

We show that using the minimal value $\text{cost}(P, g)$ of such an approximation, even without knowing g, suffices to get a (k, ε)-coreset, using the BalancedPartition(P, ε, σ) algorithm, given as Algorithm 1.

3. k-Segment mean

Let $P = \{(t_1, p_1), \cdots, (t_n, p_n)\}$ be a subset of $\mathbb{R}^{d+1}$ where $t_i \in \mathbb{R}$ and $p_i \in \mathbb{R}^d$ for every $i \in [n] = \{1, \cdots, n\}$. The *fitting cost* (henceforth simply "cost") from P to a k-segment f is the sum of squared distances

$$\text{cost}(P, f) = \sum_{(t,p) \in P} \| (p - f(t)) \|^2, \tag{2}$$

where here, as in the paper, $\|X\|^2 = \sum_{ij}(X_{ij})^2$ is the sum of squared entries of a matrix or a vector X (known as the Frobenius norm for a matrix or the ℓ_2 Euclidean norm for a vector).

A *k-segment mean* of P is a k-segment $f^* : \mathbb{R} \to \mathbb{R}^d$ that minimizes $\text{cost}(P, f)$ over every k-segment $f : \mathbb{R} \to \mathbb{R}^d$. For $\alpha \geq 1$, an *α-approximation for the k-segment mean of P* is a k-segment f such that $\text{cost}(P, f) \leq \alpha \cdot \text{cost}(P, f^*)$. For $\alpha, \beta > 0$, an (α, β)-approximation for the k-segment mean of P is a $(k \cdot \beta)$-segment g such that $\text{cost}(P, g) \leq \alpha \cdot \text{cost}(P, f^*)$.

One of our main tool for computing approximations to the k-segment mean is the *singular value decomposition* (SVD) which is defined as follows. For integers $n, d \geq 1$ we denote by $\mathbb{R}^{n \times d}$ the set $n \times d$ matrices whose entries are in $\mathbb{R}$. A *unitary* matrix is a matrix whose columns are orthonormal vectors. The thin SVD of a matrix $X \in \mathbb{R}^{n \times d}$ is $X = U\Sigma V^T$ where both $U \in \mathbb{R}^{n \times d}$ and $V \in \mathbb{R}^{d \times d}$ are unitary matrices, and $\Sigma \in \mathbb{R}^{d \times d}$ is a diagonal matrix of non-negative and non-increasing diagonal entries. As we show in Appendix A, we can use the SVD to get a $(1, 0)$-coreset for 1-segments.

GR: remind why we need balanced partition.

4. Balanced Partition

A (k, ε)-coreset D for a set P approximates the fitting cost of a any query k-segment to P up to a small multiplicative error of $1 \pm \varepsilon$. However, as proved in Appendix A, such a coreset cannot be just a weighted subset of $\mathbb{R}^{d+1}$. Instead, we define a more involved data structure D that represents the coreset, and define a new cost function $\text{cost}'(D, f)$ that approximates the cost of P to a k-segment f. We also assume that the time (first coordinate) is discrete between 1 to n. This means that the projecting of P on any line can be described exactly in $O(d)$ space using only the first and last projected point, which motives the following structure of D.

The set D consists of tuples of the type (C, g, b, e). Each tuple corresponds to a different time interval $[b, e]$ in $\mathbb{R}$ and represents the set $P(b, e)$ of points of P in this interval. The set C is a $(1, \varepsilon)$-coreset for $P(b, e)$. Our first observation is that if all the points of the k-segment f are on the same segment in this time interval, i.e, $\{f(t) \mid b \leq t \leq e\}$ is a linear segment, then the cost from $P(b, e)$ to f can be approximated well by C, up to $(1 + \varepsilon)$ multiplicative error. We refer to these tuples as *coreset segments* in the description of the algorithm.

The second observation is that if we project the points of $P(b, e)$ on their 1-segment mean g, then the projected set L of points will approximate well the cost of $P(b, e)$ to f, even if f corresponds to more than one segment in the time interval $[b, e]$. Unlike the previous case, the error here is additive. However, the third observation is that, since f is a k-segment there will be at most $k - 1$ time intervals $[b, e]$ that will intersects more than two segments of f.

We will compute such a small structure D that approximates $\text{cost}(P, f)$ for every k-segment f using the above definition of $\text{cost}'(D, f)$. Such a set D will be called a (k, ε)-coreset as follows.

Definition 5 ((k,ε)-coreset) *Let $P \subseteq \mathbb{R}^{d+1}$, $k \geq 1$ be an integer, and let $\varepsilon \in (0, 1/10)$. The set D is a (k,ε)-*coreset *for P if for every k-segment f we have*

$$(1-\varepsilon)\text{cost}(P,f) \leq \text{cost}'(D,f) \leq (1+\varepsilon)\text{cost}(P,f).$$

Our coreset construction is based on an input parameter $\sigma > 0$ such that for an appropriate σ the output is a (k,ε)-coreset. Recall that for $\alpha, \beta > 0$, an (α,β)-approximation for the k-segment mean of P is a $(k\cdot\beta)$-segment g such that $\text{cost}(P,g) \leq \alpha\cdot\text{cost}(P,f^*)$. We show that using the value $\text{cost}(P,g)$ of such an approximation, even without knowing g, suffices to get a (k,ε)-coreset. In the next section we will compute such an (α,β)-approximation for small α and β.

The size of the resulting coreset depends on α and β. In particular, for $\alpha = \beta = 1$ the following lemma implies that there exists a (k,ε)-coreset of size $O(k/\varepsilon^2)$ for every input set P.

Algorithm 2: BalancedPartition(P,ε,σ).

Input: A set $P = \{(1,p_1),\cdots,(n,p_n)\}$ in $\mathbb{R}^{d+1}$
an error parameters $\varepsilon \in (0,1/10)$ and $\sigma > 0$.
Output: A set D that satisfies Lemma 6.

```
Q := ∅; D = ∅ ;
p_{n+1} := an arbitrary point in R^d ;
for i := 1 to n + 1 do
    Q := Q ∪ {(i, p_i)};
    f* := a 2-approximation to the 1-segment mean of Q. /* See Corollary 19 */
    λ := cost(Q, f*) ;
    if λ > σ or i = n + 1 then
        T := Q \ {(i, p_i)} /* Define the new coreset segment data up to i */
        C := a (1, ε/4)-coreset for T /* See Claim 17 */
        g := a 2-approximation to the 1-segment mean of T /* See Corollary 19
             */
        b := i − |T|;
        e := i − 1;
        D := D ∪ {(C, g, b, e)}/* Add a new coreset segment */
        Q := {(i, p_i)}/* Start aggregating a new coreset segment */
return D
```

Lemma 6 *Let $P = \{(1,p_1),\cdots,(n,p_n)\}$ such that $p_i \in \mathbb{R}^d$ for every $i \in [n]$. Suppose that $h : \mathbb{R} \to \mathbb{R}^d$ is an (α,β)-approximation for the k-segment mean of P, and let*

$$\sigma = \frac{\varepsilon^2 \text{cost}(P,h)}{100k\alpha}.$$

Let D be the output of a call to BalancedPartition(P,ε,σ)*; See Algorithm 2.*

Then D is a (k,ε)-coreset for P of size

$$|D| = O(k)\cdot\left(\frac{\alpha}{\varepsilon^2}+\beta\right),$$

and can be computed in $O(dn/\varepsilon^4)$ time.

Proof Let $m = |D|$ and f be a k-segment. We denote the ith coreset segment in D by (C_i, g_i, b_i, e_i) for every $i \in [m]$. For every $i \in [m]$ we have that C_i is a $(1, \varepsilon/4)$-coreset for a corresponding subset $T = T_i$ of P. By the construction of D we also have $P = T_1 \cup \cdots \cup T_m$.

Using Definition 3 of $\text{cost}'(D, f)$, $\text{Good}(D, f)$ and L_i, we thus have

$$\begin{aligned}
&|\text{cost}(P, f) - \text{cost}'(D, f)| \\
&= |\sum_{i=1}^{m} \text{cost}(T_i, f) - \left(\sum_{i \in \text{Good}(D,f)} \text{cost}(C_i, f) + \sum_{i \in [m] \setminus \text{Good}(D,f)} \text{cost}(L_i, f) \right)| \\
&= |\sum_{i \in \text{Good}(D,f)} (\text{cost}(T_i, f) - \text{cost}(C_i, f)) + \sum_{i \in [m] \setminus \text{Good}(D,f)} (\text{cost}(T_i, f) - \text{cost}(L_i, f)) \\
&\leq \sum_{i \in \text{Good}(D,f)} |\text{cost}(T_i, f) - \text{cost}(C_i, f)| + \sum_{i \in [m] \setminus \text{Good}(D,f)} |\text{cost}(T_i, f) - \text{cost}(L_i, f)|,
\end{aligned} \tag{3}$$

where the last inequality is due to the triangle inequality. We now bound each term in the right hand side.

For every $i \in \text{Good}(D, f)$ we have that C_i is a $(1, \varepsilon/4)$-coreset for T_i, so

$$|\text{cost}(T_i, f) - \text{cost}(C_i, f)| \leq \frac{\varepsilon \text{cost}(T_i, f)}{4}. \tag{4}$$

For every $i \in [m] \setminus \text{Good}(D, f)$, we have

$$\begin{aligned}
|\text{cost}(T_i, f) - \text{cost}(L_i, f)| &= \left| \sum_{(p,t) \in T_i} \|p - f(t)\|^2 - \sum_{t=b_i}^{e_i} \|g_i(t) - f(t)\|^2 \right| \\
&= \left| \sum_{(p,t) \in T_i} \left(\|p - f(t)\|^2 - \|g_i(t) - f(t)\|^2 \right) \right| && (5) \\
&\leq \sum_{(p,t) \in T_i} \left| \|p - f(t)\|^2 - \|g_i(t) - f(t)\|^2 \right| && (6) \\
&\leq \sum_{(p,t) \in T_i} \left(\frac{12\|g_i(t) - p\|^2}{\varepsilon} + \frac{\varepsilon \|p - f(t)\|^2}{2} \right) && (7) \\
&= \frac{12\text{cost}(T_i, g_i)}{\varepsilon} + \frac{\varepsilon \text{cost}(T_i, f)}{2} \leq \frac{24\sigma}{\varepsilon} + \frac{\varepsilon \text{cost}(T_i, f)}{2}, && (8)
\end{aligned}$$

where (6) is by the triangle inequality, and (7) is by the weak triangle inequality (see (Feldman et al., 2013, Lemma 7.1)). The inequality in (8) is because by construction $\text{cost}(T, f^*) \leq \sigma$ for some 2-approximation f^* of the 1-segment mean of T. Hence, $\text{cost}(T, g_i) \leq 2\text{cost}(T, f^*) \leq 2\sigma$.

Plugging (8) and (4) in (3) yields

$$\begin{aligned}
|\text{cost}(P, f) - \text{cost}'(D, f)| &\leq \sum_{i \in \text{Good}(D,f)} \frac{\varepsilon \text{cost}(T_i, f)}{4} + \sum_{i \in [m] \setminus \text{Good}(D,f)} \left(\frac{24\sigma}{\varepsilon} + \frac{\varepsilon}{2} \text{cost}(T_i, f) \right) \\
&\leq \left(\frac{\varepsilon}{4} + \frac{\varepsilon}{2} \right) \text{cost}(P, f) + \frac{24k\sigma}{\varepsilon},
\end{aligned}$$

where in the last inequality we used that fact that $|[m] \setminus \text{Good}(D, f)| \leq k - 1 < k$ since f is a k-segment. Substituting σ yields

$$|\text{cost}(P, f) - \text{cost}'(D, f)| \leq \frac{3\varepsilon}{4}\text{cost}(P, f) + \frac{\varepsilon\text{cost}(P, h)}{4\alpha} \leq \frac{3\varepsilon}{4}\text{cost}(P, f) + \frac{\varepsilon\text{cost}(P, f)}{4} = \varepsilon\text{cost}(P, f).$$

Bound on $|D|$: Let $j \in [m-1]$, consider the values of T, Q and λ during the execution of Line 8 when $T = T_j$ is constructed. Let $Q_j = Q$ and $\lambda_j = \lambda$. The cost of the 1-segment mean of Q_j is at least $\lambda_j/2 > \sigma/2 > 0$, which implies that $|Q_j| \geq 3$ and thus $|T_j| \geq 1$. Since Q_{j-1} is the union of T_{j-1} with the first point of T_j we have $Q_{j-1} \subseteq T_{j-1} \cup T_j$. By letting g^* denote a 1-segment mean of $T_{j-1} \cup T_j$ we have

$$\text{cost}(T_{j-1} \cup T_j, g^*) \geq \text{cost}(Q_{j-1}, g^*) \geq \lambda_j/2 > \sigma/2.$$

Suppose that for our choice of $j \in [m-1]$, the points in $T_{j-1} \cup T_j$ are served by a single segment of h, i.e, $\{h(t) \mid b_{j-1} \leq t \leq e_j\}$ is a linear segment. Then

$$\text{cost}(T_{j-1}, h) + \text{cost}(T_j, h) = \text{cost}(T_{j-1} \cup T_j, h) \geq \text{cost}(T_{j-1} \cup T_j, g^*) > \sigma/2. \tag{9}$$

Let $G \subseteq [m-1]$ denote the union over all values $j \in [m-1]$ such that j is both even and satisfies (9). Summing (9) over G yields

$$\text{cost}(P, h) = \sum_{j \in [m]} \text{cost}(T_i, h) \geq \sum_{j \in G} (\text{cost}(T_{j-1}, h) + \text{cost}(T_j, h)) \geq |G|\sigma/2. \tag{10}$$

Since h is a (βk)-segment, at most $(\beta k) - 1$ sets among $T_1, \cdots, T_m$ are not served by a single segment of h, so $|G| \geq (m - \beta k)/2$. Plugging this in (10) yields $\text{cost}(P, h) \geq (m - \beta k)\sigma/4$. Rearranging,

$$m \leq \frac{4\text{cost}(P, h)}{\sigma} + \beta k = O\left(\frac{k\alpha}{\varepsilon^2}\right) + \beta k. \tag{11}$$

Running time: In Theorem 18 it was shown how to compute a $(1, \varepsilon)$-coreset C in time $O(dn/\varepsilon^4)$ for n points using the algorithm in Ghashami and Phillips (2014). This algorithm is dynamic and supports insertion of a new point in $O(d/\varepsilon^4)$ time. Therefore, updating the 1-segment mean f^* and the coreset C can be done in $O(d/\varepsilon^4)$ time per point, and the overall running time is $O(nd/\varepsilon^4)$. ■

5. (α, β)-Approximation and the Bicriteria Algorithm

We now describe the Bicriteria(P, k) algorithm. This algorithm allows us to estimate the minimal k-segmentation cost for the signal. It is described as Algorithm 3.

Theorem 7 *Let $f : \mathbb{R} \to \mathbb{R}^d$ be the output of a call to* Bicriteria(P, k). *Then*

(i) *f is a (βk)-segment for some $\beta = O(\log n)$.*

(ii) $\text{cost}(P, f) \leq \alpha\text{cost}(P, f^*)$, *where $\alpha = \log_2 n$, and f^* is a k-segment mean of P.*

Algorithm 3: BICRITERIA(P, k)

Input: A set $P \subseteq \mathbb{R}^{d+1}$ and an integer $k \geq 1$

Output: An $(O(\log n), O(\log n))$-approximation to the k-segment mean of P.

if $n \leq 2k+1$ **then**
  $f :=$ a 1-segment mean of P;
  return f;
Set $t_1 \leq \cdots \leq t_n$ and $p_1, \cdots, p_n \in \mathbb{R}^d$ such that $P = \{(t_1, p_1), \cdots, (t_n, p_n)\}$
$m \leftarrow \{t \in \mathbb{R} \mid (t, p) \in P\}$
Partition P into $4k$ sets $P_1, \cdots, P_{2k} \subseteq P$ such that for every $i \in [2k-1]$:
  (i) $|\{t \mid (t,p) \in P_i\}| = \left\lfloor \frac{m}{4k} \right\rfloor$, and (ii) if $(t,p) \in P_i$ and $(t',p') \in P_{i+1}$ then $t < t'$.
for $i := 1$ *to* $4k$ **do**
  Compute a 2-approximation g_i to the 1-segment mean of P_i
$Q :=$ the union of $k+1$ signals P_i with the smallest value $\text{cost}(P_i, g_i)$ among $i \in [2k]$.
$h :=$ BICRITERIA$(P \setminus Q, k)$; Repartition the segments that did not have a good approximation
Set

$$f(t) := \begin{cases} g_i(t) & \exists (t,p) \in P_i \text{ such that } P_i \subseteq Q \\ h(t) & \text{otherwise} \end{cases}.$$

return f;

(iii) *f can be computed in $O(dn)$ time.*

Proof ***(i)*** In every recursive iteration of the algorithm we remove $(k-1)$ subsets of P, whose overall size is

$$|Q| \geq (k-2) \cdot \left\lfloor \frac{n}{2k} \right\rfloor \geq (k-2) \cdot \left(\frac{n}{2k} - 1 \right) = \frac{n}{2} - \frac{n}{k} - (k-2) \geq \frac{n}{2} - \frac{n}{3} - \frac{n}{12} = \frac{n}{12},$$

where in the last inequality we used the assumption $k \in [3, n/12]$. Hence, the size of P reduced by a constant fraction in each recursive iteration and we have $O(\log n)$ iterations.

Each subset P_i in Q contributes at most 2 segments to f, so the number of segments in f increases by $O(k)$ in each of the $O(\log n)$ recursive iterations. Hence, the final output f has $O(k \log n)$ segments.

(ii) Consider the value of P during one of the recursive iterations. Since f^* is a k-segment, every set in $P_1, \cdots, P_{2k}$ is served by one segment of f^*, except at most $k-1$ such subsets. Let $M \subseteq [2k]$ denote the indexes of these (at most $k-1$) subsets, and let $W = [2k] \setminus M$ denote the rest, such that $Q = \bigcup_{i \in W} P_i$. Hence,

$$\text{cost}(P, f^*) \geq \sum_{i \in W} \text{cost}(P_i, f^*) \geq \sum_{i \in W} \min_g \text{cost}(P_i, g) \geq \frac{1}{2} \sum_{i \in [2k] \setminus M} \text{cost}(P_i, g_i), \tag{12}$$

where the minimum is over every 1-segment $g : \mathbb{R} \to \mathbb{R}^d$. Since

$$|[2k] \setminus M| = 2k - |M| \geq 2k - (k-1) = k+1,$$

we have

$$\sum_{i\in[2k]\setminus M} \text{cost}(P_i, g_i) \geq \sum_{i\in W} \text{cost}(P_i, g_i) = \sum_{i\in W} \text{cost}(P_i, f) = \text{cost}(Q, f).$$

Plugging the last inequality in (12) yields $\text{cost}(Q, f) \leq 2\text{cost}(P, f^*)$. Summing over all iterations proves the claim.

(iii) In each recursive iteration, the dominated running time is in the "for" loop in Lines 8–9. We compute a 2-approximation g_i for the 1-segment mean of a set P_i of m points in $O(md)$ time using Corollary 19. Hence, the overall time to compute Lines 8–9 is $O(nd)$. Since the size of P reduced by a constant fraction in each recursive iteration, the overall running time is dominated by the first iteration which takes $O(nd)$ time. ■

6. (k, ε)-Coreset

We now define the k-segment coreset, present a coreset construction algorithm, prove bounds on how well the coreset represents data with respect to the fitting cost to a k-segment query, and establish the running time complexity of the algorithm.

Algorithm 4: Coreset(P, k, ε)

Input: A set $P = \{(1, p_1), \cdots, (n, p_n)\}$ in $\mathbb{R}^{d+1}$.
Output: A (k, ε)-coreset (C, w) that satisfies Theorem 8.
1 Compute $h \leftarrow$ Bicriteria(P, k) ; See Algorithm 3
2 Set $\sigma \leftarrow \frac{\varepsilon^2 \text{cost}(P,h)}{100k \log_2 n}$
3 Set $D \leftarrow$ BalancedPartition(P, ε, σ) ; See Algorithm 2
4 **return** D

Theorem 8 *Let $P = \{(1, p_1), \cdots, (n, p_n)\}$ such that $p_i \in \mathbb{R}^d$ for every $i \in [n]$. Let D be the output of a call to* Coreset(P, k, ε)*; see Algorithm 2.*
Then D is a (k, ε)-coreset for P of size

$$|D| = O(k) \cdot \left(\frac{\log n}{\varepsilon^2}\right),$$

and can be computed in $O(dn/\varepsilon^4)$ time.

Proof By Theorem 7, h is an (α, β)-approximation for the k-segment mean of P for $\alpha = \log_2 n$ and $\beta = O(\log n)$. Theorem 8 then follows by substituting α and β in Theorem 6. ■

7. Weak (k, ε) Coreset for Efficient Segmentation

When computing an optimal k-segmentation for our data, we are bounded by the scale of the data in yet another aspect – the number of possible locations for each segment endpoint

is $O(N)$. This means we cannot run algorithms with a linear complexity in the data size, let alone a quadratic one, as the original method of Bellman (1961). While the coreset we propose handles gracefully k-segmentations whose endpoints lie on the coreset segments boundaries, it does not handle the more general case, where we want endpoints that do not coincide with out coreset segment endpoints. In the endpoint limited case, we use the same dynamic programming framework suggested by Bellman in Bellman (1961). Since the number of possible segments endpoints is $O\left(\frac{(k\alpha)}{\varepsilon^2}+\beta k\right)$, the number of steps in the algorithm is

$$O\left(\left(\frac{(k\alpha)}{\varepsilon^2}+\beta k\right)^2 k\right). \tag{13}$$

We now describe an additional new approximation tool in use by our algorithm when computing efficient k-segmentation. During the computation of an optimal segmentation, exhaustive search must be performed when updating the segmentation for the $k+1$-segments induction step. Since an $O(N)$ operation such as this is too costly to compute, we require a way of approximating this search.

For an integer $n \geq 1$ we denote $[n] = \{1, \cdots, n\}$. Let $k, n \geq 1$ be a pair of integers. A function $f : \mathbb{R} \to [0, \infty)$ is *non-decreasing* over $[n]$ if $f(i) \leq f(j)$ for every $i \leq j$ in $[n]$, and *non-increasing* if $f(i) \geq f(j)$ for every $i \leq j$ in $[n]$. A function is *monotonic* if it is either non-increasing or non-decreasing. A function $g : \mathbb{R} \to [0, \infty)$ is *k-piecewise monotonic* if $[n]$ can be partitioned into k consecutive sub-intervals $[n] = [i_1] \cup ([i_2] \setminus [i_1]) \cdots \cup ([n] \setminus [i_{k-1}])$ such that g is monotonic over each one of them.

Algorithm 5: PiecewiseCoreset(n, s, ε)

Input: An integer $n \geq 1$, a function $s : [n] \to (0, \infty)$ and an error parameter $\varepsilon > 0$.
Output: A vector $w = (w_1, \cdots, w_n)$ that satisfies Lemma 9.

Set $t := \sum_{j=1}^{n} s_j$ and $B := \emptyset$;
for $i = 1$ *to* n **do**
  Set $b_i := \left\lceil \frac{\sum_{j=1}^{i} s_i}{\varepsilon t} \right\rceil$ `/* Hence, ` $b_i \leq \lceil 1/\varepsilon \rceil$ `*/`
  if $b_i \notin \{b_j \mid j \in B\}$ **then**
    $B := B \cup \{i\}$
for *each* $j \in B$ **do**
  Set $I_j := \{i \in [n] \mid b_i = b_j\}$;
$w_j := \begin{cases} \frac{1}{s_j} \sum_{i \in I_j} s_i & j \in B \\ 0 & \text{otherwise.} \end{cases}$;
return $(w_1, \cdots, w_n)$

Lemma 9 *Let $k, n \geq 1$ be a pair of integers, $\varepsilon > 0$ and let $f, s : [n] \to (0, \infty)$ be a pair of k-piecewise monotonic functions. Let $w = (w_1, \cdots, w_n) \in \mathbb{R}^n$ denote the output of a call to*

PiecewiseCoreset$(n, s, \varepsilon/(4k\sum_{i=1}^{n} s_i))$; *see Algorithm 5. If for every* $i \in [n]$

$$f(i) \leq s_i \sum_{j=1}^{n} f(j) \tag{14}$$

then

$$(1-\varepsilon)\sum_{i=1}^{n} f(i) \leq \sum_{i=1}^{n} w_i f(i) \leq (1+\varepsilon)\sum_{i=1}^{n} f(i).$$

Proof

For every $i \in [n]$ let

$$h_i = \frac{f(i)}{s_i \sum_{j=1}^{n} f(j)}.$$

We will prove that for a vector w that is returned from a call to PiecewiseCoreset(n, s, ε) we have

$$\left| \sum_i \frac{s_i}{t} \cdot h_i - \sum_{j \in B} \frac{w_j s_j}{t} \cdot h_j \right| \leq 4\varepsilon k, \tag{15}$$

Multiplying this by $t\sum_{i=1}^{n} f(i)$ yields

$$|\sum_{i=1}^{n} f(i) - \sum_{j \in B} w_j f(j)| = |\sum_{i=1}^{n} f(i) - \sum_{j=1}^{n} w_j f(j)| \leq 4kt\varepsilon \sum_i f(i).$$

Replacing ε by $\varepsilon/(4kt)$ proves Lemma 9.

Since both s and f are k-piecewise monotonic, h is $2k$-monotonic. Hence, there is a partition $\Pi = \{[i_1], [i_2] \setminus [i_1], \cdots, [n] \setminus [i_{2k-1}]\}$ of $[n]$ into consecutive $2k$ intervals such that h_i is monotonic over each of these intervals.

Let $I_j = \{i \in [n] : b_i = b_j\}$ for every $j \in B$. For every $I \in \Pi$ we define $\text{Good}(I) = \{j \in B \mid I_j \subseteq I\}$. Their union is denoted by $\text{Good} = \sum_{I \in \Pi} \text{Good}(I)$. Hence,

$$\begin{aligned}
\left| \sum_{i \in [n]} \frac{s_i}{t} \cdot h_i - \sum_{j \in B} \frac{w_j s_j}{t} \cdot h_j \right| &= \left| \sum_{i \in [n]} \frac{s_i}{t} \cdot h_i - \sum_{j \in B} \frac{\sum_{i \in I_j} s_i}{t} \cdot h_j \right| = \sum_{j \in B} \sum_{i \in I_j} \frac{s_i}{t} \cdot (h_i - h_j) \\
&\leq \left| \sum_{j \in B \setminus \text{Good}} \sum_{i \in I_j} \frac{s_i}{t} \cdot (h_i - h_j) \right| \qquad (16) \\
&\quad + \sum_{I \in \Pi} \left| \sum_{j \in \text{Good}(I)} \sum_{i \in I_j} \frac{s_i}{t} \cdot (h_i - h_j) \right|. \qquad (17)
\end{aligned}$$

We now bound (16) and (17). Put $j \in B$. By Line 5 of the algorithm we have $|I_j \cap B| = 1$ and $\sum_{i \in I_j} s_i/t \leq \varepsilon$. Hence,

$$\left| \sum_{i \in I_j} \frac{s_i}{t} \cdot (h_i - h_j) \right| \leq \varepsilon(\max_{i \in I_j} h_i - \min_{i \in I_j} h_i) \leq \varepsilon, \tag{18}$$

where the last inequality holds since $h_i \leq 1$ for every $i \in [n]$, by (14). Since each set $I \in \Pi$ contains consecutive numbers, we have $|B \setminus \text{Good}| \leq 2k$. Using this and (18), we bound (16) by

$$\left| \sum_{j \in B \setminus \text{Good}} \sum_{i \in I_j} \frac{s_i}{t} \cdot (h_i - h_j) \right| \leq |B \setminus \text{Good}| \cdot \varepsilon \leq |\Pi| \varepsilon \leq 2\varepsilon k. \tag{19}$$

Put $I \in \Pi$ and denote the numbers in $\text{Good}(I)$ by $\text{Good}(I) = \{k, k+1, \cdots, \ell\}$. Recall that h is monotonic on I. Without loss of generality, assume that h is non-decreasing on I. Therefore, summing (18) over $\text{Good}(I)$ yields

$$\begin{aligned} \left| \sum_{j \in \text{Good}(I)} \sum_{i \in I_j} \frac{s_i}{t} (h_i - h_j) \right| &\leq \sum_{j=k}^{\ell} \left| \sum_{i \in I_j} \frac{s_i}{t} (h_i - h_j) \right| \leq \sum_{j=k}^{\ell} \varepsilon (\max_{i \in I_j} h_i - \min_{i \in I_j} h_i) \\ &\leq \varepsilon \sum_{j=k}^{\ell-1} (\min_{i \in I_{j+1}} h_i - \min_{i \in I_j} h_i) = \varepsilon (\min_{i \in I_\ell} h_i - \min_{i \in I_1} h_i) \leq \varepsilon, \end{aligned}$$

where in the last derivation we used the fact that $h_i \leq 1$ for every $i \in [n]$. Summing over every $I \in \Pi$ bounds (17) as,

$$\sum_{I \in \Pi} \left| \sum_{j \in \text{Good}(I)} \sum_{i \in I_j} \frac{s_i}{t} \cdot (h_i - h_j) \right| \leq |\Pi| \cdot \varepsilon \leq 2\varepsilon k.$$

Plugging (19) and the last inequality in (16) and (17) respectively proves (15) as

$$\left| \sum_{i \in [n]} \frac{s_i}{t} \cdot h_i - \sum_{j \in B} \frac{w_j s_j}{t} h_j \right| \leq 4\varepsilon k$$

■

For every $p, q \in \mathbb{R}^d$ we denote $D(p, q) = \|p - q\|^2$, where $\|p - q\|$ is the Euclidean distance between p and q.

Lemma 10 *Let $p_1, \cdots, p_n$ be a set of points on a line in $\mathbb{R}^d$ such that $\|p_1 - p_2\| = \cdots = \|p_{n-1} - p_n\| = \Delta$ for some $\Delta \geq 0$ and the first coordinate of p_i is i for every $i \in [n]$. Let $\ell : \mathbb{R} \to \mathbb{R}^d$ be a function such that $\{(x, \ell(x)) \mid x \in \mathbb{R}\}$ is a line in $\mathbb{R}^{d+1}$. Then for every $i \in [n]$*

$$\|p_i - \ell(i)\|_2^2 \leq \frac{4 \sum_{j \in [i]} \|p_i - \ell(i)\|_2^2}{i}.$$

Proof Since P is contained in a line, it can be shown Feldman et al. (2006) that there is a point $q \in \mathbb{R}^d$ and a positive number $w > 0$ such that for every $i \in [n]$

$$\|p_i - \ell(i)\|_2 = w \|p_i - q\|_2. \tag{20}$$

Let $\tilde{D} : [0,\infty) \to [0,\infty)$ be a monotone non-decreasing function and $r \in [0,\infty)$ such that $D(xe^{\delta}) \leq e^{r\delta}D(x)$ for every $x, \delta > 0$. It can be shown that for $\rho = \max\left\{2^{r-1}, 1\right\}$ and every $a, b, c \in M$ in a metric space (M, dist) we have

$$\tilde{D}(\text{dist}(a,c)) \leq \rho(\tilde{D}(\text{dist}(a,b)) + \tilde{D}(\text{dist}(b,c)));$$

See Feldman and Schulman (2012). In particular, for the case $M = \mathbb{R}^d$, $\text{dist}(a,b) = w\|a-b\|$, we denote $D(a,b) = \tilde{D}(w\|a-b\|)$ to obtain

$$D(a,c) \leq \rho(D(a,b) + D(b,c)). \tag{21}$$

Let $m = \frac{1}{i}\sum_{j\in[i]} D(p_j, q)$ and $i \in [n]$. We will prove that

$$D(p_i, q) \leq 4m\rho^2. \tag{22}$$

In particular, for $\tilde{D}(x) = x^2$ we have $r = 2$, $\rho = 1$ and

$$\begin{aligned}\|p_i - \ell(i)\|_2^2 = \tilde{D}(\|p_i - \ell(i)\|) = \tilde{D}(w\|p_i - q\|) = D(p_i, q)\\ \leq 4m = \frac{4\sum_{j\in[i]} \|p_i - \ell(i)\|_2^2}{i},\end{aligned} \tag{23}$$

where the second equality is by (20), and (23) is by (22).

Indeed, let $Q = \{j \in [i] \mid D(p_j, q) \leq 2m\}$. By Markov's inequality,

$$|Q| \geq \frac{i}{2}. \tag{24}$$

Hence, there are $p_s, p_t \in Q$ such that $s - t \geq i/2$. Using this and (21)

$$D(p_s, p_t) \leq \rho(D(p_s, q) + D(q, p_t)) \leq 2\rho m. \tag{25}$$

Since $s - t \geq i/2$,

$$\Delta\|p_i - p_s\| = \Delta(i - s) \leq \Delta(i - i/2) = \Delta i/2 \leq \Delta(s - t) = \Delta\|p_s - p_t\|.$$

Since $\tilde{D}$ is non-decreasing, the last equation implies $D(p_i, p_s) \leq D(p_s, p_t)$. Together with (25) we get $D(p_i, p_s) \leq 2\rho m$. Using the last inequality and the fact that $p_s \in Q$ proves (22) as

$$D(p_i, q) \leq \rho(D(p_i, p_s) + D(p_s, q)) \leq \rho(2\rho m + 2m) \leq 4m\rho^2.$$

■

A function $g : \mathbb{R} \to \mathbb{R}^d$ is a 2-*piecewise linear function* if the set $\{(x, g(x)) \mid x \in \mathbb{R}\}$ is the union of two linear segments in $\mathbb{R}^{d+1}$.

Corollary 11 *Let $(w_1, \cdots, w_n) \in \mathbb{R}^n$ be the output of a call to* PiecewiseCoreset$(n, s, \frac{c\varepsilon}{\log n})$ *where c is a sufficiently large universal constant, $n \geq 1$, $\varepsilon > 0$ and s is the function that maps every $i \in [n]$ to $s_i = \max\left\{\frac{4}{i}, \frac{4}{n-i+1}\right\}$.*

Then for every set $(1, p_1), \cdots, (n, p_n)$ of n points that is contained in a line in $\mathbb{R}^{d+1}$ and every 2-piecewise linear function $g : \mathbb{R} \to \mathbb{R}^d$ the following hold:

1. *w has $\|w\|_0 = O\left(\frac{\log n}{\varepsilon}\right)$ non-zeroes entries.*

2. *w can be computed in $O(\log n) \cdot \|w\|_0$ time and $\|w\|_0$ space.*

3. $$(1-\varepsilon)\sum_{i=1}^{n}\|g(i)-p_i\|^2 \leq \sum_{i=1}^{n} w_i^2\|g(i)-p_i\|^2 \leq (1+\varepsilon)\sum_{i=1}^{n}\|g(i)-p_i\|^2.$$

Proof ***(i)*** Put $\varepsilon' = c\varepsilon \log n$. By Line 8 of the algorithm, $\|w\|_0 = |B|$. Since B consists of distinct integers $b_i \in [1, 1/\varepsilon' + 1]$ we have $\|w\|_0 = |B| = O(1/\varepsilon') = O(\log(n)/\varepsilon)$.
(ii) Since b_i is monotonic over $i \in [n]$, we can use binary search on $[n]$ to compute the smallest $i \in [n]$ such that $b_i \notin B$. In each of the $O(\log n)$ iterations we compute b_j for some $j \in [n]$. Since $\sum_{j=1}^{i} s_i$ is a sum of two harmonic series, b_j can be computed in $O(1)$ time. As explained in (i), $|B| = O(\log(n)/\varepsilon)$ so the overall time is $O(\log(n)/\varepsilon') = O(\log^2 n/\varepsilon)$. We only need to store w during this recursion, which takes $\|w\|_0$ space.
(iii) Put $i \in [n]$ and let $f(i) = \|p_i - g(i)\|^2$. Since $(1, p_1), \cdots, (n, p_n)$ are on a line, we have that $\|p_1 - p_2\| = \cdots = \|p_{n-1} - p_n\| = \Delta$ for some $\Delta \geq 0$. Since g is 2-piecewise linear function, there is a line $\{x, \ell(x)\}$ for some $\ell : \mathbb{R} \to \mathbb{R}^d$ such that $\ell(j) = g(j)$ for every $j \in [i]$ or every $j \in \{i, i+1, \cdots, n\}$. Without loss of generality, we assume the first case. By Lemma 10,

$$f(i) = \|p_i - g(i)\|_2^2 = \|p_i - \ell(i)\|_2^2 \leq \frac{4\sum_{j\in[i]}\|p_i - \ell(i)\|_2^2}{i} \leq s_i \sum_{j\in[i]}\|p_i - \ell(i)\|_2^2 = s_i \sum_{j\in[i]} f(j). \tag{26}$$

Since g is 2-piecewise linear and $p_1, \cdots, p_n$ are points on a line, we have that f is 4-monotonic over $[n]$. The function s is 2-monotonic. We also have that

$$\frac{c\varepsilon}{\log n} \leq \frac{\varepsilon}{4k\sum_{i=1}^{n} s_i}$$

for a sufficiently small constant c. Plugging this and (26) in Lemma 9 then proves the theorem as

$$(1-\varepsilon)\sum_{i=1}^{n} f(i) \leq \sum_{i=1}^{n} w_i f(i) \leq (1+\varepsilon)\sum_{i=1}^{n} f(i).$$

■

We show how to compute a $(1+\varepsilon)$-approximation to the k-segment mean of the original signal P using its coreset. The technique can be used to solve any other optimization problem over k-segments, assuming that we have an existing algorithm for a weighted signal. For example, if priors are given (weights for each segment) or we want to minimize the cost over some subset of k-segments (e.g., (k, m)-segment mean).

We assume that we are given a possibly inefficient algorithm Slow-Segmentation ("black box") that will be used to extract the will compute the k-segment mean of a small

set that is based on the coreset. The algorithm Slow-Segmentation gets a set Q of pairs $((t,p),w)$ where $t \in \mathbb{R}$, (t,p) is a point in $\mathbb{R}^{d+1}$, and $w > 0$ denote its weight. The algorithm then returns the k-segment mean of Q, i.e., the k-piecewise linear function that minimizes the *weighted cost*

$$\text{cost}_W(Q,f) := \sum_{((t,p),w)\in Q} w\| (p - f(t)) \|^2,$$

We will run this algorithm only on a small set Q, whose size is roughly the size of the coreset. In what follows we describe the algorithm Fast-Segmentation that uses the coreset and Slow-Segmentation to get a fast approximation of the k-segment mean of the original set P.

Algorithm overview The input to the algorithm Fast-Segmentation is a signal P of n points in $\mathbb{R}^d$, an error parameter $\varepsilon > 0$, and an integer $k \geq 1$. In addition, the algorithm gets a pointer to the algorithm Slow-Segmentation.

Algorithm 6: Fast-Segmentation$(P, k, \varepsilon,$ Slow-Segmentation$)$

Input:

- A set $P = \{(1,p_1), \cdots, (n,p_n)\}$ in $\mathbb{R}^{d+1}$,
- an integer $k \geq 1$,
- an error parameter $\varepsilon > 0$, and
- an algorithm Slow-Segmentation(Q,k) that computes the k-segment mean of a given weighted set Q.

Output: A $(1+\varepsilon)$-approximation f to the k-segment mean of P.

$D \leftarrow$ Coreset(P,k,ε); See Algorithm 4.
Identify $D = \{(C_1,g_1,b_1,e_1),..,(C_m,g_m,b_m,e_m)\}$
$Q \leftarrow \emptyset$
for $i \leftarrow 1$ *to* m **do**
  $P_i \leftarrow \{(b_i, g_i(b_i)), \cdots, (e_i, g_i(e_i))\}$
  $(w_1, \cdots, w_n) \leftarrow$ PiecewiseCoreset$(|P_i|, s, c\varepsilon/\log(n))$, where c and s are defined in Corollary 11.
  $Q \leftarrow Q \cup \left\{ (t,p), w_j^2) \mid (t,p) \text{ is the } j\text{th point of the signal } P_i \right\}$
$h \leftarrow$ Slow-Segmentation(Q,k)
for $i \leftarrow 1$ *to* m **do**
  $T_i \leftarrow \{b_i, \cdots, e_i\}$
  if $\{h(t) \mid t \in T_i\}$ *consists of at most* 2*-segments* **then**
    $f(t) \leftarrow h(t)$ for every $t \in T_i$
  else
    $f(t) \leftarrow g_i(t)$ for every $t \in T_i$
  return f

Recall that for a k-segment $f : \mathbb{R} \to \mathbb{R}^d$ and $i \in [m]$ we say that C_i is served by one segment of f if $\{f(t) \mid b_i \leq t \leq e_i\}$ is a linear segment. The next lemma states the weighted

set Q that is computed in Line 7 of Algorithm 6 is a weak coreset in the following sense. For every k-segment f such that each cell C_i is served by at most two segments of f, the cost of P and the weighted cost of Q to f are approximately the same. In Theorem 7 we prove that a k-segment mean has this property, and thus can be computed from this coreset Q.

Lemma 12 (Weak coreset) *Let f be a k-segment such that C_i is served by at most two segments of f, for every $i \in [m]$. Then*

$$\min_f \text{cost}(P, f) \leq \min_f \text{cost}_W(Q, f) \leq (1+\varepsilon) \min_f \text{cost}(P, f).$$

Proof Put $i \in [m]$ and $P_i = \left\{(t_1, p_1), \cdots, (t_{|P_i|}, p_{|P_i|})\right\}$. Since C_i is served by at most two segments of f, then P_i is also served by at most two segments of f. By Corollary 11,

$$(1-\varepsilon)\sum_{j=1}^{|P_i|} w_j \|f(t_j) - p_j\|^2 \leq \sum_{j=1}^{|P_i|} w_j^2 \|f(t_j) - p_j\|^2 \leq (1+\varepsilon)\sum_{j=1}^{|P_i|} \|f(t_j) - p_j\|^2.$$

Hence, letting $Q_i = \left\{(t_j, p_j), w_j^2) \mid w_j > 0, j \in [|P_i|]\right\}$, by Line 7 of Algorithm 6 we obtain

$$\begin{aligned}|\text{cost}(P_i, f) - \text{cost}_W(Q_i, f)| &= \left|\sum_{j=1}^{|P_i|} \|f(t_j) - p_j\|^2 - \sum_{j=1}^{|P_i|} w_j^2 \|f(t_j) - p_j\|^2\right| \\ &\leq \varepsilon \sum_{j=1}^{|P_i|} \|f(t_j) - p_j\|^2 = \varepsilon \text{cost}(P_i, f).\end{aligned}$$

Summing over every $i \in [m]$ yields

$$|\text{cost}(P, f) - \text{cost}_W(Q, f)| \leq \varepsilon \text{cost}(P, f).$$

■

Theorem 13 *Let $P = \{(1, p_1), \cdots, (n, p_n)\}$ be a set in $\mathbb{R}^{d+1}$, $\varepsilon \in (0, 1/2)$, and $k \geq 1$ be an integer. Let $f : \mathbb{R} \to \mathbb{R}^d$ be the output of a call to* Fast-Segmentation$(P, k, \varepsilon,$ SegAlg$)$. *Then f is a $(1+\varepsilon)$-approximation to the k-segment mean of P, i.e.,*

$$\text{cost}(P, f) \leq (1+\varepsilon) \min_{f'} \text{cost}(P, f'),$$

where the minimum is over every k-segment $g : \mathbb{R} \to \mathbb{R}^d$.

Proof Let h be the k-segment that is computed in Line 8 of the algorithm Fast-Segmentation(P, k), and let $i \in [m]$. We first prove that $\text{cost}(P_i, f) \leq \text{cost}(P_i, h)$ by case analysis: (i) $f(t) = h_i(t)$ for every $t \in T_i$, and (ii) $f(t) = g_i(t)$ for every $t \in T_i$.
Case (i): In this case $\text{cost}(P_i, f) = \text{cost}(P_i, h)$ by definition of P_i.
Case (ii): In this case $\text{cost}(P_i, f) = \text{cost}(P_i, g_i)$. By its construction, g_i is a 2-approximation

for the 1-segment mean of P_i. Since the points of P_i lie on a line, we thus have $\text{cost}(P_i, g_i) = 0$. Hence,

$$\text{cost}(P_i, f) = \text{cost}(P_i, g_i) = 0 \leq \text{cost}(P_i, h).$$

Summing $\text{cost}(P_i, f) \leq \text{cost}(P_i, h)$ over $i \in [m]$ yields

$$\text{cost}(P, f) \leq \text{cost}(P, h). \tag{27}$$

Suppose that h^* minimizes $\text{cost}(P, f')$ over every k-segment $f' : \mathbb{R} \to \mathbb{R}^d$. Similarly to (27), it can be shown that there is a k-segment f^* such that

$$\text{cost}(P, f^*) \leq \text{cost}(P, h^*),$$

and C_i is served by at most two segments of f^*, for every $i \in [m]$. We then have

$$\begin{aligned}
\text{cost}(P, f) &\leq \frac{\text{cost}_W(Q, f)}{1+\varepsilon} && (28)\\
&\leq \frac{\text{cost}_W(Q, h)}{1+\varepsilon} && (29)\\
&\leq \frac{\text{cost}_W(Q, f^*)}{1+\varepsilon} && (30)\\
&\leq \frac{(1-\varepsilon)\text{cost}(P, f^*)}{1+\varepsilon} && (31)\\
&\leq (1+10\varepsilon)\text{cost}(P, f^*), && (32)
\end{aligned}$$

where (29) holds by (27), Eq. (28) and (31) hold by Lemma 12, Eq. (30) is by the optimality of h, and (32) holds since $\varepsilon \leq 1/2$. Replacing ε with $\varepsilon/10$ proves this theorem. ■

7.1 Efficient k-Segment Mean Computation

However, since we cannot allow linear time search over the data, we add the additional constraints that there cannot be more than one k-segment endpoint inside each coreset-segment, and that each of the k-segments starts and terminates at an endpoint of the coreset segments. This allows us to use the coreset obtained by Algorithm 2 for cost computations coreset, and perform the computation of all linear segment costs required in Bellman (1961) on a sublinear number of sampling points, reducing overall algorithm complexity from $O(kN^2)$ to $O(k^3 log^2 N)$. By construction of the piecewise coresets, and the segments C_i computed in Algorithm 2, the cost computed with these limitations on the endpoints is an ε approximation of the cost of our solution on the real data. Specifically, our solution is an ε-approximation to the real optimal solution.

The modifications required compared to the algorithm of Bellman (1961) for this case are as follows

- During the search over $u_{k'}$, $u_{k'}$ is allowed only to be at locations which are part of the piecewise coreset of some segment in D.

- For each line segment $(u_{k'}, b)$, its fitting solution and cost is obtained by concatenating row-wise the matrices C_i from each segment i of D completely contained inside $(u_{k'}, b)$, along with the sampling points inside $(u_{k'}, b)$ from partially contained segments of D, into a single matrix $C_{(u_{k'},b)}$, and solving for the linear segment using $C_{(u_{k'},b)}$.
- $h\left(u_{k'}, u_{k'}\right)$ is defined to be infinite if two segment endpoints are inside a coreset segment.

The algorithm is described as Algorithm 7. Let $L_{coreset}$ denote the maximum number

Algorithm 7: Solving for k-segmentation using a coreset

for $b = 1, 2, \ldots, m$ **do**
  Update the 1-segment solution for each subsegment starting at $t = 1$

$$f_1(b) = h\left(1, b\right)$$

end for
for $k' = 1, 2, \ldots, K$ **do**
  for $b = 1, 2, \ldots, m$ **do**
    for $u_{k'} = 1, 2, \ldots, b$ **do**
      Update the k'-segment solution by updating the cost $f_{k'}$ based on the $(k'-1)$-segment solution $f_{k'-1}$

$$f_{k'}(b) = \min_{1 \geq u_{k'} \geq b} \left[h\left(u_{k'}, b\right) + f_{k'-1}(u_{k'})\right],$$

      where $h\left(u_{k'}, b\right)$ is computed using the appropriate matrix $C_{(u_{k'},b)}$.
    end for
  end for
end for

of inner-points per segment obtained from Algorithm 5. The number of segment fitting cost computations done is

$$O\left(\left(\frac{(k\alpha)}{\varepsilon^2} + \beta k\right)^2 L_{coreset}^2 k\right) \tag{33}$$

Theorem 14 *Given a coreset D as described in Algorithm 2, and a set of piecewise-coresets computed as in Algorithm 5 for each segment, Algorithm 7 finds an ε-approximation of the k-segment mean in time $O(\text{polylog}(n)\,\text{poly}(k))$*

Proof Computation time is determined by the number of sampling points over the whole signal. Since each segment has n points at most, we have $O(\frac{logn}{\varepsilon})$ sampling points according to Corollary 11. Since there are $\left(\frac{(k\alpha)^2}{\varepsilon^2} + \beta k\right)$ segments according to Equation 11, we have overall $O(\frac{logn}{\varepsilon}\left(\frac{(k\alpha)}{\varepsilon^2} + \beta k\right))$ sampling points. Therefore our algorithm requires

$O\left(\left(\frac{logn}{\varepsilon}\left(\frac{(k\alpha)}{\varepsilon^2}+\beta k\right)\right)k\right)$ estimations of linear segment fittings. Each line segment estimation involves constructing a matrix composed out of $O\left(\frac{(k\alpha)}{\varepsilon^2}+\beta k\right)$ complete segments plus possibly $O(\frac{logn}{\varepsilon})$ sampling points, and inverting it. We note that each segment (partial or full) contributes $O(logn)$ rows to the matrix, and that its width is $O(d)$. Hence, inverting it is $O(\text{polylog}(n)\,\text{poly}(k))$, therefore the algorithm takes $O(\text{polylog}(n)\,\text{poly}(k))$ to complete. The approximation property of the algorithms comes from the approximation of the coresets ■

8. Parallel and Streaming Implementation

One major advantage of coresets is that they can be constructed in parallel as well as in a streaming setting. The main observation is that the union of coresets is a coreset — if a data set is split into subsets, and we compute a coreset for every subset, then the union of the coresets is a coreset of the whole data set. This allows us to have each machine separately compute a coreset for a part of the data, with a central node which approximately solves the optimization problem; see (Feldman et al., 2013, Theorem 10.1) for more details and a formal proof.

When discussing streaming coresets, one must define the merging and reduction operations used in streaming, and show that the coresets create are still efficient and accurate. We build our merge and reduce operations as a modification of the coreset algorithm as given in Algorithm 4, so that it compacts coreset segments rather than signal points. For this we modify Algorithms 2,3 as we now describe.

First, we look at Algorithm 3, we modify it in the following way. In line 6 of the algorithm, the original segments from both child coresets are taken. Partitioning is done by unifying existing coreset segments into sections P_i. Iterating over Theorem 7, we note that part i is kept by the reduction of parts at each turn. Looking at the proof of part ii we note that we only use the coreset segments' cost as represented for 1-segments, and this can be computed by the C matrices, starting from the end of Equation 12. This holds also for the joined and compacted matrices.

Next, we look at Algorithm 2, and modify it to utilize the child coresets' coreset segments in order to construct a new set of coreset segments for the combined span. This requires several modification the the algorithm – notibly, the accumulation of a new coreset segment Q is done solely in terms of adding new child coreset segments. We note that f^* and λ in lines 5 and 6 respectively can be computed for concatenations of coreset segments, in terms of their $(1,\varepsilon/4)$-coresets. We note that C,g can be computed using the C matrices of the child coresets. We do so by concatenating the C child matrices, and recomputing the SVD for the concatenated matrix.

Specifically, let (U_1,S_1,V_1) and (U_2,S_2,V_2) be the SVD of matrices $P_1,P,2$ corresponding to the coreset segments creation. It's easy to show that

$$\left\|\left(S_1V_1^T\right)\begin{pmatrix} a \\ b \\ -I \end{pmatrix}\right\|_F^2 + \left\|\left(S_2V_2^T\right)\begin{pmatrix} a \\ b \\ -I \end{pmatrix}\right\|_F^2 =$$
$$\left\|\begin{pmatrix} S_1V_1^T \\ S_2V_2^T \end{pmatrix}\begin{pmatrix} a \\ b \\ -I \end{pmatrix}\right\|_F^2 = \left\|U_J^T\begin{pmatrix} S_1V_1^T \\ S_2V_2^T \end{pmatrix}\begin{pmatrix} a \\ b \\ -I \end{pmatrix}\right\|_F^2 = \tag{34}$$
$$\left\|S_JV_J^T\begin{pmatrix} a \\ b \\ -I \end{pmatrix}\right\|_F^2,$$

where (U_J, S_J, V_J) is the SVD of $\begin{pmatrix} S_1V_1^T \\ S_2V_2^T \end{pmatrix}$, and we used the properties of the Frobenius norm, the isometry properties of unitary matrices, and properties of (U_J, S_J, V_J), respectively. Once C_J is computed g_J can be computed easily.

Looking at the proof of Lemma 6, we note that the treatment of good coreset segments remains the same. The coreset segments that do not belong to $Good(D, f)$ still amount to the same cost bounds, due to the construction of g_J.

For efficient k-segmentation we run a k-segment mean algorithm on our small coreset instead of the original large input. Since the coreset is small we can apply dynamic programming (as in Bellman (1961)) in an efficient manner. In order to compute an $(1+\varepsilon)$ approximation to the k-segment mean of the original signal P, it suffices to compute a $(1+\varepsilon)$ approximation to the k-segment mean of the coreset, where cost is replaced by cost$'$. However, since D is not a simple signal, but a more involved data structure, it is not clear how to run existing algorithms on D. We demonstrate several possibilities for this, including a novel coreset for that purpose. In particular, we can run naive dynamic programming Bellman (1961) on the coreset and get a $(1+\varepsilon)$ approximate solution in an efficient manner, as we summarize as follows.

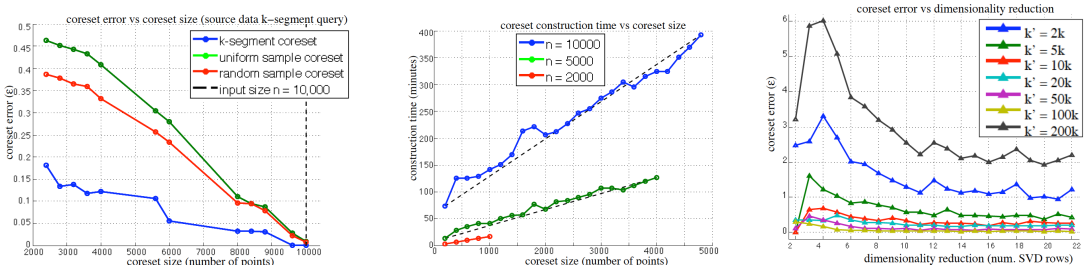
Theorem 15 *Let P be a d-dimensional signal. A $(1+\varepsilon)$ approximation to the k-segment mean of P can be computed in $O\left(ndk/\varepsilon + d(klog(n)/\varepsilon)^{O(1)}\right)$ time .*

9. Experimental Results

We now demonstrate the results of our algorithm on four data types of varying length and dimensionality. We compare our algorithms against several other segmentation algorithms. We also show that the coreset effectively improves the performance of several segmentation algorithms by running the algorithms on our coreset instead of the full data.

9.1 Segmentation of Large Datasets

We first examine the behavior of the algorithm on synthetic data which provides us with easy ground-truth, to evaluate the quality of the approximation, as well as the efficiency, and the scalability of the coreset algorithms. We generate synthetic test data by drawing a discrete k-segment P with $k = 20$, and then add Gaussian and salt-and-pepper noise.



(a) Coreset size vs coreset error (b) (k, ε)-coreset size vs construction time (c) Coreset dimensionality vs coreset error

Figure 2: Figure 2a shows the coreset error (ε) decreasing as a function of coreset size. The dotted black line indicates the point at which the coreset size is equal to the input size. Figure 2b shows the coreset construction time in minutes as a function of coreset size. Trendlines show the linear increase in construction time with coreset size. Figure 2c shows the reduction in coreset error as a function of the dimensionality of the 1-segment coreset, for a fixed input size (note that in practice dimensionality is often reduced down to $\mathbb{R}^2$.

We then benchmark the computed (k, ε)-coreset D by comparing it against piecewise linear approximations with (1) a uniformly sampled subset of control points U and (2) a randomly placed control points R. For a fair comparison between the (k, ε)-coreset D and the corresponding approximations U, R we allow the same number of coefficients for each approximation. Coresets are evaluated by computing the fitting cost to a query k-segment Q that is constructed based on the a-priori parameters used to generate P.

Approximation Power: Figure 2a shows the aggregated fitting cost error for 1500 experiments on synthetic data. We varied the assumed k' segment complexity. In the plot we show how well a given k' performed as a guess for the true value of k. As Figure 2a shows, we significantly outperform the other schemes. As the coreset size approaches the size P the error decreases to zero as expected.

Coreset Construction Time: Figure 2b shows the linear relationship between input size and construction time of D for different coreset size. Figure 2c shows how a high dimensionality benefits coreset construction. This is even more apparent in real data which tends to be sparse, so that in practice we are typically able to further reduce the coreset dimension in each segment.

Scalability: The coresets presented in this work are parallelizable, as discussed in Section 8. We demonstrate scalability by conducting very large scale experiments on both real and synthetic data, running our algorithm on a network of 255 Amazon EC2 vCPU nodes. We compress a 256,000-frame *bags-of-words* (BOW) stream in approximately 20 minutes, representing an almost-perfect scalability. For a comparable single node running on the same data dataset, we estimate a total running time of approximately 42 hours.

9.2 Real Data Experiments

We compare our coreset against uniform sample and random sample coresets, as well as two other segmentation techniques: Ramer-Douglas-Peucker (RDP) algorithm Ramer (1972); Douglas and Peucker (1973), and the Dead Reckoning (DR) algorithm Trajcevski et al. (2006). We also show that we can combine our coreset with segmentation algorithms, by running the algorithm on the coresets itself. We emphasize that segmentation techniques (RDP, DR) were purposely chosen as simple examples and are not intended to reflect the

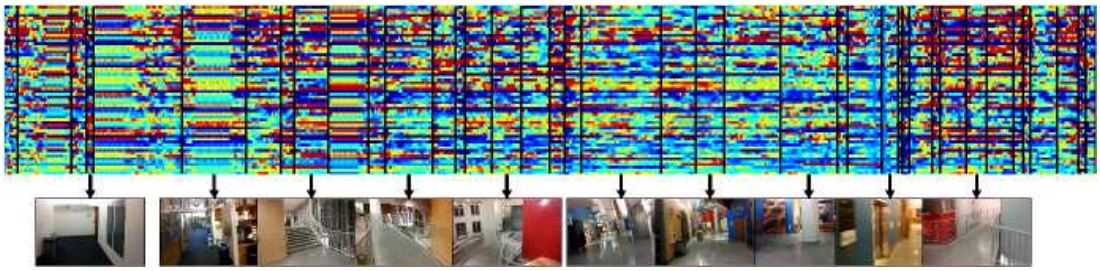


Figure 3: Segmentation from Google Glass. Black vertical lines present segment boundaries, overlayed on top of the bags of word representation. Icon images are taken from the middle of each segment.

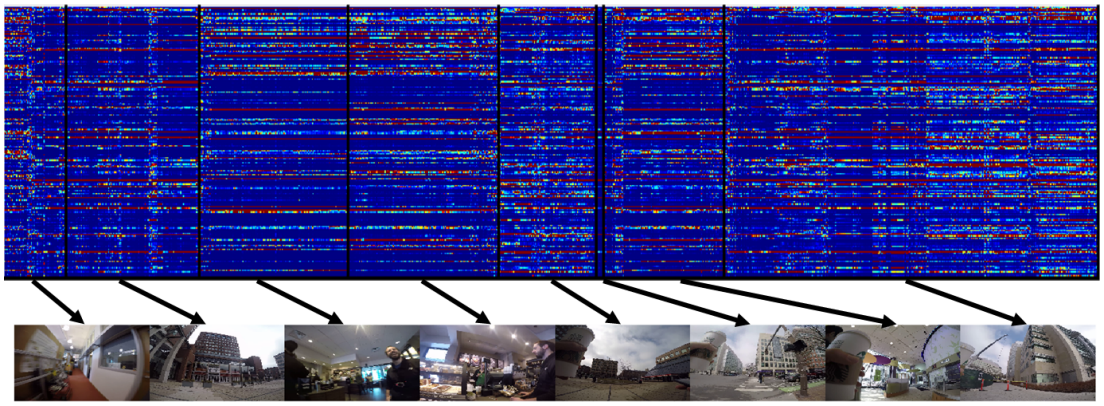


Figure 4: Segmentation based on the Places CNN model. Black vertical lines present segment boundaries, overlayed on top of the scene votes. Icon images are taken from the middle of each segment. The two meeting places (coffeeshop and restaurant) are easily captured as their own segment, where the other segments include time spent inside the laboratory and walking in the street.

state of the art – the point is to demonstrate how the k-segment coreset can be used to improve on any given algorithm.

To demonstrate the general applicability of our techniques, we run our algorithm using financial (1D) time series data, as well as GPS data (2D). For the 1D case we use Bitcoin price data from the Mt.Gox Bitcoin exchange. Bitcoin is of general interest because its price has grown exponentially with its popularity in the past two years. Bitcoin has also sustained several well-documented market crashes BBC (2013),CNBC (2013) that we can relate to our analysis. For the 2D case we use GPS data from a taxi fleet of 343 taxis in San Francisco. This is of interest because a taxi-route segmentation has an intuitive spacial interpretation that we can easily evaluate, and on the other hand GPS data forms an increasingly large information source which we are interested of analysing.

Figure 5a shows the results for the Bitcoin data. Notable market crash events are highlighted by local price highs (green) and lows (red). We observe that running the simple DR algorithm on our k-segment coreset to compute a segmentation captures these events quite well. Figures 5b,5c show example results for a single taxi. Again, we observe that computing a DR segmentation produces segments with a meaningful spatial interpretation. Figure 6 shows a plot of coreset errors for the first 50 taxis (right), and the table gives a summary of experimental results for the Bitcoin and GPS experiments.

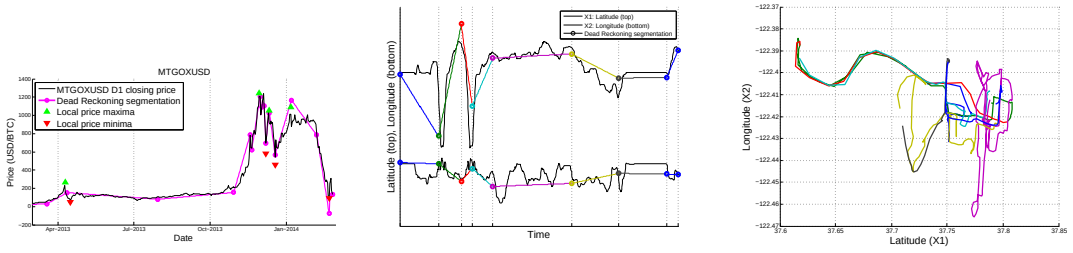
9.3 Semantic Video Segmentation

In addition, we demonstrate use of the proposed coreset for video streams summarization and compression. While different choices of frame representations for video summarization are available Sivic and Zisserman (2003); Li et al. (2009); Lu and Grauman (2013), we used BOWs based on color-augmented SURF features, quantized into 5000 visual words, trained on the ImageNet 2013 datasetDeng et al. (2009). The resulting signals are compressed in a streaming coreset. Computation in on a single core runs at 6Hz; A parallel version achieves 30Hz on a single i7 machine, processing 6 hours of video in 4 hours on a single machine, i.e. faster than real-time.

In Figure 3 we demonstrate segmentation of a video feed taken from Google Glass. We visualize the BOWs, as well as the segments suggested by the k-segment mean algorithm Bellman (1961) run on the coreset. Inspecting the results, most segment transitions occur at scene and room changes.

We note that computing the optimal segmentation cannot be done in real-time. We note that semantic segmentation of video is still unsolved and in particular, it can not be done in real-time. Our method for segmentation runs in real-time and can further be used to automatically summarize the video by associating representative frames with segments. To evaluate the "semantic" quality of our segmentation, we compared the resulting segments to uniform segmentation by contrasting them with a human annotation of the video into scenes. Our method gave a 25% improvement (in the Rand index Rand (1971)) over a 3000 frames sequence.

To give an example at a higher level, we used our algorithm to compress and then segment a stream of scene classification vote vectors based on the Places model Zhou et al. (2014). Our stream comes from a GoPro video camera strapped to a person, going from a lab space, to a couple of meetings, and back to the lab, totalling 60,000 frames and vector dimensionality of 205. As can be seen in Figure 4, the resulting 8-segmentation clearly shows the transitions between scene types, and would match the intuitive summary for this video.



(a) MTGOXUSD daily price data (b) GPS taxi data (c) GPS taxi data

Figure 5: (a) shows the daily Bitcoin price data from 2013 on, overlayed with a DR segmentation computed on our coreset. The red/green triangles indicate prominent market events. (b) 5c shows normalized GPS data overlayed with a DR segmentation computed on our coreset. (c) shows a lat/long plot (right) demonstrating that the segmentation yields a meaningful spacial interpretation.

Average ε	Bitcoin data	GPS data
***k*-segment coreset**	**0.0092**	**0.0014**
Uniform sample coreset	1.8726	0.0121
Random sample coreset	8.0110	0.0214
RDP on original data	0.0366	0.0231
RDP on *k*-segment	**0.0335**	**0.0051**
DeadRec on original data	0.0851	0.0417
DeadRec on *k*-segment	**0.0619**	**0.0385**

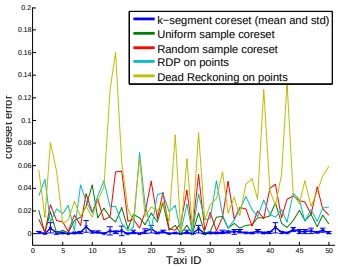


Figure 6: Table: summary of experimental results with Bitcoin and GPS data. Plot: visualization of GPS error and standard deviation results for the first 50 taxis.

10. Conclusions

In this paper we demonstrated a new framework for segmentation and event summarization of video data from robot cameras. We have shown the effectiveness and scalability of the algorithms proposed, and its applicability for large distributed video analysis with multiple devices. In the context of video processing, we demonstrate how using the right framework for analysis and clustering, even relatively straightforward representations of image content lead to a meaningful and reliable segmentation of video streams at real-time speeds.

Appendices

A 1-Segment Coreset

A $(1,\varepsilon)$-coreset approximates $\text{cost}(P,f)$ for every 1-segment f up to a factor of $1\pm\varepsilon$ as defined below.

Definition 16 ($(1,\varepsilon)$-coreset) *Let P and C be two sets in $\mathbb{R}^{d+1}$ and let $\varepsilon, w > 0$. The pair (C,w) is a* $(1,\varepsilon)$-coreset *for P, if for every 1-segment $f:\mathbb{R}\to\mathbb{R}^d$ we have*

$$(1-\varepsilon)\text{cost}(P,f) \leq w\cdot\text{cost}(C,f) \leq (1+\varepsilon)\text{cost}(P,f).$$

For example, (P,w) is a $(1,\varepsilon)$-coreset for P with $\varepsilon=0$ and $w=1$. However, a coreset is efficient if its size $|C|$ is much smaller than P.

It is easy to compute $\text{cost}(P,f)$ exactly by a matrix ΣV^T of $(d+2)$ rows using SVD, as shown in Algorithm 8. In our coreset construction we use additional matrices Q and Y to turn this matrix into a subset C of $\mathbb{R}^{d+1}$ so that the cost $\text{cost}(P,f)=\text{cost}(C,f)$ is still a

point-wise cost, although a weighted one. This allows us to improve the result later in this section, to get a less trivial coreset C of only $O(1/\varepsilon^2)$ rows.

Algorithm 8: 1-SegmentCoreset(P)

Input: A set $P = \{(t_1, p_1), \cdots, (t_n, p_n)\}$ in $\mathbb{R}^{d+1}$.

Output: A $(1, 0)$-coreset (C, w) that satisfies Claim 17.

1 Set $X \in \mathbb{R}^{n\times(d+2)}$ to be matrix whose ith row is $(1, t_i, p_i)$ for every $i \in [n]$.

2 Compute the thin SVD $X = U\Sigma V^T$ of X.

3 Set $u \in \mathbb{R}^{d+2}$ to be the leftmost column of ΣV^T.

4 Set $w := \frac{\|u\|^2}{d+2}$. /* $w > 0$ since $\|\Sigma\| = \|X\| > 0$ */

5 Set $Q, Y \in \mathbb{R}^{(d+2)\times(d+2)}$ to be unitary matrices whose leftmost columns are $u/\|u\|$ and $(\sqrt{w}, \cdots, \sqrt{w})/\|u\|$ respectively.

6 Set $B \in \mathbb{R}^{(d+2)\times(d+1)}$ to be the $(d + 1)$ rightmost columns of $YQ^T\Sigma V^T/\sqrt{w}$.

7 Set $C \subseteq \mathbb{R}^{d+1}$ to be the union of the rows in B ;

8 **return** (C, w).

Claim 17 *Let P be a set of n points in $\mathbb{R}^{d+1}$. Let (C, w) be the output of a call to* 1-SegmentCoreset(P)*; see Algorithm 8. Then (C, w) is a $(1, 0)$-coreset for P of size $|C| = d + 1$. Moreover, C and w can be computed in $O(nd^2)$ time.*

Proof

Let $f : \mathbb{R} \to \mathbb{R}^d$ be a 1-segment. Hence, there are row vectors $a, b \in \mathbb{R}^d$ such that $f(t) = a + b\,t$, for every $t \in \mathbb{R}$. By definition of Q and Y we have $YQ^Tu/\|u\| = (\sqrt{w}, \cdots, \sqrt{w})^T/\|u\|$. The leftmost column of $YQ^T\Sigma V^T$ is thus $YQ^Tu = (\sqrt{w}, \cdots, \sqrt{w})^T$. Therefore,

$$\begin{aligned}
\text{cost}\,(P, f) &= \sum_{(t,p)\in P} \|f(t) - p\|^2 = \sum_{(t,p)\in P} \|a + bt - p\|^2 = \sum_{(t,p)\in P} \left\| \begin{bmatrix}1 & t\end{bmatrix} \begin{bmatrix}a\\ b\end{bmatrix} - p \right\|^2 \\
&= \left\| X \begin{bmatrix}a\\ b\\ -I\end{bmatrix} \right\|^2 = \left\| U\Sigma V^T \begin{bmatrix}a\\ b\\ -I\end{bmatrix} \right\|^2 = \left\| YQ^T\Sigma V^T \begin{bmatrix}a\\ b\\ -I\end{bmatrix} \right\|^2 = \left\| \begin{bmatrix}\sqrt{w} & \\ \vdots & \sqrt{w}B\\ \sqrt{w} & \end{bmatrix} \begin{bmatrix}a\\ b\\ -I\end{bmatrix} \right\|^2 \\
&= w \left\| \begin{bmatrix}1 & \\ \vdots & B\\ 1 & \end{bmatrix} \begin{bmatrix}a\\ b\\ -I\end{bmatrix} \right\|^2 = w \sum_{(t,p)\in B} \|(a + bt - p)\|^2 = w \cdot \text{cost}(C, f).
\end{aligned}$$

Construction Time. The matrices Q and Y can be computed in $O(dn^2)$ time using the QR decomposition of $\begin{bmatrix}(1, \cdots, 1)^T & I\end{bmatrix}$ and $\begin{bmatrix}u & I\end{bmatrix}$. Computing the thin SVD of an $n \times d$ matrix X also takes $O(nd^2)$ time. Hence, the overall running time is $O(nd^2)$ Pearson (1901). ■

The size $d + 1$ and running time of the above $(1, 0)$-coreset C might be too large, for example when d is in the order of n, or we are dealing with high dimensional space such as images or text. On the other side, in the rest of the paper the construction of $(1, \varepsilon)$-coresets suffices. Using recent results from Feldman et al. (2013) and Ghashami and Phillips (2014), the following theorem yields faster and smaller coreset constructions when $d \gg 1/\varepsilon$.

Theorem 18 *Let $P \subseteq \mathbb{R}^{d+1}$ and let $\varepsilon > 0$. A $(1,\varepsilon)$-coreset $C \subseteq \mathbb{R}^{d+1}$ for P of size $|C| = O(1/\varepsilon^2)$ can be computed in $O(nd/\varepsilon^4)$ time.*

Proof It was proven in Feldman et al. (2013) that a coreset for P and a family of query shapes, where each shape is spanned by $O(1)$ vectors in $\mathbb{R}^d$, can be computed by projecting P on a $(1/\varepsilon^2)$ dimensional subspace S that minimizes the sum of squared distances to P up to a $(1+\varepsilon)$ factor. The resulting coreset approximates the sum of squared distances to every such shape up to a factor of $(1+\varepsilon)$. The size of this coreset is n, the same as the input size, however the coreset is contained in an $O(1/\varepsilon^2)$ dimensional subspace. We then compute a $(1,0)$-coreset C for this low dimensional set of n points in $s = O(1/\varepsilon^2)$ space using Claim 17. This will take additional $O(ns^2)$ time and the resulting coreset will be of size $O(s)$.

The subspace S can be computed deterministically in $O(nd/\varepsilon^4)$ using a recent result of Ghashami and Phillips (2014). ■

As proven below, the 1-segment mean of C is an approximation to the 1-segment mean of P. So, using C we can compute a fast approximation for the 1-segment mean of P.

Corollary 19 *Let $\varepsilon \in (0,1)$. A $(1+\varepsilon)$-approximation to the 1-segment of P can be computed in $O(nd/\varepsilon^4)$ time.*

Proof Using Theorem 18 we compute a $(1,\varepsilon)$-coreset C of size $|C| = O(1/\varepsilon^2)$ in $O(nd/\varepsilon^4)$ time. Then, using the singular value decomposition it is easy to compute a 1-segment mean f of C in $O(d\cdot|C|^2) = O(d/\varepsilon^4)$ time. Hence, the overall running time is $O(nd/\varepsilon^4)$.

Let f^* be a 1-segment mean of P and f be an arbitrary 1-segment. Since C is a $(1,\varepsilon)$-coreset for P,

$$\text{cost}(P,f) \le (1+\varepsilon)\text{cost}(C,f) \le (1+\varepsilon)\text{cost}(C,f^*) \le (1+\varepsilon)^2\text{cost}(P,f^*) \le (1+3\varepsilon)\text{cost}(P,f^*),$$

where in the last inequality we use the assumption $\varepsilon < 1$. Replacing ε with $\varepsilon/3$ in the above proof proves the corollary. ■

In the previous section we showed that a 1-segment coreset (C,w) of size independent of n exists for every signal P. Unfortunately, the next example shows that, in general, for $k \ge 3$ such a coreset C must contain all the n points of P. This result justifies the more complicated definition of a (k,ε)-coreset in the next section; See Definition 3.

Claim 20 *For every integers $n,c,d \ge 1$ there is a set P of n points in $\mathbb{R}^{d+1}$ such that the following holds. If $C \subseteq \mathbb{R}^{d+1}$ and $|C| < n$ then there is a 3-segment f such that either*

$$\text{cost}(C,f) \ge c\cdot\text{cost}(P,f) \;\; \textit{or} \;\; \text{cost}(P,f) \ge c\cdot\text{cost}(C,f).$$

Proof Let $P = \{(i,0,\cdots,0)\}_{i=1}^n$, a constant-0 signal. Consider the 3-segment $f:\mathbb{R}\to\mathbb{R}^d$ such that $f(t) = (0,\cdots,0)$ for every $t\in\mathbb{R}$. We have $\text{cost}(P,f) = 0$. If $\text{cost}(C,f) > 0$ then $\text{cost}(C,f) \ge c\cdot\text{cost}(P,f)$ as desired.

Otherwise, $\text{cost}(C, f) = 0$. Let $(t, p) \in P \setminus C$ and consider a 3-segment $g : \mathbb{R} \to \mathbb{R}^d$ such that $g(t) = f(t) = (0, \cdots, 0)$ for every $t \in \mathbb{R} \setminus \{t\}$ and $g(t) \neq p$. Hence,

$$\begin{aligned}\text{cost}(C, g) &= \sum_{(t',p')\in C} \|p' - g(t')\|^2 = \sum_{(t',p')\in C\setminus(t,p)} \|p' - g(t')\|^2 \\ &= \sum_{(t',p')\in C} \|p' - f(t')\|^2 = \text{cost}(C, f) = 0.\end{aligned}$$

Since $\text{cost}(P, g) = \|p - g(t)\|^2 > 0$ the last two inequalities imply $\text{cost}(P, g) \geq c \cdot \text{cost}(C, g)$. ■

References

P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Geometric approximations via coresets. *Combinatorial and Computational Geometry - MSRI Publications*, 52:1–30, 2005.

Sunil Bandla and Kristen Grauman. Active learning of an action detector from untrimmed videos. In *ICCV*, 2013.

BBC. Bitcoin panic selling halves its value, 2013. URL `http://www.bbc.com/news/technology-22105322`.

Richard Bellman. On the approximation of curves by line segments using dynamic programming. *Commun. ACM*, 4(6):284, 1961.

Winston Churchill and Paul Newman. Continually improving large scale long term visual navigation of a vehicle in dynamic urban environments. In *Proc. IEEE Intelligent Transportation Systems Conference (ITSC)*, Anchorage, USA, September 2012.

CNBC. Bitcoin crash spurs race to create new exchanges, April 2013. URL `http://www.cnbc.com/id/100633793`.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Computer Vision and Pattern Recognition*, 2009.

David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

D. Feldman and M. Langberg. A unified framework for approximating and clustering data. In *STOC*, 2010. Manuscript available at arXiv.org.

D. Feldman, A. Fiat, and M. Sharir. Coresets for weighted facilities and their applications. In *Proc. 47th IEEE Ann. Symp. on Foundations of Computer Science (FOCS)*, pages 315–324, 2006. ISBN 0-7695-2720-5.

D. Feldman, A. Sugaya, and D. Rus. An effective coreset compression algorithm for large scale sensor networks. In *IPSN*, pages 257–268, 2012a.

D. Feldman, C. Sung, and D. Rus. The single pixel gps: learning big data signals from tiny coresets. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 23–32. ACM, 2012b.

D. Feldman, M. Schmidt, and C. Sohler. Turning big data into tiny data: Constant-size coresets for k-means, PCA and projective clustering. *SODA*, 2013.

Dan Feldman and Leonard J. Schulman. Data reduction for weighted and outlier-resistant clustering. In *SODA*, pages 1343–1354, 2012.

M. Ghashami and J. M. Phillips. In *Relative Errors for Deterministic Low-Rank Matrix Approximations (to appear)*, 2014.

Anna C Gilbert, Sudipto Guha, Piotr Indyk, Yannis Kotidis, S Muthukrishnan, and Martin J Strauss. Fast, small-space algorithms for approximate histogram maintenance. In *STOC*, pages 389–398. ACM, 2002.

Yogesh Girdhar and Gregory Dudek. Efficient on-line data summarization using extremum summaries. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3490–3496. IEEE, 2012.

Sudipto Guha, Nick Koudas, and Kyuseok Shim. Approximation and streaming algorithms for histogram construction problems. *ACM Transactions on Database Systems (TODS)*, 31(1):396–438, 2006.

Dorit S Hochbaum. *Approximation algorithms for NP-hard problems*. PWS Publishing Co., 1996.

Yunpeng Li, David J. Crandall, and Daniel P. Huttenlocher. Landmark classification in large-scale image collections. In *ICCV*, pages 1957–1964, 2009.

Zheng Lu and Kristen Grauman. Story-driven summarization for egocentric video. In *CVPR*, pages 2714–2721, 2013.

Rohan Paul, Daniela Rus, and Paul Newman. Visual precis generation using coresets. In *ICRA*. IEEE Press, 2014. accepted.

Karl Pearson. On lines and planes of closest fit to systems of points in space. *London, Edinburgh and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901. Sixth Series.

Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244 – 256, 1972. ISSN 0146-664X. doi: http://dx.doi.org/10.1016/S0146-664X(72)80017-0. URL `http://www.sciencedirect.com/science/article/pii/S0146664X72800170`.

W.M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, volume 2, pages 1470–1477, October 2003. URL http://www.robots.ox.ac.uk/~vgg.

Goce Trajcevski, Hu Cao, Peter Scheuermann, Ouri Wolfson, and Dennis Vaccaro. On-line data reduction and the quality of history in moving objects databases. In *MobiDE*, pages 19–26, 2006.

Bolei Zhou, Àgata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Inf. Proc. Sys.*, pages 487–495, 2014. URL http://papers.nips.cc/paper/5349-learning-deep-features-for-scene-recognition-using-places-database.