Document downloaded from:

http://hdl.handle.net/10251/38225

This paper must be cited as:

Leiva Torres, LA.; Vidal, E. (2013). Warped K-Means: An algorithm to cluster sequentially-distributed data. Information Sciences. 237:196-210. doi:10.1016/j.ins.2013.02.042.

The final publication is available at

http://dx.doi.org/10.1016/j.ins.2013.02.042

Copyright Elsevier

Warped K-Means: An Algorithm to Cluster Sequentially-Distributed Data

Luis A. Leiva*, Enrique Vidal

Institut Tecnològic d'Informàtica, Universitat Politècnica de València;
Camí de Vera, s/n – Edif. 8G Acc. B, 46022 Valencia (Spain)

Abstract

Many devices generate large amounts of data that follow some sort of sequentiality, e.g., motion sensors, e-pens, eye trackers, etc. and often these data need to be compressed for classification, storage, and/or retrieval tasks. Traditional clustering algorithms can be used for this purpose, but unfortunately they do not cope with the sequential information implicitly embedded in such data. Thus, we revisit the well-known K-means algorithm and provide a general method to properly cluster sequentially-distributed data. We present Warped K-Means (WKM), a multi-purpose partitional clustering procedure that minimizes the Sum of Squared Error criterion, while imposing a hard sequentiality constraint in the classification step. We illustrate the properties of WKM in three applications, one being the segmentation and classification of human activity. WKM outperformed five state-of-the-art clustering techniques to simplify data trajectories, achieving a recognition accuracy of near 97%, which is an improvement of around 66% over their peers. Moreover, such an improvement came with a reduction in the computational cost of more than one order of magnitude.

Keywords: Partitional clustering, sequential data, trajectory segmentation, data compression, data simplification

1. Introduction

There is a notably increasing spectrum of devices that capture dense volumes of sequentially-generated information (e.g., motion sensors, video cameras, active RFID tags, electronic pens, etc.). Therefore, a clear interest exists in new compression techniques and methods to simplify the structure of this kind of data, so that original objects can be represented by more compact structures that are better tailored for classification, storage, and retrieval tasks. This can be approached as a clustering problem, i.e., organizing such object sequences into subgroups whose members are similar in some way.

The motivation to using these simplified structures can be as elemental as reducing the number of data samples to save storage space in motion capture databases [5], to more complex applications such as detecting actions in hours of video data [25]. In the context of this work, however, we focus on the identification of elementary parts within a single data trajectory. For instance, many applications track the movement of mobile objects (e.g. vehicular traffic trajectories, GPS signals, eye-gaze movements, etc.), recording their position as it evolves over time. Here, data imprecision appears due to sampling and/or measurement errors, during pre-processing for preserving data anonymity, and so on [7]. Therefore, sequential clustering can be an interesting means to reduce these uncertainties.

When clustering sequential data[1] there exists a strong constraint, often related to time, that can be exploited to a great advantage. By ignoring this constraint, classical clustering techniques fail to cope with the underlying sequential data structure, as illustrated in Figure 2d.

*Corresponding author. Tel.: +34 963877069; Fax: +34 963877239.
Email addresses: luileito@iti.upv.es (Luis A. Leiva), evidal@iti.upv.es (Enrique Vidal)

[1]Since we are interested in sequences whose elements typically are vectors, we will use the terms *sequence*, *trajectory*, or *trace* interchangeably in the context of this paper.

As pointed out in the next section, previous work on sequential data clustering, as approached in this paper, is surprisingly scarce. While there are many works that solve sequential supervised machine learning problems (c.f. [15]), the unsupervised case still poses interesting challenges among the research community (c.f. [55]). Our contribution to the field is thus a general model for clustering sequentially-distributed data that does not require (labeled) training data or manually adjusting many control parameters; and we qualify its feasibility by means of both quantitative and qualitative experiments.

This paper is organized as follows. First, we review previous work and set our approach in context, highlighting how does it differ from existing techniques. Second, we provide a brief background on data clustering procedures. Then, we describe the basis of sequential clustering and present our approach. Third, we illustrate the capabilities of our approach by describing a series of experiments with e-pen data and eye movement simulations, followed by a formal experimentation on human action recognition using a real-world dataset. Fourth, we discuss a series of envisioned application domains where our approach suitably fits. Finally, we close the paper with the main conclusions.

2. Related Work

Clustering techniques are used extensively in diverse fields well beyond pattern recognition, such as medicine, psychology, geography, biology, or marketing. The importance and interdisciplinary nature of clustering is evident through its vast literature (c.f. [31, 32]). Cluster analysis provides an unsupervised classification scheme to efficiently organize large datasets [19]. Additionally, cluster analysis can supply a means for assessing dimensionality [3] or identifying outliers [39].

The scalability of clustering algorithms has been investigated by diverse authors, e.g., [12, 22, 50, 63], proving that clustering can be conveniently deployed even when the amount of resources is limited (e.g., in terms of memory and CPU). An extreme condition of this resource-scarced case is the clustering of data streams, where samples must be processed in chunks only once, as they arrive, without being able to access them again. This problem is well understood today, and there are several techniques to solve it such as [1, 2, 16, 26, 36, 64].

In these works, however, although data could be considered to be sequentially given, no sequentiality restrictions are imposed by the algorithms that handle the data. Therefore, the goals of these lines of research are quite different from those we are pursuing. Instead, our work is closer to a different framework which has come to be loosely referred to as "sequential clustering".

Figure 1 shows a taxonomy of works that, for one reason or another, are considered within such a sequential clustering framework, which, as it can be seen, is by no means homogeneous. In this taxonomy, our contribution appears as a general purpose, unsupervised approach, based on optimizing a well defined criterion function, which aims at clustering points of a single sequence into clusters that preserve the sequential ordering of these points. Different branches of this taxonomy are discussed below, along with a brief comparison to our proposal.

In the first branch of the taxonomy, labeled "full sequences", a number of authors have considered the problem of (conventional) clustering of data elements which are themselves full sequences of points. For instance, full human action trajectories [23] and human motion data [8], hurricane tracking data, stock rates, and animal movements [38], or other kinds of time-series data streams [10]. These works deal with issues such as how to establish a dissimilarity measure between (maybe incomplete) sequences or how to cope with the computational costs entailed by handling the complex data elements (vector sequences). Therefore, they have little relation with the approach presented here, which aims at discovering subsequences of a *single* trajectory and where the data elements are individual *points* (vectors) of the given trajectory.

Nevertheless, in some of the works of the "full sequences" branch (e.g., [38, 49]), rather than full sequences, the elements to be clustered are *subsequences*, which are determined by some trajectory segmentation technique. These segmentation techniques can themselves be considered to belong to a different branch of the taxonomy, closer to our proposal, and will be discussed later.

The second branch in the taxonomy, labeled "single sequence", does correspond to the kind of problems we are interested in; that is, sequential clustering of points belonging to a single sequence. Approaches in

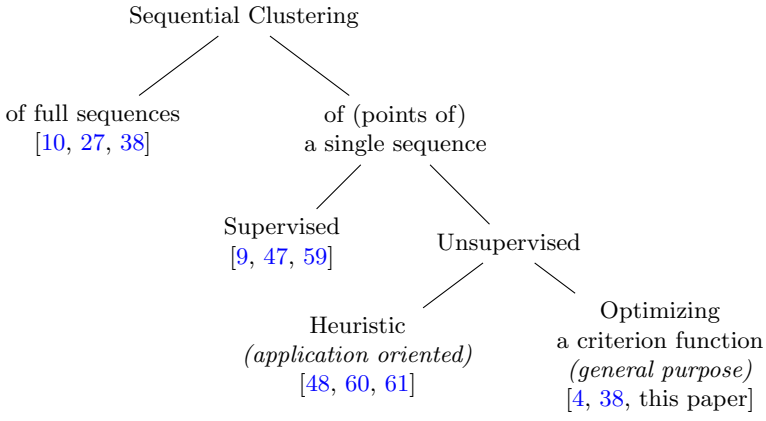


Figure 1: **Taxonomy of related work**. For the sake of brevity, we cite in this figure at most three of the references that relate to our work.

this branch can be further divided into supervised and unsupervised catgories. In the "supervised" case, different kinds of models such as Hidden Markov Models [9, 52, 57] or Neural Networks [29, 35, 59] are trained for different segments using a suitable amount of manually segmented and labeled trajectories. This clearly differs from our proposal, since we are only interested in *unsupervised* methods, which do not require the availability of (otherwise costly) training data. The "unsupervised" branch can still be further divided into heuristic techniques and general purpose methods based on optimizing a suitable criterion function.

The "heuristic" (or application oriented) branch encompasses techniques such as those presented in [48, 60, 61]. These techniques generally require an important number of input parameters to be tuned. For instance, filtering, smoothing and segmentation thresholds, window sizes and batch intervals, the specification of a expected geometry, etc. Clearly, in practice, this renders these approaches difficult to use in applications different from those which they were originally developed for.

Finally, the branch labeled "optimizing a criterion function" corresponds to *general purpose* approaches such as the the work presented in this paper and the segmentation techniques used in [4, 38, 49]. Some (or all) of these techniques can be more properly viewed from a *curve fitting* point of view, which is considered to be conceptually different from a clustering framework. Moreover, this point of view has been thoroughly studied in other works (see, e.g., [43, 51]). Most approaches in this line aim at computing piecewise approximations to the given curves and therefore we will not dwelve further into this line in the present paper. Even more, most of these curve fitting methods have been shown to be impractical for real-time applications and even computationally intractable [13].

On the other hand, in [4, 38, 49], the desired number of segments is not explicitly specified, although it is implicitly determined by the choice of internal thresholds or regularization parameters. In contrast, in our approach, the number of segments (or clusters) is explicitly specified as its (unique) input parameter, as in classical clustering algorithms. We believe this is an important feature for a general-purpose clustering algorithm. In fact, specifying the number of clusters is needed in many applications. For instance, in gesture recognition, template matching algorithms require gesture strokes to be segmented using the same number of keypoints for the whole gesture vocabulary. This is also the case in the human action recognition experiments which will be presented in this paper. In a similar vein, path similarity analysis (e.g., comparison of spatial patterns of attention of different users, search and retrieval of moving object trajectories, etc.) requires trajectories having the same number of segments.

In sum, the following four key features differentiate our approach from other trajectory clustering algorithms: simplicity, flexibility, optimization-driven procedure, and sequentiality constraints. Figure 2 illustrates the kind of results achieved with the proposed approach applied to sequential data and how it can cleanly overcome the most important drawbacks of classical clustering.

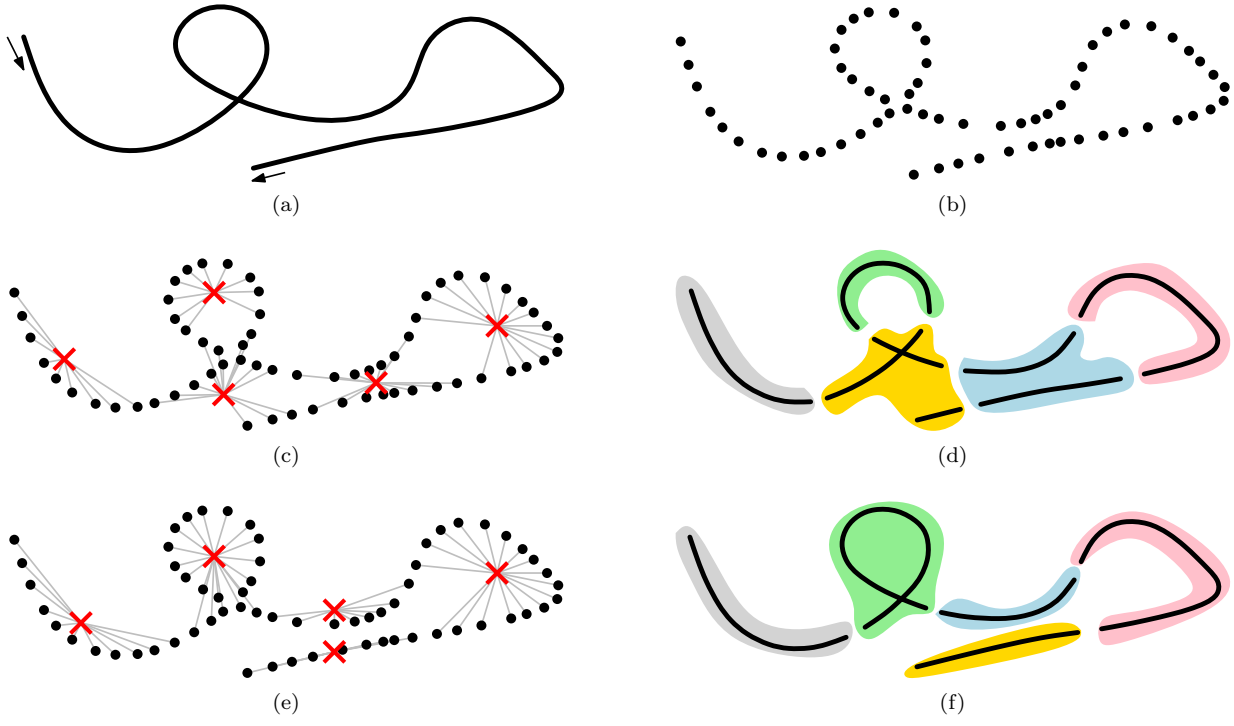


Figure 2: **A 2D example**. An arbitrary shape (2a) is digitized (2b) and reduced to 5 elemental units. Classical clustering algorithms do not deal with the temporal information and, therefore, units are ill-defined (2c), leading to an inconsistent clustering (2d). Our approach, however, provides a simple framework to easily cope with the sequentiality of data (2e, 2f).

3. Background

The fundamental data clustering problem may be defined as discovering "natural" groups or clubbing similar objects together. In this paper, data clustering is viewed as a data partitioning problem [17, 44, 62] as opposed to the hierarchical approach [24, 46, 58], since we are interested in a *partition* of the data and not in a *structure* (dendrogram) thereof.

Partitional clustering divides a dataset $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ of n d-dimensional feature vectors into a set $\prod = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ of k disjoint homogeneous classes with $1 < k \ll n$. One way to tackle this problem is to define a criterion function that measures the quality of the clustering partition and then find a partition $\prod^*$ that optimizes such a criterion function.

The most popular algorithm for partitional clustering in scientific and industrial applications is by far the K-means (or C-means) algorithm, which can be considered as a simplified case of Expectation-Maximization (EM) clustering. K-means, including its multiple variants such as Fuzzy C-Means [20], K-Medoids [34], etc., is based on the firm foundation of variance analysis. It requires the number of clusters k to be an input parameter (though there are many studies for choosing k automatically [11, 14, 21, 28, 30]), which is tightly coupled to the nature of the involved task.

The criterion function that K-means tries to minimize is the Sum of Quadratic Errors (SQE), denoted simply as J in the literature, which emphasizes the local structure of the data [56]:

$$J = \sum_{j=1}^{k} H_j, \tag{1}$$

where

$$H_j = \sum_{\boldsymbol{x} \in \mathcal{C}_j} \| \boldsymbol{x} - \boldsymbol{\mu}_j \|^2 \tag{2}$$

represents the heterogeneity (or distortion) of cluster $\mathcal{C}_j$, and

$$\boldsymbol{\mu}_j = \frac{1}{n_j} \sum_{\boldsymbol{x} \in \mathcal{C}_j} \boldsymbol{x} \tag{3}$$

is the cluster mean, with $n_j = |\mathcal{C}_j|$ being the number of samples in cluster j.

The K-means algorithm is known for its simplicity, relative robustness, and fast convergence to local minima. The most common version of this algorithm, generally attributed to Lloyd [42], uses a heuristic minimum distance criterion, where in each iteration all the samples are assigned to their closest cluster means and convergence is achieved when the assignments no longer change. There exists, however, a more interesting version, often attributed to Duda and Hart [18, 19], which uses a sample-by-sample iterative optimization refinement scheme. At each step, the SQE is evaluated and the considered sample is reallocated to a different cluster if and only if that reassignment decreases J. Clearly, such a greedy optimization guarantees that the resulting partition corresponds to a local minimum of the SQE. In this paper we follow this approach. The variation in the SQE produced when moving a sample $\boldsymbol{x}$ from cluster j to cluster l can be obtained in a single computational step as [19]:

$$\Delta J(\boldsymbol{x}, j, l) = \frac{n_l}{n_l + 1} \| \boldsymbol{x} - \boldsymbol{\mu}_l \|^2 - \frac{n_j}{n_j - 1} \| \boldsymbol{x} - \boldsymbol{\mu}_j \|^2 \tag{4}$$

If this increment is negative, the new means, $\boldsymbol{\mu}'_j$, $\boldsymbol{\mu}'_l$ and the SQE, J', can then be incrementally computed as follows [19]:

$$\begin{aligned} \boldsymbol{\mu}'_j &= \boldsymbol{\mu}_j - \frac{\boldsymbol{x} - \boldsymbol{\mu}_j}{n_j - 1} \\ \boldsymbol{\mu}'_l &= \boldsymbol{\mu}_l + \frac{\boldsymbol{x} - \boldsymbol{\mu}_l}{n_l + 1} \\ J' &= J + \Delta J(\boldsymbol{x}, j, l) \end{aligned} \tag{5}$$

4. Sequential Clustering

If the data in a dataset X are sequentially distributed, it can be said that such data describe a *trace* or *trajectory* in the d-dimensional vector space where samples are represented:

$$X = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \tag{6}$$

We define a sequential clustering into k classes as the mapping

$$\mathrm{b} : \{1, \ldots, k\} \mapsto \{1, \ldots, n\}$$

where b_j is the (left) *boundary* of cluster j; i.e., the index of the first sample in this cluster. See Figure 3 for a graphical example.

Using this convenient notation, the j-th (sequential) cluster of X can be written as follows:

$$\mathcal{C}_j = \{\boldsymbol{x}_{\mathrm{b}_j}, \ldots, \boldsymbol{x}_{\mathrm{b}_j + n_j - 1}\} \tag{7}$$

where n_j can now be trivially computed as:

$$n_j = \mathrm{b}_{j+1} - \mathrm{b}_j \tag{8}$$

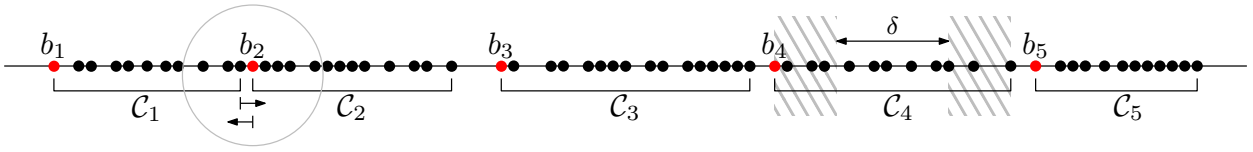


Figure 3: **Overview of WKM basis**. A sample $\boldsymbol{x} \in \mathcal{C}_j$ is only allowed to move either to cluster $\mathcal{C}_{j-1}$ or $\mathcal{C}_{j+1}$. If a move proves advantageous, that is, the increment in SQE is negative, the sample is reallocated and the two cluster means involved in such a reallocation are incrementally recomputed according to Eq. (5). Otherwise, the next cluster is inspected, in order to preserve the clustering sequentiality. The algorithm provides an optional parameter δ which specifies the maximum amount of samples that will be inspected in each iteration. This way, if $\delta = 1$ only two samples are considered per iteration, while if $\delta = 0$ full search is carried out.

This way, equations (2) and (3) can be rewritten as

$$H_j = \sum_{i=\mathrm{b}_j}^{\mathrm{b}_{j+1}-1} \| \boldsymbol{x}_i - \boldsymbol{\mu}_j \|^2 \tag{9}$$

$$\boldsymbol{\mu}_j = \frac{1}{n_j} \sum_{i=\mathrm{b}_j}^{\mathrm{b}_{j+1}-1} \boldsymbol{x}_i \tag{10}$$

so that (4) and (5) can be directly used as such in this new formulation.

5. Warped K-Means

To begin, we capitalize in the sequentiality of the data and use the trace segmentation (TS) algorithm [37] to produce a suitable initial partition. TS has been employed by many authors as a preprocessing technique for speech and on-line handwriting recognition (e.g., [41, 53, 54]). TS creates a resampled sequence, or "trace", by using piecewise linear interpolation based on distances accumulated along the original trace, which results in a non-linearly distributed initial partition. This technique is described in Algorithm 1.

Then, we iterate over data points using a K-means-like optimization procedure, described in Algorithm 2 and referred to as *Warped K-Means* (WKM). This name is inspired by the idea that the original tajectory is delusively distorted, or "unfolded" (see Figure 3).

As in classical K-means, WKM reallocates samples based on the analysis of effects on the objective function J, caused by moving a sample from its current cluster to a potentially better one. But now a hard sequentiality constraint is imposed. The first half of samples in cluster j are only allowed to move to cluster $j-1$, and, respectively, the last half of samples are only allowed to move to cluster $j+1$. A sample will be reallocated if and only if the corresponding SQE increment is beneficial (i.e., negative). This process is iterated until no transfers are performed.

Because of this constraint, along with the sequential ordering of samples within each cluster, typically only the samples close to the cluster boundaries get reallocated. To take advantage of this observation, we introduce an optional parameter $\delta \in [0, 1]$ which allows us to fine-tune the WKM behavior and at the same time achieve further reductions in the computational cost. It allows testing only those samples that are more or less close to cluster boundaries. In the extreme case of $\delta = 0$ the algorithm is conservative: *all* samples in a cluster are visited to see if they should be reallocated. In the other extreme, if $\delta = 1$ WKM is optimistic: only the boundary and the last sample in each cluster will be checked. In general, the effect of δ is illustrated in Figure 3.

In each cluster j, the samples close to its boundary b_j are visited first to see if they can be advantageously reallocated to the previous cluster, $j-1$ ("reallocate backwards" loop); then the samples close to the boundary of the next cluster, $j+1$, are similarly considered ("reallocate forwards" loop). It is worth

TS Boundary Initialization Algorithm

Input: Trajectory $X = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$; No. Clusters $k \geq 2$
Output: Boundaries $\mathrm{b}_1, \ldots, \mathrm{b}_k$

$L_1 = 0$
for $i = 2$ **to** n **do** /* accumulated trace length */
  $L_i = L_{i-1} + \| \boldsymbol{x}_i - \boldsymbol{x}_{i-1} \|$
$\lambda = \frac{L_n}{k}$ /* segment length */
$i = 1$
for $j = 1$ **to** k **do**
  while $\lambda\ (j-1) > L_i$ **do** /* interpolate */
    $i{+}{+}$
  $\mathrm{b}_j = i$ /* define boundaries */

Algorithm 1: **Boundaries initialization**. Each boundary is evenly allocated according to a piecewise linear interpolation on accumulated distances, resulting in a non-linearly distributed boundary allocation.

WKM Algorithm

Input: Trajectory X; No. Clusters $k \geq 2$ [; Proportion $\delta = 0.0$]
Output: Boundaries $\mathrm{b}_1, \ldots, \mathrm{b}_k$; Centroids $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$; Distortion J

Initialize boundaries $\mathrm{b}_1, \ldots, \mathrm{b}_k$ /* use TS (Algorithm 1) */
for $j = 1$ **to** k **do**
  Compute $\boldsymbol{\mu}_j$, n_j, J /* use Eq. (1), (8), (9), and (10) */
repeat
  $transfers = false$
  for $j = 1$ **to** k **do**
    if $j > 1$ **then** /* reallocate backward 1st half */
      $first = \mathrm{b}_j$; $last = first + \lfloor \frac{n_j}{2}(1-\delta) \rfloor$
      for $i = first$ **up to** $last$ **do**
        if $n_j > 1$ **and** $\Delta J(\boldsymbol{x}_i, j, j-1) < 0$ **then**
          $transfers = true$
          $\mathrm{b}_j \mathrel{+}= 1$; $n_j \mathrel{-}= 1$; $n_{j-1} \mathrel{+}= 1$
          Update μ_j, μ_{j-1}, J /* according to Eq. (5) */
        else break /* SQE did not improve */
    if $j < k$ **then** /* reallocate forward 2nd half */
      $last = \mathrm{b}_{j+1} - 1$; $first = last - \lfloor \frac{n_j}{2}(1-\delta) \rfloor$
      for $i = last$ **down to** $first$ **do**
        if $n_j > 1$ **and** $\Delta J(\boldsymbol{x}_i, j, j+1) < 0$ **then**
          $transfers = true$
          $\mathrm{b}_{j+1} \mathrel{-}= 1$; $n_j \mathrel{-}= 1$; $n_{j+1} \mathrel{+}= 1$
          Update μ_j, μ_{j+1}, J /* according to Eq. (5) */
        else break /* SQE did not improve */
until $\neg transfers$

Algorithm 2: **Warped K-Means**.

noting that, with $\delta < 1$, the proportion of samples processed in each backward or forward sequential chunk is typically less than the number corresponding to the given value of δ. This is because the reallocation process is aborted as soon as the SQE does not improve for that chunk, in order to preserve the sequentiality of our clustering procedure.

Note also that if some samples are reallocated during the forward processing of cluster j, then we do not need to re-check them in the backward processing of cluster $j+1$. This is easily verifiable with an auxiliary variable that stores the index of the last reallocated sample. This detail, however, is not shown in the previous pseudo-code for the sake of clarity.

The computational cost of a complete iteration of WKM over the whole sequence X depends on the number of samples n, the sample vector dimension d, and the number of clusters k. As previously discussed, it can also depend on the value of δ. On the one hand, if $\delta = 1$, the complexity of WKM is reduced to $\Theta(kd)$ per iteration, in comparison to $\Theta(nkd)$ in the case of classical K-means. On the other hand, if $\delta = 0$, the best- and worst-case complexities are $\Omega(kd)$ and $O(nd)$, respectively. Therefore, for all the values of δ and in all cases, each iteration of WKM is expected to be (much) faster than conventional K-means algorithms. Moreover, according to empirical observations, the convergence tends to require significantly less iterations than such K-means algorithms.

Overall, the main advantages of our proposal can be summarized as follows:

- **Consistent results**: It always guarantees the convergence to a good local minimum, i.e., a low distorted partition of the original dataset that preserves sequence ordering.
- **Robust solution**: Each run for a given k always yields the same clustering configuration — thanks to the initialization algorithm and the minimization criterion for sample reallocation.
- **Low computational cost**: Much lower than that of classical clustering algorithms since, instead of the usual all-against-all search strategy, WKM only needs to check two clusters in each step; which results in a really fast convergence.
- **No extra mandatory parameters**: Our solution requires the same input data and same input parameters as in K-means, though an optional δ parameter can be specified to tune both the algorithm behavior and its cost.

As we shall demonstrate in Section 7, the WKM algorithm is also suitable for online learning tasks over large datasets, due to the following facts: *1)* the computational cost of updating the centroids is independent of the number of samples and *2)* the final partition can be updated while new samples arrive without affecting too much the previous data structure [40], in terms of clustering configuration (centroids and boundaries).

6. Illustrative Experiments with E-pen Data

A first series of tests were carried out to illustrate the capabilities of WKM, in comparison to classical clustering. In these experiments, electronic pen trajectories corresponding to *continuous* handwriting (i.e., without lifting the pen from the tablet) of some isolated words were captured. Then, a classical K-means (*Lloyd* version) and WKM were both applied to these trajectories with the aim of finding homogeneous clusters that would roughly correspond to different natural parts of the written words.

Figure 4 shows the results of two of these tests. As expected, classical K-means fails to provide meaningful results, while, for the proper values of k, WKM quite accurately discovers the underlying morphological units (letters or letter groups). Similar results are obtained with many other examples, not shown here for the sake of brevity[2]. What is more, these results illustrate possible applications of WKM in the field of on-line handwritten text recognition, as we shall comment later in Section 9.

7. Online Clustering Experiments

Here we test WKM in a simulated incremental setting, where samples must be sequentially processed. As such, we built an application that traverses trajectory data samples from beginning to end, updating the

[2]Web-based interactive prototype available at http://personales.upv.es/luileito/wkm/.

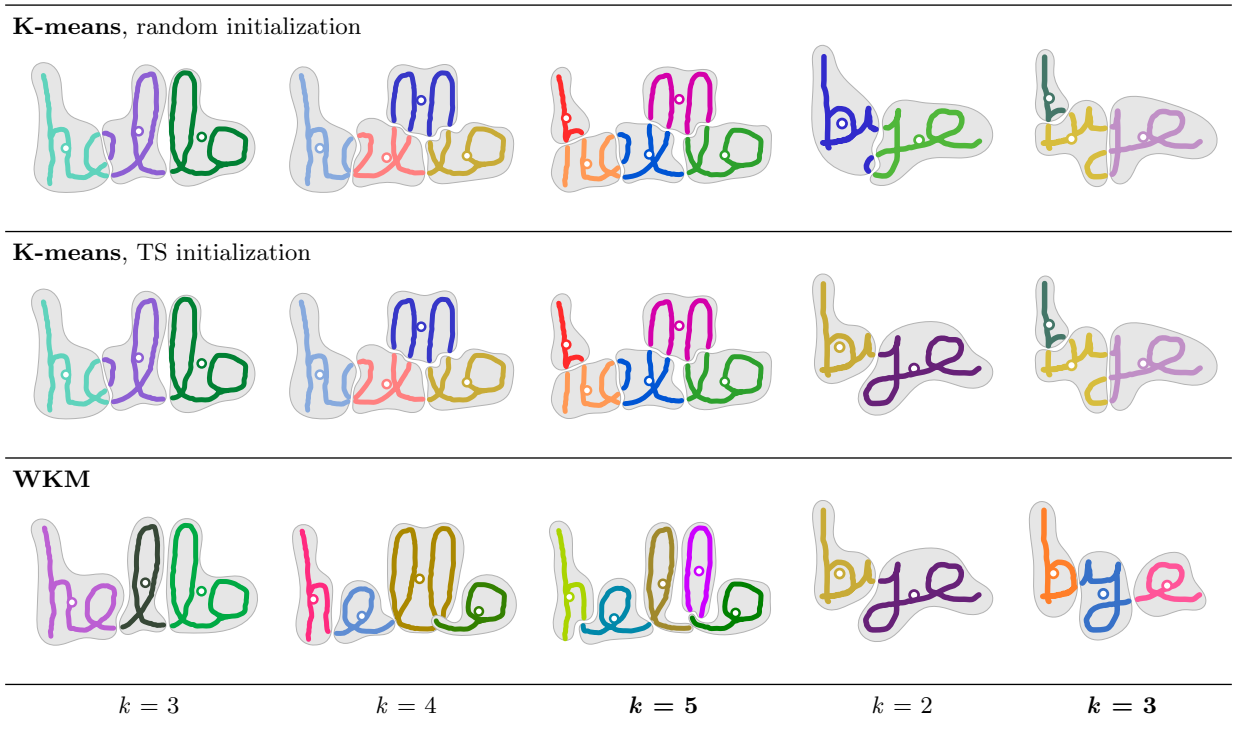


Figure 4: **E-pen data experiments**. Two single pen strokes corresponding to the words "hello" and "bye" are segmented into different number of clusters using classical K-means (*Lloyd* version) and the proposed WKM algorithm. By preserving sequentiality, only WKM succeeds to discover the underlying morphological units (i.e., letters or letter groups).

clustering configuration as the samples are processed. This setting can therefore be seen as using a sliding window with infinite memory (window size), since the application processes all previously seen samples after each window displacement.

7.1. Method

We simulated the calibration procedure of an eye-tracker, manually drawing 2D trajectories with a computer mouse. Such a calibration procedure consists in asking the user to look at one calibration circle at a time, using typically a grid of 9 circles. Therefore, these experiments comprised identifying nine areas, where eye fixations have been produced. In all experiments, we used one of these generated trajectories that had 3000 points.

7.2. Procedure

For these experiments, given that points are processed as they arrive, instead of providing the desired number of final k segments as an input parameter to the application, we set it to be the desired average number of points per cluster, which was 330 points for $k = 9$ in this simulation. This way, the number of segments would sequentially increase from 1 to 9 according to the number of processed samples so far. Also, in order to adapt WKM to this online scenario, we set the window displacement to be a configurable parameter for the application, expressed in number of points.

7.3. Results

Figure 5 shows a graphical summary of these experiments using a window displacement of 200 points. In the figure, the centroid color opacity is proportional to the time it was computed, i.e., darker colors represent

the most recent results. As can be noticed, centroids are indeed placed inside the calibration circles when all points are processed. The accompanying lines in the figure indicate the variation of each centroid position through time. We considered a centroid to be stable if after some point in time (and until the end of the experiment) the aforementioned variation of its position with respect to its final position was less than 10 px.

Together with Figure 7, it can be seen that centroids tend to stabilize quickly, after a number of samples have been processed. Then, the variation of centroid positions is as low as 2 px on average. Such a variation of centroid positions was measured as the Euclidean distance between the current position of the centroids and their final position. Given that each calibration circle had a diameter of 100 px, we can assert that such a variation of centroid positions represents an error of roughly 2% on average after stabilization. Table 1 summarizes these experiments for different window displacements. Overall, in addition to these low variations of centroid positions, we observed that the smaller the window displacement, the sooner the centroids stabilize.

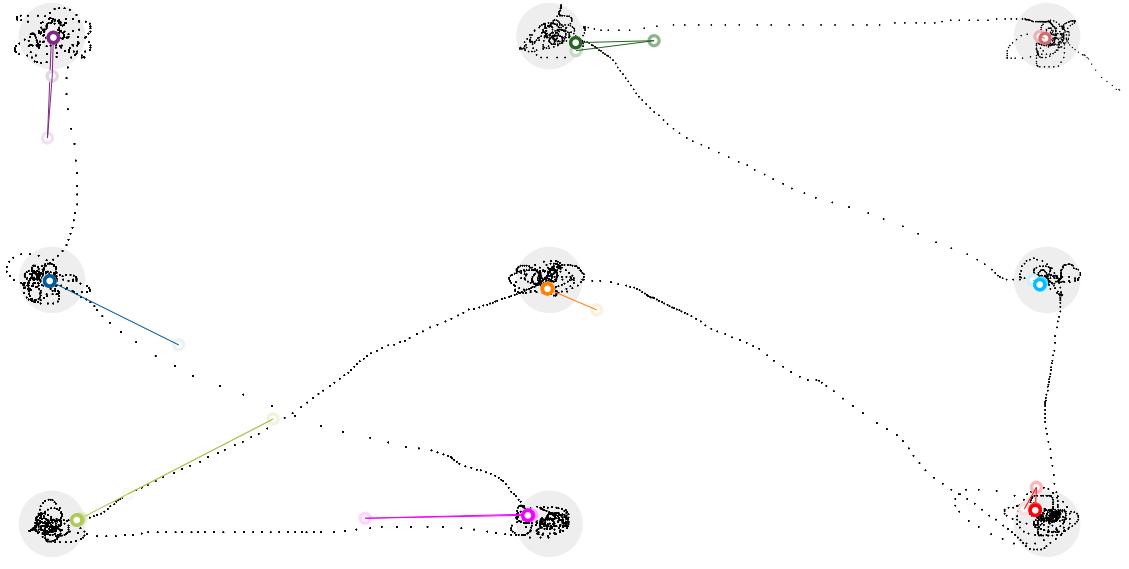
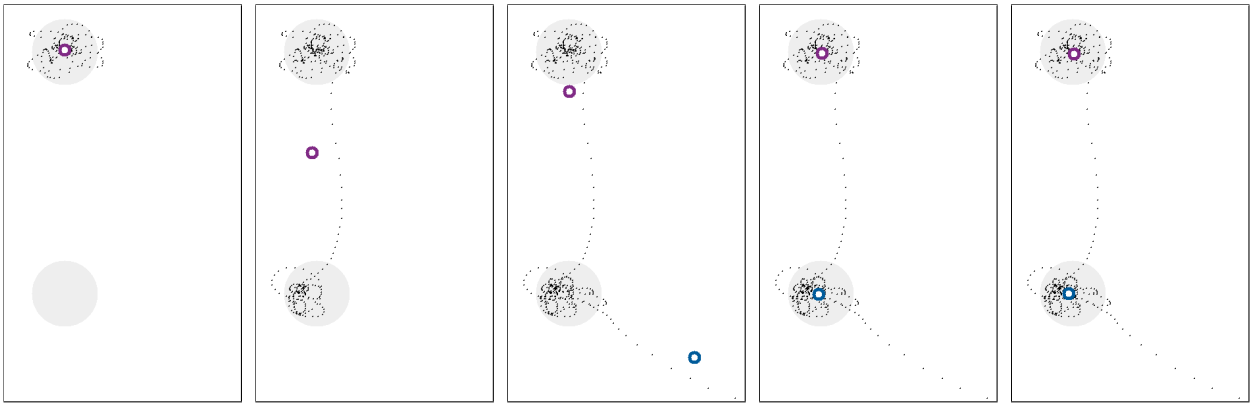


Figure 5: **Online clustering scenario**. We simulated the calibration procedure of an eye-tracker. Sample coordinates were processed in an incremental fashion, as they arrived. Color circles represent the centroids. Lines indicate the (temporal) variation of centroid positions.

Table 1: **Online clustering results**. We experimented with different window displacements. We report here the mean (and SD) number of points until stabilization, as well as the variation of centroid positions after such stabilization, measured in px.

Window displacement	**# Points until stabilization**	**Centroid variation after stabilization**
50	283.33 (136.9)	1.85 (1.6)
100	377.78 (109.3)	1.97 (1.6)
200	517.11 (165.0)	1.89 (1.6)
400	761.56 (115.3)	2.21 (1.6)
800	1470.25 (179.1)	1.99 (1.1)



(a) $7\%, k = 1$ (b) $14\%, k = 1$ (c) $21\%, k = 2$ (d) $28\%, k = 3$ (e) $35\%, k = 3$

Figure 6: **Detailed view of top-left corner of Figure 5**. Centroids stabilize as soon as the number of processed points increases, since the number of segments k to cluster points into also increases (c.f., 6c, 6d). The percentage indicates the number of processed points with respect to the final trajectory length.

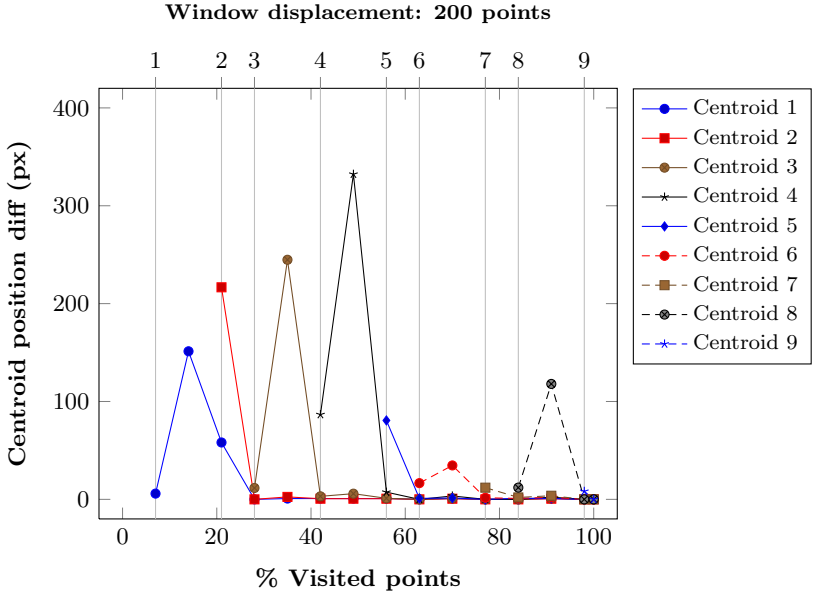


Figure 7: **Evolution of centroids**. Minor grid lines denote an update in the number of clusters being considered up to the percentage of visited points indicated by the horizontal axis. The vertical axis denotes the variation of centroid positions regarding the final clustering configuration.

Aggarwal et al. showed that, in an incremental setting, traditional clustering algorithms produce clusters of poor quality when the data evolves over time [2]. This experimentation has demonstrated that, in contrast, WKM can update the clustering configuration while new samples arrive, without affecting too much the previous results. In addition, these experiments suggest that WKM could be adapted for a pure streaming-only scenario. For instance, the application could store periodically sufficient statistics as a "history" or summary of the data samples, and then operate over them. This extension to WKM, however, is left as an opportunity for future work.

8. Human Action Recognition Experiments

A classification task was chosen to test our algorithm more formally. We used the Localization Data for Person Activity dataset [33] form the UCI Machine Learning Repository [6]. In this corpus, each human action is essentially represented as a time series of `x,y,z` coordinates of different body parts of the person performing the action.

We chose this corpus to show the capabilities of WKM as a simple and accurate classifier for a complex, real-world task. To this end, each human action is represented as a vector of a fixed number of "elementary actions", where each elementary action is, in turn, a cluster mean vector obtained by clustering the original sequence of action samples (`x,y,z` coordinates) into a fixed number of clusters. Once each action is represented as a fixed dimension vector, many simple classifiers can be adequately used, among which we chose the well-known Nearest-Neighbor (NN) classifier.

8.1. Corpus

The selected dataset is a sequential, time-series corpus which contains recordings of five people performing 11 different (labeled) actions; e.g., walking, lying, sitting on the ground, etc. While performing the same scenario up to five times, each person wore four sensors (active RFID tags) placed at both ankles, the belt, and the chest. Overall, a set of 164860 instances were captured. Each instance is a localization data for one of the tags, having 8 attributes: person label, tag ID, timestamp, formatted date, x-coord, y-coord, z-coord, and type of activity. More details of the dataset can be found in [33].

8.2. Methodology

In order to characterize the activities, we merged the `x,y,z` coordinates of all sensors into a single 12-dimensional feature vector sample $\boldsymbol{x} = (x_1, y_1, z_1, \ldots, x_4, y_4, z_4)^T$. This way, a trajectory was defined as the sequence $X = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, where n is the number of samples in X.

Unfortunately the dataset did not include the same number of instances per sensor. Therefore some of the composed trajectories had extremely different number of 12-dimensional vectors (e.g., some had just two vectors and others had more than 800). We needed thus to build a more comparable dataset; so, while composing each trajectory we verified that it had at least ten samples.

Eventually we obtained 125 trajectories of 162 samples on average (SD=138.6), belonging to one of the following 5 classes: 'falling', 'lying', 'on-all-fours', 'sitting', and 'walking'. There were 25 trajectories per class.

8.2.1. Vector Representation of Action Data

We ran our WKM implementation to cluster each trajectory into a variable number of segments $k \in \{2, 4, \ldots, 20\}$ and with different values of $\delta \in \{0, 0.2, \ldots, 1\}$. The cluster means obtained by k-clustering each action data sequence were stacked into a $3 \cdot 4 \cdot k$ dimensional feature vector, i.e., a $12\,k$-dimensional vector. For those trajectories with less samples than the desired number of segments, (i.e., when $k > n$) we used singleton clusters instead (i.e. $k = n$) and the missing dimensions were filled with zeros. As we will see below, this fact had clear repercussions when classifying some trajectories with $k > 10$ (which was the minimum number of vectors in all trajectories), specially in terms of classification error.

8.2.2. Nearest Neighbor Classifier

The simple and well-known 1-NN classifier with Euclidean distance was adopted to classify vector-represented action trajectories. Each class was represented by a number of prototype trajectories and each test trajectory was classified into the class of its nearest neighbor prototype.

In these experiments, we employed the C++ ANN library [45] for NN searching, with its basic, exact search option. Given the relatively small number of available trajectories overall, we adopted the leaving-one-out training and testing procedure.

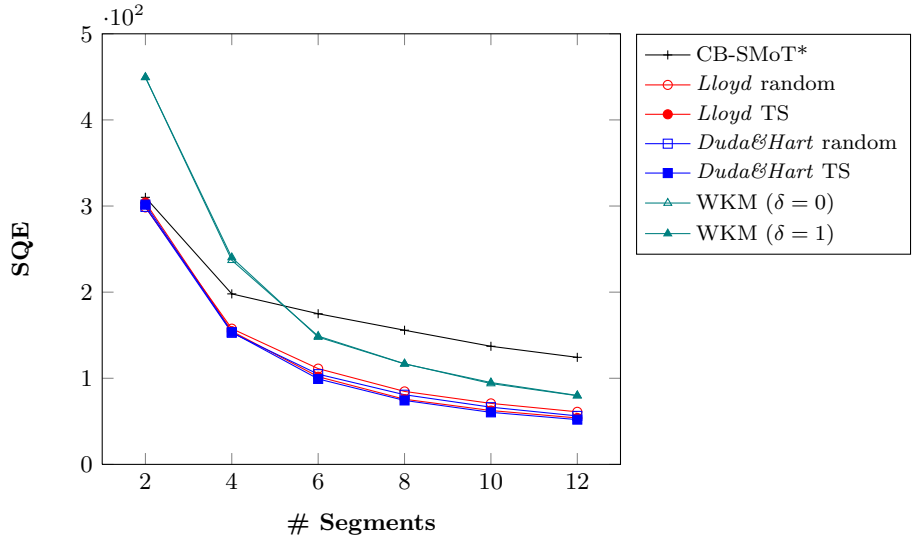


Figure 8: **Squared error comparison**. Each value is averaged for all trajectories (activity × person × trial). As expected, the segmentations achieved by WKM have higher distortion than those of classical K-means, since the former imposes a strong sequential restriction while the latter does not.

8.2.3. Methods Used for Comparison

We compared WKM with the two well-known versions of K-means: the greedy *Duda&Hart*'s algorithm [19] and the popular *Lloyd*'s version [42], using both random and TS initializations. When initializing randomly we performed up to 5 times each experiment, in order to mitigate the effects of chance, and computed the average values.

We also compared our approach with another method previously proposed for trajectory segmentation based on clustering. Among many possibilities, we chose the well-known CB-SMoT technique, which identifies stop and moves in a trajectory, being stops the important parts [48]. Because of the reasons given in Section 2, other methods proved difficult or impossible to use in the human action recognition task considered here. We implemented a modified version of CB-SMoT (referred to as CB-SMoT* from here onwards) in which the (many) original input parameters, needed to provide semantic trajectory information, were reduced to a single parameter: the distance threshold used to calculate the neighborhood of a point. This parameter was chosen in such a way that the algorithm returned a set of k stops. This way, all methods being compared took exactly the same input parameters and returned the same kind of outcome.

8.3. Results

The first experiment was aimed at studying the behavior of the different algorithms in terms of distortion (SQE) for an increasing number of clusters. Results are shown in Figure 8. As expected, the SQE decreases monotonically with increasing number of clusters. It is interesting to note that K-means algorithms achieve a (slightly) lower SQE than WKM, which is explained by the lack of sequentiality restrictions, that otherwise WKM imposes on the data.

In the next experiment we studied the ability of all techniques being compared to behave as preprocessors, in order to obtain vector-represented trajectories for classification purposes. We considered the case when a trajectory is segmented into just one single cluster ($k = 1$) as the baseline; that is, each trajectory is represented by a 12-dimensional vector corresponding to the average of all its trajectory samples. In that case the classification error was as low as 9.6%, which is reasonable given the nature of the activities involved (e.g., the position of "lying" and "sitting" should differ greatly at least in the average $\mathbf{z}$ coordinate of each sensor.)

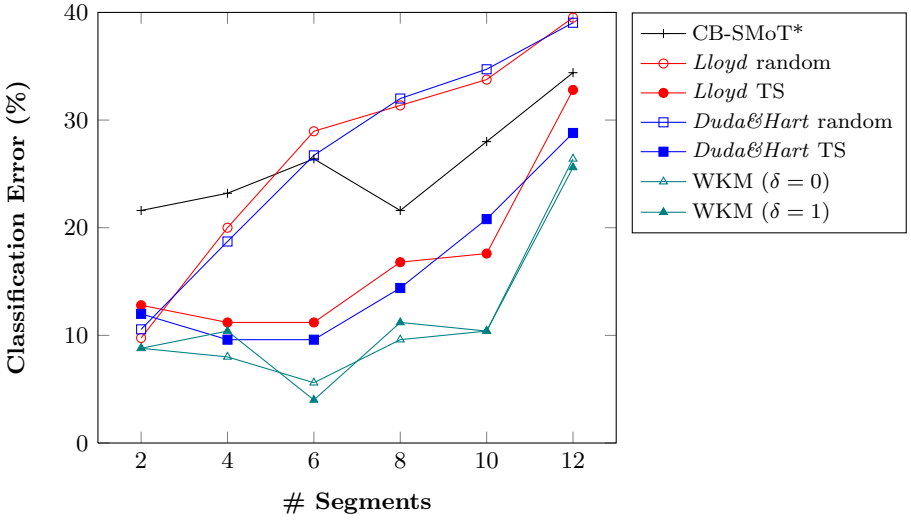


Figure 9: **Classification error comparison**. We tested CB-SMoT*, an adaptation of a comparable technique, and performed variations to three clustering alternatives: *Duda&Hart*'s algorithm and *Lloyd*'s version, using both random initialization and trace segmentation, and the WKM algorithm using two extreme δ parameters.

Results for other values of k are shown in Figure 9. As expected, certain segmentations performed better than others for each algorithm, but a particularly adequate number of elementary actions seems to be 6 in most cases. Interestingly, WKM is the method that better puts this fact forward. We observed that accuracy degraded noticeably for $k > 10$, to the point that for $k = 20$ error rates were above 50% for all classifiers — for the reason explained in Section 8.2.1. In addition, the randomly initialized versions of K-means performed worse than when they are initialized with TS.

In order to better understand the impact of the optional δ parameter of WKM, we repeated the previous experiment for different values of δ. Figure 10 shows the influence of δ in the recognition accuracy. We observed that by tuning this parameter WKM results can be further improved for a given number of segments. All in all, the best achieved result is an error rate of 3.2%, which corresponds to six elementary actions and $\delta = 0.8$.

Finally, regarding the computational cost of each algorithm, as shown in Figure 11, WKM behaves much better than its peers. It should be noted that CB-SMoT* does not optimize a criterion function, and therefore its computational cost was dropped from this analysis.

Table 2 summarizes the results discussed so far. The randomly-initialized K-means algorithms do not help overcoming the trivial baseline (just one cluster). For CB-SMoT*, the best classification error was achieved for $k = 8$, although it is higher that the rest of the methods considered. In contrast, WKM achieved a recognition accuracy of 96.8% (4 errors in 125 trials), which represents around 66% of improvement over the best result among their peers. Most interestingly, this improvement is achieved along a huge computational cost reduction (more than one order of magnitude) with respect to the K-means algorithms. We conclude that WKM was the best performer among its peers, and that results confirmed our expectations.

8.4. Discussion

One reason why using a cluster representation may have advantages over working with the original data is the evident size reduction, which in turn may enhance the ease of storage, transmission, analysis, and indexing. Moreover, extending this notion to the analysis of trajectories reverts in another significant advantage: having a good and compact representation of a data sequence makes it more invariant to noise or distortions in such data.

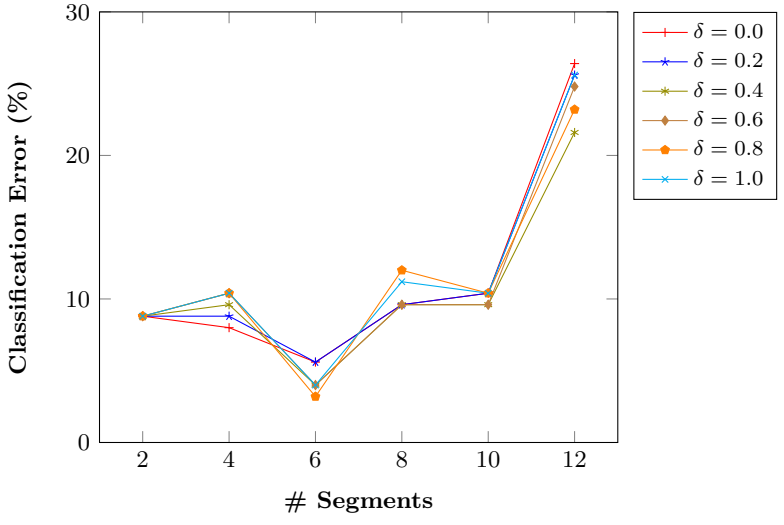


Figure 10: **WKM classification error**. We tried different values of the δ parameter for each tested number of segments. The best accuracy was achieved when using $k = 6$ in all cases.

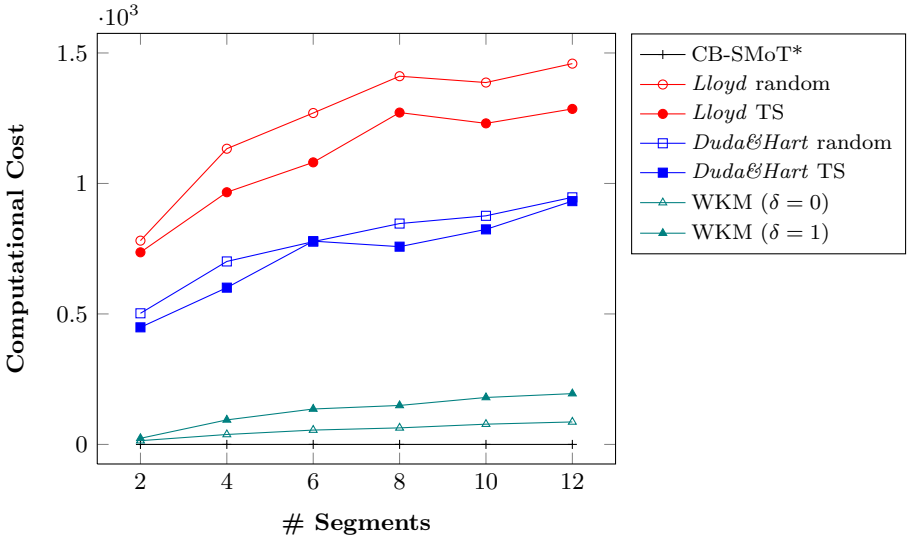


Figure 11: **Computational cost comparison**. Cost is estimated as number of times Eq. (4) is computed. For *Lloyd* versions, cost was computed as the number of times the algorithm tested if cluster means did change. For CB-SMoT*, we assumed it has zero cost, since it does not optimize a criterion function.

Table 2: **Summary of results**. Bold value indicates that it is the best result among all of the methods being compared.

Algorithm	**Best k**	**% Error**	**SQE**	**Cost**
Baseline	1	9.6	629.4	—
CB-SMoT*	8	21.6	155.7	—
Lloyd random	2	9.8	298.3	796.6
Lloyd TS	6	11.2	101.8	1080.8
Duda&Hart random	2	10.6	299.0	502.4
Duda&Hart TS	6	9.6	**99.2**	778.9
WKM $\delta = 0.0$	6	5.6	148.9	**54.9**
WKM $\delta = 0.8$	6	**3.2**	147.6	71.3
WKM $\delta = 1.0$	6	4.0	147.8	135.7

As can be observed in the figures, WKM gives very competitive error rates at a low computational cost. Since CB-SMoT* does not optimize a criterion function, it was found to be less accurate for classification tasks, with statistically significant differences at the $\alpha = 0.05$ level. In addition, all WKM results performed better than the baseline technique and all other algorithms being compared for such classification tasks.

Differences between all methods being compared were found to be statistically significant in terms of classification error, SQE, and cost [$\chi^2_{(7,N=125)}$ tests, $p < .0001$]. Two-tailed pairwise t test comparisons (using Bonferroni correction) of WKM against the comparable clustering alternatives revealed that WKM provides a significantly lower error rate [$p < .001$] and a significantly lower computational cost [$p < .0001$]. Therefore, these results show that WKM is an interesting approach for reducing both the classification error and the computational cost regarding to using other comparable alternatives when clustering sequential data.

It is worth pointing out that the algorithms initialized with TS allow to find the "natural" number of segments. However, as shown in Figure 9, for WKM this number in turn corresponds to the lowest classification error rate in all cases (see Table 2). As stated in Section 5, a critical step for (adequately) clustering sequential data with WKM is the initialization of segment boundaries. We used the TS technique, although other algorithms that ensure a sequential distribution may be also helpful. For instance, we could use an equispaced boundary initialization instead.

Additionally, we have shown that WKM ensures monotonic improvement and finite assignments in a sequential fashion, which translates to convergence to a good local minimum in which trajectory segments are well-defined. This can be leveraged in some interesting applications, as we shall comment as follows.

9. Envisioned Application Domains

We depict here some usage scenarios where our approach suitably fits. One may note that the applications described below are generic enough to ensure a broad generalization scope.

1. **Motion Segmentation**. The storage and transmission of motion tracking content is a problem due to their tremendous size and the noise due to imperfections in the capture process. Thus, one could use WKM for a more compact representation of these (large) data, as we have demonstrated experimentally in Section 8.

2. **Eye/Mouse Tracking**. Identification of fixations and saccades in eye movements is an essential part in the analysis of visual behavior. In the same way, identifying their equivalents in pointing devices (e.g., mice, styli, or touchscreens) are key issues in many research fields such as usability evaluation or interaction design. Applications in these domains often need to analyze in detail representative subsets of the tracking

data, which are then arranged into sequences. We feel that here WKM can be of great help, as it has been shown in Section 7.

3. **Online Handwriting**. Today, many pen-based computers and related devices have enabled a natural way of writing by storing (and later analyzing) the strokes of "digital ink". As illustrated in Section 6, the obtained (well-formed) segments capture pen-stroke or even full-character regularities which can be advantageously exploited by existing handwritten recognition approaches to increase their accuracy.

4. **Video Retrieval**. Many research on content-based video retrieval represents the content of videos as a set of frames, leaving out time-based features. However, for some domains, including non-linear edition and video surveillance, it may be helpful to cluster temporal features from the video for later indexing. Other applications in this field comprise automatic key frame extraction or motion behavior classification.

5. **Time Series Analysis**. In general, any discipline that deals with ordered data sequences can benefit from our approach. For instance, identify stock events through time in economics, discover places of interest in geolocation applications, or extract meaningful changes on the orography of continents in geology.

10. Conclusion

We have introduced WKM, a novel revisitation of K-means clustering for grouping sequentially-distributed data. It guarantees the convergence to a (local) minimum distortion configuration, in terms of the SQE criterion, while preserving the sequentiality of the original trajectory. We have demonstrated its feasibility and robustness through a series of experiments. An important advantage of WKM is that it exhibits a very low computational complexity, which results in a really fast convergence. Furthermore, since the algorithm imposes a hard sequential constraint, segments can be updated while new samples arrive without affecting too much the previous clustering configuration.

The WKM algorithm requires the same user input as in classical partitional clustering algorithms. In addition, WKM provides an optional tuning δ parameter that allows to control the number of samples that will be tested between iterations. From the experiments we can conclude that WKM behaves considerably well. It achieves very competitive results for action recognition and subtrajectory identification, better than other comparable techniques.

We have depicted a wide range of possible scenarios where this approach could be used, and how research communities could benefit from it. Finally, we feel that this work may encourage researchers to apply WKM to a wealth of new problems and/or domains.

Acknowledgments

This work has been partially supported by Casmacat (FP7-ICT-2011-7, project 287576), tranScriptorium (FP7-ICT-2011-9, project 600707), STraDA (MINECO, TIN2012-37475-C02-01), and ALMPR (GVA, Prometeo/2009/014) projects.

References

[1] Ackermann, M. R., Märtens, M., Raupach, C., Swierkot, K., Lammersen, C., Sohler, C., 2012. StreamKM++: A clustering algorithm for data streams. ACM Journal of Experimental Algorithmics 17 (1), 1–30.

[2] Aggarwal, C. C., Han, J., Wang, J., Yu, P. S., 2003. A framework for clustering evolving data streams. In: Proceedings of Very Large Databases (VLDB). pp. 81–92.

[3] Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P., 1998. Automatic subspace clustering of high dimensional data for data mining applications. In: Proceedings of the 1998 ACM SIGMOD international conference on Management of data. pp. 94–105.

[4] Anagnostopoulos, A., Vlachos, M., Hadjieleftheriou, M., 2006. Global distance-based segmentation of trajectories. In: Proceedings of the 12th International Conference on Knowledge Discovery and Data Mining (KDD). pp. 34–43.

[5] Arikan, O., 2006. Compression of motion capture databases. ACM Transactions on Graphics 25 (3), 890–897.

[6] Asuncion, A., Newman, D. J., 2007. UCI machine learning repository. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[7] Athavale, S., Sao, N., 2002. Data mining on moving object trajectories. International Journal of Computer Science & Engineering Technology 2 (4), 1040–1042.

[8] Barbic, J., Safonova, A., Pan, J.-Y., Faloutsos, C., Hodgins, J. K., Pollard, N. S., 2004. Segmenting motion capture data into distinct behaviors. In: Proceedings of Graphics Interface (GI). pp. 185–194.
[9] Bashir, F. I., Khokhar, A. A., Schonfeld, D., 2007. Object trajectory-based activity classification and recognition using hidden Markov models. IEEE Transactions on Image Processing 16 (7), 1912–1919.
[10] Beringer, J., Hüllermeier, E., 2006. Online clustering of parallel data streams. Data and Knowledge Engineering 58 (2), 180–204.
[11] Bezdek, J. C., Pal, N. R., 1998. Some new indexes of cluster validity. IEEE Transactions on System, Man and Cybernetics 28 (3), 301–315.
[12] Bradley, P. S., Fayyad, U., Reina, C., 1998. Scaling clustering algorithms to large databases. In: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD). pp. 9–15.
[13] Buchin, M., Driemel, A., van Kreveld, M., Sacristán, V., 2010. An algorithmic framework for segmenting trajectories based on spatio-temporal criteria. In: Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS). pp. 202–211.
[14] Davies, D. L., Bouldin, D. W., 1979. A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence 1 (4), 224–227.
[15] Dietterich, T. G., 2002. Machine learning for sequential data: A review. In: Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition. pp. 15–30.
[16] Domingos, P., Hulten, G., 2003. A general framework for mining massive data streams. Journal of Computational and Graphical Statistics 12 (1), 945–949.
[17] Dubes, R. C., 1993. Handbook of Pattern Recognition & Computer Vision. World Scientific Publishing Co., Inc., Ch. Cluster analysis and related issues, pp. 3–32.
[18] Duda, R. O., Hart, P. E., 1973. Pattern Classification and Scene Analysis. John Wiley & Sons.
[19] Duda, R. O., Hart, P. E., Stork, D. G., 2001. Pattern Classification. John Wiley & Sons, Ch. Unsupervised Learning and Clustering, pp. 517–599.
[20] Dunn, J. C., 1973. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. Journal of Cybernetics 3, 32–57.
[21] Dunn, J. C., 1974. A cluster separation measure. Journal of Cybernetics 4, 95–104.
[22] Farnstrom, F., Lewis, J., Elkan, C., 2000. Scalability for clustering algorithms revisited. SIGKDD Explorations Newsletter 2 (1), 51–57.
[23] Fod, A., Mataric, M., Jenkins, O., 2002. Automated derivation of primitives for movement classification. Autonomous Robots 12 (1), 39–54.
[24] Fraley, C., 1996. Algorithms for model-based gaussian hierarchical clustering. Tech. Rep. 311, Department of Statistics, University of Washington.
[25] Gaidon, A., Harchaoui, Z., Schmid, C., 2011. Actom sequence models for efficient action detection. In: Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition (CVPR). pp. 3201–3208.
[26] Guha, S., Meyerson, A., Mishra, N., Motwani, R., O'Callaghan, L., 2003. Clustering data streams: Theory and practice. IEEE Transactions on Knowledge and Data Engineering 15 (3), 515–528.
[27] Guralnik, V., Karypis, G., 2001. A scalable algorithm for clustering sequential data. In: Proceedings of IEEE International Conference on Data Mining. pp. 179–186.
[28] Hamerly, G., Elkan, C., 2001. Learning the k in k-means. In: Proceedings of the seventeenth annual conference on neural information processing systems. pp. 281–288.
[29] Hofmann, Buhmann, 1998. Competitive learning algorithms for robust vector quantization. IEEE Transactions on Signal Processing 46 (6), 1665–1675.
[30] Hubert, L., Arabie, P., 1985. Comparing partitions. Journal of Classification 2 (1), 193–218.
[31] Jain, A. K., 2010. Data clustering: 50 years beyond K-means. Pattern Recognition Letters 31 (8), 651–666.
[32] Jain, A. K., Murty, M. N., Flynn, P. J., 1999. Data clustering: A review. ACM Computing Surveys 31 (3), 1–60.
[33] Kaluža, B., Mirchevska, V., Dovgan, E., Luštrek, M., Gams, M., 2010. An agent-based approach to care in independent living. In: Proceedings of the International Joint Conference on Ambient Intelligence (AmI). pp. 177–186.
[34] Kaufman, L., Rousseeuw, P., 1990. finding Groups in Data: An Introduction to Cluster Analysis. John Wiley & Sons.
[35] Kohonen, T., 1990. Improved versions of learning vector quantization. In: International Joint Conference on Neural Networks (IJCNN). pp. 545–550.
[36] Kranen, P., Assent, I., Baldauf, C., Seidl, T., 2010. The ClusTree: indexing micro-clusters for anytime stream mining. Knowledge Information Systems 29 (2), 249–272.
[37] Kuhn, M. H., Tomaschewski, H., Ney, H., 1981. Fast nonlinear time alignment for isolated word recognition. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). pp. 736–740.
[38] Lee, J.-G., Han, J., Whang, K.-Y., 2007. Trajectory clustering: a partition-and-group framework. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data. pp. 593–604.
[39] Leiva, L. A., 2011. Mining the browsing context: Discovering interaction profiles via behavioral clustering. In: Adjunct Proceedings of the 19th conference on User Modeling, Adaptation, and Personalization (UMAP). pp. 31–33.
[40] Leiva, L. A., Vidal, E., 2011. Revisiting the K-means algorithm for fast trajectory segmentation. In: Proceedings of the 38th International Conference and Exhibition on Computer Graphics and Interactive Techniques (SIGGRAPH).
[41] Liu, C.-L., Jaeger, S., Nakagawa, M., 2004. Online recognition of chinese characters: The state-of-the-art. IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (2), 198–213.
[42] Lloyd, S., 1982. Least squares quantization in PCM. IEEE Transations on Information Theory 28 (2), 129–137.
[43] Mann, R., Jepson, A. D., El-Maraghi, T., 2002. Trajectory segmentation using dynamic programming. In: Proceedings of

the 16th International Conference on Pattern Recognition (ICPR). pp. 331–334.
[44] McQueen, J., 1967. Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability. pp. 281–297.
[45] Mount, D., Arya, S., 1998. ANN: library for approximate nearest neighbor searching. Available: http://www.cs.umd.edu/∼mount/ANN/.
[46] Murtagh, F., 1984. A survey of recent advances in hierarchical clustering algorithms which use cluster centers. Computing Journal 26 (1), 354–359.
[47] Niebles, J. C., Wang, H., Fei-Fei, L., 2008. Unsupervised learning of human action categories using spatial-temporal words. International Journal of Computer Vision 79 (3), 299–318.
[48] Palma, A. T., Bogorny, V., Kuijpers, B., Alvares, L. O., 2008. A clustering based approach for discovering interesting places in trajectories. In: Proceedings of the ACM symposium on Applied computing (SAC). pp. 863–868.
[49] Panagiotakis, C., Pelekis, N., Kopanakis, I., Ramasso, E., Theodoridis, Y., 2012. Segmentation and sampling of moving object trajectories based on representativeness. IEEE Transactions on Knowledge and Data Engineering 24 (7), 1328–1343.
[50] Patra, B., 2011. Convergence of distributed asynchronous learning vector quantization algorithms. The Journal of Machine Learning Research 12 (1), 3431–3466.
[51] Perez, J., Vidal, E., 1994. Optimum polygonal approximation of digitized curves. Pattern Recognition Letters 15 (8), 743–750.
[52] Peshkin, L., Gelfand, M. S., 1999. Segmentation of yeast DNA using hidden Markov models. Bioinformatics 15 (12), 980–986.
[53] Pieraccini, R., Billi, R., 1983. Experimental comparison among data compression techniques in isolated word recognition. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). pp. 1025–1028.
[54] Seni, G., Srihari, R. ., Nasrabadi, N. M., 1996. Large vocabulary recognition of on-line handwritten cursive words. IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (7), 757–762.
[55] Trahanias, P., Skordalakis, E., 1989. An efficient sequential clustering method. Pattern Recognition 22 (4), 449–453.
[56] Veenman, C. J., Reinders, M. J. T., Baker, E. L., 2002. A maximum variance cluster algorithm. IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (9), 1273–1280.
[57] Wang, T.-S., Shum, H.-Y., Xu, Y.-Q., Zheng, N.-N., 2001. Unsupervised analysis of human gestures. In: Proceedings of the Second IEEE Pacific Rim Conference on Multimedia: Advances in Multimedia Information Processing (PCM). pp. 174–181.
[58] Ward, J. H., 1963. Hierarchical grouping to optimize an objective function. Journal of American Statistics Association 58 (301), 235–244.
[59] Xu, L., Krzyzak, A., Oja, E., 1993. Rival penalized competitive learning for clustering analysis, RBF net, and curve detection. IEEE Transactions on Neural Networks 4 (4), 636–649.
[60] Yan, Z., Giatrakos, N., Katsikaros, V., Pelekis, N., Theodoridis, Y., 2011. Setrastream: semantic-aware trajectory construction over streaming movement data. In: Proceedings of the 12th international conference on Advances in spatial and temporal databases (SSTD). pp. 367–385.
[61] Yoon, H., Shahabi, C., 2008. Robust time-referenced segmentation of moving object trajectories. In: Proceedings of the Eighth IEEE International Conference on Data Mining. pp. 1121–1126.
[62] Yu, J., 2005. General C-means clustering model. IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (8), 1197–1211.
[63] Zhang, T., Ramakrishnan, R., Livny, M., 1997. BIRCH: A new data clustering algorithm and its applications. Data Mining and Knowledge Discovery 1 (2), 141–182.
[64] Zhou, A., Cao, F., Qian, W., Jin, C., 2007. Tracking clusters in evolving data streams over sliding windows. Knowledge and Information Systems 15 (2), 181–214.