A Semi-Supervised Approach for Kernel-Based Temporal Clustering

by

Rodrigo Araujo

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2015

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Temporal clustering refers to the partitioning of a time series into multiple non-overlapping segments that belong to k temporal clusters, in such a way that segments in the same cluster are more similar to each other than to those in other clusters. Temporal clustering is a fundamental task in many fields, such as computer animation, computer vision, health care, and robotics. The applications of temporal clustering in those areas are diverse, and include human-motion imitation and recognition, emotion analysis, human activity segmentation, automated rehabilitation exercise analysis, and human-computer interaction. However, temporal clustering using a completely unsupervised method may not produce satisfactory results. Similar to regular clustering, temporal clustering also benefits from some expert knowledge that may be available. The type of approach that utilizes a small amount of knowledge to "guide" the clustering process is known as "semi-supervised clustering."

Semi-supervised temporal clustering is a strategy in which extra knowledge, in the form of pairwise constraints, is incorporated into the temporal data to help with the partitioning problem. This thesis proposes a process to adapt and transform two kernel-based methods into semi-supervised temporal clustering methods. The proposed process is exclusive to kernel-based clustering methods, and is based on two concepts. First, it uses the idea of instance-level constraints, in the form of *must-link* and *cannot-link*, to supervise the clustering methods. Second, it uses a dynamic-programming method to search for the optimal temporal clusters. The proposed process is applied to two algorithms, aligned cluster analysis (ACA) and spectral clustering. To validate the advantages of the proposed temporal semi-supervised clustering methods, a comparative analysis was performed, using the original versions of the algorithm and another semi-supervised temporal cluster. This evaluation was conducted with both synthetic data and two real-world applications. The first application includes two naturalistic audio-visual human emotion datasets, and the second application focuses on human-motion segmentation. Results show substantial improvements in accuracy, with minimal supervision, compared to unsupervised and other temporal semi-supervised approaches, without compromising time performance.

Acknowledgements

I would like to thank, first and foremost, my supervisor, Prof. Mohamed Kamel, for his guidance, patience and support over these years. Without his encouragement, this thesis would not have been possible. I am also grateful to my Ph.D. examining committee members, Dr. Ling Guan, Dr. Fakhri Karray, Dr. Otman Basir, and Dr. William W. Melek, for providing valuable feedback and comments to my thesis. Acknowledgement also goes to Dr. Mohamed Cheriet and Abdelhamid Daouadji for the joint work produced and the support during the time I spent in Montreal at ETS.

I also wish to thank many colleagues at the Centre for Pattern Analysis and Machine Intelligence (CPAMI), especially my officemates, Yun Qian (Mike) Miao and Mehrdad Gangeh, for the valuable discussions, insights and feedbacks about my research and life in general. I would also like to thank Aya and Pouria for the valuable collaborative work and discussions. A special thanks to Ahmed, Sepideh, Safaa, Yibo (Bob) Zhang, Allaa, Bahador, Jamil, Dr. Alaa Khamis, Dr. Farook Sattar, Roy, and CPAMI secretaries, Anne Dracopoulos and Rosalind Klein.

I would like to acknowledge Dr. George Cavalcanti for the support and inspiration that helped me following my academic path. My deepest gratitude to the members of the writing centre, Janne Janke and Jane Russwurm for making my writing better. Also, thanks to Cait Glasson and Ketri Grise for the great suggestions and editing tips of my papers and thesis.

I would like to thank all my Brazilian friends in Canada, Priscila, Plinio, Aline, Felipe, Cibele, Cleyton, Paulinha, André, Marcela, Fábio, Chrys, Léo, Diogo, and Alline, for not only keep my life balanced by the joy they bring, but also for the words of encouragement and motivation that kept me on track.

Finally, I would like to thank my wife, Monica, for her unconditional support and love, without which this work would not be possible. I also wish to thank my father João, and my mother Ilse for providing me with the best education, my sister Raquel, and my forever nanny Maria for the love and support.

Dedication

This thesis is dedicated to my wife, Monica, who has always stood by me and dealt with all my absences with such a positive attitude.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ACA	Aligned cluster analysis
AU	Action units
DP	Dynamic programming
DPSearch	Dynamic programming search
DTAK	Dynamic time alignment kernel
DTW	Dynamic time warping
EP	Exhaustive propagation
FACS	Facial action coding system
H	Cluster alternation rate
HACA	Hierarchical aligned cluster analysis
HMRF	Hidden Markov random fields
LBP	Local binary pattern
MCVar	Mean cluster variance
MOCAP	Motion capture
SC	Spectral clustering
SSACA	Semi-supervised aligned cluster analysis
SSACA+EP	Semi-supervised aligned cluster analysis with exhaustive propagation
SSDPSearch	Semi-supervised dynamic programming search
SSTSC	Semi-supervised temporal spectral clustering
SSTSC+EP	Semi-supervised temporal spectral clustering with exhaustive propagation
STS	Subsequence time series
TC	Temporal clustering
TDCK	Temporal-driven constrained k-means

Chapter 1

Introduction

Clustering algorithms are known for their problem-solving applications when there is little a priori knowledge of the data being analyzed. For example, with a clustering problem, there is no need to know the classes beforehand. These are some well-known advantages of unsupervised methods in general. However, there are situations in which some limited knowledge of the data or application is available, and yet, there is no mechanism allowing the use of that knowledge in a clustering algorithm. Semi-supervised clustering is a strategy that incorporates limited supervision to guide the clustering process. This supervision is usually modelled by using class labels or constraints; as a result, some refer to this learning problem as "constrained clustering."

By incorporating supervision into the clustering process, it is possible to improve accuracy and help the clustering algorithm in situations where the data is not well separated. Some popular unsupervised algorithms that have been adapted into a semi-supervised framework are constrained k-means clustering with background knowledge [3], semi-supervised kernel mean shift clustering [4], semi-supervised kernel k-means [5], and Constrained spectral clustering [6]. However, none of these methods are designed to handle temporal data.

Use of temporal data broadens the range of applications which can benefit from clustering. Traditional areas that incorporate temporal data and clustering are data mining, visualization, and segmentation. The problem of temporal clustering is the focus of this thesis, and is particularly important in applications such as human-motion analysis, audio-visual emotion analysis, and animal behaviour analysis. However, temporal clustering using completely unsupervised methods may not produce satisfactory results. In some cases prior high-level knowledge may be known about the data, or some labeled data may be

available. This information can be used to aid the clustering algorithm. In this thesis, this problem is referred to semi-supervised temporal clustering, which can also be described as the combination of temporal and semi-supervised clustering.

1.1 Proposed Work

This thesis proposes an approach to creating semi-supervised methods which perform temporal clustering. This approach is exclusive to kernel-based clustering methods, and it is based on two concepts. First, it uses the idea of instance-level constraints, in the form of *must-link* and *cannot-link*, as a way to add supervision to the clustering methods. Second, it uses a dynamic-programming method, inspired by [7], to search for the optimal temporal clusters. The proposed approach was applied to two methods, namely, aligned cluster analysis (ACA) and spectral clustering (SC). To validate the advantage of the proposed temporal semi-supervised clustering methods, they were compared with their original versions and with another semi-supervised temporal cluster, using both synthetic and real-world data.

1.2 Summary of Contributions

The main contributions of this thesis can be summarized as follows:

- Proposal of semi-supervised aligned cluster analysis (SSACA), an extension of the temporal clustering method aligned cluster analysis (ACA).
- Proposal of semi-supervised temporal spectral clustering (SSTSC), an application of the dynamic programming search optimization proposed in [8] to the problem of constrained spectral clustering.
- Introduction of exhaustive propagation (EP) into the proposed methods to create SSACA+EP and SSTSC+EP, helping improving accuracy.
- Application of semi-supervised temporal clustering to spontaneous continuous emotion segmentation.
- Application of semi-supervised temporal clustering to human motion segmentation.

1.3 Thesis Organization

This thesis is organized as follows.

Chapter 2 gives an overview of the fundamentals necessary to understand the concepts of semi-supervised temporal clustering. Sections 2.1 and 2.2 explain, respectively, some time series representations and some similarity measures. Section 2.3 discusses some of the main tasks in the field of time series. Section 2.4 describes time series segmentation and explains popular methods of carrying out segmentation. Section 2.5 gives an overview of clustering methods, and Section 2.6 explains the kernel k-means clustering algorithm and other traditional algorithms. Section 2.7 introduces semi-supervised clustering and how constraints are used as supervision. Finally, Section 2.8 explains temporal clustering.

Chapter 3 introduces the proposed methods and offers some considerations about the use of constraints. Chapter 3 also provides a theoretical comparison with subsequence time series (STS). Section 3.1 derives the problem of temporal clustering, and Section 3.2 revisits the ACA method to give some background. Section 3.3 explains the process of adding supervision in the form of pairwise constraints to ACA, using the semi-supervised kernel k-means framework. Sections 3.4 and 3.5 introduce the two proposed methods, SSACA and SSTSC, respectively. Section 3.6 explains how an exhaustive constraint propagation is added to SSACA and SSTSC to create SSACA+EP and SSTSC+EP, which are explained in Sections 3.7 and 3.8, respectively. Finally, Sections 3.9 and 3.10 discuss some theoretical considerations related to the proposed methods.

Chapter 4 evaluates the performance of the proposed methods when applied to synthetic data. Section 4.1 presents the setup of the experiments and some quantitative and qualitative evaluation measures. Section 4.2 shows some comparative results of the proposed methods in a synthetic dataset and a complete analysis of how different variables can influence these methods.

Chapter 5 evaluates the performance of the proposed methods when applied to real-world applications. Before analyzing the results, this chapter gives an overview of the problem of emotion analysis and human motion segmentation. Section 5.1 introduces emotion analysis in the context of the visual modality, and how facial expression relates to emotion states. It also describes the fundamental steps required to develop an automatic facial expression recognition system, and differentiates the two most common ways to interpret emotion: categorical and dimensional models. Finally, Section 5.1.2 shows results of the comparison between the proposed methods and other related methods on the VAM Corpus dataset, and Section 5.1.5 shows results and comparisons on the AVEC emotion dataset. Section 5.2 discusses the problem of human motion segmentation and shows the

results of the comparisons between the proposed methods and other related methods when applied to a motion capture dataset.

Finally, Chapter 6 concludes the thesis, and discusses future extensions of this research, as well as additional applications that might benefit from the proposed methods. Also provided in this chapter is a list of publications produced by this research.

Chapter 2

Background and Literature Review

This chapter presents an overview of some fundamental concepts of temporal clustering and semi-supervised methods. The overview starts with some concepts of time series. Later, some background on clustering algorithms is discussed, with a particular emphasis on kernel-based methods, which are the focus of this thesis. This chapter also discusses the concept of semi-supervised clustering and how constraints can be used to aid the clustering process. The chapter closes with a review of related work.

2.1 Time Series Representation

According to [9], a time series is a collection of chronological observations. Representation of time series poses a fundamental problem, especially when the goal is to reduce their dimensionality of the time series. The simplest method of representing a time series is by sampling. Sampling involves selecting data representatives at fixed intervals, such that a more concise representation can be obtained. However, if the sampling rate is too low, this approach can result in distortions in the shape of the representation. An alternative method of representation is to use the average value of the data points within fixed-sized windows. This method is also known as piecewise aggregate approximation (PAA)[10]. Similar variations on this method have also been proposed, such as adaptive piecewise constant approximation (APCA)[11], in which the length of the window is adaptable, and piecewise linear representation (PLR).

Another type of time series representation involves transformation of the time series from the time domain to a different domain. Discrete Fourier transform (DFT) is a popular

transformation technique, first used by [12]. Discrete wavelet transform (DWT), Haar transform [13], and singular value decomposition (SVD) [14] offer alternative approaches to DFT.

Another family of representation consists of transforming numeric time series into a symbolic representations. SAX [15] is an example of this type of approach. This method first transform the data into a PAA representation, and then symbolizes the PAA into a discrete string. A benefit of this type of representation is its ability to produce distance measures that lower bounds the distance measures in the original series. In other words, SAX allows the use of various algorithms, such as clustering, classification, and anomaly detection, in the new representation, while still obtaining the same results.

2.2 Similarity Measures

Similarity measure plays a major role in solving many time series tasks. The most commonly used distance measure for time series is the Euclidean distance, also known as L_2-norm. Given two time series T and Q of size n, the Euclidean distance is given by:

$$Dist(P, Q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}. \tag{2.1}$$

[16] describes this family of distance as *lock-step measures*, which refer to all the distance measures comparing the ith point of one time series to the ith point of another time series. They also include the other Lp norms, such as L_1-norm (city block distance); L_{inf}-norm; and DISSIM [17]. Another category of distance is called *elastic measures*, which include distance measures that allow for comparison of either one-to-many points or one-to-none (e.g., LCSS). Some of the distance measures which fall into this category are Longest Common SubSequence (LCSS) [18], Edit Sequence on Real Sequence, Swale, Edit Distance with Real Penalty (ERP) [19], and dynamic time warping (DTW) [20]. In contrast with *lock-step measures*, which only compare time series of the same sizes, *elastic measures* have the capability to measure time series of different sizes and handle local time shifting – i.e., time series that are similar, but are out of phase.

More recently, two other categories of similarity measures were proposed. The first of these categories is threshold-based measure, and the second is pattern-based measure. TQuEST [21] distance is an example of threshold-based distance. TQuEST transforms the time series into threshold-crossing time intervals, wherein the points within each time

interval have values greater than a threshold τ. Then, each interval is treated in a two-dimensional space, the first dimension being starting time, and the second being the ending time. Later, the similarity is defined as the Minkowski sum of the two sequences of time interval points. SpADe [22] is an example of pattern-based similarity measure, the second of the recent measures. SpADe attempts to detect matching segments within the entire time series by shifting and scaling in both temporal and amplitudinal dimensions. The similarity is represented by the set of matching pattern with the greatest degree of likeness.

2.2.1 Dynamic Time Warping

Dynamic time warping is one of the most frequently used measures in time series. This technique was proposed by [20] as an application for speech recognition. DTW allows for the measurement of similarities between temporal sequences which varies in time or speed, and can be computed by dynamic programming. Dynamic programming is a method for solving complex problems by recursively breaking them down into simpler subproblems. The complexity of DTW is $O(n^2)$; however, there are lower bound implementations with amortized complexities [23].

In order to align a sequence P of length n, and a sequence Q of length m using DTW, certain steps are necessary. The first step is to create a n-by-m matrix M, where the (i^{th}, j^{th}) element of the matrix $m_{i,j}$ is a squared distance (Euclidean distance) $d(p_i, q_i) = (p_i - q_i)^2$ corresponding with the alignment between points p_i and q_j. Second, in order to find the best alignment between the two sequences, the algorithm must retrieve a path through the matrix that minimizes the total cumulative distance. The optimal path is the one that minimizes the warping cost

$$DTW(P,Q) = min\left\{\sqrt{\sum_{k=1}^{K} w_k}\right\}, \tag{2.2}$$

where w_k is an element $(i,j)_k$ of the warping path W, which is a contiguous set of matrix elements that defines a mapping between P and Q. Its k^{th} element is defined as $W_k(i_k, j_k)$ and $W = w_1, w_2, \ldots, w_k, \ldots, w_K$, where $\max(m,n) \leq K < m + n - 1$.

The warping path is subject to some constraint sets. The first constraint set is called boundary condition, and it requires that the warping start and finish diagonally. The second set of constraints is the continuity, meaning that if $w_k = (a,b)$, then $w_{k-1} = (a',b')$, where $a - a' \leq 1$ and $b - b' \leq 1$. The continuity constraint only allows to the warping path,

cells that are adjacent. The third constraint set only allows $a - a' \geq 0$ and $b - b' \geq 0$, which forces the points in W to be monotonically spaced in time.

The optimization can be executed by dynamic programming, in order to evaluate the following recurrence equation

$$\gamma(i,j) = d(p_i, q_j) + min\left\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\right\}, \tag{2.3}$$

where $d(p_i, q_j)$ is the distance found in the current cell, and $\gamma(i,j)$ is the cumulative distance of $d(i,j)$, and the minimum cumulative distance of the three adjacent cells. Although DTW creates a distance-like measure between time sequences, one drawback of using it as an approach is that it does not guarantee the triangle inequality property.

2.2.2 Dynamic Time Alignment Kernel (DTAK)

Dynamic time alignment kernel (DTAK) [24] is another technique that aligns time series, and offers a metric for calculating distances between them. DTAK is an extension of DTW; however, in contrast with DTW, DTAK satisfies the Cauchy-Schwartz inequality, and can also be solved efficiently by dynamic programming. Much like DTW, when given a sequence $X = [x_1, \ldots, x_{n_x}] \in \mathbb{R}^{d \times n_x}$ and a sequence $Y = [y_1, \ldots, y_{n_y}] \in \mathbb{R}^{d \times n_y}$, DTAK creates a cumulative kernel matrix $U \in \mathbb{R}^{n_x \times n_y}$ which can be computed recursively by

$$u_{ij} = \max \begin{cases} u_{i-1,j} + k_{ij} \\ u_{i-1,j-1} + 2k_{ij} \\ u_{i,j-1} + k_{i,j}, \end{cases} \tag{2.4}$$

such that $u_{11} = 2k_{11}$ where $k_{ij} = \phi(\mathbf{x}_i^T)\phi(\mathbf{y}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{y}_j\|^2}{2\sigma^2}\right)$, which is the frame kernel matrix $\mathbf{K} \in \mathbb{R}^{n_x \times n_y}$. The distance between X and Y is given by

$$\tau(\mathbf{X}, \mathbf{Y}) = \frac{u_{n_x n_y}}{n_x + n_y}. \tag{2.5}$$

[25] points out one of the drawbacks of DTAK: it is not necessarily a strictly positive definite kernel, and a regularization of the kernel matrix therefore needs to be performed.

2.3 Time Series Tasks

Time series is widely studied in the field of data mining, and includes several related research areas, such as finding similar time series, subsequence searching in time series, dimensionality reduction, and segmentation.

A summary list of some of the main time series tasks includes [15]:

- **Indexing**: Consists of finding the most similar data series in a database, given a query time series P and a similarity measure.
- **Clustering**: Consists of organizing time series into similar groups, such that time series in the same group are more similar to each other than to those in other groups, given a similarity measure.
- **Classification**: Consists of assigning a given unlabelled time series P to a predefined class.
- **Summarization**: Consists of creating a smaller (possibly graphical) representation of a very large time series P, such that it retains its essential features.
- **Anomaly Dectection**: Consists of finding anomalies (unexpected, surprising, inconsistent) in the sections of a given time series P, given a model of "normal" behaviour.
- **Segmentation**: Consists of partitioning a given time serie P into k segments that are internally homogeneous [26].

2.4 Time Series Segmentation

A popular solution for segmentation in time series comes from solving the change detection problem. Change-point detection consists of an analysis of changes on the distribution of the points within a window of temporal observations, in order to locate their boundaries. The standard solution as [9] describes consists of fixing the number of change-points, then identifying their positions, and finally, determining functions for curve-fitting the intervals between successive change-points. A typical strategy for triggering a change-point is when the probability distributions of time series samples, over past and present intervals becomes significantly different [27] [28]. However, this technique detects only local boundaries, and does not provide a global model for temporal events.

Another group of methods uses non-parametric approaches, such as kernel density estimation, which are designed with no particular parametric assumptions. However, these estimations exhibit reduced accuracy in high-dimensional problems. According to [29], a recently-introduced strategy estimates the ratio of the probabilities directly without going through density estimation [30] (e.g., kernel mean matching [31], the logistic-regression method, and the Kullback-Leibler importance estimation procedure (KLIEP)), and has shown promising results.

Another popular way of performing time series segmentation is through a switching linear dynamic system (SLDS). Linear dynamical systems (LDSs) are useful in describing dynamical phenomena. However, such phenomena change over time, and as a result, the models that represent the phenomena also change. Switching several linear dynamic systems over time allows for description of the dynamics of the time series. The switching states in SLDS inference provide the segmentation of an input sequence implicitly. Because of the intractability of finding exact inferences, some approximations have been proposed: in [32], which casts the SLDS model as a dynamic Bayesian network; in [33], which presents a data-driven MCMC (DD-MCMC) sampling method for approximate inference in SLDSs; and [34], which uses a nonparametric Bayesian approach for learning switching dynamical processes by extending the HDP-HMM formulation.

Other approaches tackle segmentation from the perspective of analyzing periodicity of cyclic events [35] [36]. However, cyclic motion analysis only extracts segments of repetitive motion, which may leave some portion of the signal unsegmented and therefore not modelled.

Segmentation in video can be done by clustering frames and grouping those that are assigned to the same cluster to form a segment. One example of this technique is proposed by [37], which uses an action-based distance to group actions in a video by clustering similar frames. However, this approach performs segmentation as a later step of clustering; consequently it lacks a mechanism by which to incorporate the dynamics of the temporal events in the clustering process, as [38] indicates. Another method to segment time series involves obtaining segments that are internally homogeneous, and aim to minimize the overall cost of the segmentation. In other words, the optimal k-segmentation of time series s using costing function $Cost_F(s_1 s_2 \dots s_k)$ is minimal among all possible k-segmentations [39] [40]. This problem can be, however, stated as a clustering problem – if, for example the overall cost function is defined as the minimization of the distances of the centre of the clusters.

Some methods, however, attempt to solve the problem of segmentation as a global model for temporal events. These methods actually solve the problem by minimizing the

errors across various segments for each k cluster, in order to find the segments. This subset of segmentation method relates to this thesis, and it is discussed in great detail in Section 2.8. First, however, some background on clustering algorithms must be given to further contextualize the challenges addressed by this type of approach.

2.5 Clustering

Data clustering is the main task of unsupervised learning. Unlike supervised methods, unlabelled data is used as the basis of learning: the data samples involved in the learning process have no categories previously assigned. Clustering consists of grouping data points into the same group, or "cluster," as it is usually called, such that data points in the same cluster are more similar to each other (internal criterion) than to those in another cluster (external criterion). Figure 2.1 depicts an example of a cluster.

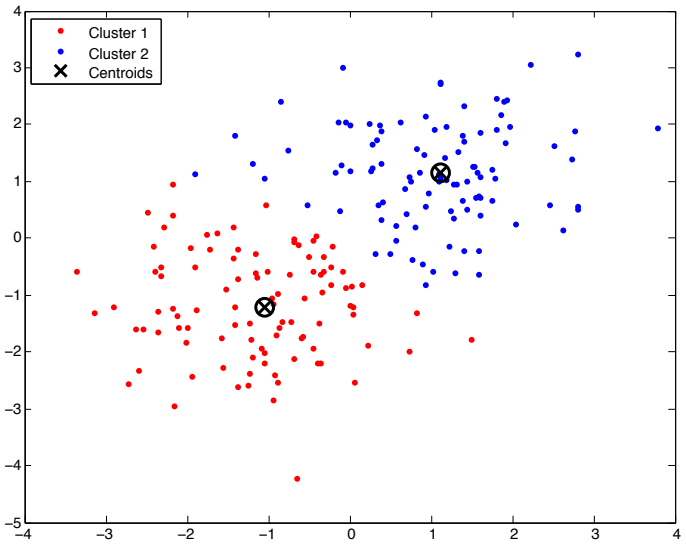


Figure 2.1: 2D random points clustered by k-means algorithm. $\otimes$ represent the centroid of each cluster.

2.5.1 Clustering Objective Function

Consider a set $\mathcal{X}$ of n elements $x_1, \ldots, x_n$ that needs to be clustered in k subsets $\mathcal{C}_1, \ldots, \mathcal{C}_k$, where elements in the same cluster are more similar to each other than the elements in a different cluster. In order to make this statement a well-defined problem, it is necessary to define an objective function that measures the quality of any partition of the data [41]. The challenge subsequently becomes a matter of finding the partitions that optimize the objective function.

One simple way of defining an objective function for clustering is to use the sum-of-squared-error. Let n_i be the number of samples in $\mathcal{C}_i$, and let μ_i be the mean of those samples defined by

$$\mu_i = \frac{1}{n_i} \sum_{x \in \mathcal{C}_i}^{N} x. \tag{2.6}$$

The sum-of-squared errors is defined by

$$J_c = \sum_{i=1}^{k} \sum_{x \in \mathcal{C}_i} \|x - \mu_i\|^2. \tag{2.7}$$

For a given cluster $\mathcal{C}_i$, the mean vector μ_i is the best representative of samples $\mathcal{X}_i$, since it minimizes the sum of the squared lengths of the error vectors $x - \mu_i$ in the subset $\mathcal{C}_i$. Therefore, J_c is the total squared error incurred to represent the n samples $x_1, \ldots, x_n$ by the k centres $\mu_1, \ldots, \mu_k$. The optimal partition is the one that minimizes J_c, whose value depends on how the samples are grouped into clusters and the number of clusters. This type of clustering is called minimum variance partition.

2.5.2 Similarity Measures

The clustering problem consists of finding a natural group of data that satisfies both an internal and external criterion. In order to define that samples in one cluster are more similar to one another than the samples in other clusters, a similarity measure must be defined. The most straightforward way to defining similarity or dissimilarity between two points is to use the distance between them. The distances most often used for a numerical data $x, y \in \mathbb{R}^m$ are based on L_2, L_1 and L_∞ norm.

L_2 norm is the Euclidean, distance defined by

$$d(x, y) = ((\sum_{i=1}^{m} |x_i - y_i|^2)^{\frac{1}{2}}), \tag{2.8}$$

L_1 norm, which is also known as city block metric, and is defined by

$$d(x, y) = (\sum_{i=1}^{m} |x_i - y_i|), \tag{2.9}$$

and L_∞ norm or Chebyshev distance, defined by

$$d(x, y) = \max_i(|x_i - y_i|). \tag{2.10}$$

A non-metric similarity function $s(x, y)$ is another type of measurement that can be used is to compare the angular similarity of two vectors x and y. Conventionally, this function is symmetric and its value is large when x and y are similar in some way. The function can, for example, be used to measure the angle between vectors x and y. In fact, this measurement is the cosine of the angle between them.

$$s(x, y) = \frac{x^t y}{\|x\|\|y\|} \tag{2.11}$$

Other types of similarity measurements are specific to categorical (nominal) data, including Hamming distance (the number of attributes taking different values) and edit distances. Yet, another class of similarity measures is based on kernels. Next section discusses the use of kernel as a similarity measurement.

2.5.3 Kernel as Similarity Measure

Let us denote a set of n objects by $\mathcal{S} = (x_i, \ldots x_n)$. Suppose that each object x_i is an element of a set $\mathcal{X}$, and therefore, a representation $\phi(x) \in \mathcal{F}$ is defined for each object. In a classical representation the data set $\mathcal{S}$ is denoted as $\phi(\mathcal{S}) = (\phi(x_1, \ldots, \phi(n_n))$. All the methods explained above are based on this classical, individual representation of the data as feature vectors. However, there is a whole family of methods based on a set of pairwise comparisons. In other words, they are represented as real-valued comparison functions

$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, and the data set $\mathcal{S}$ is represented by the $n \times n$ matrix of pairwise comparisons $k_{i,j} = k(x_i, x_j)$ [42]. This type of representation produces squared matrices, which are completely independent of the nature of the objects being analyzed. As a result, data of different natures can be analyzed with this unified framework, which is part of a family of algorithms known as "kernel methods."

2.6 Clustering Algorithms

Clustering algorithms are mainly categorized into partitional and hierarchical [1]. The choice of which type of algorithms to use depends on the nature of the data being analyzed and the prior knowledge available about the data. Figure 2.2 provides an overview of the main types of clustering algorithms. This subsection will briefly introduces these two categories.

2.6.1 Partitional Clustering

Partitioning algorithms, also known as "flat clustering", divide data points into non-overlapping clusters in order to optimize a certain criterion function such as the one defined in Section 2.5.1. A standard algorithm that gives an effective representation of the partitioning category is the k-means algorithm. The goal of k-means is to group the data points into k partitions such that the distance between the points in each cluster and its centroid is minimized. Refer to Algorithm 1, which outlines the algorithm's pseudocode.

The algorithm performs a greedy minimization of the criterion function; however, there is no guarantee of reaching a global minimum. The quality of the obtained solution depends on the initial partitioning of the data points. An heuristic for obtaining a better (albeit not optimal) solution is to repeat the algorithm, starting from different initial solutions, and to return the solution with the minimum value of the criterion function.

Another partitioning algorithm is called k-medoids. K-medoids is a variation of k-means, which instead of using the mean point to define the centre of the clusters, it uses actual points to represent the cluster. As a result, this method is more robust to noises and outliers. K-medoids determines the k centres, which are actual points, by minimizing the sum of the distances between each point and the nearest centre. The Partitioning around Medoids (PAM) is a method that implements this approach, and its time complexity is $O(K(N-K)^2 I)$, where N is the number of points, K is the number of clusters, and I the number of iterations to converge [43].

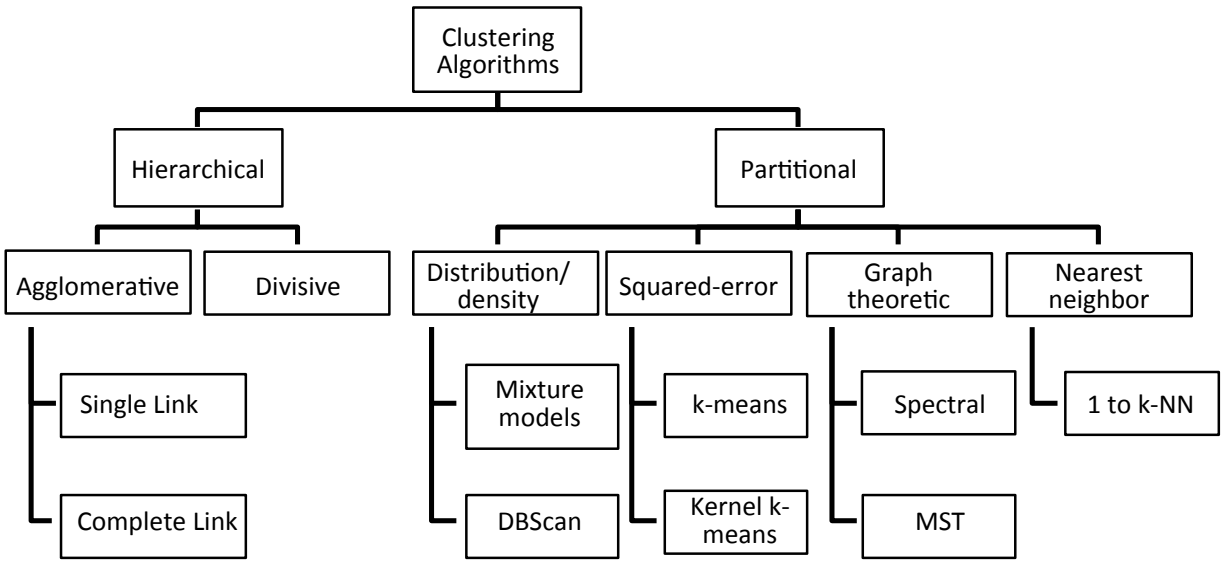


Figure 2.2: Taxonomy of clustering algorithms. [1]

Kernel Clustering

As discussed in Section 2.5.3, there is a family of methods (kernel methods) that use square matrices that are symmetric-positive-definite. In other words, if k is an $n \times n$ matrix of pairwise comparisons, it should satisfy $k_{i,j} = k_{j,i}$ for any $1 \leq i, j \leq n$, and $\mathbf{c}^\top k\mathbf{c} \geq 0$ for any $\mathbf{c} \in \mathbf{R}^n$, as described in [42]. This family of methods includes approaches such as SVM [44], PCA, kernel k-means [45], Spectral Clustering [46], non-negative matrix factorization (NMF) [47], and kernel Self-Organizing Maps [48].

Kernel k-means is one of the simplest kernel clustering algorithms, and is a non-linear extension of the k-means. In kernel k-means, the Euclidean distance function is simply replaced by a non-linear distance function, which has the capacity of projecting the data into higher dimensions.

Kernel k-means is a partitioning clustering algorithm, in which n data points are divided

Algorithm 1 k-means

Choose randomly k points centres $\mathbf{C_1}$,...,$\mathbf{C_k}$.
while the clusters are changing **do**
  Reassign the data points.
  for $\mathbf{i} = 1 \rightarrow \mathbf{n}$ **do**
    Assign data point $\mathbf{x_i}$ to the cluster whose centre $\mathbf{C_j}$ is closest
  end for
end while
Update the cluster centrers.
for $\mathbf{j} = 1 \rightarrow \mathbf{k}$ **do**
  $\mathbf{n_j}$ = number of points in C_j;
  $\mathbf{c_j} = \frac{1}{n_j} \sum_{x_i \in C_j} x_i$
end for

into k disjoint groups by minimizing the within-cluster variation. It finds the partition of the data that is a local optimum of the following energy function:

$$J_{kkm}(\mathbf{G}) = \sum_{c=1}^{k} \sum_{i=1}^{n} g_{ci} \underbrace{\|\phi(\mathbf{x}_i) - \mathbf{z}_c\|^2}_{dist^2_\phi(\mathbf{x}_i,\, \mathbf{z}_c)} = \|\mathbf{X} - \mathbf{ZG}\|_F^2, s.t. \mathbf{G}^T 1_k = 1_n, \tag{2.12}$$

where $\mathbf{x}_i \in \mathbb{R}^d$ (see notation[1].) is a vector representing the ith data point, $\phi(.)$ is a nonlinear mapping function, and $\mathbf{z}_c \in \mathbb{R}^d$ represents the centroid of the data points in cluster c. Matrix $\mathbf{G} \in \{0,1\}^{k \times n}$ indicates, in a binary format, the cluster to which $\mathbf{x}_i$ belongs, and $\mathbf{Z} \in \mathbb{R}^{d \times n}$ is the actual value of the means.

The squared distance between the ith element and the centre of the cluster c represented by $dist^2_\phi(\mathbf{x}_i, \mathbf{z}_c)$ can be calculated as:

$$dist^2_\phi(\mathbf{x}_i, \mathbf{z}_c) = k_{i,i} - \frac{2}{n_c} \sum_{j=1}^{n} g_{cj} k_{ij} + \frac{1}{n_c^2} \sum_{j_1, j_2 = 1}^{n} g_{cj_1} g_{cj_2} k_{j_1 j_2}, \tag{2.13}$$

where $n_c = \sum_{j=1}^{n} g_{cj}$ represents the total number of samples that are in cluster c, and the kernel function k is given by $k_{ij} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$.

[1]Bold capital letters denote a matrix X, bold lower-case letters a column vector x

Kernel k-means follows the same approach as regular k-means, and assigns a data point $\mathbf{x}_i$ to the cluster whose mean $\dot{\mathbf{z}}$, which was computed in the previous step, it is closest to, that is

$$g_{c_i^*} = 1, \text{ where } c_i^* = \arg\min_c dist_\phi^2(\mathbf{x}_i, \dot{\mathbf{z}}_c). \tag{2.14}$$

The algorithm subsequently updates the new cluster centres. Although kernel k-means cannot explicitly compute the centres, there is no need to do so, because the distances $dist_\phi^2(\mathbf{x}_i, \dot{\mathbf{z}}_c)$ between the samples and the means can be retrieved from the kernel matrix.

2.6.2 Hierarchical Clustering

Hierarchical algorithms construct the clusters by building a nested structure or tree, also known as dendrogram (Figure 2.3). The root of the tree represents the top cluster, and the leaf nodes represent the singleton clusters.

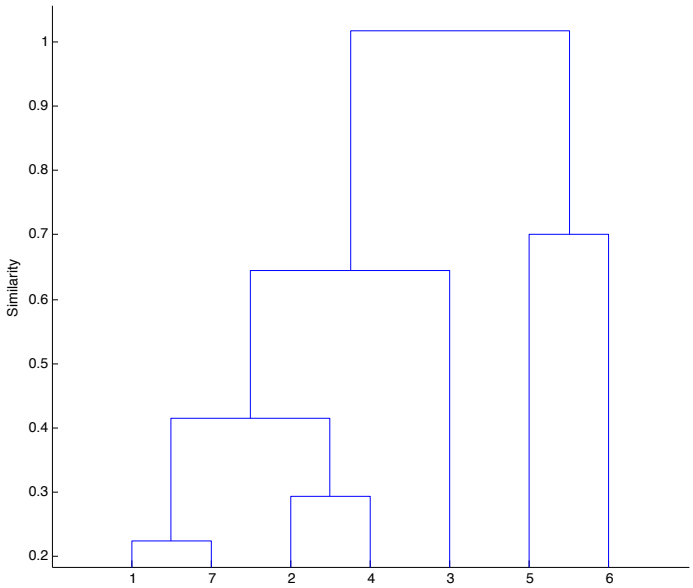


Figure 2.3: Dendrogram

Unlike partitioning algorithms, hierarchical clustering does not require the number of clusters to be specified. However, a flat-partition can be obtained by traversing the root

of the dendrogram until a number of clusters arise. A flat-partition can also be acquired by defining a threshold of the combination similarity between clusters and cutting off the tree at this specific point.

The strategy for construction of the tree can be generated in two ways: agglomerative hierarchical clustering (AHC) or divisive hierarchical clustering (DHC). AHC begins with singleton clusters, and then merges the most similar pairs of clusters until all data is grouped in one cluster. DHC operates in the opposite direction, with all the data points as one cluster, and then a partition of the dissimilar groups into separate clusters until each point becomes a singleton.

Of the two approaches AHC is more commonly used in applications ostensibly because the DHC criterion is less natural and more computationally expensive. Instances of AHC differ in their methods of calculating the similarities and distances between pairs of clusters. The most common methods are:

1. **Single-link clustering:** The distance between two clusters is computed as the distance between the two closest elements in the two clusters.

$$d(C_i, C_j) = \min\{d(x_i, x_j) | \forall x_i \in C_i, \forall x_j \in C_j \tag{2.15}$$

2. **Complete-link clustering:** The distance between two clusters is computed as the maximum distance between the farthest elements in two clusters.

$$d(C_i, C_j) = \max\{d(x_i, x_j) | \forall x_i \in C_i, \forall x_j \in C_j \tag{2.16}$$

3. **Average-link clustering:** The distance between two clusters is computed as the average distance between objects from the first cluster and objects from the second cluster.

$$d(C_i, C_j) = \frac{1}{|C_i| \times |C_j|} \sum_{\substack{x_i \in C_i \\ x_j \in C_j}} d(x_i, x_j) \tag{2.17}$$

The dendrogram that is generated using the hierarchical algorithm displays the structure of the data distribution. This type of structure allows for the number of clusters to be chosen a posterior, unlike from k-means and k-median, which require a priori number of clusters.

Hierarchical clustering algorithms possess some limitations. AHC follows a greedy process in order to decide whether to merge or split clusters. The decision, once made, is never reconsidered in later steps. Another limitation of AHC is its computational complexity ($O(N^3)$) in the worst-case for computing pairwise similarities and iterations.

AHC includes many other approaches beyond those described above. Some of these alternatives use hybrid algorithms, such as BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [49], to address the aforementioned limitations. Other algorithms include CURE (Clustering Using REpresentatives), ROCK (RObust Clustering using linKs), CHAMELEON, and SOTM (Self-organizing Hierarchical Variance Map) [50].

2.7 Semi-Supervised Clustering

In machine learning, unsupervised methods such as clustering algorithms have been used in a variety of tasks, though mainly in exploratory data analysis. Some examples of these tasks include group discovery, data compression, dimensionality reduction, and outlier detection. In some certain tasks, some expert knowledge may be available, such as labels, relationship between data, or global assumption that are part of the domains. The process of leveraging extra knowledge is known as "semi-supervised learning". Semi-supervised learning exists at the intersection between supervised and unsupervised learning, and falls into two categories:

1. **Semi-supervised classification:** Tasks falling into this category are, essentially, supervised in the presence of low quantities of supervision. Semi-supervised classification uses the addition of unlabelled data to help supervised classifier learning. Semi-supervised classification nevertheless suffers from the same limitations as other supervised methods, including a priori knowledge of the number of classes and a minimum number of training samples for each class.

2. **Semi-supervised clustering:** Also known as constrained clustering, semi-supervised clustering is essentially an unsupervised method that uses the addition of some supervision to support the clustering process. Classes are not known beforehand; therefore, there is no need for a minimum number of data samples for each class.

The reason why semi-supervised clustering is also known as "constrained clustering" is because constraints are the typical approach to leveraging supervision in the clustering

process. These constraints usually come in pairs, and are commonly known as "pairwise constraints."

Methods such Constrained K-means Clustering with Background Knowledge represent a category of semi-supervised clustering algorithms. COP-KMEANS, as the authors of [3] call their algorithm, is a modification of the popular k-means algorithm that uses instance-level constraints in the form of *must-link* and *cannot-link* constraints. During the assignment phase of k-means, in which points are assigned to the closest clusters, COP-KMEANS ensures that no constraint is broken, guiding the clustering process towards a more appropriate partitioning of the data.

Basu et al. [51] propose a probabilistic model for semi-supervised clustering, based on Hidden Markov random fields (HMRFs). Their model is a combination of a constraint-based method, which uses constraints to modify the objective function, and a distance-based method, which trains similarities to satisfy constraints. This method partitions the data into k clusters, such that the total distortion between the points and the representatives of the clusters is minimized, and a minimum number of constraints are violated in the process.

Anand et al. [4] propose a kernel mean shift algorithm that incorporates pairwise constraints. Their method first maps the data to a high-dimensional kernel space. Then, by using a linear transformation, the points are projected to a space where the constraints are enforced. The authors show that this transformation can be achieved implicitly by modifying the kernel matrix. This method is another effort directed towards transforming an established clustering algorithm into semi-supervised clustering.

2.7.1 Pairwise Constraints

The use of extra knowledge in the clustering process can be of great value for improving accuracy, or even for creating clusters with desirable geometric properties [52]. One way of incorporating this extra knowledge is through the introduction of constraints. Constraints can be introduced at the cluster level (group of instances), in ways such as constraining the size of a cluster [53], constraining neighbours in a cluster to be within a certain threshold distance [54], or forcing time contiguity of instances in a temporal clustering [55]. Constraints can also be introduced at the instance level. The most common instance-level constraints are *must-link* and *cannot-link* [56]. A *must-link* constraint denotes that two instances x and y must be in the same cluster, whereas a *cannot-link* constraint denotes that two instance x and y must not be in the same cluster. Figure 2.4 illustrates the concept of *must-links* and *cannot-links*. In many situations where there is no domain expert

available, feedback in the form of these two constraints is easier to acquire, compared to the actual label. Since *must-link* constraints share symmetrical, reflexive, and transitive properties, it can be inferred that, if $(x, y) \in \mathcal{M}$ and $(y, z) \in \mathcal{M} \Rightarrow (x, z) \in \mathcal{M}$, where $\mathcal{M}$ represents a set of *must-links*. Therefore, it is possible to infer extra pairs of constraints by applying transitive closure.

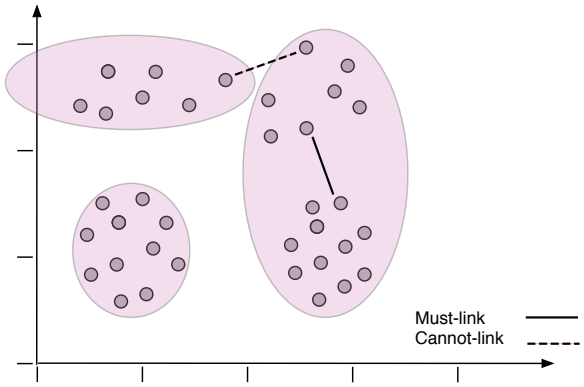


Figure 2.4: Pairwise constraints in the form of *must-link* and *cannot-link*

The use of pairwise constraints can also contribute to seeding the cluster initialization. For example, during cluster initialization, points that are *must-links* should start in the same cluster, whereas points that are *cannot-links* should start in different clusters.

2.8 Temporal Clustering

Temporal clustering (TC) can be defined as the partitioning of multiple time series into a set of non-overlapping segments that belong to k temporal clusters [38]. Temporal clustering is similar to normal clustering in that both require a similarity measures, clustering algorithms, and evaluation criteria; however, the temporal nature of the data requires special treatment when it comes to one or more of these components. The two major ways of handling time series are either to modify existing static data-clustering algorithms to handle time, or to convert the time series into a form that works with static algorithms. The former approach relies, in most cases, on modifying the similarity measure to an appropriate measure of time series, such as DTW and DTAK. The latter maps the time series into a different representation or domain that embeds the temporal information – such as Wavelets, Fourier, and Haar transform [9] – or into a number of model parameters, and then applies the conventional static data clustering algorithm.

Temporal clustering can be used as a tool to solve segmentation problems. Applying a clustering approach to the segmentation problem requires global modelling of all temporal segments in a time series, instead of only finding local boundaries. As a result, simply using some of the traditional time series segmentation techniques, such as change point detection, may not produce satisfactory results. HACA [25] and ACA [7] solve this problem by minimizing errors across various segments for each k cluster, yet, all of these methods lack mechanisms for adding supervision.

Most of the time series clustering techniques, such as the ones reviewed in [57], assume that the time series are already segmented, which underlies the main difference between the traditional time series clustering techniques and TC.

2.9 Related Work

Semi-supervised temporal clustering, as its name suggests, combines temporal and semi-supervised clustering in order to perform temporal segmentation. Much like general clustering approaches, temporal segmentation using clustering may not produce satisfactory results, due to being a completely unsupervised approach. In some situations, however, prior high-level knowledge exists about the segments, or some labeled data is available. This information can be used to aid the clustering algorithm.

However, there are few methods discussed in the literature dealing with both temporal and semi-supervised aspects. Of the few methods available, temporal-driven constrained k-means (TDCK) [55] shows the best results compared to other methods such as TCK-means [58], Constrained k-means, and Temporal-Driven k-means, as described in [59]. TDCK offers a framework to model external information and add it to an unsupervised algorithm. Originally created as a solution to the problem of detecting typical evolution patterns, such as country evolutions, TDCK also provides benefits to applications for social network analysis, such as detecting social roles. Intuitively, TDCK adds *must-links* between all the pairs of observations belonging to the same entity. This mechanism, however, restricts the method to lower control at the instance level, and does not allow the use of *cannot-links*.

[60] describes a similar method, which combines discriminative cluster analysis (DCA) with a temporal and a semi-supervised term. The paper describes a semi-supervised temporal clustering algorithm used to group large amounts of multimodal data into different activities. Similar to the approach used in [55], the constraints penalize non-smooth changes (over time) on the assigned clusters. However, there is no control over the granularity of the temporal term, and there is also no robust metric between time series.

2.9.1 Temporal-Driven Constrained K-means

The goal of TDCK is to cluster observations $x_i \in \mathcal{X}$, which are descriptions of entities at a given timestamp written as triples (entity, timestamp, description): $x_i = (x_i^\phi, x_i^t, x_i^d)$, where $x_i^d \in \mathcal{D}$ is the vector in the multidimensional description space that describes the entity $x_i \in \Phi$ in time $x_i^t \in \mathcal{T}$, while taking into account both the temporal component and the multidimensional description [59]. TDCK-means searches to minimize the following objective function:

$$\sum_{\mu_j \in \mathcal{M}} \sum_{x_i \in \mathcal{C}_j} \left(\|x_i - \mu_j\|_{TA} + \sum_{\substack{x_k \notin \mathcal{C}_| \\ x_k^\phi = x_i^\phi}} w(x_i, x_k) \right), \tag{2.18}$$

where $\|\cdot\|_{TA}$ is a temporal dissimilarity measure, $w(x_i, x_j)$ is the cost function that determines the penalty of clustering adjacent observations of the same entity into different clusters, and C_j is the set of observations in cluster j.

The temporally-aware dissimilarity measure combines distances, both in the multidimensional space $\mathcal{D}$ and in the time space $\mathcal{T}$, and is described as follows:

$$\begin{aligned} \|x_i - x_j\|_{TA} = 1 - \left(1 - \gamma_d \frac{\|x_i^d - x_j^d\|^2}{\Delta x_{max}^2}\right) \times \\ \left(1 - \gamma_t \frac{\|x_i^t - x_j^t\|^2}{\Delta t_{max}^2}\right), \end{aligned} \tag{2.19}$$

where γ_d is the weight given to the multidimensional component of the temporally-aware dissimilarity measure, and γ_t is the weight of the temporal component. The cost function $w(x_i, x_j)$ encourages temporally-adjacent observations that belong to the same entity to be assigned to the same cluster. The cost function is defined as:

$$w(x_i, x_j) = \beta \times e^{-\frac{1}{2}\left(\frac{\|x_i^t - x_j^t\|}{\delta}\right)^2} \mathbb{1}[x_i^\phi = x_j^\phi] \tag{2.20}$$

[59] compares TDCK to other methods available in the literature – such as TCK-means [58], Constrained k-means, Temporal-Driven k-means – and proves it to be the most effective approach to temporal clustering.

Chapter 3

Proposed Semi-Supervised Temporal Clustering

The problem addressed in this research is different from simple clustering of time series, which refers to the problem of clustering time series when they are already segmented. More specifically, it is a temporal clustering problem, which is explained in Section 3.1. The main goal of this chapter is to explain the process of adding side information to a temporal clustering algorithm. The approaches proposed in this chapter are an extension of aligned cluster analysis (ACA), a temporal-clustering method, which is described in Section 3.2. The particular mechanism of which side information is added to the clustering method is based on the semi-supervised kernel k-means framework, which is explained in Section 3.3. Sections 3.4 and 3.5 detail the two proposed methods, semi-supervised aligned cluster analysis (SSACA) and semi-supervised temporal spectral clustering (SSTSC), and Section 3.6 discusses a constraint-propagation mechanism used to extend SSACA and SSTSC. The extended methods are discussed in Sections 3.7 and 3.8. Finally, Section 3.9 elaborates on the use of constraints on semi-supervised methods, and Section 3.10 explains the differences between the proposed methods and subsequence time series (STS).

3.1 The Problem of Temporal Clustering

Given a time series $X = [x_1, \ldots, x_n] \in \mathbf{R}^{d \times n}$, find a vector $\mathbf{s} \in \mathbb{R}^m$ that contains the start positions of m segments, where m is the total number of segments, and each segment $X_{[s_j, s_{j+1})}$ starts at the position s_j, and ends at the position $s_{j+1} - 1$, such that similar

segments are grouped into k clusters, as illustrated in Figure 3.1. In other words, temporal clustering (TC) performs temporal segmentation while simultaneously grouping segments into similar clusters. Although TC is a relatively less explored problem, it is especially important for applications such as human motion analysis, audio-visual emotion analysis, animal behaviour analysis, and speaker diarization.

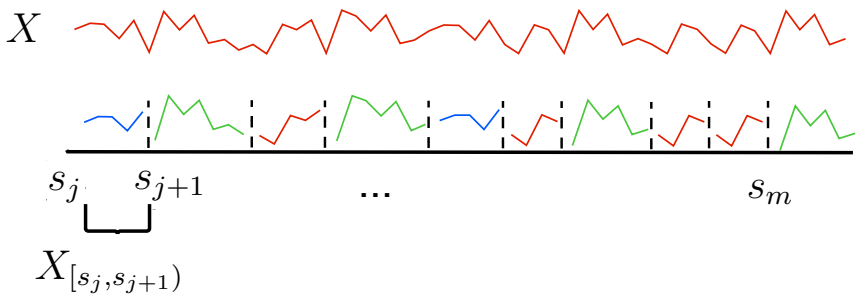


Figure 3.1: Temporal Clustering

Aligned Cluster Analysis, which is discussed in Section 3.2, is a method that tackles this problem by minimizing an objective function through dynamic programming (DP).

3.2 Aligned Cluster Analysis (ACA)

Aligned cluster analysis (ACA) [7] is an extension of the kernel k-means clustering algorithm that takes into account temporal ordering of frames. ACA combines kernel k-means with dynamic time alignment kernel (DTAK) (see Section 2.2.2 for details)

The goal of ACA is to decompose a segment $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$ into m disjoint segments, where each segment belongs to a single k cluster. Each segment is constrained by a maximum length n_{max}, which controls the temporal granularity of the segmentation, and it is defined as an input parameter. The ith segment $\mathbf{Y}_i = \mathbf{X}_{[s_i, s_i+1)} = [\mathbf{x}_{s_i}, \ldots, \mathbf{x}_{s_i+1}]$ begins at position s_i and ends at $s_{i+1} - 1$, such that $n_i = s_{i+1} - s_i \leq n_{max}$. An indicator matrix $\mathbf{G} \in \{0, 1\}^{k \times m}$ assigns each segment to a cluster: $g_{ci} = 1$ if $\mathbf{Y}_i$ belongs to cluster c; otherwise, $g_{ci} = 0$.

ACA achieves temporal clustering by minimizing:

$$
\begin{aligned}
J_{aca}(\mathbf{G}, \mathbf{s}) = \sum_{c=1}^{k} \sum_{i=1}^{m} g_{ci} \underbrace{\|\psi(\mathbf{X}_{[\mathbf{s_i}, \mathbf{s_{i+1}})}) - \mathbf{z_c}\|^2}_{dist^2_{\psi}(\mathbf{Y}_i,\, \mathbf{z}_c)} = \\
\|[\psi(\mathbf{Y}_1), \ldots, \psi(\mathbf{Y}_m) - \mathbf{ZG}\|_F^2, \\
s.t. \mathbf{G}^T 1_k = 1_m \text{ and } s_{i+1} - s_i \in [1, n_{max}],
\end{aligned}
\tag{3.1}
$$

where $\mathbf{G} \in \{0,1\}^{k \times m}$ is a cluster indicator matrix, and $\mathbf{s} \in \mathbb{R}^{m+1}$ is the vector that contains the start[1] of each segment.$\mathbf{Y} = \mathbf{X}_{[s_i, s_{i+1})}$ indicates a segment. In the case of ACA, the $dist^2_{\psi}(\mathbf{Y}_i, \mathbf{z}_c)$ is the squared distance between the ith segment and the centre of cluster c in the nonlinear mapped feature space represented by $\psi(.)$; that is,

$$
dist^2_{\psi}(\mathbf{Y}_i, \mathbf{z}_c) = \tau_{i,i} - \frac{2}{m_c} \sum_{j=1}^{m} g_{cj} \tau_{ij} + \frac{1}{m_c^2} \sum_{j_1, j_2 = 1}^{m} g_{cj_1} g_{cj_2} \tau_{j_1 j_2}, \tag{3.2}
$$

where $m_c = \sum_{j=1}^{m} g_{cj}$ represents the total number of segments in cluster c, and the dynamic kernel function τ is given by $\tau_{ij} = \psi(\mathbf{Y}_i)^T \psi(\mathbf{Y}_j)$. Equation 2.5 shows this calculation using DTAK.

Optimizing the temporal clustering problem described in Equation 3.2 is NP-hard. However, by applying a coordinate-descent scheme that uses dynamic programming to compute $\mathbf{s}$ and $\mathbf{G}$ with a winner-takes-all strategy, it is possible to produce an efficient solution.

3.2.1 Optimization of ACA

The strategy proposed by [25] consists of breaking the problem into the following subproblems at each iteration:

$$
\mathbf{G}, \mathbf{s} = \arg\min_{\mathbf{G}, \mathbf{s}} J_{aca}(\mathbf{G}, \mathbf{s}) = \arg\min_{\mathbf{G}, \mathbf{s}} \sum_{c=1}^{k} \sum_{i=1}^{m} g_{ci} dist^2_{\psi}(\mathbf{Y}_i, \dot{\mathbf{z}}_c) \tag{3.3}
$$

[1] There is a dummy position $s_{m+1} = n + 1$ kept for the last segment.

where $\dot{\mathbf{z}}_c$ is the implicitly computed cluster mean from the segmentation $(\dot{\mathbf{G}}, \dot{\mathbf{s}})$, calculated in the previous step. Trying to find all the possible segments by brute force is an infeasible task: for a segment of length n, the number of possible segments $\mathbf{s}$ is $O(2^n)$. However, using a dynamic-programming-based algorithm, it is possible to make the complexity polynomial instead of exponential.

Equation 3.3 can be rewritten as

$$J_{aca}(\mathbf{G}, \mathbf{s}) = \sum_{c=1}^{k} \sum_{i=1}^{m} g_{ci} dist_{\psi}^{2}(\mathbf{Y}_i, \mathbf{z}_c). \tag{3.4}$$

To take advantage of the relationship between $\mathbf{G}$ and $\mathbf{s}$ an auxiliary function, $J(\cdot) : [1, n] \rightarrow \mathbb{R}$,

$$J(v) = \min_{\mathbf{G},\mathbf{s}} J_{aca}\left(\mathbf{G}, \mathbf{s}\right)|_{\mathbf{x}_{[1,v]}} \tag{3.5}$$

was created. This auxiliary function relates the minimum energy directly with the tail position v of the subsequence $\mathbf{X}_{[1,v)} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_v]$. Moreover, $J(\cdot)$ satisfies the principle of optimality, i.e.,

$$J(v) = \min_{1<i\leq v} \left(J(i-1) + \min_{\mathbf{G},\mathbf{s}} J_{aca}(G, s)|_{\mathbf{x}_{[1,v]}} \right), \tag{3.6}$$

which suggests that an optimal decomposition of subsequence $\mathbf{X}_{[1,v]}$ is only achieved when the two sides $\mathbf{X}_{[1,i-1]}$ and $\mathbf{X}_{[i,v]}$ are optimal, and their sum is minimal. By using Bellman's equation,

$$J(v) = \min_{v-n_{max}<i\leq v} \left(J(i-1) + \min_{g} \sum_{c=1}^{k} g_c dist_{\psi}^{2}(\mathbf{X}_{[i,v]}, \dot{\mathbf{z}}_c) \right), \tag{3.7}$$

where $dist_{\psi}^{2}(\mathbf{X}_{[i,v]}, \dot{\mathbf{z}}_c)$ is the squared distance between the segment $\mathbf{X}_{[i,v]}$ and the centre of the cluster c:

$$\begin{aligned} dist_{\psi}^{2}(\mathbf{X}_{[i,v]}, \dot{\mathbf{z}}_c) = \tau(\mathbf{X}_{[i,v]}, \mathbf{X}_{[i,v]}) - \frac{2}{\dot{m}_c} \sum_{j=1}^{\dot{m}} \dot{g}_{cj} \tau(\mathbf{X}_{[i,v]}, \dot{\mathbf{Y}}_j) + \\ \frac{1}{\dot{m}_c^2} \sum_{j_1, j_2=1}^{\dot{m}} \dot{g}_{cj_1} \dot{g}_{cj_2} \tau(\dot{\mathbf{Y}}_{j_1}, \dot{\mathbf{Y}}_{j_2}). \end{aligned} \tag{3.8}$$

Once $v = n$, the function $J(n)$ possesses the optimal segmentation that is being sought. The inner values i_v^* and $\mathbf{g}_v^* = \arg\min_{i,g} J(v)$ are the intermediate values of head position and label of the last segment, respectively.

The ACA algorithm is divided into two phases: forward step and backward step. Before beginning the forward step, the sequence X of length n is randomly segmented. In the forward step, the algorithm scans from the beginning ($v = 1$) of the sequence to its end ($v = n$). For each v, the respective $J(v)$ is computed as described in Equation 3.7. For every position $i - n_{max} < i < v - 1$, the DTAK between the segment $X_{[i,v]}$ and each segment Y_j of each of the clusters, which is precomputed in the last iteration. Because the mean of each cluster cannot be explicitly calculated, the values that are stored internally are the head position i_v^*, the label $\mathbf{g}_v^*$, and $J(v)$ that has the lowest value. Once the forward step is complete, the algorithm starts the backward step, which traces back from the end of the sequence ($v = n$) and cuts off the segment whose head $s = i_v^*$. The indicator vector $g = g_v^*$ can be retrieved from the stored positions. This operation is repeated for the entire sequence ($v = i_v^* - 1$). Both steps are repeated until $J(n)$ converges.

3.3 Applying Semi-Supervised Framework to ACA

A theoretical equivalence between weighted kernel k-means and graph clustering has been shown, unifying several vector- and graph-based approaches [5]. One implication of this unification is that a single algorithm for semi-supervised clustering includes several objective functions in the same framework as special cases. Weighted Kernel k-means objective function captures a number of semi-supervised clustering objective functions, such as HMRF k-means. HMRF k-means, with a certain distance and class of constraint penalty function, can be expressed as a special case of weighted kernel k-means.

The semi-supervised clustering objective of HMRF can be expressed as

$$\begin{aligned} J_{HMRF}(\mathbf{Z}, \mathbf{G}) = & \sum_{c=1}^{k} \sum_{i=1}^{n} g_{ci} \|\mathbf{x}_i - \mathbf{z}_c\|^2 \\ & - \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{M} \\ g_i = g_j}} w_{ij} + \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{C} \\ g_i = g_j}} w_{ij}, \end{aligned} \tag{3.9}$$

where $\mathcal{M}$ is the set of *must-link* constraints, $\mathcal{C}$ is the set of *cannot-link* constraints, w_{ij} is the constant cost for violating or complying with a constraint $\mathbf{x}_i$ and $\mathbf{x}_j$, and g_i refers

to the cluster label of $\mathbf{x}_i$. There are three terms in this objective function. The first term is the unsupervised k-means term of the objective function. The distance $\|\mathbf{x_i} - \mathbf{z_c}\|^2$ can be represented as a matrix of pairwise squared Euclidean distances among the data points (see [5] for proof). We refer later to this distance matrix as $\mathbf{S}$. The second term is based on the *must-link* constraints, which state that, every $\mathbf{x}_i$ and $\mathbf{x}_j$ that are *must-links*, and are in the same cluster, have satisfied that constraint, and, as a result, the objective function is rewarded by subtracting some pre-specified weight. Similarly, the third term in the objective function states that, every $\mathbf{x}_i$ and $\mathbf{x}_j$ that are *cannot-links*, and are in the same cluster, have violated that constraint, so the objective function is penalized by some pre-specified penalty weight. Later, the second and third terms of the function are incorporated in a single matrix $\mathbf{W}$.

As previously mentioned, for the equivalence of the HMRF k-means and the weighted kernel k-means to hold, it is necessary to construct a specific kernel matrix and set weights in a certain way. A kernel matrix $\mathbf{K}$ should have two components: $\mathbf{K} = \mathbf{S} + \mathbf{W}$. $\mathbf{S}$ is the similarity matrix and comes from the unsupervised term, while $\mathbf{W}$ is the constraint matrix, which incorporates the penalties and rewards of the constraints. Thus, this objective function is mathematically equivalent to the weighted kernel k-means objective function. In other words, weighted kernel k-means can be used to decrease the objective function.

It may be the case that $\mathbf{K}$ is not positive semidefinite, a requirement for the kernel k-means to converge. This problem can be avoided by performing diagonal shifting [5], which does not change the global optimal solution. However, in practice, it was observed that diagonal shifting was not necessary in order to guarantee convergence.

ACA, as discussed in Section 3.2, is derived from kernel k-means. In fact, ACA is a segment-based kernel k-means controlled by a length constraint n_{max}. If $n_{max} = 1$, each segment consists of a single frame, and ACA becomes a special case of kernel k-means.

3.4 Semi-Supervised ACA (SSACA)

In semi-supervised aligned cluster analysis (SSACA), the equivalence of ACA and weighted kernel k-means is used to incorporate the kernel k-means semi-supervised framework into ACA. Therefore, the two cost terms, which are derived from the *must-link* and *cannot-link*, are incorporated into the objective function. The *must-links* and *cannot-links* are informed as inputs to the algorithm, defined by the user in the form of pairs of segments (vectors containing the start and end of each segment). Thus, the objective function of SSACA can be written as

$$J_{ssaca}(\mathbf{G}, \mathbf{s}) = \sum_{c=1}^{k} \sum_{i=1}^{m} g_{ci} \underbrace{\|\psi(\mathbf{X}_{[\mathbf{s_i}, \mathbf{s_{i+1}})}) - \mathbf{z_c}\|^2}_{dist^2_{\psi}(\mathbf{Y}_i,\, \mathbf{z}_c)} - \sum_{\substack{\mathbf{Y}_i, \mathbf{Y}_j \in \mathcal{M} \\ g_i = g_j}} w_{ij} + \sum_{\substack{\mathbf{Y}_i, \mathbf{Y}_j \in \mathcal{C} \\ g_i = g_j}} w_{ij}, \tag{3.10}$$

where $\mathcal{M}$ is the set of *must-link* constraints, $\mathcal{C}$ is the set of *cannot-link* constraints, and w_{ij} is the constant cost for violating or respecting a constraint between $\mathbf{Y}_i$ and $\mathbf{Y}_j$. $\mathbf{G} \in \{0,1\}^{k \times m}$ is an indicator matrix that assigns each segment to a cluster: $g_{ci} = 1$ if $\mathbf{Y}_i$ belongs to cluster c; otherwise, $g_{ci} = 0$, and $\mathbf{s} \in \mathbb{R}^{m+1}$ is the vector that contains the start of each segment, and as described in Equation 3.3. The distance $dist^2_{\psi}(\mathbf{Y}_i, \mathbf{z}_c)$ is explained in Equation 3.2. Like the objective function of HMRF k-means, the objective function of SSACA can be represented by three terms, an unsupervised term, a *must-link* term, and a *cannot-link* term. These terms can be written in matrix form.

$\mathbf{T} = [\tau_{ij}]_{m \times m} \in \mathbb{R}^{m \times m}$ is the segment kernel matrix that represents the similarity of the segments using DTAK, and is the matrix representation on the unsupervised term of the objective function. Because the constraints are held in the segment level, there are two kernel matrices: $\mathbf{K}$ and $\mathbf{T}$. $\mathbf{K}$ is the frame kernel matrix, $\mathbf{K} = \phi(\mathbf{X})^T \phi(\mathbf{X}) \in \mathbb{R}^{n \times n}$, and each entry k_{ij} defines the similarity between two frames, $\mathbf{x_i}$ and $\mathbf{x_j}$. To use the same framework, the kernel matrix should be constructed as $\tilde{\mathbf{T}} = \mathbf{T} + \mathbf{W}$, where $\mathbf{T}$ represents the similarity matrix in the segment level described previously, and $\mathbf{W}$ is the constraint matrix. $\mathbf{W} = [\overline{w}_{ij}]_{m \times m} \in \mathbb{R}^{m \times m}$ represents the *must-link* and *cannot-link* term of the objective function, where $\overline{w}_{ij} = w_{ij}$ for *must-links* and $\overline{w}_{ij} = -w_{ij}$ for *cannot-links*. Figure 3.2 illustrates the different matrices.

The SSACA algorithm in its entirety consists of two main steps. The first step is the inclusion of the constraints in the similarity matrix, represented by step 1 of Algorithm 2. The second step, explained above, is the search process for optimizing the objective function, represented by step 5 of Algorithm 2. This search process is repeated until no more changes in the final partition of the data are observed, or until the process reaches a maximum number of iterations.

The search algorithm that searches for the segmentation $(\dot{\mathbf{G}}, \dot{\mathbf{s}})$ which minimizes J is discussed in the next section.

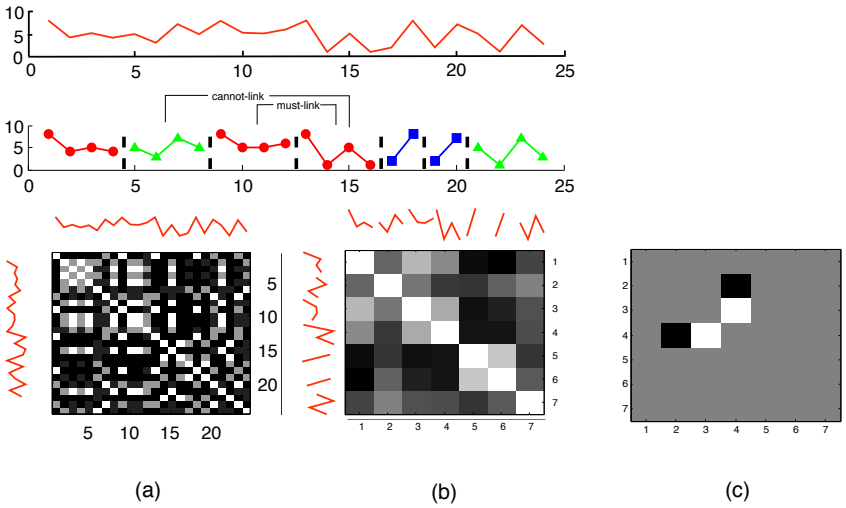


Figure 3.2: (a) shows the frame kernel matrix $\mathbf{K}$, where each entry k_{ij} defines the similarity between two frames, $\mathbf{x_i}$ and $\mathbf{x_j}$. (b) shows the segment kernel matrix calculated by DTAK. (c) shows the constraint matrix $\mathbf{W}$.

3.4.1 Optimizing SSACA

The optimization of SSACA's objective function is a non-convex problem. ACA adopts a dynamic-programming-based (DP-based) algorithm, which has a complexity of $O(n^2 n_{max})$ to examine all possible segmentations as its primary method. In SSACA, an adaptation of the DP-based search into a semi-supervised framework is proposed, in which pairwise constraints are incorporated to the algorithm. The same general strategy adopted in ACA to break the problem into subproblems is adopted for SSACA. At every iteration, the following subproblem is solved:

$$\begin{aligned}\mathbf{G}, \mathbf{s} = \arg\min_{\mathbf{G},\mathbf{s}} J_{ssaca}(\mathbf{G}, \mathbf{s}) = \arg\min_{\mathbf{G},\mathbf{s}} \sum_{c=1}^{k}\sum_{i=1}^{m} g_{ci} dist_{\psi}^{2}(\mathbf{Y}_i, \dot{\mathbf{z}}_c) \\ - \sum_{\substack{\mathbf{Y}_i,\mathbf{Y}_j \in \mathcal{M} \\ g_i = g_j}} w_{ij} + \sum_{\substack{\mathbf{Y}_i,\mathbf{Y}_j \in \mathcal{C} \\ g_i = g_j}} w_{ij},\end{aligned} \tag{3.11}$$

where $\dot{\mathbf{z}}_c$ is the implicitly-computed cluster mean from the segmentation $(\dot{\mathbf{G}}, \dot{\mathbf{s}})$, calculated in the previous step.

Algorithm 2 SS Aligned Cluster Analysis (SSACA)

input: $\mathbf{K} \in \mathbb{R}^{n \times n}$: input frame kernel matrix, $\mathbf{T} \in \mathbb{R}^{m \times m}$: input segment kernel matrix, $\mathbf{W} \in \mathbb{R}^{m \times m}$: constraint penalty, k: number of clusters, $\mathcal{M}$: set of *must-link* constraints, $\mathcal{C}$: set of *cannot-link* constraints, $\dot{s}$: initial segmentation , t_{max}: optional maximum number of iterations.

output: $\mathrm{G} \in \{0,1\}^{k \times n}$: Final partitioning of the points, s: Final segmentation.

1: Form the kernel matrix $\tilde{\mathbf{T}} = \mathbf{T} + \mathbf{W}$.
2: Diagonal-shift $\tilde{\mathbf{T}}$ by adding σI to guarantee positive definiteness of $\tilde{\mathbf{T}}$.
3: Get initial clusters $G^{(0)}$ using constraints.
4: **while** $G \neq G^0$ **or** $iter > t_{max}$ **do**
5: $\quad G, s = \text{SSDPSearch}(G^{(0)}, \dot{s}, \mathbf{K}, \tilde{\mathbf{T}}, \mathcal{M}, \mathcal{C}, k)$.
6: $\quad G^0 \leftarrow G$;
7: $\quad iter = iter + 1$;
8: **end while**
9: Return G, s;

SSACA can be rewritten as the sum of the following distances:

$$J_{ssaca}(\mathbf{G}, \mathbf{s}) = \sum_{c=1}^{k} \sum_{i=1}^{m} g_{ci} dist_{\psi}^{2}(\mathbf{Y}_i, \mathbf{z}_c) - \sum_{\substack{\mathbf{Y}_i, \mathbf{Y}_j \in \mathcal{M} \\ g_i = g_j}} w_{ij} + \sum_{\substack{\mathbf{Y}_i, \mathbf{Y}_j \in \mathcal{C} \\ g_i = g_j}} w_{ij}. \tag{3.12}$$

Similar to ACA, an auxiliary function $J(\cdot) : [1, n] \rightarrow \mathbb{R}$ (see Equations 3.5 and 3.6) that satisfies the principle of optimality can be introduced to form a direct relation between the minimum energy and the tail position v of the subsequence $\mathbf{X}_{[1,v)} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_v]$. By using Bellman's equation, the following optimization problem can be derived:

$$\begin{aligned} J(v) = \min_{v - n_{max} < i \leq v} \Bigg(& J(i-1) + \min_{g} \sum_{c=1}^{k} g_c dist_{\psi}^{2}(\mathbf{X}_{[i,v]}, \dot{\mathbf{z}}_c) \\ & - \sum_{\substack{\mathbf{Y}_i, \mathbf{Y}_j \in \mathcal{M} \\ g_i = g_j}} w_{ij} + \sum_{\substack{\mathbf{Y}_i, \mathbf{Y}_j \in \mathcal{C} \\ g_i = g_j}} w_{ij} \Bigg), \end{aligned} \tag{3.13}$$

where $dist_{\psi}^{2}(\mathbf{X}_{[i,v]}, \dot{\mathbf{z}}_c)$ is the squared distance between the segment $\mathbf{X}_{[i,v]}$ and the centre of the cluster c, as given by Equation 3.8.

Directly solving Equation 3.13 results in a complexity of $O(n^2 n_{max}^2)$, which corresponds to computing $\tau(\mathbf{X}_{[i,v]}, \dot{\mathbf{Y}})$ for all i, v, and j. However, by using dynamic programming, it is possible to minimize J more efficiently. The algorithmic implementation of Equation 3.13 that uses dynamic programming to solve J by using a cumulative kernel matrix, proposed by [25], was adapted to make use of the extra information given by the pairwise constraints.

Calculating $\tau(\mathbf{X}_{[i,v]}, \dot{\mathbf{Y}})$ effectively involves using a cumulative $\mathbf{U} \in \mathbb{R}^{n_v \times \dot{n}_j}$, where $n_v = v - i + 1$ and $\dot{n}_j$ are respective sizes of the segments $\mathbf{X}_{[i,v]}$ and $\dot{\mathbf{Y}}$. In the efficient implementation, $\mathbf{U} \in \mathbb{R}^{n_{max} \times n_{max} \times n}$ is a matrix that retains the kernel calculations of all previously calculated segments; therefore, the calculation uses the previous results, e.g. $\tau(\mathbf{X}_{[i,v-1]}, \dot{\mathbf{Y}})$, to quickly calculate the next segment $\tau(\mathbf{X}_{[i,v]}, \dot{\mathbf{Y}})$. In other words,

$$\tau(\mathbf{X}_{[i,v]}, \dot{\mathbf{Y}}) = \frac{\mathbf{U}(n_v, \bar{v}, \dot{s}_{j+1} - 1)}{n_v + \dot{n}_j},$$
$$\text{where } \mathbf{U}(n_v, \bar{v}, \dot{s}) = \max \begin{cases} \mathbf{U}(n_v, \bar{v}, \dot{s} - 1) + k_{v\dot{s}} \\ \mathbf{U}(n_v - 1, \overline{v-1}, \dot{s} - 1) + 2k_{v\dot{s}} \\ \mathbf{U}(n_v - 1, \overline{v-1}, \dot{8}) + k_{v\dot{s}}, \end{cases} \tag{3.14}$$

where $\dot{s} \in [\dot{s}_j, \dot{s}_{j+1})$ and $\mathbf{U}$ is initialized at $\mathbf{U}(1, v, s) = 2k_{v\dot{s}_j}$ for each $v \in [1, n]$ and $j \in [1, \dot{m}]$. However, since some segments are known a priori based on the pairwise constraints, the algorithm does not need to check for every $v \in [1, n]$. Instead, when v, which is the tail part of the segment being calculated, is part of a known segment $\overline{\mathbf{X}_{[i,v]}}$, the algorithm can skip all the intermediate steps that process the interval from $[i, v]$ in order to calculate $\mathbf{U}$ and retrieve the segment similarities directly from the initial segmentation kernel matrix $\mathbf{T}$, which contains the correct segmentation of the known segments. Empirically speaking, the algorithm can skip some of the processing when the search process reaches a known segment. The search process follows the same strategy as ACA, and divides the process into two phases:

1. **Forward phase:** The forward phase consists of scanning the entire sequence, from $v = 1$ to $v = n$, where $v \cap \overline{\mathbf{X}}_{[\mathbf{i},\mathbf{v}]} = \emptyset$. At every interval $i - n_{max} < i < v - 1$, the sequence $\mathbf{X}_{[i,v]}$ has its similarity measured against the segments Y_j precomputed in the previous iteration by DTAK, as described in Equation 3.14. The computed values for the head position i_v^*, the label $\mathbf{g}_v^*$, and $J(v)$ with the lowest values are stored in a table, which will be traced back in the backward phase.

2. **Backward phase:** The backward phase consists of tracing backwards from the end of the sequence ($v = n$) and cutting off the sequence where the head $s = i^*$. The

indication vector $g = g_v^*$ can be indexed from the stored positions. This operation is repeated until v, which is given by $(v = i_v^* - 1)$, is $v = 1$.

This process is repeated until $J(n)$ converges, or, in other words, until no more changes in the segmentation S and indication matrix G occur. The complete process is depicted in Figure 3.3.

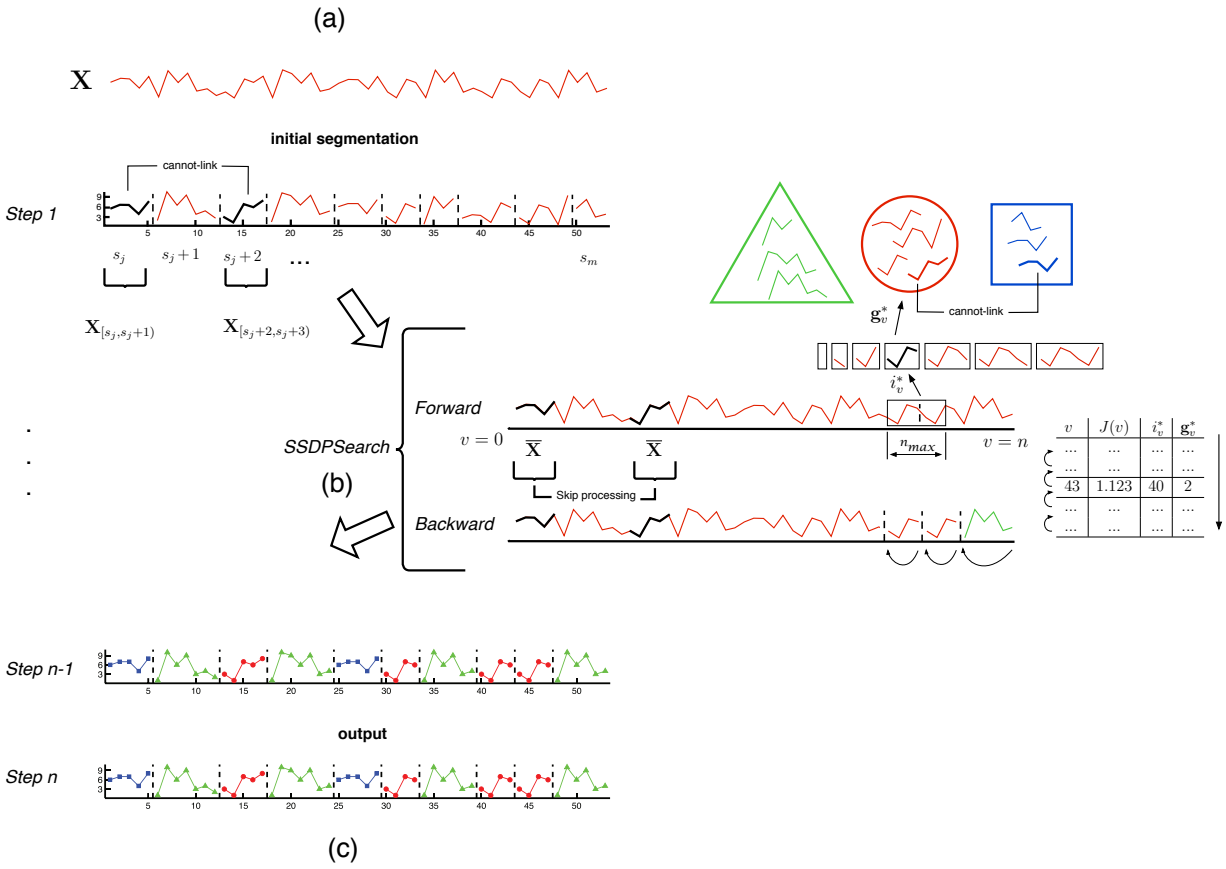


Figure 3.3: Optimization of SSACA in a 1D sample. (a) Initial segmentation of sequence $\mathbf{X}$ into m subsequences s. The *cannot-link* constraint assists the segments $\mathbf{X}_{[s_j,s_j+1)}$ and $\mathbf{X}_{[s_j+2,s_j+3)}$. (b) Search process using forward phase and backward phase applied to every step of the optimization. The table keeps track of the head position i_v^*, the label $\mathbf{g}_v^*$, and the $J(v)$ with the lowest values. (c) Converged segmentation.

The proposed modified algorithm that performs these tasks is SS DPSearch, outlined in Algorithm 3. SS DPSearch optimizes SSACA with respect to $\mathbf{G}$ and $\mathbf{s}$, while rewarding

or penalizing the distance between segments, $\tau(X_{[i,v]}, \dot{Y}_j)$, according to the given pairwise constraints $\mathcal{M} \in \mathbb{Z}^{n_{ml} \times 2}$ and $\mathcal{C} \in \mathbb{Z}^{n_{cl} \times 2}$.

Algorithm 3 SS DPSearch

parameter: n_{max}, k, n_{ml}, n_{cl}
input: G $\in \{0,1\}^{k \times \dot{m}}$, $\dot{\mathrm{s}} \in \mathbb{R}^{(\dot{m}+1)}$, $\mathbf{K} \in \mathbb{R}^{n \times n}$, $\mathbf{T} \in \mathbb{R}^{m \times m}$, $\mathcal{M} \in \mathbb{Z}^{n_{ml} \times 2}$, $\mathcal{C} \in \mathbb{Z}^{n_{cl} \times 2}$
output: G $\in \{0,1\}^{k \times m}$, $\dot{\mathrm{s}} \in \mathbb{R}^{(m+1)}$

```
headTail = getHeadTails(M, C);
{Forward step }
for v = 1 to n do
    J(v) ← ∞;
    if v ≥ headTail(:,1) and v < headTail(:,2) then
        continue;
    end if
    if isTail(v) then
        for j = 1 to ṁ do
            Retrive directly from T(X_[i,v], Ẏ_j);
        end for
        c* ← arg min_c dist_ψ(X_[i,v], ż_c); J ← dist_ψ(X_[i,v], ż_c*);
        J([i, v]) ← J, g*_[i,v] ← e*_c, i*_[i,v] ← i;
    else
        for n_v = 1 to min(n_max, v) do
            { Same as DPSearch [25]}
        end for
    end if
end for
{Perform backward step}
while v > 0 do
    Create a segment Y = X_[i*_v,v] with the label g*_v;
    v ← i*_v − 1;
end while
```

As parameters, SS DPSearch receives the length constraint n_{max} and the number k of clusters. As inputs, SS DPSearch requires an initial indicator matrix $\mathbf{G} \in \{0,1\}^{k \times m}$ that assigns segments to a cluster, an initial segmentation $\dot{\mathrm{s}} \in \mathbb{R}^{(\dot{m}+1)}$, and the frame kernel matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$. In addition to the inputs of the original approach, there are pairs of *must-links* $\in \mathbb{Z}^{n_{ml} \times 2}$, *cannot-links* $\in \mathbb{Z}^{n_{cl} \times 2}$, and $\mathbf{T} \in \mathbb{R}^{m \times m}$, which is the segment kernel matrix.

3.4.2 Complexity Analysis of SSACA

The bottleneck of the search exists in the update of matrix $\mathbf{U}$. When updating $\mathbf{U}$, the algorithm requires that all the segments $\sum_{j=1}^{\dot{m}} \dot{s}_{j+1} - \dot{s_j} \equiv n$, which correspond with the total number of frames, be analyzed. Therefore, the estimated time-cost of DPSearch is $O(n^2 n_{max})$. Considering the number (t) of iterations, the overall time complexity becomes $O(n^2 n_{max})t$.

The time-complexity of SSACA is efficient because it skips the computation of the frames of some segments. For instance, if two segments S_j and S_{j+1} are used as *cannot-link* constraints, they are automatically a known segment; as a result, all the frames that belong to these two sequences can be skipped. The $\mathbf{U}$ matrix, which is calculated in a cumulative way by adding a new column every time a new frame v is presented, can be directly fetched specifically for this segment from the kernel segment matrix $\mathbf{T}$ without any loss of accuracy.

The complexity of SSACA is influenced by the number of constraints n_c. Each constraint represents a pair of sequences S_j, S_{j+1}, which are formed by sets of frames n_v; these specific frames will be called constrained frames n_{cf}. Therefore, the more constraints available, the fewer the frames to be computed. The complexity notation becomes $O((n - (n_c n_v) + n_c)^2 n_{max} t)$, where $n_c n_v$ is the number of constrained frames, or, in a simpler fashion the complexity can be reduced to $O((n - n_{cf})^2 n_{max})$. The complexity is still quadratic; however, as the number of constraints increases, the complexity exponentially decreases.

The decrease in complexity of SSACA can be translated in terms of time-processing. Figure 3.4 shows the time of SSACA, as well as some variations of SSACA that are discussed in Section 3.6, compared to ACA.

The time-processing of the semi-supervised methods starts off more slowly. This behaviour is caused by some extra checking done by the algorithm, specifically, in lines 1 and 4 of Algorithm 3. However, the overhead caused by this extra checking is compensated for by skipping the processing of the known segments. The gains in the time-processing start to become visible as the number of constraints increase. The time-processing displayed in Table 3.4 does not account for the extra processing needed for the initial seeding, which consists of making sure that, when first initializing the clusters, the constrained segments are grouped consistently, according to their *must-link* and *cannot-links* definitions. However, in practice, inconsistent cluster initializations still managed to return good results, as evidenced by the experiments of section 5.1.5.

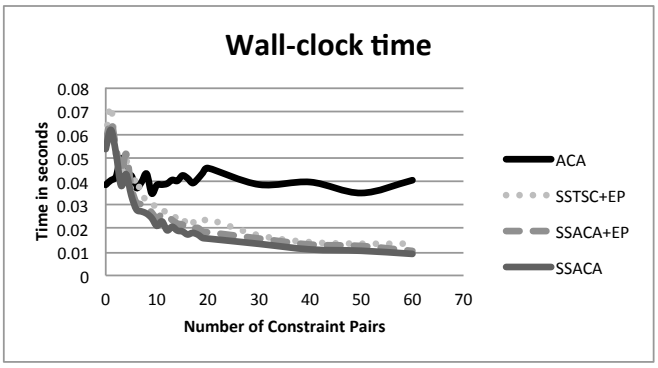


Figure 3.4: Wall time of the proposed semi-supervised methods compared to a completely unsupervised method.

3.5 Semi-Supervised Temporal Spectral Clustering

Spectral clustering is a graph-clustering algorithm that uses the eigenvectors of the similarities to reduce their dimensionality before clustering them. In other words, the clustering process takes place in a lower dimension. Two conditions are necessary in order to transform a simple spectral clustering to a semi-supervised temporal spectral clustering. The first of these conditions is the addition of extra knowledge, and the second is the consideration of time awareness. The same analogy used in ACA and kernel k-means to derive SSACA can be applied to spectral clustering. Extra knowledge is added in the form of a constraint matrix derived from *must-links* and *cannot-links*, and time awareness is introduced by the DTAK measure embedded in the SSDPSearch.

To perform semi-supervised temporal spectral clustering, the algorithm needs to first find the eigenvectors corresponding to the k largest eigenvalues of $\tilde{\mathbf{T}}$, in order to derive the low-dimension feature vector, where $\tilde{\mathbf{T}}$ is a similarity graph or a similarity matrix modified by the pairwise constraints. Next, the algorithm performs the temporal clustering search, which adds the temporal aspect to this approach, using the dynamic-programming search, SSDPSearch. The first steps of the algorithm are described in Algorithm 4.

The kernel created in step 3 acts as the input for the SSDPSearch. The mechanism of adding extra information in the form of constraints is explained in section 3.8, along with the complete algorithm.

Algorithm 4 First Steps Spectral Clustering

1: Find k largest eigenvectors $v_1 \dots v_k$ of $\tilde{D}^{\frac{-1}{2}}\tilde{T}\tilde{D}^{\frac{-1}{2}}$, where $\tilde{D}$ is a diagonal matrix with its (i,i)-element equal to the sum of the i-th row of $\tilde{T}$;
2: Form $E = [v_1, \dots, v_k]$, and normalize each row to the unit length. The rows of E represent the low-dimensional features of segment $X_{[s_i, s_{i+1}]}$;
3: Create a kernel with the new feature vector $E_i (i = 1, \dots, m)$.

3.6 Exhaustive and Efficient Constraint Propagation

Pairwise constraints are used to penalize or reward constrained segments, in order to adjust the similarity matrix for the kernel k-means clustering algorithm. This technique, affects only the constrained segment similarities, however. To make the propagation of constraints more efficient, the idea of exhaustive and efficient constraint propagation (E^2CP) is borrowed from Lu and Ip [6] and adapted for the proposed framework. The rationale behind this method is to spread the effects of the constraints throughout the similarity matrix.

E^2CP tackles the problem of constraint propagation by decomposing it into sets of label propagation subproblems. Given the dataset $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$, a set of *must-link* $\mathcal{M}$, and a set of *cannot-link* $\mathcal{C}$, all the pairwise constraints can be represented as a single matrix $\mathbf{W} = [W_{ij}]_{n \times n}$, where

$$W_{ij} = \begin{cases} +1, & (x_i, x_j) \in \mathcal{M} \\ -1, & (x_i, x_j) \in \mathcal{C} \\ 0, & \text{otherwise.} \end{cases} \tag{3.15}$$

Each j-th column of $W_{\cdot j}$ can be seen as a two-class, semi-supervised learning problem, where the *positive class* ($W_{ij} > 0$) represents the segments that should be in the same cluster, and the *negative class* ($W_{ij} < 0$) represents the segments that should not be in the same cluster. If $(W_{ij}) = 0$, x_i x_j are not constrained. Each column and row are solved by label propagation in parallel [61], to ensure that all the segments will be affected by the propagation. The algorithm can be described as follows:

1. Create the similarity matrix T or a symmetric k-NN graph.
2. Create the matrix $\bar{\mathcal{L}} = D^{\frac{-1}{2}} T D^{\frac{-1}{2}}$, where D is a diagonal matrix with its (i,i)-element equal to the sum of the i-th row of T.

3. Iterate $F_v(t+1) = \alpha\bar{\mathcal{L}}F_v(t) + (1-\alpha)W$ for vertical constraint propagation until convergence, where $F_v(t) \in \mathcal{F}$, and α is a parameter between 0 and 1.

4. Iterate $F_h(t+1) = \alpha F_h(t)\bar{\mathcal{L}} + (1-\alpha)F_v^*$ for horizontal constraint propagation until convergence, where $F_h(t) \in \mathcal{F}$ and F_v^* is the limit of $\{F_{v(t)}\}$.

5. Output $F^* = F_h^*$ as the final representation of the pairwise constraints, where F_h^* is the limit of $\{F_{h(t)}\}$.

Intuitively, the similarities receive information from their neighbours at each iteration, and the parameter α controls the relative amount of information passed from that neighbours. Without loss of generality, [61] shows that $\{F(t)\}$ can be calculated in a closed form. The output F^* represents an exhaustive set of pairwise constraints, with associated confidence scores $|F^*|$. At this point, the similarities in T can be adjusted with the output scores of F^*, as described in Equation 3.16. The resulting similarity matrix $\tilde{T}$ has the same properties as the kernel matrix specified in the kernel k-means framework; it is both nonnegative and symmetric [6].

$$\tilde{T_{ij}} = \begin{cases} 1-(1-F_{ij}^*)(1-T_{ij}), & F_{ij}^* \geq 0 \\ (1+F_{ij}^*)T_{ij}, & F_{ij}^* < 0 \end{cases} \tag{3.16}$$

3.7 Semi-Supervised ACA with Exhaustive Propagation (SSACA+EP)

To improve the performance of SSACA, exhaustive propagation (EP) is added to the SSACA framework to create SSACA+EP. Step 1 of Algorithm 2 is replaced with two steps. First, the initial weights $\mathbf{W}$ are propagated to F^*, as described in the five steps of section 3.6. Later, the matrix $\tilde{\mathbf{T}}$ is created according to Equation 3.16. The rest of the algorithm remains unchanged. The complete algorithm can be seen in Algorithm 5.

3.8 Semi-Supervised Temporal Spectral Clustering with EP (SSTSC+EP)

In order to apply the same methodology as used in SSACA, and to comply with the semi-supervised kernel k-means framework, it is necessary to again form a kernel matrix as a

Algorithm 5 SSACA + EP (Exhaustive propagation)

input: $\mathbf{K} \in \mathbb{R}^{n \times n}$: input frame kernel matrix, $\mathbf{T} \in \mathbb{R}^{m \times m}$: input segment kernel matrix, $\mathbf{W} \in \mathbb{R}^{m \times m}$: constraint penalty, k: number of clusters, $\mathcal{M}$: set of must-link constraints, $\mathcal{C}$: set of cannot-link constraints, $\dot{s}$: initial segmentation, t_{max}: optional maximum number of iterations.

output: G $\in \{0,1\}^{k \times n}$: Final partitioning of the points, s: Final segmentation.

1: Propagate the constraints $F^* \leftarrow F$ by label propagation using $\mathbf{W}$.
2: Form the matrix $\tilde{\mathbf{T}}$ according to Equation 3.16.
3: Diagonal-shift $\tilde{\mathbf{T}}$ by adding σI to guarantee positive definiteness of $\tilde{\mathbf{T}}$.
4: Get initial clusters $G^{(0)}$ using constraints.
5: **while** $G \neq G^0$ **or** $iter > t_{max}$ **do**
6: $\quad G, s = \text{SSDPSearch}(G^{(0)}, \dot{s}, \mathbf{K}, \tilde{\mathbf{T}}, \mathcal{M}, \mathcal{C}, k)$.
7: $\quad G^0 \leftarrow G$;
8: $\quad iter = iter + 1$;
9: **end while**
10: Return G, s;

combination of similarity matrix and constraint. The output $\tilde{\mathbf{T}}$ generated by the exhaustive propagation method is a combination of the propagated constraints F^* and the similarity matrix $\mathbf{T}$, as described in Equation 3.16. $\tilde{\mathbf{T}}$ becomes, therefore, the new modified similarity matrix that will be used for the spectral clustering process. The complete algorithm is displayed in Algorithm 6.

3.9 Do Constraints Always Improve Performance?

The general assumption is that, if given constraints are drawn from correctly-labeled data and are free from noise, the performance of the algorithm will always increase. This observation is indeed true when performance is averaged over many different constraint sets. However, for a single set of constraints, no change or even a decrease in performance may occur. The reason why the average over different sets of constraints does not show this behaviour is simply that the magnitude of the increases is much higher than the negative losses. This phenomenon has been analyzed and empirically demonstrated by [62].

Some characteristics of the constraints can affect the accuracy of performance in either a positive or negative way. [62] introduces two measures that can qualify whether a set of constraints may increase or decrease the performance of the clustering algorithm:

Algorithm 6 SS Temporal Spectral Clustering + EP

input: $\mathbf{K} \in \mathbb{R}^{n \times n}$: input frame kernel matrix, $\mathbf{T} \in \mathbb{R}^{m \times m}$: input segment kernel matrix, $\mathbf{W} \in \mathbb{R}^{m \times m}$: constraint penalty, k: number of clusters, $\mathcal{M}$: set of must-link constraints, $\mathcal{C}$: set of cannot-link constraints, $\dot{s}$: initial segmentation.

output: G $\in \{0,1\}^{k \times n}$: Final partitioning of the points, s: Final segmentation.

1: Propagate the constraints $F^* \leftarrow F$ by label propagation using $\mathbf{W}$.
2: Form the matrix $\tilde{\mathbf{T}}$ according to Equation 3.16.
3: Diagonal-shift $\tilde{\mathbf{T}}$ by adding σI to guarantee positive definiteness of $\tilde{\mathbf{T}}$.
4: Get initial clusters $G^{(0)}$ using constraints.
5: Create a new kernel $\hat{\mathbf{T}}$ based on $\tilde{\mathbf{T}}$ using Algorithm 4.
6: Return $G, s = \text{SSDPSearch}(G^{(0)}, \dot{s}, \mathbf{K}, \hat{\mathbf{T}}, \mathcal{M}, \mathcal{C}, k)$.

informativeness and coherence.

Informativeness is defined as the amount of information available in the constraint set which the algorithm cannot determine using only its bias, created by the objective function. For example, a k-means-based algorithm is naturally biased to cluster together data that are closer in distance, and separate points that are further away; however, in a highly-informative constraint set, information that contradicts this bias would be more readily available through the constraints.

Coherence is defined as the amount of agreement within the constraints, with respect to a given distance measure. For example, in a situation where two pairs of constraints are parallel to each other in a Euclidean space, both pairs are expected to be part of a *must-link* set. However, if one pair is part of a *must-link* set, and the other is part of a *cannot-link* link set, these two pairs are incoherent with respect to Euclidean distance metrics.

By quantifying these measures, [62] shows empirically that highly-informative and coherent constraints are most likely to improve clustering-accuracy performance, whereas incoherent sets of constraints may drop the performance of the clustering algorithm. This thesis' focus is not to deeply analyze this phenomenon; however, it is important to acknowledge its relevance to semi-supervised temporal clustering.

3.10 Differentiating SSACA from Subsequence Time Series

Subsequence Time Series clustering was a popular technique in the late 1990s and early 2000s, used as a subroutine in many data mining algorithms. However, after research findings published by Keogh and Lin [2], STS lost its popularity. Their research showed that clustering time series subsequences extracted by a sliding window produced meaningless results. Because our proposed method is applied to time series, and includes a sliding window component in its process, it is understandable that concerns about the meaningfulness of proposed method might be raised. This section will respond to those concerns and demonstrate the fact that SSACA suffers from none of the same weaknesses as STS.

To show that SSACA is different from subsequence clustering, it is necessary to define subsequence clustering and whole clustering. According to [2], the definitions of these two processes are the following:

Subsequence Clustering: Given a single time series, subsequences are consecutively extracted by a sliding window, and then clustered.

Whole Clustering: Given a set of individual time series, much like in a conventional clustering algorithm, where similar objects are clustered together, the goal is to group similar time series into clusters.

A time series $T = [t_i, \ldots, t_n]$ can be defined as a set of n ordered points. A subsequence C_m of T is a sampling of length $w < n$ of contiguous positions from T; that is, $C = [t_m, \ldots, t_{m+w-1}]$ for $1 \leq m \leq n - w + 1$. A set S of all subsequences of a time series T can be constructed by a sliding window that spans the entire time series, starting from $[t_1, \ldots, t_w]$ and proceeding all the way to $[t_{n-w}, \ldots, t_n]$, where $S \in \mathbb{R}^{1 \times n}$ is a matrix of the subsequences generated by the sliding window. Figure 3.5 illustrates these definitions.

Keogh and Lin's work also shows that, if the subsequences defined in S are clustered, the produced outputs are independent of the input; in other words, the centre of the generated clusters are no different from randomly generated cluster centres. They demonstrate the existence of this behaviour in several databases, distance measures, and clustering algorithms. To pinpoint the differences between STS and SSACA, one of Keogh and Lin's experiments is reproduced here using the Cylinder-Bell-Funnel data [63], but subjected to the process of SSACA. This dataset consists of random instantiations of the eponymous

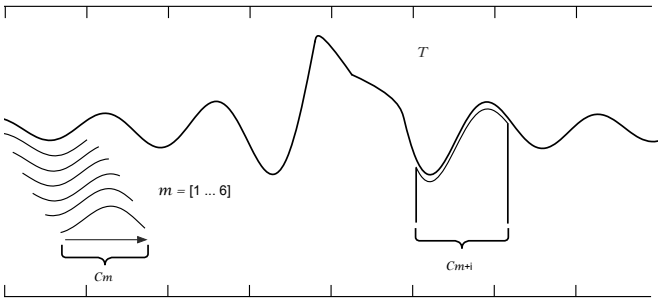


Figure 3.5: Sample time series T of length n, a subsequence in position $m+i$, and the first 6 subsequences extracted by a sliding window. Figure based on [2].

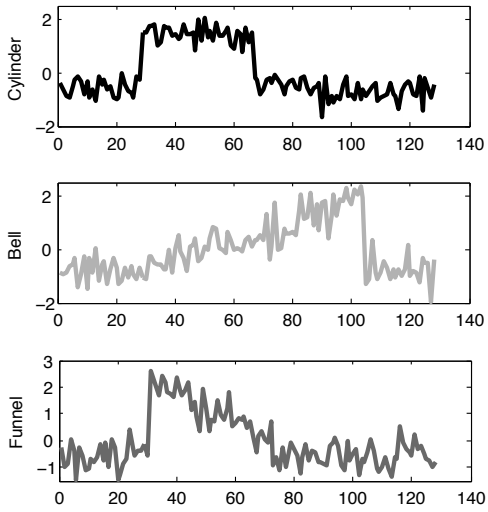


Figure 3.6: Samples of the three different patterns (Cylinder, Bell, and Funnel) of the CBF dataset.

patterns, with Gaussian noise added, and contains three different patterns, which can be seen in Figure 3.6. Each time series has a length of 128.

This analysis used a total of 128 time series, wherein 42 series corresponded with the Cylinder pattern, 42 with the Bell pattern, and 44 with the Funnel pattern. For the control experiment, whole clustering was performed on the set of 128 individual time series. To make this experiment compatible with the proposed method, a kernel k-means was used as the clustering algorithm, and DTAK as the distance measure. The centre of each cluster can be seen in Figure 3.7. Note that the centre of the clusters naturally resembles the samples of each pattern.

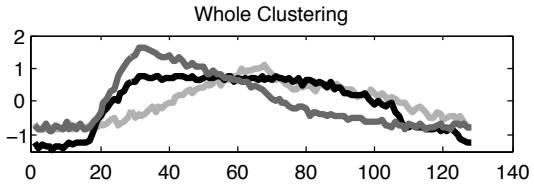


Figure 3.7: Cluster centers of the CBF dataset generated by kernel k-means. The shapes are similar to approximations of the original pattern.

For the subsequence analysis, the 128 time series were concatenated into a single time series in order to be able to extract subsequences. Then, subsequences were extracted using a sliding window of size 128 ($w = 128$), and clustered using kernel k-means with $k = 3$. The results were three close-to-perfect sine waves (see Figure 3.8), the same results reported by Keogh and Lin, in their explanation of the lack of meaningfulness derived from STS clustering.

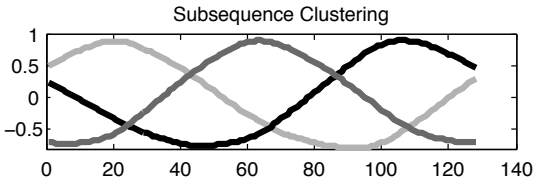


Figure 3.8: Cluster centers of the CBF dataset generated by sliding windows using kernel k-means. The shapes of the centers look like sine waves.

In stark contrast to STS, ACA attempts to cluster and segment at the same time, starting by pre-segmenting the time series into k partitions. After pre-segmenting, ACA searches within the different consecutive incremental segments of a subsequence C_m =

$[t_1][t_1t_2][t_1t_2t_3]\ldots[t_1\ldots t_w]$ for the segment that minimizes the distance to the centroids of the pre-segmented partitions. Later, the selected segment is assigned to the new cluster, and this process is repeated until no more changes are found. Therefore, the sliding window technique in ACA is a mechanism to find the size of the sequence that best fits the analyzing cluster.

To demonstrate that the technique used by the method proposed in this thesis, which is based on ACA, is different from STS clustering, one step of the ACA process was examined. Since the mechanism of temporal clustering and segmentation of ACA consists of several iterations of these steps, if step one is proven not to retrieve a meaningless result, the argument can thus be extrapolated to the whole method.

Given a subsequence C_m, generated by an incremental sliding window of final size w, clustered against an initially segmented time series T, there is an output subsequence that is closer to one of the cluster centres than to a sine wave.

The initial segmentation of the concatenated CBF dataset was clustered into k partitions. Then, centroids were calculated by averaging the segments that belonged to each k(th) cluster. The distances from one set of complete subsequences $C_m = [t_1][t_1t_2][t_1t_2t_3]\ldots[t_1\ldots t_w]$, which contains incrementing sized segments, to the centroid of each cluster were computed to find the subsequence that minimized the distance to the centroids, exactly as defined in ACA. The resulting subsequence is displayed in Figure 3.9, and is closer to one of the centroids of the CBF dataset than to a sine wave.

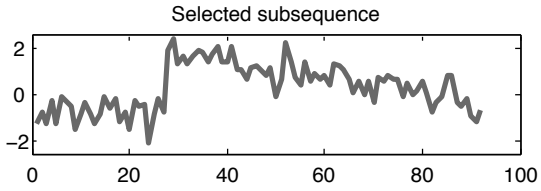


Figure 3.9: Retrieved subsequence generated by ACA. This sample is similar to the Funnel pattern of the CBF dataset.

The initial pre-segmentation of the time series was purposely chosen as 128, which is the size of each concatenated time series T, in order to give the algorithm a "head start." Obviously, the centroids retrieved by ACA in its first step were essentially whole clusterings of individual time series, as shown in Figure 3.7. However, the approach to initial segmentation of the time series does not matter. Any initial segmentation would ultimately lead to a converged result – not necessarily a global optimal, but at least a local optimal, as guaranteed by k-means based algorithms. The intent of this experiment is to

show that ACA starts with whole clustering, and only later draws on the sliding window mechanism. Therefore, the sliding window mechanism is merely a way of searching for the best segment that fits the analyzing cluster centres.

As explained previously, the mechanism used in ACA generates a retrieved segment that is closer to the centre of one of the patterns, and does not resemble the sine wave produced by the STS clustering in any shape or form. Therefore, the mechanism of clustering and segmenting used in ACA is based on whole clustering of segments of variable sizes.

Chapter 4

Experimental Analysis of the Proposed Methods

One of the goals of this chapter is to evaluate the proposed methods in comparison with other approaches, including the two baseline methods, ACA, a version of spectral clustering (SC) used in [25], and the native semi-supervised temporal method TDCK-means [55]. The reason for choosing TDCK as the comparative method is the fact that TDCK is a temporal semi-supervised method (see details in Section 2.9.1). Compared to other methods available in the literature, such as TCK-means [58], Constrained k-means, and Temporal-Driven k-means, TDCK shows the best results, as attested to by [59]. This thesis uses a synthetic dataset to compare the methods and confirm the results of [59].

The other goal is to analyze the behaviour of the proposed method in different situations, which includes varying the number of constraints and clusters. In addition, this chapter also seeks to provide an understanding of the effect of the exhaustive propagation on the accuracy of the algorithms, and the effect of the initial segmentation.

To achieve these goals a synthetic dataset was created. A synthetic dataset is flexible enough to provide a better test bed for analyzing different setup variations. The experiments on the synthetic dataset are presented and discussed in Section 4.2.

4.1 Setup of the Experiments

The methods being compared were tested on different variations of the synthetic datasets. All the datasets had their own ground truth, making it possible to calculate the accuracy

of the result segmentations by comparing them to a "gold standard". The methodology for calculating the accuracy and other evaluation measures are described in Section 4.1.1. All the accuracy results and other qualitative measures were the average over multiple runs, where in each run, random initializations and random pairs of constraints were selected. This randomization process diminished the bias caused by specific sets of constraints and initializations.

Must-link constraints were subjected to augmentation by applying transitive closure in all the experiments performed using semi-supervised methods (SSACA, SSACA+EP, and SSTSC+EP). Pairwise constraints were used to seed the initial segmentation of the proposed methods.

As discussed in Chapter 3, main parameters needed to be selected for the compared methods. The first one was the n_{max} parameter, which controls the granularity of the temporal term. This term was chosen empirically, according to the dataset. The second parameter is related to the frame kernel matrix. The kernel used on the experiments was the Gaussian kernel, $k_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{y}_j\|^2}{2\sigma^2}\right)$, and the parameters σ was set to 1. The third parameter is the propagation rate α, which applies only to the methods with exhaustive propagation (i.e., SSACA+EP and SSTSC+EP). The value of the propagation rate was selected by grid-search in a portion of the data. For this experiment, the propagation rate was $\alpha = 0.40$. The last set of parameters is exclusively for the TDCK algorithm, i.e., γ_t γ_d β δ. These four parameters were also chosen by grid search. In this experiment, the dimensional weight was set to $\gamma_d = 1$, and the temporal weight was set to $\gamma_t = 0.25$. The scaling factor was set to $\beta = 0.3$, and the width of the function was set to $\delta = 3$. Details about TDCK parameters are included in Section 2.9.1.

TDCK can be applied to multiple entities; however, because the proposed experiments consider a single entity at a time, all temporal data are treated as a single entity.

4.1.1 Evaluation Measures

Three major evaluation measures were considered to analyze the behaviour of the compared methods. The first measure was the clustering accuracy, which represents a quantitative value of the performance of the algorithms. The second measure was the mean cluster variance, which represented a qualitative measure of the "tightness" of the clusters. The third measure was a modified version of Shannon entropy, which was reserved for evaluating the variability of the datasets. Besides these three measures measures, the objective function values were also used to evaluate the approaches.

Clustering Accuracy

Since the datasets have ground truth, a natural way to evaluate the quality of results from the different algorithms is to compare the produced clustered segments to the ground truth. The result of this comparison is referred as "clustering accuracy". To compute the clustering accuracy, the metric from [25] was used. A confusion matrix was computed between the returned segmentation results of the different algorithms (G_{alg}, H_{alg}) and the ground truth (G_{tru}, H_{tru}), i.e.,

$$C = G_{alg} H_{alg} H_{tru}^T G_{tru}^T \in \mathbb{R}^{k \times k}, \tag{4.1}$$

where G_{alg} was the segment cluster and H_{alg} was the sample-segment indicator. Each entry c_{c_1,c_2} in the confusion matrix corresponded with the total number of frames of cluster segment c_1 that were shared by cluster segment c_2 in the ground truth. Then, the calculated confusion matrix was subjected to the Hungarian algorithm to find the optimal cluster correspondence. The accuracy was calculated by the following equation:

$$\text{Accuracy} = \max_{\text{P}} \frac{\text{Tr}(C\text{P})}{\text{Tr}(C 1_{k \times k})}, \tag{4.2}$$

where P was a permutation matrix constrained to $P \in \{0,1\}^{k \times k}$.

Mean Cluster Variance

Mean cluster variance is used mainly to quantify the dispersion in clusters. This measure can be used to evaluate the influence of extra knowledge added to the clustering process (semi-supervised clustering). The mean cluster variance was calculated as follows:

$$\text{MCVar} = \frac{1}{m} \times \sum_{c=1}^{k} \sum_{\mathbf{x}_i \in g_c} ||\mathbf{Y}_i - \mathbf{z}_c||^2, \tag{4.3}$$

where m was the number of segments, and $||\mathbf{Y}_i - \mathbf{z}_c||^2$ was the squared distance between the segment $\mathbf{Y}_i$ and the centre of the cluster c.

Cluster Alternation Rate

This measure was not used to evaluate the quality of the clusters, but rather to measure the variability of the datasets in terms of the number of alternations between clusters. This rate was calculated based on a modified version of Shannon entropy, adapted from [55], and was calculated as follows:

$$H = -\sum_{c=1}^{k} P(g_c) \log_2 P(g_c) \left(1 + \frac{n_{ch} - n_{min}}{m - 1}\right), \tag{4.4}$$

where $P(g_c)$ represented the probability of a cluster c in the observed series, n_{ch} represented the number of cluster changes, n_{min} was the minimal required number of changes, and m was the total number of segments. The last term in brackets constituted a penalty factor that increased the measurement value as the number of changes grew.

4.2 Synthetic Dataset

One of the datasets used for the analysis was a randomly-generated synthetic dataset that created time series by sampling 2D Gaussian distributions, according to the setup used in [25]. Figure 4.1 shows a sample of the dataset.

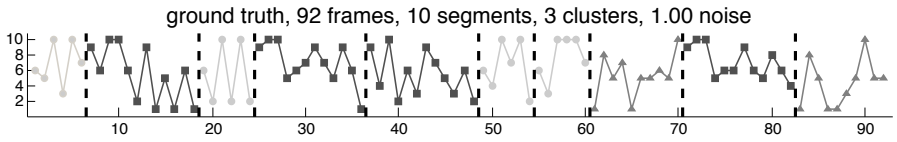


Figure 4.1: A sample time series from the synthetic dataset.

4.2.1 Baseline Algorithm

Comparative studies in the literature [55] [59] show that the semi-supervised method, TDCK, outperforms other similar methods. However, in order to confirm its superiority, a similar comparison was performed on the synthetic data. Compared to simple k-means, temporal-driven k-means, constrained k-means, and temporal Constrained k-means, TDCK had the best results, as Table 4.1 attests.

Table 4.1: Average accuracy results of some temporal-based k-means methods.

Average Accuracy				
TDCK-means [55]	TCK-means	Constrained k-means	Temp-driven K-means	Simple K-means
0.38 ± 0.05	0.36 ± 0.05	0.34 ± 0.04	0.30 ± 0.04	0.28 ± 0.02

The results of this comparison in Table 4.1 confirmed the superiority of TDCK for the synthetic dataset. Therefore, in the rest of the experiments, TDCK was used as the semi-supervised baseline method.

4.2.2 Analysis of the Number of Constraints

The goal of this experiment was to analyze how the addition of extra knowledge in the form of pairwise constraints affected the performance of semi-supervised clusters. This experiment used a randomly-generated synthetic time series composed of 276 frames and 30 segments ($m = 30$). The number of constraints was varied progressively from zero to 20. For each random pair of constraints added to the experiment, it was necessary to ensure that the pair of segments designated by that pair of constraints complied with the labels. For example, in a simple case where there were two pairs of constraints (i.e., segment 1 must link with segment 2, and segment 2 must link with segment 4), by transitiveness, it can be inferred that segment 4 should have the same label as segment 1 and 2. Consequently, in the experiment setup, it was necessary to create an initial segmentation that complied with the inferred labels. The single process of keeping these compatibilities correct can get very expensive as the number of constraints increases.

In a sample of 30 segments ($m = 30$), the total number of non-repeatable possible pairs of constraints was $m * (m - 1)/2$. In this particular experiment, it equated to 435 pairs. For every pair of constraints that were allowed in the experiment, the correct segmentation for two segments was provided. The minimum number of constraints required in order to cover the correct segmentation of a whole sequence would correspond with the set of subsequent pairs of constraints with no intersection (overlapping) between them. Figure 4.2 illustrates an example where one pair of *must-link* constraint accounts for the correct segmentation of the whole time series.

On the same 30-segment sample ($m = 30$), it was possible to predict the correct segmentation with as little as 8 pairs of constraints in the best case scenario; however, this

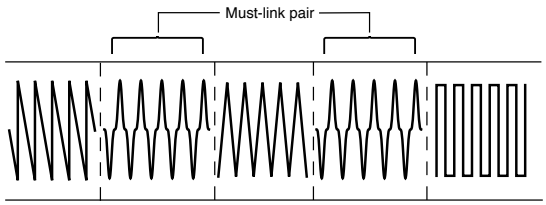


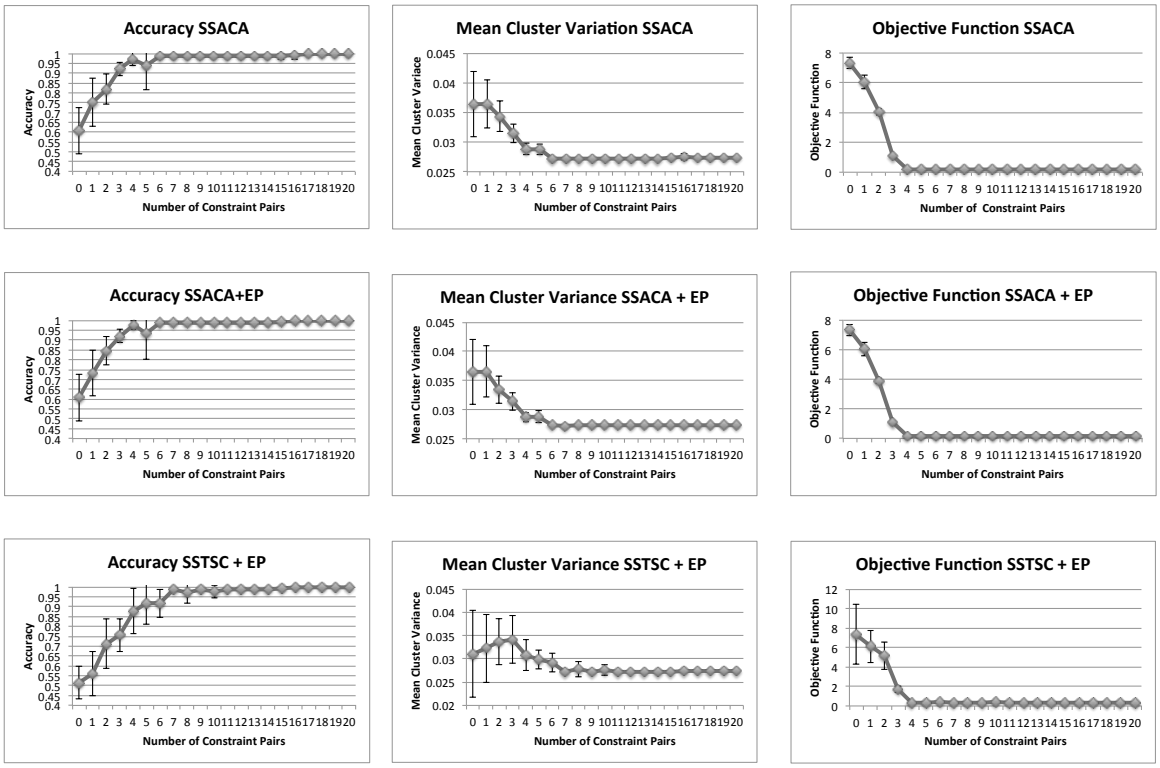
Figure 4.2: A sample of a time series with one pair of constraint.

specific arrangement is unrealistic in real life. In this experiment, the pairs of constraints were generated randomly; therefore, to guarantee the full coverage of all the segments, it would be necessary to have 379 pairs out of the 435 possible, which represent the worst case scenario. The value 379 represents the number of pairs of constraints that completely covers 28 segments ($28 * 27/2 = 378$) plus one extra pair to cover the last 2 segments.

Segmentation was only one of the two problems being solved. The second one was the clustering itself. Even with a perfect segmentation, it was still necessary to organize these segments into coherent groups, where similar segments were in the same group and dissimilar segments were in different groups.

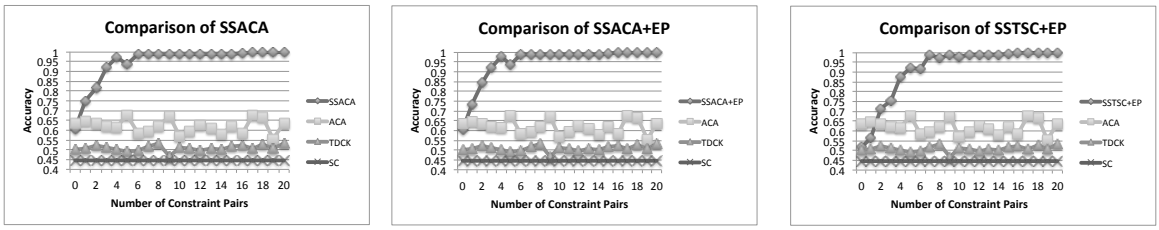
The accuracy, mean cluster variation, and the value of the objective function of the methods were evaluated as the dataset was grouped into three clusters ($k = 3$). Column 4.3a of Figure 4.3 depicts the average accuracy results of ten runs of each of the proposed methods: SSACA, SSACA+EP, and SSTSC+EP. Column 4.3b shows the mean cluster variance, and Column 4.3c shows the objective function values. In terms of accuracy, the results showed that, in all three methods, the performance rose quickly as the constraints were progressively added. In this particular example, as constraints reached as low as 2 pairs, the accuracy had already reached above 85% for SSTSC+EP, and above 90% for SSACA and SSACA+EP. High variability was noticeable in the results, as can be observed by the standard deviation displayed in the figures. This behaviour was particularly normal for k-means-based algorithms; such algorithms rely on random initializations, which may lead to different results for different runs.

Conversely, the mean cluster variance decreased as the number of constraints increased. This behaviour shows that the pairwise constraints were "helping" the cluster method create tighter clusters. Similarly, the values of the objective functions decreased as the number of constraints increased, because of the adjustments made to the similarities of the segments by the constraints. The *must-link* constraints made distances between two segments in the same cluster smaller, and *cannot-link* constraints made distances between two segments larger.



(a) Accuracy (b) Mean Cluster Variance (c) Objective Function

Figure 4.3: Constraint analysis of SSACA, SSACA+EP, SSTSC+EP. (a) Accuracy average, (b) Mean cluster variance, (c) Objective function values.



(a) SSACA (b) SSACA+EP (c) SSTSC+EP

Figure 4.4: Comparison of SSACA, SSACA+EP, SSTSC+EP with ACA, SC, and TDCK.

Figure 4.4 shows the average accuracy of the proposed methods compared to two unsupervised methods (ACA and SC) and one temporal semi-supervised method (TDCK). The average was the result of 10 runs applied to 276 synthetically-generated frames, organized into 30 segments (m = 30) and three clusters (k = 3). Figure 4.4a shows the average accuracy of SSACA, Figure 4.4b shows the average accuracy of SSACA+EP, and Figure 4.4c shows the average results of SSTSC+EP. Note that the overall accuracy of the proposed methods was boosted by the addition of very few constraints, surpassing the accuracy of the other methods with only a small amount of side information in the form of pairwise constraints.

4.2.3 Analysis of the Number of Clusters

The goal of this experiment was to analyze how the number of clusters affected the results of the proposed semi-supervised methods. For this experiment, 30 randomly-generated synthetic time series samples were used, ranging from 180 to 350 frames and made up of 30 segments ($m = 30$) and 6 pairs of constraints ($n_{const} = 6$). Then, the number of clusters was varied from 3 to 6 ($k = [3 \ldots 6]$) in order to observe the behaviour of the proposed methods. For this analysis, the accuracy (Acc.), the mean cluster variance (MCVar), the value of the objective function (Obj.), and the cluster alternation rate (H) of the samples were evaluated. The results shown in Table 4.2 are the average of all the evaluation measures over the 30 runs. As the number of clusters increased, the average cluster alternation rate of the analyzed samples also increased – meaning that the variability of the samples increased, or, in other words, that the number of alternations of clusters rose. In addition, the number of elements for each cluster decreased as the number of clusters increased. With fewer elements per cluster, the pool of elements used to calculate the centroids was smaller, increasing the chances of finding an incorrect cluster. As a result, the overall performance of all methods dropped consistently across the different numbers of clusters.

Still, the relative increase in accuracy and quality of clusters that semi-supervised methods caused was not compromised. Figure 4.5 shows the relative difference among the methods.

Table 4.2: Analysis of number of clusters

	ACA	SSACA	SC	SSACA +EP	SSTSC +EP	TDCK
k	3	3	3	3	3	3
n_{const}	6	6	6	6	6	6
H	2.4222					
Acc.	0.5880	0.9214	0.4928	0.9294	0.8344	0.4600
MCVar	0.0389	0.0300	0.0417	0.0294	0.0290	0.0673
Obj.	8.0691	0.5437	n/a	0.5283	0.6852	75.8242
k	4	4	4	4	4	4
H	3.2157					
Acc.	0.4839	0.8248	0.4129	0.8446	0.7724	0.3815
MCVar	0.0501	0.0398	0.0529	0.0386	0.0376	0.0476
Obj.	8.3097	0.6442	n/a	0.5948	0.6993	116.6765
k	5	5	5	5	5	5
H	3.7623					
Acc.	0.4616	0.7247	0.4011	0.7259	0.7013	0.3479
MCVar	0.0621	0.0494	0.0746	0.0491	0.0475	0.0566
Obj.	7.5798	0.5075	n/a	0.4857	0.5308	115.0031
k	6	6	6	6	6	6
H	4.1516					
Acc.	0.4347	0.7081	0.3774	0.7177	0.7045	0.3244
MCVar.	0.0711	0.0604	0.0814	0.0590	0.0541	0.0464
Obj.	7.5697	0.5237	n/a	0.5756	0.6136	145.5870

4.2.4 Analysis of the Influence of Exhaustive Propagation

The goal of this experiment was to analyze the effect of exhaustive propagation of constraints on the accuracy of a semi-supervised temporal clustering algorithm, compared to regular instance-based constraints. This experiment analyzed the averaged results of 100 runs of a randomly-generated time series, containing 78 segments ($m = 78$) and four clusters ($k = 4$), while varying the number of constraints from 0 to 60.

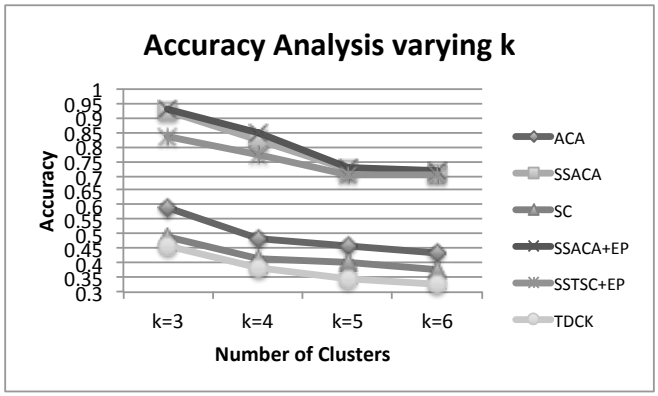


Figure 4.5: Analysis of variation of the number of clusters applied to a synthetic dataset.

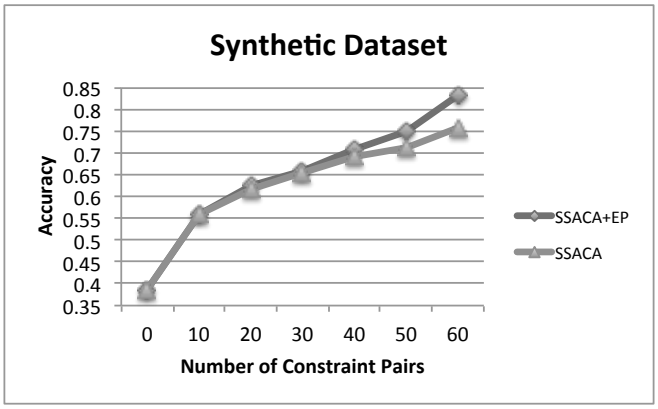


Figure 4.6: Analysis of the effect of exhaustive propagation.

Figure 4.6 shows that, after 30 pairs of constraints, the effect of the exhaustive propagation became noticeable. The higher the number of constraints, the more they reassured the neighbourhood similarities affected by the propagation of the constraints. Therefore, greater improvements were only noticeable when the number of constraints was higher.

4.2.5 Analysis of the Influence of Constrained Initial Segmentation

The use of constraints in a clustering problem can provide several benefits, not only by helping to "guide" the clustering algorithm, but also by helping with initial segmentation and with seeding the initial clusters. In the case of a temporal segmentation problem, the pairwise constraints are provided in the form of segments. These constrained segments can be used to help with initial segmentation. The logical consequence of adding more constraints is the improvement of initial segmentation, which is a contribution of the approach proposed in this thesis. The question that remains is whether the overall gain, as the constraints increase, comes only from supporting the initial segmentation, or if there is improvement that comes from the direct influence of the similarity manipulation of the constraints in the clustering process. The goal of this experiment was to answer this question.

A randomly-generated synthetic time series composed of 20 frames, 7 segments ($m = 7$), and three clusters ($k = 3$) was used for this experiment. The number of constraints was varied progressively from zero to 15. For each random pair of constraints added to the experiment, the accuracy of the temporal segmentation was calculated based on the initially-constrained segmentation only, and on the complete semi-supervised method. The former case is referred in the figures as "(Avg. Acc. Constrained Initial Segmentation)" and the latter as "Avg. Acc. (method name)". Figure 4.7 shows the comparison for SSACA, Figure 4.8 shows the comparison for SSACA+EP, and Figure 4.9 shows the comparison for SSTSC.

In Figure 4.7, the average gains in accuracy of SSACA are greater than the gains provided by the simple, constrained initial segmentation. In this particular example, there was an increase of 8 points of extra improvement in the total accuracy when providing 1 or 2 constraints. Similar results can be observed in Figure 4.8, which depicts the accuracy of SSACA+EP compared to its initial segmentation. Figure 4.9, which depicts the results of the comparison of SSTSC and its initial constrained segmentation, shows even better results, with improvements of 11 points in the overall performance, when given 1 pair of constraint, even after the initial segmentation improvements. All these results show that

the improvements are not limited to the gains of the initial segmentation. In other words, there is actual "learning" in the clustering process provided by the pairwise constraints.

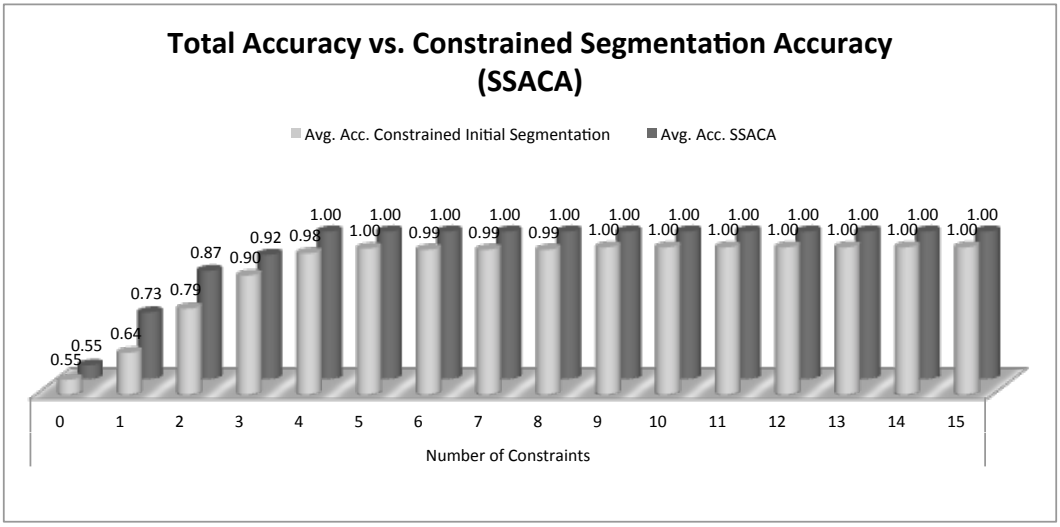


Figure 4.7: Comparison between the accuracy of only using the initial constrained segmentation and the accuracy of SSACA plus the initial constrained segmentation.

The same experiment was repeated in a larger dataset, so that a larger number of constraints could be observed without reaching total coverage of the initial segmentation. In this second experiment, the same randomly-generated synthetic time series was used, but this time, it was composed of 4,474 frames, 78 segments ($m = 78$), and four clusters ($k = 4$). The results showed that, with a larger number of constraints, the gains in performance discounting the influence of the constrained initial segmentation were even higher. In Figure 4.10, which depicts a comparison of the results of SSACA and its initial constrained segmentation, there was an increase of up to 12 points of extra improvement in the total accuracy. The improvement of SSACA+EP, compared to its initial constrained segmentation was an increase of 12 points using 10 constraints (0.33% of constraints), as shown in Figure 4.11. The improvements of SSTSC+EP reached 6 points using 1% of the constraints (see Figure 4.12). Although the absolute number of constraints was higher compared to the experiment with the smaller dataset, relatively speaking, it represented a smaller percentage of the total number of possible constraints.

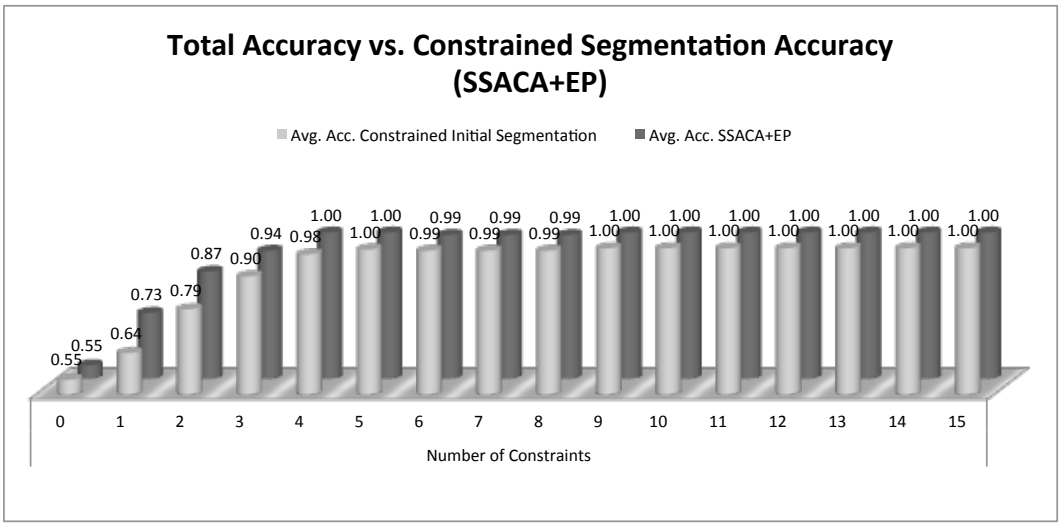


Figure 4.8: Comparison between the accuracy of only using the initial constrained segmentation and the accuracy of SSACA+EP plus the initial constrained segmentation.

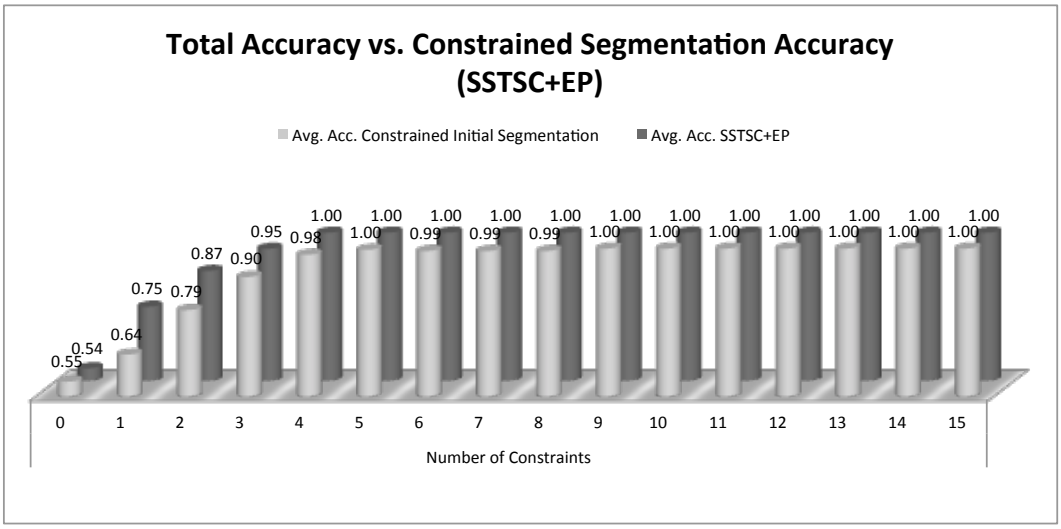


Figure 4.9: Comparison between the accuracy of only using the initial constrained segmentation and the accuracy of SSTSC+EP plus the initial constrained segmentation.

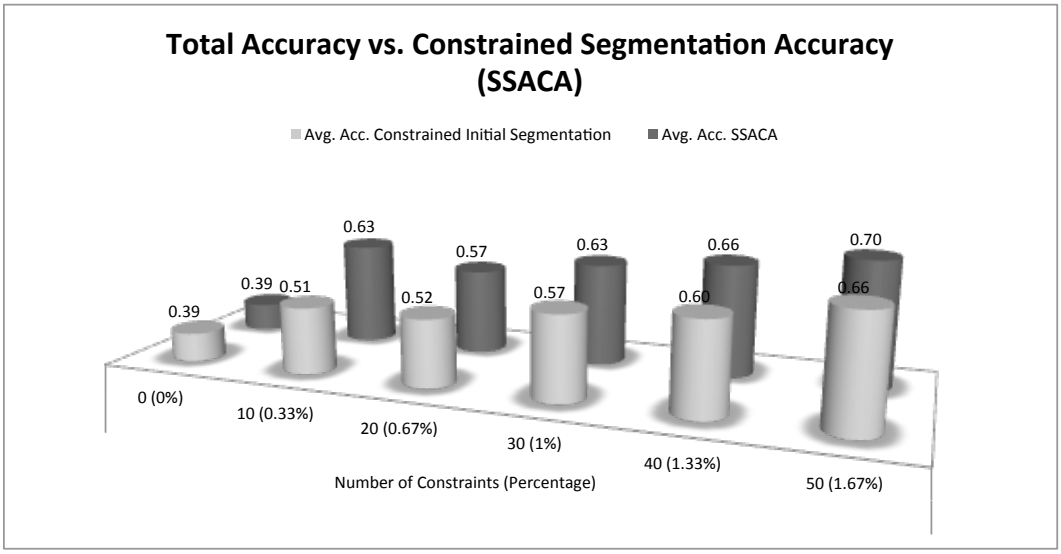


Figure 4.10: Comparison between the accuracy of only using the initial constrained segmentation and the accuracy of SSACA (initial constrained segmentation + similarity manipulations).

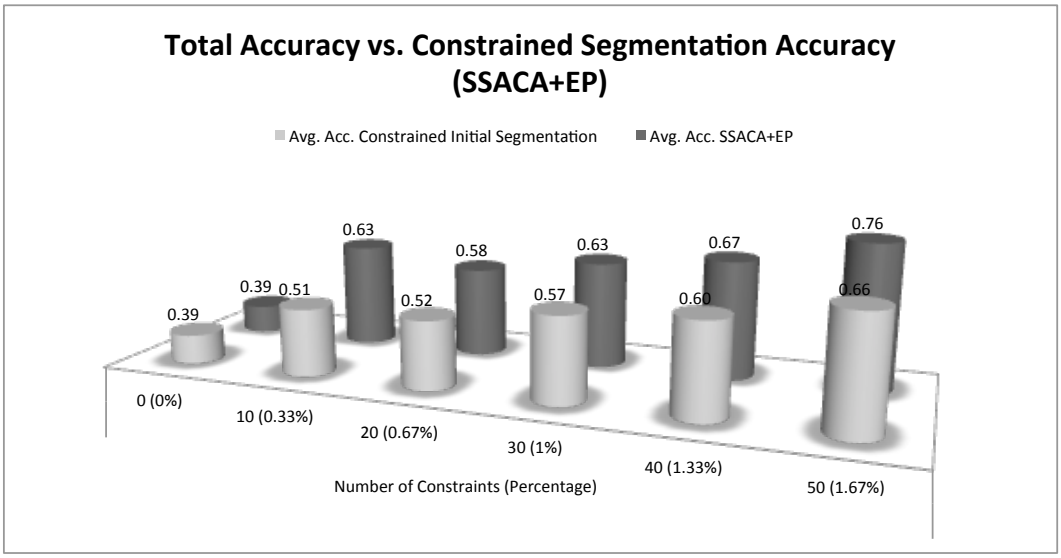


Figure 4.11: Comparison between the accuracy of only using the initial constrained segmentation and the accuracy of SSACA+EP (initial constrained segmentation + similarity manipulations).

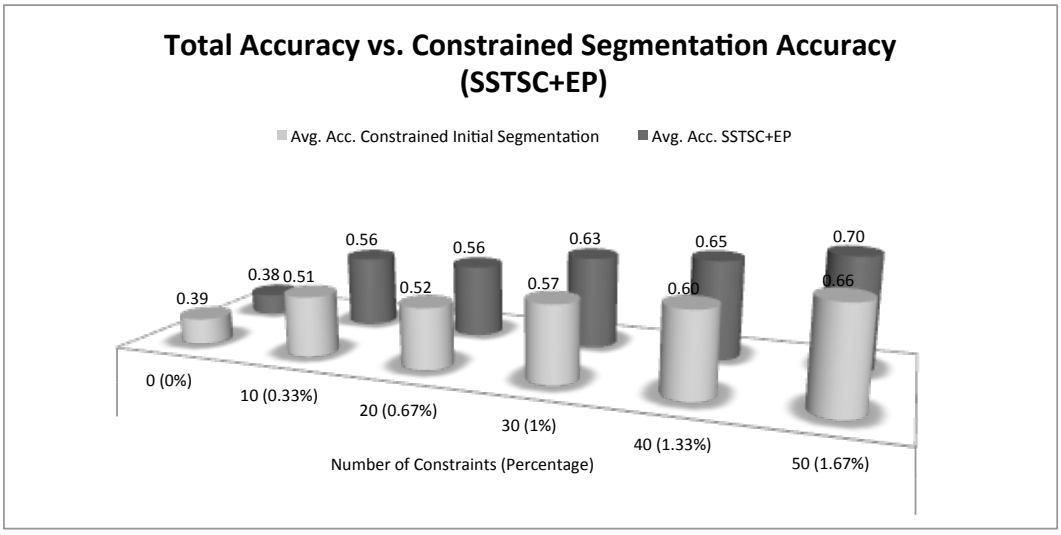


Figure 4.12: Comparison between the accuracy of only using the initial constrained segmentation and the accuracy of SSTSC+EP (initial constrained segmentation + similarity manipulations).

Chapter 5

Experiments on Emotion Analysis and Human Motion Segmentation

This chapter provides some experiments on real-world applications. The first application is on emotion segmentation. Two naturally-occurring human emotion datasets are analyzed, which include the VAM corpus and the AVEC dataset. The second application is on human motion segmentation, and the dataset used is the Carnegie Mellon University Motion Capture.

An overview of emotion analysis and human motion segmentation is provided, before each experiment is presented and discussed. Section 5.1 gives an introduction to emotion analysis, and reviews some of the work in the field of facial expression analysis, more specifically, the methods based on clustering approaches. It also explains the problem of facial expression analysis, and the fundamental steps involved in the process, which includes face acquisition, facial data extraction and representation, and facial expression interpretation. Finally, the experiments are presented and analyzed by the end of the section. Section 5.2 explains the problem of human segmentation, discusses some related work, and analyzes the experiments performed in a human motion dataset.

5.1 Emotion Analysis

Analysis of naturally-occurring human emotions is the main focus of recent research in the field of affective computing. Emotional analysis is a vital step in building efficient and more realistic intelligent human-computer interfaces. Facial expressions and speech are the two

modalities most commonly used to analyze emotions in human interaction [64] [65] [66]. While facial expression is considered to be the primary modality in human communication, according to [67], speech is the fastest and most natural method of communication between humans.

Facial expressions consist of movements of the muscles of the face. These movements represent a response to physical or emotional internal states of a person, and play a major role in interpersonal and non-verbal communication. Facial expression analysis has been studied for many years in the fields of psychology and behavioural science. More recently, a great deal of effort has been put into automatic recognition and analysis of these expressions in images and videos. The automation of facial expression analysis can provide many useful applications, from straightforward ones, such as automatically assessing the credibility of statements in interrogations, to those less evident, such as warning a drowsy driver that he or she is about to fall asleep, or judging how difficult a student finds a video lecture.

Focus is now directed towards recognition in terms of dimensional and continuous description, rather than a small number of discrete emotional categories. Numerical representation of emotions in a multi-dimensional space is considered to be a more appropriate representation of the gradated nature of emotions. Moreover, human natural affective behaviour is multimodal, subtle, and complex, making it challenging to map the affective human state into a single label or discrete number of classes [68].

The bulk of the approaches found in the literature use supervised learning, despite the fact that the labelling process is demanding. With the abundance of data available in this domain, and the burdensome nature of the labelling process, there should be a natural inclination for researchers to pursue unsupervised methods for these applications more intensively. However, purely unsupervised methods may not produce desirable results, due to the complexity of emotion analysis. A semi-supervised approach offers a balance between the demands of process and accuracy of results.

There are few attempts in the literature based on unsupervised methods. This observation is true for both of the two most-used modalities: facial expressions and speech. A possible reason for the shortage of unsupervised work is the lack of the temporal aspect of the traditional clustering algorithms, preventing the analysis of dimensional and continuous emotion.

Wollmer et al. [69] study the estimation of emotions from speech in the valence and activation dimensions using Long-Short-Term Recurrent Neural Networks. Nicolaou et al. [70] propose the use of Output-Associative Relevance Vector Machine (OA-RVM) for dimensional and continuous estimation of emotions from facial expressions. Grimm et al. [71] compares the performance of Support Vector Regression, Fuzzy k-Nearest Neighbour,

and Rule-based Fuzzy Logic classifiers as estimators of spontaneously expressed emotions in speech from three continuous-valued emotion primitives.

Hoey [72] presents an early study that uses unsupervised methods to categorize facial expressions. He proposes a hierarchical dynamic bayesian network for unsupervised classification of expression sequences from video input. He shows that learning the high-level temporal structure of the environment helps the unsupervised learning process of the facial expressions. His approach categorizes five basic emotions on a posed database.

[73] presents an approach using existing clustering techniques to obtain temporal clustering by grouping frames capturing consistent shapes. These temporal cuts detect non-rigid changes in the shape of objects. One example illustrating this approach uses sequences of facial expressions, grouping frames into smiling and serious clusters. The results show accuracy similar to supervised methods, and achieve moderate intersystem agreement with FACS. One of the challenges that this study presents is how to increase accuracy for subtle facial actions.

De la Torre et al. [74] propose a temporal segmentation method of facial gestures in order to cluster similar facial actions. They apply their approach to spontaneous facial expressions, using two steps. First, they cluster the facial gestures by shape in a way that is invariant to specific geometric transformations, and second they group clusters effectively into temporally coherent chunks. They use this approach in two applications: both to detect unusual or rare facial expressions and actions and to preprocess videos for manual FACS annotation. For clustering, they use a spectral graph clustering algorithm.

Another work that focuses on unsupervised approach for recognition is that of Zhou et al. [75]. The goal of their work is to discover facial events directly from naturally occurring videos, using unsupervised temporal clustering instead of FACS or other label schemes. The first algorithm they use for this task is ACA, and the second is a multi-subject correspondence algorithm for matching expressions. ACA is an extension of kernel k-means to cluster time series, combined with Dynamic Time Alignment Kernel (DTAK).

The only research that applies semi-supervised clustering to the problem of emotion segmentation are studies co-authored by the author of this thesis [76] [77] [78]. These researches are part of the experiments presented in Chapter 5.

5.1.1 Facial Expression Analysis

According to Tian et al. [79], facial expression analysis refers to the process by which computer systems attempt to automatically analyze and recognize facial motions and facial feature changes from visual information.

Facial motions not only represent internal emotional states, but also represent physical states, such as intentions, and cognitive processes. Computer analysis systems should be able to analyze facial movements regardless of the context involved. However, the interpretation of facial movements depends heavily on context. For example, a facial expression recognition system running in a car safety environment will be programmed to identify facial movements that indicate signs of fatigue on the driver's face, so that the system can warn the driver to stop and rest.

The ability to automatically recognize facial motions broadens the horizon for many new real-world applications in fields such as medicine, education, and human-computer interaction. The applications developed so far, many of which are mentioned in previous sections, are part of a nascent movement, with many more sure to follow.

Fundamental Steps of an Automatic Facial Expression Recognition System

Automatic facial expression recognition systems share many of their fundamental steps with general computer vision systems. In fact, the three fundamental steps in both processes are the same. The three steps are: (1) face acquisition, (2) facial data extraction and representation, and (3) facial expression recognition.

Face acquisition is the process of capturing images of faces through the use of a sensor device. Acquisition can be as simple as gathering a sequence of digitized images that has already been digitized. Two main approaches are used for this step. The first approach finds the face or faces in the image, and then keeps track of them in the remainder of the video sequence. The second appoach detects the faces at each frame.

The next step is facial data extraction and representation. Data extraction and representation is the process of deriving a set of features which represents the original digital image in a different format. Usually, features are a more compact way of representing digital images. Two main types of data extraction and representation are found in the literature: geometric and appearance-based approaches. Geometric-based approaches define features as shapes or locations of facial feature points (such as the corners of the mouth or eyes) [65]. Appearance-based approaches define features based on texture changes (wrinkles, furrows, lines, etc.) [80].

Each method has its own advantages and disadvantages. Appearance-based approaches cope with variations of skin patterns or markings well, but are susceptible to illumination changes. Geometric-based approaches demand less computation, but they only consider some specific fiducial points and, as a result, may miss some important information.

The last step is facial expression recognition. Facial expression recognition is the process of assigning a label to a pattern based on its features. Most of the work in this area uses supervised learning methods to classify the patterns, either by individual movements of the face, also known as action units (AU), or by prototypic emotional categories. Another way of recognizing the emotions is to continuously predict the intensity of dimensional models by performing regression. Dimensional models are discussed in Section 5.1.1.

Face Acquisition

Face acquisition is the first phase of a facial expression analysis system, and can be classified into one of two types of imaging: static images and video sequences. From the data modality point-of-view, face acquisition can be either 2D or 3D. There is a current upward trend in the number of publications that uses 3D data [81, 82] due to its robustness to changes in lighting conditions and even pose variations. However, there are some limitations of 3D imaging, including the cost of the computational time, the cost of memory allocation, and the cost of the acquisition itself (although the price of 3D scanners is constantly becoming cheaper). Despite the increase in 3D acquisition, the majority of the databases still use 2D data [83, 84], which requires less computational effort and offers greater ease of acquisition.

There are some databases available in the literature that can be used for facial expression analysis. Table A.11 lists some of the publicly-available databases and their characteristics.

Head Orientation

Head orientation represents the position of the face in relation to the camera. The frontal view is the most natural position, and the one that provides more visible facial features. However, in real-world applications, a frontal view is not always possible. Some databases, such as Multi-PIE and Face Database MPI, even provide multiple views in order to make systems more robust for such view variations. In contrast, some studies claim that non-frontal-view facial expression recognition outperforms frontal-view facial expression recognition. The experiments of Hu et al. [85] suggest that, although a frontal-view face has the most visible facial features, the displays of facial features are always symmetrical at the frontal view, causing a certain degree of redundancy of information.

Moore and Bowden [86] raise the question of whether Hu et al.'s conclusion is related to the geometric features they used and consequently proposes an appearance-based approach to answer the same question. According to their results, the frontal pose is optimal for

facial expression recognition, but its efficacy is dependent on feature selection. They state that weaker features perform better with non-frontal pose, and that some expressions may perform better with non-frontal views.

Pantic and Patras [87] go even further in their approach to the problem of non-frontal view facial expressions by exploring temporal segments of the video (i.e., onset, apex, and offset) of action units. However, their method operates under two assumptions: the input video sequence is a non-occluded, near-profile view of the face, with possible in-image-plane head rotations and an initial frame showing a neutral expression. They do not address the problem initial location of the facial points, and their method cannot deal with spontaneously-occurring facial behaviour.

In a more recent paper, Rudovic et al. [88] propose a regression-based scheme for multi-view facial expression recognition for 2D geometric features. They point out that 2D-based methods need to train view-specific classifiers, and as a result, the number of classes increases proportionally with the number of different views and facial expressions. Consequently, the demand for training data for specific views increases. A similar amount of training data is needed in order to avoid bias, but there is an imbalance between the data available from frontal-view and multi-view, making it possible for a good frontal-view classifier that generalizes poorly for non-frontal views. Rudovic et al. address this problem by mapping facial points from non-frontal to frontal views, using regression models that perform facial expression recognition with state-of-the-art methods. Although some experiments add some noise to the testing data, the process is not analogous to a real-world application, which has no controlled environment.

Building a facial expression system that is robust to non-frontal-view face is a necessary task in order to achieve a truly automated facial expression recognition system – one that can operate in a less constrained environment and is able to deal with spontaneous facial behaviour.

Posed vs Spontaneous Expressions

Spontaneous and posed facial differ substantially in terms of which muscle is moving, and consequently affecting the overall dynamics of facial movements. As a result, fine-motor control of deliberate facial actions are often less symmetrical than spontaneous. Tian et al. [79] give an example of one specific facial movement that can be different in spontaneous and deliberate expression. Many people can raise their outer brows spontaneously while leaving their inner brows at rest, but performing this movement voluntarily is uncommon. [89] offers more details on the differences.

Some effort has been put into the creation of spontaneous expression datasets. Some of the public databases are RU-FACS (interview-based), Prkachin-Solomon (pain induction), MMI, Cohn-Kanade+ (only part of the dataset), and Belfast.

Facial Feature Extraction and Representation

Optimal features should be able to maximize variations of intra-class patterns and minimize variation of inter-class patterns. Inappropriate features will make a good classifier work poorly.

The existing approaches in the literature can be placed into one of the two main categories, denoted by facial features. The first one of these categories is geometric-based approaches, in which features are defined as shapes, specific points, or fiducial points [90, 88, 65, 91, 92]. The second category is appearance-based approaches, which treat the image as a whole. Some examples of this approach include: texture [83], local binary pattern (LBP) [80], gabor wavelet [93] and moments [84, 94]. Some works have mixed both approaches [93, 95].

A new and emerging method is the dynamic-texture-based approach, which can be seen as a generalization of appearance-based approaches. Two papers have used this method: [96] and [83]. The latter uses a temporal model that allows recognition of sequences of temporal segments, such as neutral, onset, apex, and offset. However, one of the limitations of these approaches is their need for near-frontal-view face videos.

Facial Expression Categorization

The majority of studies use supervised methods for the process of categorization of facial expression. Most systems rely on a famous code system called FACS (Facial Action Coding System) [97], published by Ekman and Friesen in 1978 and later revised in 2002 [98]. This system maps all possible muscle movements of the face into independent facial actions, which are called Action Units (AU).

Regardless of debate amongst on the psychology community about the uniqueness and universality of this system, these AUs are extensively documented, and have been used in various scientific papers [83, 84, 81], either alone or in combination, to predict or categorize different emotions.

Ekman and Friesen's work [97] defines six prototypical emotions: anger, disgust, fear, happiness, sadness and surprise. The majority of theorists accept these categories as

universal emotions. So far, generalization of expression recognition to new expression classes remains unsolved, according to [99]. However, some unsupervised approaches have been created in attempt to avoid FACS taxonomy [75].

Categorical vs Dimensional Affect

For many years, the focus of affect recognition has been on the categorization of emotions into a discrete number of prototypical emotions, as discussed in Section 5.1.1. This approach makes it difficult to map the affective human state into a single label. Recently, the focus has shifted to dimensional emotion models. These models make it possible to analyze real-life emotions, which are usually subtle, mixed, and complex, in less constrained ways.

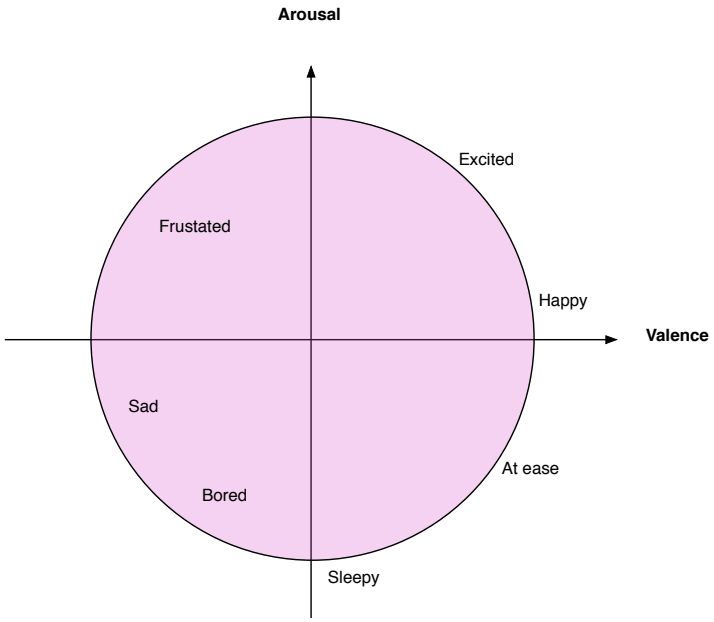


Figure 5.1: The arousal-valance (A-V) space proposed by Russell with some plotted affect words.

Affective states are characterized in terms of a small number of latent dimensions. The dimensional representation of emotions is described using the emotion space concept [100]. One popular dimensional emotion model uses three dimension primitives: valence (positive to negative), activation (calm to excited), and dominance (weak to strong) dimensions [68]. The valence dimension refers to how positive or negative the emotion is, and ranges from

very unpleasant emotion to happiness. Arousal refers to the level of excitement, and ranges from boredom to frantic excitement. The dominance dimension refers to the sense of control over the emotion. Figure 5.1 shows the Circumflex of Affect proposed by Russell [101].

5.1.2 VAM Corpus

The VAM corpus [102] is an authentic, spontaneous audio-visual dataset of real-life conversations. This dataset was recorded from the German TV talk show "Vera am Mittag", in which guests talk about their personal issues in a spontaneous, affective, and unscripted manner. Three emotion primitives describe the actual emotions of the participants: activation or arousal (calm – excited), dominance (weak – strong), and valence (positive – negative). Figure 5.2 shows sample images taken from the dataset.

Figure 5.2: Sample images from the VAM-Corpus dataset.

Each emotion primitive ranges from [-1,+1], where 0 signifies a lack of the evaluated primitive, and -1 and +1 signify the very low and very high values, respectively. For example, in the analysis of activation, -1 means that the person is very calm, 0 means that the person shows neutral emotion, and +1 shows that the person is very excited. In this experiment, the spectrum was divided into four categories: very low [-1, -0.5], low [-0.5, 0], high [0, +0.5], and very high [+0.5, 1]. For some subjects, the particular primitive being considered may not produce labels for the four categories. For instance, some speakers show small variation in emotions, which generates only two categories (low and high). Figure 5.3 shows a sample of the discretization process.

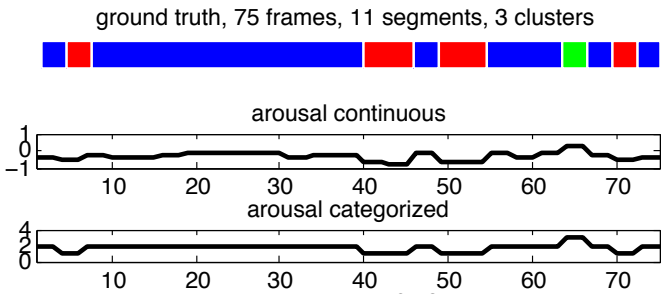


Figure 5.3: Discretization of the arousal dimensional emotion in subject 17 of VAM-Corpus dataset.

5.1.3 Features

For visual feature extraction, Local Binary Patterns (LBP) [103] were used in this work based on the approach described in [104]. A total of 1,486 facial images, which were annotated by 17 human evaluators using *Self Assessment Manikins* [105] were used. The number of frames per individual ranged from 32 to 113. The face region in each frame was first detected using the real-time face detection approach of Viola and Jones [106].The resulting face regions were then normalized. The normalization process began with identification of the coordinates of the two eyes, based on the OpenCV implementation of a Haar-cascade object detector, trained for either a left or a right eye. Then the size normalization was completed by resizing the image to create a distance between the eyes of 55 pixels. Afterwards, the whole face image was cropped to the size of 150 x 110, relative to the position of the eyes. The resulting face images were then divided into 7 x 6 sub-regions. Finally, the LBP descriptors were extracted for each region, and the histograms were mapped into uniform patterns in an (8, 2) neighbourhood. The size of the final feature vector was 2,478 (7 x 6 x 59) for each face.

5.1.4 Experimental Results and Analysis

The performance of six different algorithms for the three emotion primitives was evaluated based on the general setup described in 4.1. The whole dataset had a total of 20 subjects. However, some subjects were removed due to a lack of emotional variation. Subjects 3, 9, and 19 were removed from the valence analysis, subjects 9 and 10 from dominance, and subjects 9 and 12 from activation.

Table 5.1: Average accuracy results of VAM corpus dataset.

	Average Accuracy VAM		
	Activation	Valence	Dominance
SSACA+EP	0.63 ± 0.05	0.65 ± 0.05	0.66 ± 0.06
SSTSC+EP	0.60 ± 0.05	0.63 ± 0.04	0.64 ± 0.05
SSACA	0.63 ± 0.05	0.65 ± 0.05	0.66 ± 0.06
TDCK [55]	0.54 ± 0.06	0.55 ± 0.06	0.58 ± 0.07
ACA [7]	0.53 ± 0.09	0.55 ± 0.09	0.58 ± 0.09
SC [25]	0.55 ± 0.10	0.56 ± 0.11	0.58 ± 0.11

The frame kernel used was computed by $k_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{y}_j\|^2}{2\sigma^2}\right)$, where σ was set to be the average distance from the 5% closest neighbours, and $n_{max} = 3$. For the two methods that used constraint propagation (i.e., SSACA+EP and SSTSC+EP), the propagation rate was $\alpha = 0.40$. For the TDCK method, the dimensional weight was set to $\gamma_d = 1$, and the temporal weight was set to $\gamma_t = 0.25$. The scaling factor was set to $\beta = 0.3$, and the width of the function was set to $\delta = 3$.

Figure 5.4 shows the segmentation and the respective accuracy values of all the analyzed methods of one initialization of subject 17 compared to the ground truth. Table 5.1 shows the average result of all the analyzed subjects over 30 different initializations and sets of constraints. Regardless of the dimensions (activation, valence, and dominance), all three proposed methods (SSACA+EP, SSTSC+EP, and SSACA) had much better results compared to the alternative semi-supervised method, TDCK. Compared to the two unsupervised methods (ACA and SC), a highly significant increase in accuracies occurred using only 7% of the possible constraints.

5.1.5 AVEC Dataset

Audio/Visual Emotion Challenge (AVEC) [107] is an audio-visual emotion recognition dataset created for the emotion recognition challenge (AVEC 2012). The dataset consists of conversations among several participants and four stereotyped characters. Each character has a specific emotion stereotype: sensible, happy, angry, and sad. The dataset is labeled for arousal, valence, power, and expectancy. Some sample images of the AVEC dataset can be seen in Figure 5.5. The train partition of the database contains 31 sections, wherein

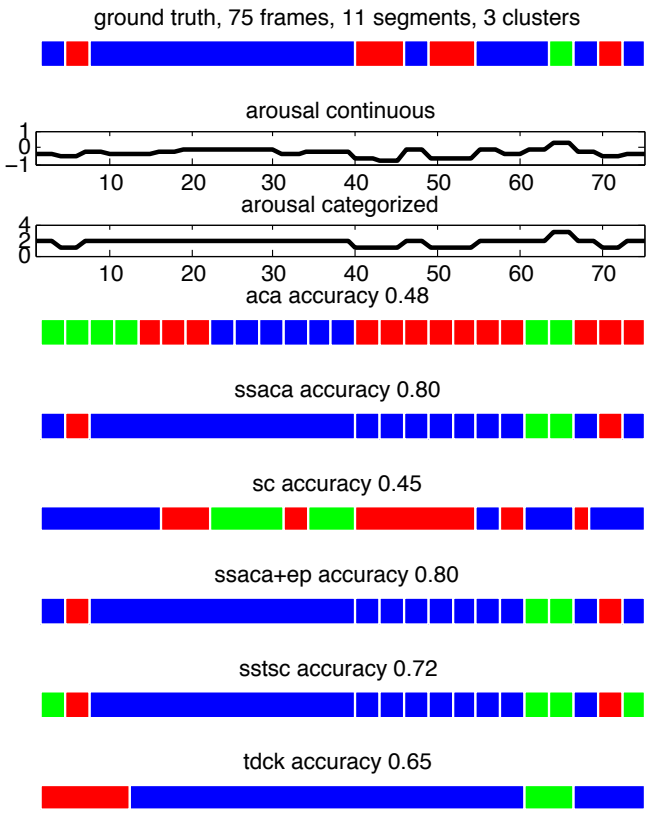


Figure 5.4: Accuracy of the analyzed methods compared to the ground truth of speaker 17 of VAM dataset.

each session contains one dialogue with a specific character. A total of seven different subjects (subject 1 was used to record eight sections) were used to record the 31 sections.

LBP were the features extracted as described in Section 5.1.3. Approximately 5% of the total number of possible constraints were applied.

5.1.6 Experimental Results and Analysis

The experiments were performed in the training portion of the AVEC dataset and were organized by subject, with each subject representing a dialog with four consecutive different characters. This setup allowed for longer conversation and higher variability. User 6 was removed from the experiments due to a high number of out-of-boundary frames during

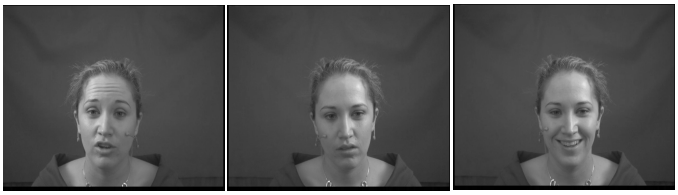


Figure 5.5: Sample images of AVEC dataset.

the face tracking process, caused by excessive head movement. Therefore, a total of six subjects were tested in this experiment. The continuous values of the affective dimensions, which ranged from [-1, +1], were discretized in 6 categories, similar to [76], and the results of the segmentation were observed for arousal, valence, power, and expectancy.

The frame kernel used was computed by $k_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{y}_j\|^2}{2\sigma^2}\right)$, where σ was set to be the average distance from the 5% closest neighbours, and $n_{max} = 43$. For the two methods that used constraint propagation (i.e., SSACA+EP and SSTSC+EP), the propagation rate was $\alpha = 0.01$. For the TDCK method, the dimensional weight was set to $\gamma_d = 1$, and the temporal weight was set to $\gamma_t = 0.25$. The scaling factor was set to $\beta = 0.3$, and the width of the function was set to $\delta = 3$.

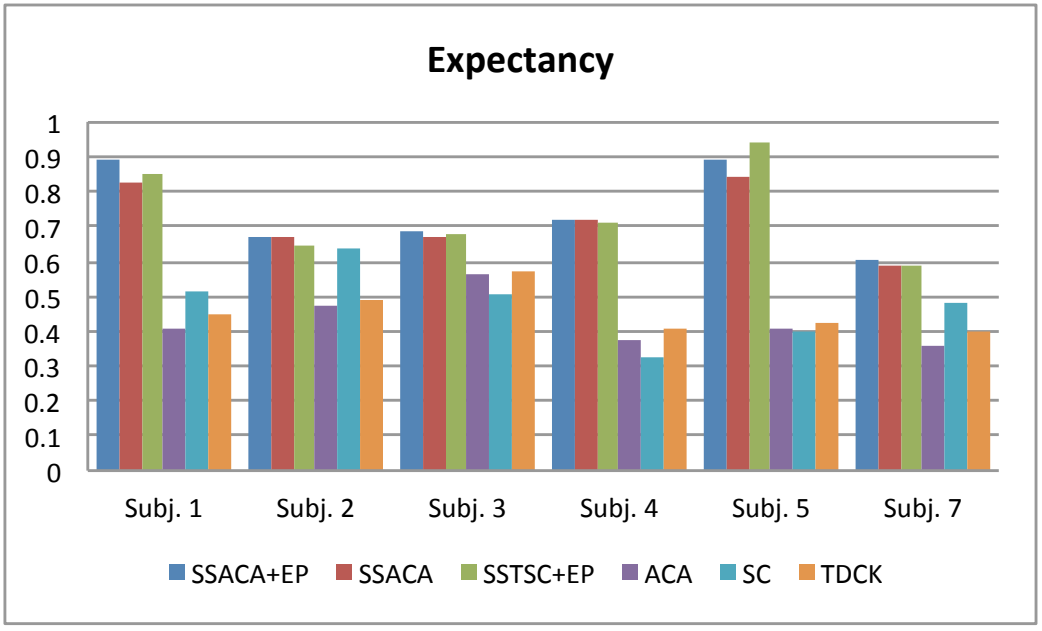


Figure 5.6: Average accuracy results per subject for expectancy.

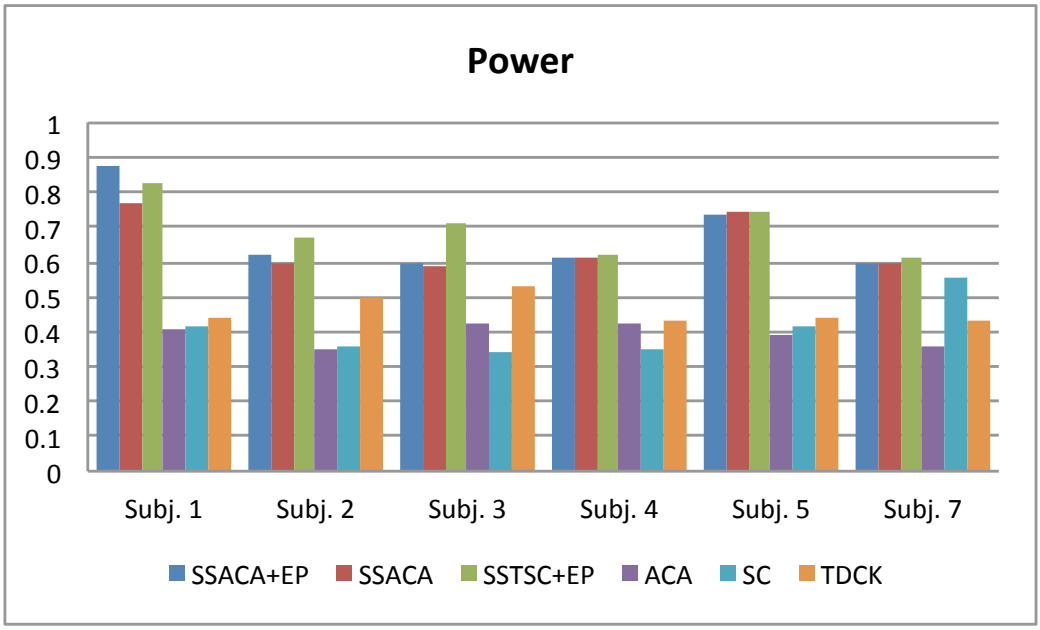


Figure 5.7: Average accuracy results per subject for power.

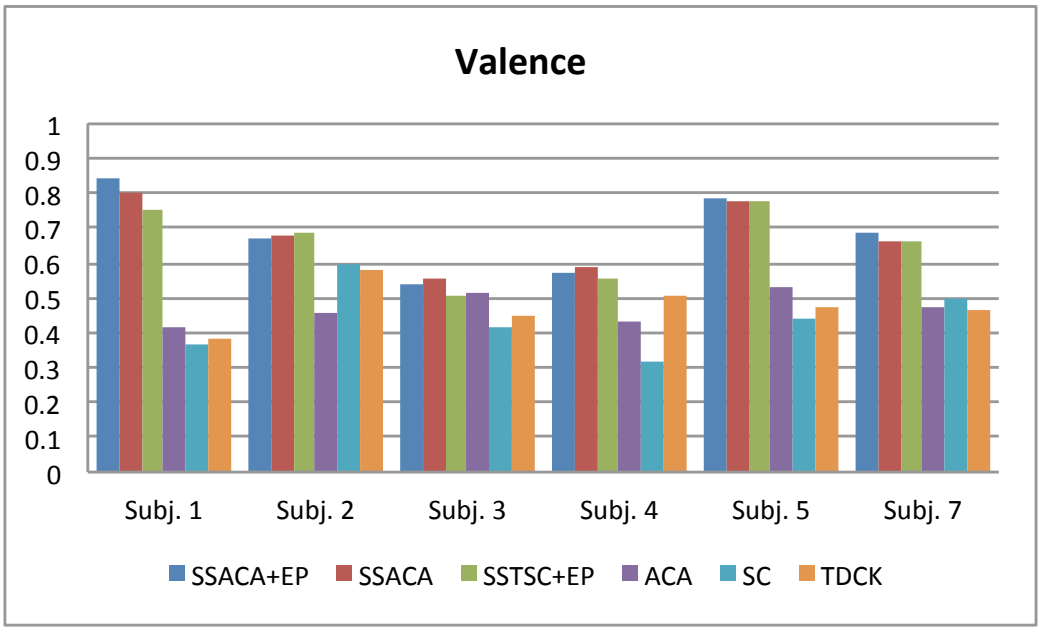


Figure 5.8: Average accuracy results per subject for valence.

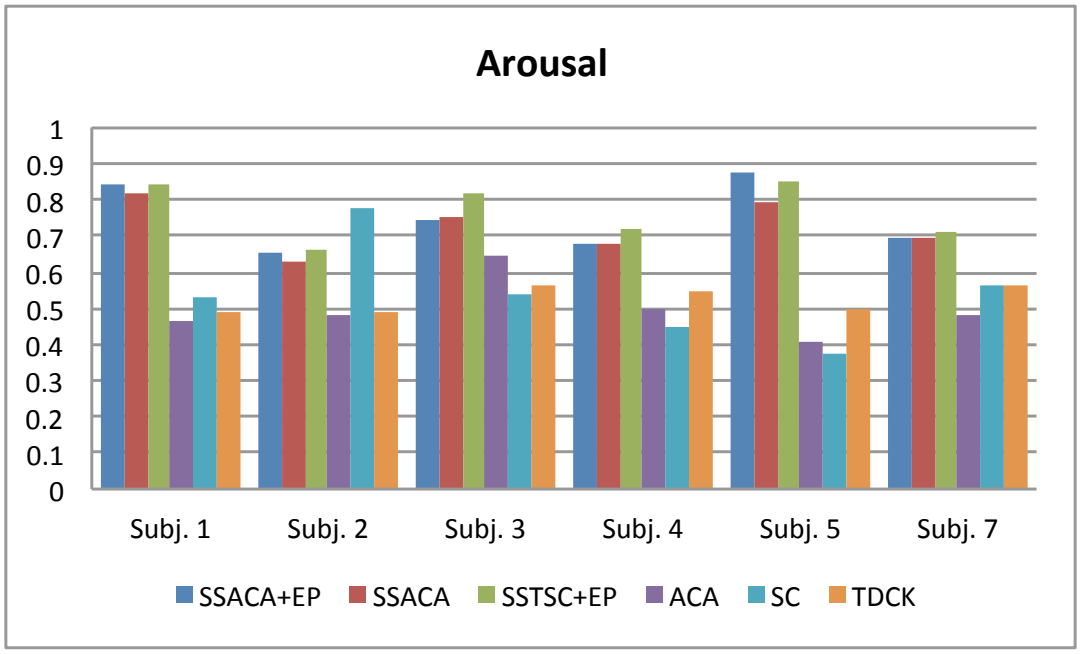


Figure 5.9: Average accuracy results per subject for arousal.

Table 5.2: Characteristics of each subject of the AVEC dataset for the Expectancy emotional dimension.

	Expectancy					
	Subj. 1	Subj. 2	Subj. 3	Subj. 4	Subj. 5	Subj. 7
# Seg. (m)	118	59	44	73	123	38
# Clus. (k)	5	3	3	4	4	5
Total # of Const.	6903	1711	946	2628	7503	703
# Known Const.	345	86	47	131	375	35
H	3.3702	2.5830	2.3829	2.6254	3.1479	3.6999

Table 5.3: Characteristics of each subject of the AVEC dataset for the Power emotional dimension.

	Power					
	Subj. 1	Subj. 2	Subj. 3	Subj. 4	Subj. 5	Subj. 7
# Seg. (m)	77	39	25	38	78	54
# Clus. (k)	5	4	5	4	4	4
Total # of Const.	2926	741	300	703	3003	1431
# Known Const.	146	37	15	35	150	72
H	4.0435	3.1098	3.3765	3.0021	3.1499	2.9016

Table 5.4: Characteristics of each subject of the AVEC dataset for the Valence emotional dimension.

	Valence					
	Subj. 1	Subj. 2	Subj. 3	Subj. 4	Subj. 5	Subj. 7
# Seg. (m)	81	41	23	32	83	48
# Clus. (k)	4	3	4	4	5	4
Total # of Const.	3240	820	253	496	3403	1128
# Known Const.	162	41	13	25	170	56
H	3.5589	2.7209	3.3637	3.1651	3.9309	3.6105

Table 5.5: Characteristics of each subject of the AVEC dataset for the Arousal emotional dimension.

	Arousal					
	Subj. 1	Subj. 2	Subj. 3	Subj. 4	Subj. 5	Subj. 7
# Seg. (m)	98	48	57	39	98	33
# Clus. (k)	4	3	3	3	5	3
Total # of Const.	4753	1128	1506	741	4753	528
# Known Const.	238	56	80	37	238	26
H	3.3379	2.7515	2.9555	2.8265	3.5142	2.8711

For this experiment, because of the higher number of constraints and segments, the seeding process that accounted for initializing the clusters in a consistent fashion became very cumbersome in order to be guaranteed. As a result, initial labels were not guaranteed to be consistent with the *must-link* and *cannot-link* constraints, which otherwise could theoretically have produced better results. Nevertheless, results showed great improvements in the proposed methods compared to the other methods.

Figure 5.6 shows the average accuracy results of 20 different initializations for all the compared methods. The results were grouped by subjects and derived by visual features for the expectancy dimension. Table 5.2 shows some characteristics generated by expectancy dimension in terms of number of segments m, number of clusters k, total number of possible constraints, total number of known constraints, and cluster alternation rate. Results show that all three proposed methods (SSACA, SSACA+EP, and SSTSC+ EP) performed much better than the methods to which they were compared (TDCK, ACA, and SC), particularly for subjects 1 and 5. These subjects were the ones with both a higher number of segments, 118, and 123, respectively, and high rate of cluster alternation. The higher the number of segments, the more constraints could be drawn from the samples, as a result, a higher absolute number of pairwise constraints could be derived. For example, in subject 1, 5% of all possible constraints accounted for 345 pairwise constraints, as opposed to 35 in subject 7. The higher number of constraints reassured the neighbourhood similarities, producing better results. This effect is even more noticeable in the methods that use the exhaustive propagation (SSACA+EP and SSTSC+EP). The numerical results with the respective standard deviations is shown in Table A.1.

Figure 5.7 shows the average accuracy results of 20 different initializations for the power dimension. The results were also grouped by subjects and derived from visual features. Table A.8 shows the characteristics generated by the power dimension. The results for this dimension were similar to the expectancy dimension in terms of the overall improvement of the proposed methods (SSACA, SSACA+EP, and SSTSC+ EP). In this dimension, subjects 1 and 5, also had the best results, consequence of the combination of high number of segments and high cluster alternation rate. This observation is on par with the results of Section 4.2.4.

Figure 5.8 shows the average accuracy results of 20 different initializations for the valence dimension. The results were also grouped by subjects and derived from visual features. Table 5.4 shows the characteristics generated by the valence dimension. The same trend showed in expectancy and valence was observed for the valence dimension, where the proposed methods produced great improvements with minimal supervision. One interesting aspect that can be pointed out, is the fact that small samples tend to produce even results across the different methods. For example, subject 3 produced only 23 segments, which

allowed for only 13 constraints. Consequently, the difference of the results across the methods were not very significant, ranging from 0.41 in the worse case and 0.54 in the best case. This detailed numerical results can be seen on Table A.3.

Finally, Figure 5.9 shows the average accuracy results of 20 different initializations for the arousal dimension. The results were also grouped by subjects and derived from visual features. Table A.10 shows the characteristics generated by the arousal. Results for this dimension showed similar trends produced by the previous dimensions. Subjects 1 and 5 produced the best results, as consequence of the higher number of segments and constraints discussed previously. The exception, however, was subject 2. In this particular case, a combination of factors contributed to lower accuracy of the semi-supervised methods and higher results for SC. The main characteristic that benefits SC as a method is having frames from the same cluster that are contiguous. This characteristic is defined by low cluster alternation rate. In contrast, having a high number of segments combined with high cluster alternation rate benefits the semi-supervised methods; however, subject 2 showed very low cluster alternation rate combined with an unbalanced distribution of segments per cluster, with 87.5% of the segments divided between two clusters, which can be seen in Table A.10.

TDCK in most of the cases performed better than the unsupervised methods, ACA and SC. However, in this type of application, emotions categories are recurrent, which means, for example, that segments with high arousal, may occur several times during a conversation, and are likely to be further apart in time. This behaviour causes TDCK to not perform as good as the proposed method, due to its assumption that benefits the creation of clusters that are closer in time. In contrast, the proposed methods do not make general assumptions based on time similarity, instead it use the constraints in the instance-level as provided.

Overall, the results show that all three proposed methods (SSACA, SSACA+EP, and SSTSC+ EP) performed much better than the methods to which they were compared (TDCK, ACA, and SC) with the addition of only 5% of the possible constraints across all emotion dimensions. SSACA+EP, which has exhaustive propagation, performed better or at least very similar to its counter-part without exhaustive propagation (SSACA). Greater differences in accuracy were more noticeable in sets with a higher number of segments, which enables higher absolute numbers of constraints, as discussed in Section 4.2.4. This effect was more visible in the results of subjects 1 and 5 in all the dimensions.

All numerical detailed results showed in Figures 5.6, 5.7, 5.8, and 5.9 can be seen in Tables A.1, A.2, A.3, A.4, A.5, and A.6 organized by users. Detailed tables with information about the number of segments and distribution of frames per cluster are included in Section A.2.

5.2 Human Motion Segmentation

The development of devices that capture motion data has increased vastly in recent years. As a result, research on motion analysis, recognition, and synthesis has grown in importance. Motion segmentation is one of the most relevant techniques in the context of these new devices, as discussed in [108]. Motion segmentation splits motion capture data into continuous segments. In a human context, the process consists of breaking motion into actions. Applications that can benefit from motion segmentation include those in the field of entertainment, computer animation, healthcare, and consumer electronics [109].

According to [8], some challenges of this task include the high level of variability in the temporal scale and periodicity of human actions, as well as the exponential nature of all possible movement combinations. Kulic and Nakamura [110] classify the segmentation algorithms according to whether they require previous knowledge of the motion primitives or not. The category of algorithms that does not need prior information about the motion primitives is called unsupervised segmentation. Some examples of unsupervised methods are velocity-based approaches. Fod et al. [111] use a Zero Velocity Crossing (ZVC) approach, in which the segmentation point is given when zero-velocity crossing is detected in a sufficient number of dimensions in the joint angle data. The first stage of the method proposed by [112] uses a similar approach. In Pomplum and Matarić [113], a new segment is defined every time the root square value of the joint velocity falls below a threshold. The drawback of these methods is that the algorithm becomes more difficult to tune as the number of joints increases.

Another family of unsupervised methods are based on variances of the feature data. In [114], for example, the algorithm searches for a set of segment points that minimize a cost function of the data variance. In [25] and [8], the segmentation is formulated as a temporal clustering problem. Their approach extends the standard kernel k-means by changing the similarity measure to a kernel distance to achieve temporal invariance. The segmentation boundaries are found by minimizing a clustering objective function across several segments.

The second category of segmentation algorithms, which require prior knowledge of the motion primitives, is known as supervised segmentation. This category of segmentation compares known motions with incoming data. [115] searches for the best match between an observed sequence and a known prototypical motion primitive. Other methods encode the known primitives in short HMMs, and later, the segmentation points are decided based on the error of the predicted motion data returned by the HMM and the actual observed motion data. If the error increases over a threshold, it triggers a segment [116].

As discussed in [109], the great diversity of applications and the considerable amount of recorded data makes it especially important to boost unsupervised motion analysis methods, while also making them more efficient. These observations are in line with the goal of semi-supervised methods, and provide even greater motivation to use the proposed methods.

Motion segmentation is usually divided into two levels. The higher level includes the detection of distinct activities, where the result segments are defined in the level of behaviour or activity (e.g., walk, run, and kick). The lower level includes the detection of motion primitives, where the result segments are defined as reoccurring units or cycles (e.g., steps). The focus of the experiments bellow is on the higher level of distinct activities.

5.2.1 CMU Motion Capture Dataset (MOCAP)

Carnegie Mellon University Motion Capture Dataset (MOCAP) [117] is a human motion dataset. The dataset is the result of twelve Vicon infrared MX-40 cameras being placed in the centre of a room, around a rectangular area of approximately 3m x 8m. Humans wearing a black jumpsuit with forty-one markers taped onto it performed several actions while the markers were captured by the cameras using infrared. The images picked up by the cameras were triangulated to produce 3D data. Only the fourteen most informative joints were considered.

Figure 5.10: Sample images of CMU MOCAP dataset.

5.2.2 Experimental Results and Analysis

The experiments were performed on fourteen sequences of subject 86, containing roughly ten actions, such as walking, punching, and kicking. The length of the sequences was reduced by a factor of five to improve scalability, and frames were grouped into twenty clusters, as described in [25]. The range of frames for each activity in the experiment was between 160 and 300, the n_{max} was set to 60, and the number of constraints were based on 20% of all possible constraints.

Table 5.6: Accuracy of 14 sequences of subject 86 from the MOCAP dataset.

	Average over 14 sequences
	Accuracy
SSACA+EP	0.91 ± 0.08
SSTSC+EP	0.90 ± 0.08
SSACA	0.91 ± 0.08
TDCK [55]	0.51 ± 0.06
ACA [7]	0.87 ± 0.09
SC [25]	0.74 ± 0.14

The frame kernel used was computed by $k_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{y}_j\|^2}{2\sigma^2}\right)$, where σ was set to be the average distance from the 3% closest neighbours, and $n_{max} = 60$. The propagation rate for EP was set to $\alpha = 0.40$. The TDCK dimensional weight was set to $\gamma_d = 1$ and the temporal weight was set to $\gamma_t = 0.25$, while the scaling factor was set to $\beta = 0.3$ and the width of the function was set to $\delta = 3$.

Some improvement in accuracy of the proposed methods can be observed in Table 5.6, although these improvements were not as significant as the ones produced in other applications. One of the reasons for these smaller improvements was that the baseline accuracy was already very high (e.g., ACA had 87% accuracy on average). The differences between methods with EP and without EP were not as significant in this experiment, due to the low number of segments. The average number of segments for this dataset was only 8.7. Despite the low increases, the proposed method performed better than the other semi-supervised method (TDCK). The better performance of the proposed methods can be attributed to the use of kernel, which maps points to a higher dimension where they can be clustered more easily.

Chapter 6

Conclusions and Future Work

This chapter summarizes and concludes this thesis, while also providing some discussion on the directions of future work and a list of publications that resulted from this research.

6.1 Conclusions

This thesis proposed an approach to creating kernel-based semi-supervised temporal clustering algorithms. This approach used instance-level constraints, in the form of *must-link* and *cannot-link*, to add supervision to the clustering process, and a dynamic-programming method to search for the optimal temporal clusters. This approach was applied to two algorithms, a kernel-based temporal clustering (ACA), and a graph-based clustering (spectral clustering), transforming both algorithms into semi-supervised temporal clustering methods.

The use of pairwise constraints at the instance level was demonstrated to be better-suited for the real-world applications presented, when compared to TDCK [59], the available alternative approach. Another benefit of pairwise constraints at the instance level is their specificity, which only considers constraints between two points, instead of making general assumptions. The proposed methods showed substantial improvements in accuracy compared to the completely unsupervised methods, with the addition of minimal supervision. Furthermore, the introduction of supervision did not affect the complexity of the core algorithm, and in fact, lowered the search complexity as the number of constraints increased.

Many applications which require temporal segmentation may benefit from this approach, particularly when extra knowledge can be provided. The trade-off between the gains in accuracy and the amount of side information provided was demonstrated to be advantageous. Good results were also shown for real-world applications, such as emotion analysis and human motion segmentation; however, applications such as activity categorization, animal behaviour analysis, speaker diarization, and neuron activity modelling could also potentially benefit from this approach.

6.2 Future Work

Some aspects of this approach which are open to improvement include the time and space complexity of the algorithms. Although the dynamic programming implementation of the objective function makes the time complexity polynomial, for very large datasets, some scalability problems may occur. The same can be said for the space complexity, which is the memory required to store data for the algorithm to run. All kernel-based approaches need to store a pairwise kernel matrix of the data instances used in the clustering process (e.g., the kernel matrix of a dataset of size n needs to be stored in $O(n^2)$ of memory). One alternative to mitigate this problem is the use of distributed algorithms for scaling the proposed methods on a MapReduce framework. Elgohary [118] proposed a way to scale kernel k-means algorithms by first creating low-dimensional embeddings based on Nström approximation and p-stable distributions, and then applying a parallelization strategy using MapReduce.

One of the limitations of the proposed approaches is the necessity of having the complete data in order to do the clustering. Therefore, the problem of clustering streaming data was not considered in this thesis. However, for future work, the proposed methods can be adapted to handle online data. Another extension includes the expansion of this approach to other kernel-based methods (e.g., kernel-mean shift [4]). A detailed analysis of the trade-off between accuracy and the complexity added in order to guarantee the consistency of the seeding process is another possibility potential direction for future research.

One area that has gained a great deal of attention in the last few years, due to the explosion of wearable technologies, is accelerometer-based human activity recognition (AHAR). This area of research has potential applications for health monitoring, context-awareness, and new forms of human-computer interaction. AHAR, which naturalistic 3D acceleration-based activity datasets are available, such as SCUT-NAA dataset [119], has the same temporal characteristics of emotion analysis and human-motion segmentation. Therefore,

AHAR is an interesting candidate for the application of semi-supervised temporal clustering.

6.3 List of Publications

Journal Paper

- R. Araujo and M. S. Kamel. A Constraint-based Approach for Temporal Clustering. *Pattern Recognition*, submitted.

Refereed Conference

- R. Araujo and M. S. Kamel. Audio-visual emotion analysis using semi- supervised temporal clustering with constraint propagation. In Aurélio Campilho and Mohamed Kamel, editors, Image Analysis and Recognition, Lecture Notes in Computer Science, pages 3–11. Springer International Publishing, 2014.
- R. Araujo and M. S. Kamel. Semi-supervised kernel-based temporal clustering. In 13th International Conference on Machine Learning and Applications, ICMLA 2014, Detroit, MI, USA, December 3-5, 2014., pages 123–128, 2014.
- R. Araujo and M. S. Kamel. A semi-supervised temporal clustering method for facial emotion analysis. In Multimedia and Expo Workshops (ICMEW), 2014 IEEE International Conference on, pages 1–6, July 2014.
- R. Araujo, Y-Q. Miao, Mohamed S. Kamel, and Mohamed Cheriet. A fast and robust feature set for cross individual facial expression recognition. In Leonard Bolc, Ryszard Tadeusiewicz, LeszekJ. Chmielewski, and Konrad Wojciechowski, ed- itors, Computer Vision and Graphics, volume 7594 of Lecture Notes in Computer Science, pages 272–279. Springer Berlin Heidelberg, 2012.
- Y-Q. Miao, R. Araujo, and M.S. Kamel. Cross-domain facial expression recognition using supervised kernel mean matching. In Machine Learning and Applications (ICMLA), 2012 11th International Conference on, volume 2, pages 326–332, Dec 2012.
- A. Sayedelahl, R. Araujo, and M.S. Kamel. Audio-visual feature-decision level fusion for spontaneous emotion estimation in speech conversations. In Multimedia and Expo Workshops (ICMEW), 2013 IEEE International Conference on, pages 1–6, July 2013.

APPENDICES

Appendix A

Detailed Results

A.1 Detailed Accuracy Results of AVEC Dataset

This section details the results per subject of the AVEC experiment described in Section 5.1.5.

A.2 Detailed Description of AVEC Dataset

This section gives additional details about the characteristics of the AVEC dataset previously discussed in Section 5.1.6.

A.3 List of Public Available Facial Expression Datasets

Table A.1: Subject 1 average accuracy results.

	AVEC Subject 1 - Average Accuracy			
	Arousal	Valence	Power	Expectancy
SSACA+EP	0.84 ± 0.07	0.84 ± 0.07	0.87 ± 0.07	0.89 ± 0.09
SSTSC+EP	0.84 ± 0.09	0.75 ± 0.11	0.83 ± 0.11	0.85 ± 0.12
SSACA	0.82 ± 0.06	0.80 ± 0.10	0.77 ± 0.13	0.83 ± 0.13
TDCK [55]	0.38 ± 0.03	0.49 ± 0.05	0.44 ± 0.05	0.45 ± 0.05
ACA [7]	0.47 ± 0.03	0.42 ± 0.03	0.41 ± 0.05	0.40 ± 0.05
SC [25]	0.53 ± 0.00	0.37 ± 0.00	0.41 ± 0.00	0.51 ± 0.00

Table A.2: Subject 2 average accuracy results.

	AVEC Subject 2 - Average Accuracy			
	Arousal	Valence	Power	Expectancy
SSACA+EP	0.66 ± 0.12	0.67 ± 0.10	0.62 ± 0.15	0.67 ± 0.10
SSTSC+EP	0.66 ± 0.13	0.69 ± 0.11	0.67 ± 0.17	0.64 ± 0.11
SSACA	0.63 ± 0.12	0.68 ± 0.09	0.60 ± 0.14	0.67 ± 0.10
TDCK [55]	0.49 ± 0.09	0.58 ± 0.07	0.49 ± 0.10	0.49 ± 0.06
ACA [7]	0.48 ± 0.04	0.46 ± 0.03	0.35 ± 0.06	0.47 ± 0.06
SC [25]	0.78 ± 0.00	0.60 ± 0.00	0.36 ± 0.00	0.64 ± 0.00

Table A.3: Subject 3 average accuracy results.

	AVEC Subject 3 - Average Accuracy			
	Arousal	Valence	Power	Expectancy
SSACA+EP	0.74 ± 0.13	0.54 ± 0.09	0.59 ± 0.13	0.68 ± 0.16
SSTSC+EP	0.82 ± 0.13	0.51 ± 0.07	0.71 ± 0.15	0.68 ± 0.12
SSACA	0.75 ± 0.14	0.55 ± 0.08	0.59 ± 0.13	0.67 ± 0.15
TDCK [55]	0.57 ± 0.07	0.45 ± 0.04	0.53 ± 0.10	0.57 ± 0.09
ACA [7]	0.65 ± 0.02	0.51 ± 0.03	0.43 ± 0.05	0.56 ± 0.02
SC [25]	0.54 ± 0.00	0.41 ± 0.00	0.34 ± 0.00	0.51 ± 0.00

Table A.4: Subject 4 average accuracy results.

	AVEC Subject 4 - Average Accuracy			
	Arousal	Valence	Power	Expectancy
SSACA+EP	0.68 ± 0.14	0.57 ± 0.10	0.62 ± 0.13	0.72 ± 0.09
SSTSC+EP	0.72 ± 0.14	0.56 ± 0.11	0.62 ± 0.12	0.71 ± 0.10
SSACA	0.68 ± 0.13	0.59 ± 0.10	0.61 ± 0.12	0.72 ± 0.09
TDCK [55]	0.54 ± 0.09	0.51 ± 0.06	0.43 ± 0.08	0.41 ± 0.06
ACA [7]	0.50 ± 0.05	0.43 ± 0.05	0.42 ± 0.06	0.38 ± 0.05
SC [25]	0.45 ± 0.00	0.32 ± 0.00	0.35 ± 0.00	0.33 ± 0.00

Table A.5: Subject 5 average accuracy results.

	AVEC Subject 5 - Average Accuracy			
	Arousal	Valence	Power	Expectancy
SSACA+EP	0.87 ± 0.07	0.79 ± 0.07	0.74 ± 0.10	0.89 ± 0.06
SSTSC+EP	0.85 ± 0.09	0.78 ± 0.08	0.75 ± 0.13	0.94 ± 0.04
SSACA	0.79 ± 0.09	0.78 ± 0.07	0.75 ± 0.12	0.85 ± 0.09
TDCK [55]	0.50 ± 0.06	0.48 ± 0.06	0.44 ± 0.08	0.42 ± 0.04
ACA [7]	0.41 ± 0.03	0.53 ± 0.05	0.39 ± 0.04	0.41 ± 0.01
SC [25]	0.37 ± 0.00	0.44 ± 0.00	0.41 ± 0.00	0.40 ± 0.00

Table A.6: Subject 7 average accuracy results.

	AVEC Subject 7 - Average Accuracy			
	Arousal	Valence	Power	Expectancy
SSACA+EP	0.69 ± 0.14	0.69 ± 0.13	0.60 ± 0.07	0.61 ± 0.13
SSTSC+EP	0.71 ± 0.11	0.66 ± 0.11	0.61 ± 0.11	0.59 ± 0.10
SSACA	0.69 ± 0.13	0.66 ± 0.13	0.60 ± 0.08	0.59 ± 0.12
TDCK [55]	0.56 ± 0.06	0.46 ± 0.05	0.43 ± 0.05	0.40 ± 0.05
ACA [7]	0.48 ± 0.07	0.47 ± 0.04	0.36 ± 0.03	0.36 ± 0.03
SC [25]	0.56 ± 0.00	0.49 ± 0.00	0.56 ± 0.00	0.48 ± 0.00

Table A.7: Cluster distribution of each subject of the AVEC dataset for the Expectancy emotional dimension.

	Expectancy					
	Subj. 1	Subj. 2	Subj. 3	Subj. 4	Subj. 5	Subj. 7
	# Segs.	# Segs.	# Segs.	# Segs.	# Segs.	# Segs.
C 1	4 (3%)	24 (41%)	20 (45%)	33 (45%)	37 (30%)	9 (24%)
C 2	44 (37%)	30 (51%)	22 (50%)	35 (48%)	59 (48%)	15 (39%)
C 3	53 (45%)	5 (8%)	2 (5%)	4 (5%)	25 (20%)	10 (26%)
C 4	14 (12%)	-	-	1 (1%)	2 (2%)	3 (8%)
C 5	3 (3%)	-	-	-	-	1 (3%)

Table A.8: Cluster distribution of each subject of the AVEC dataset for the Power emotional dimension.

	Power					
	Subj. 1	Subj. 2	Subj. 3	Subj. 4	Subj. 5	Subj. 7
	# Segs.	# Segs.	# Segs.	# Segs.	# Segs.	# Segs.
C 1	4 (5%)	3 (8%)	1 (4%)	1 (3%)	8 (10%)	2 (4%)
C 2	12 (16%)	3 (8%)	8 (32%)	5 (13%)	34 (44%)	4 (7%)
C 3	24 (31%)	16 (41%)	11 (44%)	18 (47%)	32 (41%)	25 (46%)
C 4	27 (35%)	17 (44%)	4 (16%)	14 (37%)	4 (5%)	23 (43%)
C 5	10 (13%)	-	1 (4%)	-	-	-

Table A.9: Cluster distribution of each subject of the AVEC dataset for the Valence emotional dimension.

	Valence					
	Subj. 1	Subj. 2	Subj. 3	Subj. 4	Subj. 5	Subj. 7
	# Segs.	# Segs.	# Segs.	# Segs.	# Segs.	# Segs.
C 1	7 (9%)	15 (37%)	4 (17%)	1 (3%)	4 (5%)	8 (17%)
C 2	27 (33%)	21 (51%)	9 (39%)	7 (22%)	13 (16%)	18 (38%)
C 3	32 (40%)	5 (12%)	8 (35%)	15 (47%)	30 (36%)	16 (33%)
C 4	15 (18%)	-	2 (9%)	9 (28%)	28 (34%)	6 (13%)
C 5	-	-	-	-	8 (10%)	-

Table A.10: Cluster distribution of each subject of the AVEC dataset for the Arousal emotional dimension.

	Arousal					
	Subj. 1	Subj. 2	Subj. 3	Subj. 4	Subj. 5	Subj. 7
	# Segs.	# Segs.	# Segs.	# Segs.	# Segs.	# Segs.
C 1	7 (7%)	6 (13%)	16 (28%)	6 (15%)	1 (1%)	6 (18%)
C 2	36 (37%)	24 (50%)	28 (49%)	19 (49%)	6 (6%)	16 (48%)
C 3	43 (44%)	18 (38%)	13 (23%)	14 (36%)	30 (31%)	11 (33%)
C 4	12 (12%)	-	-	-	44 (45%)	-
C 5	-	-	-	-	17 (17%)	-

Database	No. of Subjects	Elicitation	Imaging	Camera View View	Labels
AR	126	Posed	Static	Frontal	Emotions
AVEC 2012	7 (train)	Dialog	Video	Mostly Frontal	Continuous Dimensional
Belfast	125	Interview and TV	Video	Occlusion/Frontal	Emotions and dimensions
Bosphorous	105	Posed	Static	3D	FACS AU
BU-3DFE	100	Posed	Static	3D	Emotions
BU-4DFE	101	Posed	Dynamic	3D	Emotions
Cohn-Kanade	97	Posed	Video	Frontal	FACS AU
Cohn-Kanade+	123	Posed and Conversation	Video	Frontal and 15 to the side	FACS AU/ emotions
FABO	23	Posed	Video	Frontal	Landmarks/emotions
GEMEP	10	Acted	Video	Fronta	Emotions
KDEF	70	Posed	Static	Five views	Emotions
JAFFE	10	Posed	Static	Frontal	Emotions
MMI	101	Posed / spontaneous	Static /Video(5min)	Frontal 90 to the side	FACS AU
Face Database MPI	2	Posed	Video	18 intervals	FACS AU
Multi-PIE	337	Posed	Static	15 views 19 illuminations	Emotions
Prkachin-Solomon Pain	129	Pain Induction	Video	Frontal	Landmarks/AU Landmarks
Multi-PIE	72	Posed	Static	Five views	five views
RU-FACS	100	Interview	Video(2min)	Mostly Frontal	FACS AU/Emotions
RML	8	Posed	Video	Frontal	FACS AU
University of Texas Video Database	284	Viewing videoclip	Video(10min)	Frontal	Emotions
VAM-Faces	20	Interview	Video	Mostly Frontal	Continuous Dimensional

Table A.11: Publicly available facial expression databases

References

[1] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Comput. Surv.*, vol. 31, pp. 264–323, Sept. 1999.

[2] E. Keogh and J. Lin, "Clustering of time-series subsequences is meaningless: Implications for previous and future research," *Knowl. Inf. Syst.*, vol. 8, pp. 154–177, Aug. 2005.

[3] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl, "Constrained k-means clustering with background knowledge," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, (San Francisco, CA, USA), pp. 577–584, Morgan Kaufmann Publishers Inc., 2001.

[4] S. Anand, S. Mittal, O. Tuzel, and P. Meer, "Semi-supervised kernel mean shift clustering," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, pp. 1201–1215, June 2014.

[5] B. Kulis, S. Basu, I. S. Dhillon, and R. J. Mooney, "Semi-supervised graph clustering: a kernel approach," *Machine Learning*, vol. 74, no. 1, pp. 1–22, 2009.

[6] Z. Lu and H. H. S. Ip, "Constrained spectral clustering via exhaustive and efficient constraint propagation," in *Proceedings of the 11th European Conference on Computer Vision: Part VI*, ECCV'10, (Berlin, Heidelberg), pp. 1–14, Springer-Verlag, 2010.

[7] F. Zhou, F. De la Torre, and J. F. Cohn, "Unsupervised discovery of facial events," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 2574–2581, June 2010.

[8] F. Zhou, F. De la Torre Frade, and J. K. Hodgins, "Aligned cluster analysis for temporal segmentation of human motion," in *IEEE Conference on Automatic Face and Gestures Recognition*, September 2008.

[9] T. Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 164 – 181, 2011.

[10] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowledge and Information Systems*, vol. 3, no. 3, pp. 263–286, 2001.

[11] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani, "Locally adaptive dimensionality reduction for indexing large time series databases," *ACM Trans. Database Syst.*, vol. 27, pp. 188–228, June 2002.

[12] R. Agrawal, C. Faloutsos, and A. N. Swami, "Efficient similarity search in sequence databases," in *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*, FODO '93, (London, UK, UK), pp. 69–84, Springer-Verlag, 1993.

[13] K.-P. Chan and A. W.-C. Fu, "Efficient time series matching by wavelets," in *Data Engineering, 1999. Proceedings., 15th International Conference on*, pp. 126–133, Mar 1999.

[14] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," in *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, SIGMOD '94, (New York, NY, USA), pp. 419–429, ACM, 1994.

[15] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: A novel symbolic representation of time series," *Data Min. Knowl. Discov.*, vol. 15, pp. 107–144, Oct. 2007.

[16] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, "Experimental comparison of representation methods and distance measures for time series data," *Data Mining and Knowledge Discovery*, vol. 26, no. 2, pp. 275–309, 2013.

[17] E. Frentzos, K. Gratsias, and Y. Theodoridis, "Index-based most similar trajectory search," in *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pp. 816–825, April 2007.

[18] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Data Engineering, 2002. Proceedings. 18th International Conference on*, pp. 673–684, 2002.

[19] L. Chen and R. Ng, "On the marriage of lp-norms and edit distance," in *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pp. 792–803, VLDB Endowment, 2004.

[20] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series.," in *KDD Workshop* (U. M. Fayyad and R. Uthurusamy, eds.), pp. 359–370, AAAI Press, 1994.

[21] J. Aßfalg, H.-P. Kriegel, P. Kröger, P. Kunath, A. Pryakhin, and M. Renz, "Similarity search on time series based on threshold queries," in *Proceedings of the 10th International Conference on Advances in Database Technology*, EDBT'06, (Berlin, Heidelberg), pp. 276–294, Springer-Verlag, 2006.

[22] Y. Chen, M. A. Nascimento, B. Chin, O. Anthony, and K. Tung, "Spade: On shape-based pattern detection in streaming time series," in *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pp. 786–795, April 2007.

[23] C. A. Ratanamahatana and E. Keogh, "Three myths about dynamic time warping data," in *Mining, in the Proceedings of SIAM International Conference on Data Mining (2005*, pp. 506–510, 2005.

[24] H. Shimodaira, K.-i. Noma, M. Nakai, and S. Sagayama, "Dynamic Time-Alignment Kernel in Support Vector Machine," *Advances in Neural Information Processing Systems 14, NIPS2001*, vol. 2, pp. 921–928, December 2001.

[25] F. Zhou, F. De la Torre, and J. K. Hodgins, "Hierarchical aligned cluster analysis for temporal clustering of human motion," *IEEE Transactions Pattern Analysis and Machine Intelligence (PAMI)*, vol. 35, no. 3, pp. 582–596, 2013.

[26] K. T. Vasko and H. T. T. Toivonen, "Estimating the number of segments in time series data using permutation tests," in *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pp. 466–473, 2002.

[27] X. Xuan and K. Murphy, "Modeling changing dependency structure in multivariate time series," in *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, (New York, NY, USA), pp. 1055–1062, ACM, 2007.

[28] Z. Harchaoui, F. Bach, and E. Moulines, "Kernel change-point analysis," in *NIPS*, pp. 609–616, 2008.

[29] S. Liu, M. Yamada, N. Collier, and M. Sugiyama, "Change-point detection in time-series data by relative density-ratio estimation," *Neural Networks*, vol. 43, no. 0, pp. 72 – 83, 2013.

[30] M. Sugiyama, T. Suzuki, and T. Kanamori, "Density-ratio matching under the bregman divergence: a unified framework of density-ratio estimation," *Annals of the Institute of Statistical Mathematics*, vol. 64, no. 5, pp. 1009–1044, 2012.

[31] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf, "Covariate shift by kernel mean matching," in *Dataset Shift in Machine Learning* (J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. Lawrence, eds.), pp. 131–160, Cambridge, MA, USA: MIT Press, 2 2009.

[32] V. Pavlovic, J. M. Rehg, and J. MacCormick, "Learning switching linear models of human motion," in *Advances in Neural Information Processing Systems 13* (T. Leen, T. Dietterich, and V. Tresp, eds.), pp. 981–987, MIT Press, 2001.

[33] S. M. Oh, J. M. Rehg, T. Balch, and F. Dellaert, "Learning and inferring motion patterns using parametric segmental switching linear dynamic systems," *Int. J. Comput. Vision*, vol. 77, pp. 103–124, May 2008.

[34] E. Fox, E. Sudderth, M. Jordan, and A. Willsky, "Bayesian nonparametric inference of switching dynamic linear models," *Signal Processing, IEEE Transactions on*, vol. 59, pp. 1569–1585, April 2011.

[35] R. Cutler and L. S. Davis, "Robust real-time periodic motion detection, analysis, and applications," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, pp. 781–796, Aug 2000.

[36] E. Pogalin, A. Smeulders, and A. Thean, "Visual quasi-periodicity," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, June 2008.

[37] L. Zelnik-Manor and M. Irani, "Statistical analysis of dynamic actions," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, pp. 1530–1535, Sept 2006.

[38] M. Hoai and F. De la Torre, "Maximum margin temporal clustering," in *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS-12)*, vol. 22, pp. 520–528, 2012.

[39] J. Abonyi, B. Feil, S. Nemeth, and P. Arva, "Modified gath–geva clustering for fuzzy segmentation of multivariate time-series," *Fuzzy Sets Syst.*, vol. 149, pp. 39–56, Jan. 2005.

[40] J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmaki, and H. T. Toivonen, "Time series segmentation for context recognition in mobile devices," in *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pp. 203–210, 2001.

[41] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2Nd Edition)*. Wiley-Interscience, 2000.

[42] J.-P. Vert, K. Tsuda, and B. Schölkopf, "A primer on kernel methods," *Kernel Methods in Computational Biology*, pp. 35–70, 2004.

[43] X. Jin and J. Han, "K-medoids clustering," in *Encyclopedia of Machine Learning* (C. Sammut and G. Webb, eds.), pp. 564–565, Springer US, 2010.

[44] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[45] R. Chitta, R. Jin, T. C. Havens, and A. K. Jain, "Approximate kernel k-means: Solution to large scale kernel clustering," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, (New York, NY, USA), pp. 895–903, ACM, 2011.

[46] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems* (T. Dietterich, S. Becker, and Z. Ghahramani, eds.), pp. 849–856, MIT Press, 2001.

[47] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '03, (New York, NY, USA), pp. 267–273, ACM, 2003.

[48] R. Inokuchi and S. Miyamoto, "LVQ clustering and SOM using a kernel function," in *Fuzzy Systems, 2004. Proceedings. 2004 IEEE International Conference on*, vol. 3, pp. 1497–1500, July 2004.

[49] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," in *In Proc. of the ACM SIGMOD Intl. Conference on Management of Data (SIGMOD*, pp. 103–114, 1996.

[50] M. J. Kyan and L. Guan, "The self-organising hierarchical variance map," in *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, pp. 3767–3774, 2006.

[51] S. Basu, M. Bilenko, and R. J. Mooney, "A probabilistic framework for semi-supervised clustering," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, (New York, NY, USA), pp. 59–68, ACM, 2004.

[52] I. Davidson and S. Basu, "A survey of clustering with instance level constraints," 2007. This work was presented as a tutorial at the IEEE ICDM Conference 2005 and ACM KDD Conference 2006.

[53] S. Zhu, D. Wang, and T. Li, "Data clustering with size constraints," *Knowledge-Based Systems*, vol. 23, no. 8, pp. 883 – 889, 2010.

[54] I. Davidson and S. S. Ravi, "Clustering with Constraints: Feasibility Issues and the k-Means Algorithm," in *Proc. 5th SIAM Data Mining Conference*, 2005.

[55] M.-A. Rizoiu, J. Velcin, and S. Lallich, "Structuring typical evolutions using temporal-driven constrained clustering," in *Tools with Artificial Intelligence (ICTAI), 2012 IEEE 24th International Conference on*, vol. 1, pp. 610–617, 2012.

[56] K. Wagstaff and C. Cardie, "Clustering with instance-level constraints," in *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, (San Francisco, CA, USA), pp. 1103–1110, 2000.

[57] T. W. Liao, "Clustering of time series data – a survey," *Pattern Recognition*, vol. 38, no. 11, pp. 1857 – 1874, 2005.

[58] W.-H. Lin and E. Hauptmann, "Structuring continuous video recordings of everyday life using time-constrained clustering," in *IS&T SPIE Symposium on Electronic Imaging*, 2006.

[59] M.-A. Rizoiu, *Semi-supervised Structuring of Complex Data*. PhD thesis, Univ. of Lumière Lyon 2, Lyon, 2013.

[60] F. De la Torre and C. Agell, "Multimodal diaries," in *Multimedia and Expo, 2007 IEEE International Conference on*, pp. 839–842, July 2007.

[61] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in Neural Information Processing Systems 16*, pp. 321–328, MIT Press, 2004.

[62] I. Davidson, K. L. Wagstaff, and S. Basu, "Measuring constraint-set utility for partitional clustering algorithms," in *Knowledge Discovery in Databases: PKDD 2006* (J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, eds.), vol. 4213 of *Lecture Notes in Computer Science*, pp. 115–126, Springer Berlin Heidelberg, 2006.

[63] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration," *Data Min. Knowl. Discov.*, vol. 7, pp. 349–371, Oct. 2003.

[64] Y.-Q. Miao, R. Araujo, and M. Kamel, "Cross-domain facial expression recognition using supervised kernel mean matching," in *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, vol. 2, pp. 326–332, Dec 2012.

[65] R. Araujo, Y.-Q. Miao, M. S. Kamel, and M. Cheriet, "A fast and robust feature set for cross individual facial expression recognition," in *Computer Vision and Graphics* (L. Bolc, R. Tadeusiewicz, L. Chmielewski, and K. Wojciechowski, eds.), vol. 7594 of *Lecture Notes in Computer Science*, pp. 272–279, Springer Berlin Heidelberg, 2012.

[66] Z. Xie and L. Guan, "Multimodal information fusion of audiovisual emotion recognition using novel information theoretic tools," in *Multimedia and Expo (ICME), 2013 IEEE International Conference on*, pp. 1–6, July 2013.

[67] M. Pantic and L. J. M. Rothkrantz, "Automatic analysis of facial expressions: The state of the art," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1424–1445, 2000.

[68] H. Gunes and M. Pantic, "Automatic, dimensional and continuous emotion recognition," *Int. J. Synth. Emot.*, vol. 1, pp. 68–99, Jan. 2010.

[69] M. Wöllmer, F. Eyben, S. Reiter, B. Schuller, C. Cox, E. Douglas-cowie, and R. Cowie, "Abandoning emotion classes - towards continuous emotion recognition with modelling of long-range dependencies," in *in Proceedings Interspeech*, 2008.

[70] M. A. Nicolaou, H. Gunes, and M. Pantic, "Output-associative rvm regression for dimensional and continuous emotion prediction," *Image Vision Comput.*, vol. 30, pp. 186–196, Mar. 2012.

[71] M. Grimm and K. Kroschel, "Emotion estimation in speech using a 3d emotion space concept," in *Robust Speech Recognition and Understanding* (M. Grimm and K. Kroschel, eds.), pp. 281–300, I-Tech Education and Publishing, Vienna, Austria, 2007.

[72] J. Hoey, "Hierarchical unsupervised learning of facial expression categories," in *Detection and Recognition of Events in Video, 2001. Proceedings. IEEE Workshop on*, pp. 99 –106, 2001.

[73] L. Zelnik-Manor and M. Irani, "Temporal factorization vs. spatial factorization," in *Computer Vision - ECCV 2004* (T. Pajdla and J. Matas, eds.), vol. 3022 of *Lecture Notes in Computer Science*, pp. 434–445, Springer Berlin Heidelberg, 2004.

[74] F. De La Torre, J. Campoy, Z. Ambadar, and J. F. Cohn, "Temporal segmentation of facial behavior," in *International Conference on Computer Vision*, pp. 1–8, 2007.

[75] F. Zhou, F. De la Torre, and J. F. Cohn, "Unsupervised discovery of facial events," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[76] R. Araujo and M. S. Kamel, "Audio-visual emotion analysis using semi-supervised temporal clustering with constraint propagation," in *Image Analysis and Recognition* (A. Campilho and M. Kamel, eds.), Lecture Notes in Computer Science, pp. 3–11, Springer International Publishing, 2014.

[77] R. Araujo and M. S. Kamel, "A semi-supervised temporal clustering method for facial emotion analysis," in *Multimedia and Expo Workshops (ICMEW), 2014 IEEE International Conference on*, pp. 1–6, July 2014.

[78] R. Araujo and M. S. Kamel, "Semi-supervised kernel-based temporal clustering," in *13th International Conference on Machine Learning and Applications, ICMLA 2014, Detroit, MI, USA, December 3-5, 2014.*, pp. 123–128, 2014.

[79] Y.-L. Tian, T. Kanade, and J. F. Cohn, "Facial expression analysis," *Handbook of Face Recognition*, vol. 3, no. 5, pp. 247–276, 2005.

[80] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face description with local binary patterns: Application to face recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, pp. 2037–2041, Dec 2006.

[81] Y. Venkatesh, A. K. Kassim, and O. R. Murthy, "Resampling approach to facial expression recognition using 3d meshes," in *20th International Conference on Pattern Recognition*, pp. 3772–3775, Aug. 2010.

[82] X. Zhao, D. Huang, E. Dellandrea, and L. Chen, "Automatic 3d facial expression recognition based on a bayesian belief net and a statistical facial feature model," in *20th International Conference on Pattern Recognition*, pp. 3724–3727, Aug. 2010.

[83] S. Koelstra, M. Pantic, and I. Patras, "A dynamic texture-based approach to recognition of facial actions and their temporal models," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, pp. 1940–1954, Nov 2010.

[84] Y. Ji and K. Idrissi, "Using moments on spatiotemporal plane for facial expression recognition," in *Pattern Recognition (ICPR), 2010 20th International Conference on*, pp. 3806–3809, Aug 2010.

[85] Y. Hu, Z. Zeng, L. Yin, X. Wei, J. Tu, and T. Huang, "A study of non-frontal-view facial expressions recognition," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pp. 1–4, Dec 2008.

[86] S. Moore and R. Bowden, "Effects of pose on facial expression recognition," in *Proc. British Machine Vision Conference BMVC'09*, 2009.

[87] M. Pantic and I. Patras, "Dynamics of facial expression: recognition of facial actions and their temporal segments from face profile image sequences," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 36, pp. 433 –449, april 2006.

[88] O. Rudovic, I. Patras, and M. Pantic, "Regression-based multi-view facial expression recognition," in *20th International Conference on Pattern Recognition*, pp. 4121–4124, Aug. 2010.

[89] M. S. Bartlett, G. C. Littlewort, M. G. Frank, C. Lainscsek, I. R. Fasel, and J. R. Movellan, "Automatic recognition of facial actions in spontaneous expressions," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 1, pp. 22–35, Sep 2006.

[90] A. Maalej, B. B. Amor, M. Daoudi, A. Srivastava, and S. Berretti, "Local 3d shape analysis for facial expression recognition," in *20th International Conference on Pattern Recognition*, pp. 4129–4132, Aug. 2010.

[91] T. Yun and L. Guan, "Fiducial point tracking for facial expression using multiple particle filters with kernel correlation analysis," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pp. 373–376, Sept 2010.

[92] Y. Tie and L. Guan, "A deformable 3-d facial expression model for dynamic human emotional state recognition," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 23, pp. 142–157, Jan 2013.

[93] Y.-L. Tian, T. Kanade, and J. F. Cohn, "Evaluation of gabor-wavelet-based facial action unit recognition in image sequences of increasing complexity," in *Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 229–234, May 2002.

[94] P. Li, S. L. Phung, A. Bouzerdoum, and F. H. C. Tivive, "Improved facial expression recognition with trainable 2-d filters and support vector machines," in *20th International Conference on Pattern Recognition*, pp. 3732–3735, Aug 2010.

[95] Y.-L. Tian, T. Kanade, and J. F. Cohn, "Recognizing action units for facial expression analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, pp. 97–115, Feb 2001.

[96] M. Valstar, M. Pantic, and I. Patras, "Motion history for facial action detection in video," in *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, vol. 1, pp. 635 – 640 vol.1, oct. 2004.

[97] P. Ekman and W. Friesen, *Facial Action Coding System: A Technique for the Measurement of Facial Movement.* Palo Alto: Consulting Psychologists Press, 1978.

[98] P. Ekman, W. Friesen, and J. Hager, *The Facial Action Coding System.* Salt Lake City: Research Nexus eBook, second ed., 2002.

[99] G. Littlewort, J. Whitehill, T.-F. Wu, N. Butko, P. Ruvolo, J. Movellan, and M. Bartlett, "The motion in emotion a cert based approach to the fera emotion challenge," in *Proceedings of the 9th IEEE Conference on Automatic Face and Gesture Recognition, Workshop on Facial Expression Recognition and Analysis Challenge*, 2011.

[100] R. Kehrein, "The Prosody of Authentic Emotions," in *Proceedings of Speech Prosody Conference*, pp. 423–426, 2002.

[101] J. A. Russell, "A circumplex model of affect," *Journal of Personality and Social Psychology*, vol. 39, pp. 1161–1178, 1980.

[102] M. Grimm, K. Kroschel, and S. Narayanan, "The vera am mittag german audio-visual emotional speech database," in *Multimedia and Expo, 2008 IEEE International Conference on*, pp. 865–868, June 2008.

[103] C. Shan, S. Gong, and P. W. McOwan, "Facial expression recognition based on local binary patterns: A comprehensive study," *Image and Vision Computing*, vol. 27, no. 6, pp. 803 – 816, 2009.

[104] A. Sayedelahl, R. Araujo, and M. S. Kamel, "Audio-visual feature-decision level fusion for spontaneous emotion estimation in speech conversations," in *Multimedia and Expo Workshops (ICMEW), 2013 IEEE International Conference on*, pp. 1–6, July 2013.

[105] M. Grimm and K. Kroschel, "Evaluation of natural emotions using self assessment manikins," in *Automatic Speech Recognition and Understanding, 2005 IEEE Workshop on*, pp. 381–385, Nov 2005.

[106] P. Viola and M. Jones, "Robust real-time object detection," in *International Journal of Computer Vision*, 2001.

[107] B. Schuller, M. Valstar, R. Cowie, and M. Pantic, "Avec 2012: The continuous audio/visual emotion challenge - an introduction," in *Proceedings of the 14th ACM International Conference on Multimodal Interaction*, ICMI '12, (New York, NY, USA), pp. 361–362, ACM, 2012.

[108] S. Schulz and A. Woerner, "Automatic motion segmentation for human motion synthesis," in *Articulated Motion and Deformable Objects* (F. Perales and R. Fisher, eds.), vol. 6169 of *Lecture Notes in Computer Science*, pp. 182–191, Springer Berlin Heidelberg, 2010.

[109] A. Vögele, B. Krüger, and R. Klein, "Efficient unsupervised temporal segmentation of human motion," in *2014 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, July 2014.

[110] D. Kulić and Y. Nakamura, "Comparative study of representations for segmentation of whole body human motion data," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 4300–4305, Oct 2009.

[111] A. Fod, M. J. Mataric, and O. C. J. Jenkins, "Automated derivation of primitives for movement classification," *Autonomous Robots*, vol. 12, no. 1, pp. 39–54, 2002.

[112] J. F.-S. Lin and D. Kulić, "Segmenting human motion for automated rehabilitation exercise analysis," in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pp. 2881–2884, Aug 2012.

[113] M. Pomplun and M. Matarić, "Evaluation metrics and results of human arm movement imitation," in *in: Proceedings of the 1st IEEE-RAS International Conference on Humanoid Robotics*, 2000.

[114] N. Koenig and M. J. Matarić, "Behavior-based segmentation of demonstrated task," in *International Conference on Development and Learning*, (Bloomington, IN), May 2006.

[115] W. Ilg, G. H. Bakir, J. Mezger, and M. Giese, "On the representation, learning and transfer of spatio-temporal movement characteristics," *International Journal of Humanoid Robotics*, vol. 1, pp. 613–636, 12 2004.

[116] W. Takano and Y. Nakamura, "Humanoid robot's autonomous acquisition of proto-symbols through motion segmentation," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pp. 425–431, Dec 2006.

[117] M. Shell, "Carnegie mellon university motion capture database.," 2012.

[118] A. Elgohary, "Scalable embeddings for kernel clustering on mapreduce," Master's thesis, University of Waterloo, Waterloo, 2014.

[119] Y. Xue and L. Jin, "A naturalistic 3d acceleration-based activity dataset amp; benchmark evaluations," in *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pp. 4081–4085, Oct 2010.

[120] M. W. Robards and P. Sunehag, "Semi-markov kmeans clustering and activity recognition from body-worn sensors," in *Data Mining, 2009. ICDM '09. Ninth IEEE International Conference on*, pp. 438–446, Dec 2009.