Multi-Variate Time Series Similarity Measures and Their Robustness Against Temporal Asynchrony

by

Yash Garg

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2015 by the
Graduate Supervisory Committee:

Kasim Selçuk Candan, Chair
Gerardo Chowell-Punete
HangHang Tong
Hasan Davulcu
Maria Luisa Sapino

ARIZONA STATE UNIVERSITY

December 2015

ABSTRACT

The amount of time series data generated is increasing due to the integration of sensor technologies with everyday applications, such as gesture recognition, energy-optimization, health care, video surveillance. The use of multiple sensors simultaneously capturing different aspects of the real world attributes has also led to an increase in dimensionality from uni-variate to multi-variate time series. This has facilitated richer data representation but also has necessitated algorithms determining similarity between two multi-variate time series for search and analysis.

Various algorithms have been extended from uni-variate to multi-variate case, such as multi-variate versions of Euclidean distance, edit distance, dynamic time warping. However, it has not been studied how these algorithms account for asynchrony in time series. Human gestures, for example, exhibit asynchrony in their patterns as different subjects perform the same gesture with varying movements in their patterns at different speeds. In this thesis, we propose several algorithms (some of which also leverage metadata describing the relationships among the variates). In particular, we present several techniques that leverage the contextual relationships among the variates when measuring multi-variate time series similarities. Based on the way correlation is leveraged, various weighing mechanisms have been proposed that determine the importance of a dimension for discriminating between the time series as giving the same weight to each dimension can led to misclassification. We next study the robustness of the considered techniques against different temporal asynchronies, including shifts and stretching.

Exhaustive experiments were carried on datasets with multiple types and amounts of temporal asynchronies. It has been observed that accuracy of algorithms that rely on data to discover variate relationships can be low under the presence of temporal asynchrony, whereas in case of algorithms that rely on external metadata, robustness against asynchronous distortions tends to be stronger. Specifically, algorithms using external metadata have better classification accuracy and cluster separation than existing state-of-the-art work, such as EROS, PCA, and naïve dynamic time warping.

DEDICATION

To my grandparents,

Shanti Devi Agrawal & Shyamlal Shah Agrawal*,*

you will always be missed.

If it weren't for you, I wouldn't be here

ACKNOWLEDGEMENT

Research is formalized curiosity. It is poking and prying with a purpose - ***Zora***

I would like to take this opportunity to express my deep gratitude to my advisor and mentor, Dr. Kasim Selçuk Candan. Dr. Candan, this thesis is the outcome of your vision, insight and specially your boundless patience throughout this process. You gave me the opportunity to do the most interesting and challenging work of my academic career. You are truly a great mentor. You not only provided constant academic guidance but gave me suggestions on how to overcome the difficulties that I met in my life. Achieving this milestone would not have been possible without your invaluable feedback and unending support. You have been and will be my constant source of inspiration for dedication, perfection and curiosity. I will always be in-debt of you for believing in me and giving me the opportunity to pursue PhD with you.

I am and will always be in-debt of Dr. Maria-Luisa Sapino for her valuable input on my work that helped me make it complete and comprehensive. I would also like to thank Dr. Gerardo Chowell-Puente, Dr. Hasan Davulcu and Dr. HangHang Tong for being on my committee and giving their valuable guidance for shaping my thesis.

I would like thank my dad, Hukumchand Garg, and my mom, Heeramani Garg, for their unflagging love and for providing me with the means for a world class education. My sister, Reema Garg, you are a great friend, your unconditional love and support has helped me build the path towards this accomplishment.

I would like to thank my lab-mates at Emitlab - Silvestro Roberto-Poccia, Jung Hyun Kim, Xinsheng Li, Xilun Chen, Shengyu Wang, Sicong Liu and Aneesha Bhat for their insightful discussions. Special thanks to Parth Nagarkar, for his unyielding badgering to keep me motivated and for all his help in write this thesis. I am also thankful to former Emitlab members, Mithila Nagendra and Mijung Kim.

Particular note of appreciation goes to my friends, Ankit Goretela, Kunal Parikh and Parag Chandakkar, for their continued support and enthusiasm that made me sail through this journey. Ankit and Kunal, you guys quietly suffered through the hard times while editing my intangible and intrinsic writing to make it more readable and to get it to the "*English Shape*" it is today. Parag you made it easy for me to write the content that not only presents the concept, I am trying to explain but helped me make it more presentable and understandable with all the technical jargons in this thesis.

It would be selfish of me not to thank, Brinton McMillan and Micheal McAllister, for helping me through all the technical difficulties I brought upon you guys.

Time is not a line, but a series of now-points - ***Taison Deshimaru***

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

1.1 Overview

In the past decade, time series have become intrinsically involved in everyday applications with the integration of sensor infrastructures to the applications. This has drawn major emphasis on search and analysis of time series. Many applications generate large amount of time series data, at a very fast pace, that requires efficient storage and retrieval mechanism for analysis. This analysis could be similarity search over a time series database to discover existing patterns or for predicting future patterns. For example:

1. Weather monitoring systems employ time series analysis to determine weather patterns and to predict future weather patterns [35].

2. In stock markets, financial data analysis helps understand fluctuations in stock pricing, detect fraudulent trading and establish correlation between different stocks by exploiting the temporal correlation amongst the stocks [66, 67, 69].

3. Video surveillance helps investigate crowd movement patterns at different point in time to facilitate congestion management and planning of public transportation schedules. This has been made possible by temporal video processing[29].

4. Network analysis helps in load balancing, preparing for sudden burst of traffic in the network by analyzing past and current patterns. Further in-depth analysis facilitates detection of irregular pattern that might result in intrusion into the network [74].

5. Efforts have been made to design energy efficient buildings which has increased the involvement of sensor networks that tracks various attributes such as temperature, humidity, pressure, number of people in a room to give an optimal setting to reduce the carbon footprint [53].

6. Motion recognition [70] has been dealing with determining similarities amongst how different people perform same activities.

Wide application of time series analysis has not only led to an active research domain but has unearthed new paths for data analysis whilst exploiting the temporal aspect of the data, furthermore, instigating a devouring need to determine similar time series stored in an enormous data corpse. Similar time series can be retrieved by using any of the following *two* approaches, first, using k-nearest neighbor query, k-NN, [13, 36, 42] and second, by performing clustering of time series [6, 11, 48, 56, 76].

The need to define large amount of information associated with time series, effectively, has undoubtedly proven that *one*-dimensional time series **(uni-variate time series)** is not sufficient to store and represent all the necessary information associated with it. This has led to a significant increase in dimensionality of time series called *multi*-dimensional time series **(multi-variate time series)** to store information from multiple attributes at the same time.

Definition 1.1. ***Time Series or Uni-Variate Time Series****, T, is defined as a sequences of pair of data values and time at which they were recorded for a certain attributes. In general these timestamps can be of equal increments or can be varying time increments*[1]*.*

$$T = [< t_1, d_1 >, < t_2, d_2 >, ..., < t_N, d_N >] \tag{1.1}$$

[1]gap between two consecutive timestamps

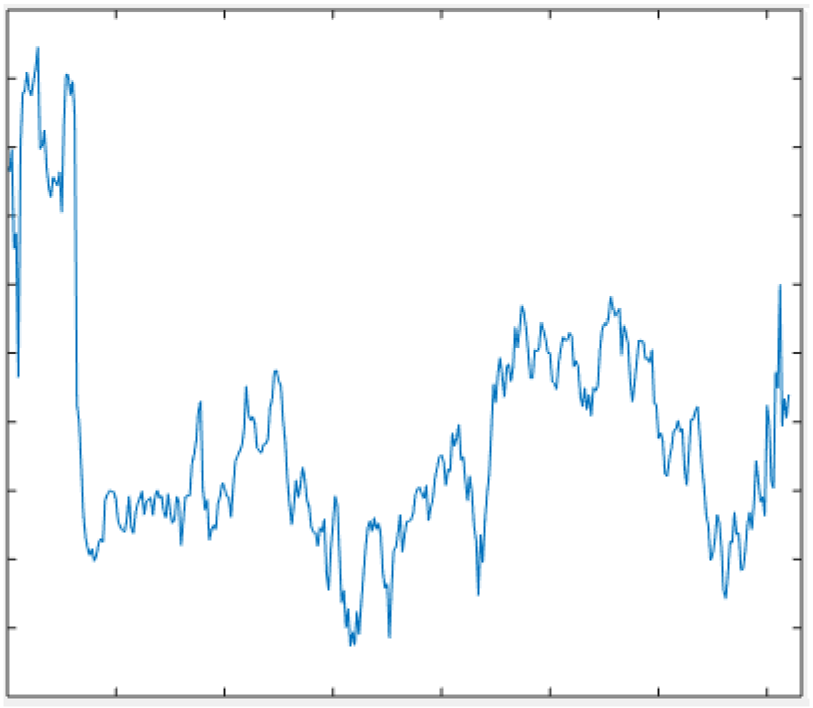


Figure 1.1: Uni-Variate Time Series for Opening Price of GOOG at NASDAQ from March 27th 2014 - August 28th, 2015. Source : [4]

where, N is defined as the length of time series i.e number of sampled pairs, d is the data value and t is the corresponding timestamps. Figure 1.1 illustrates an example of uni-variate time series.

Definition 1.2. ***Multi-Variate Time Series***, $\mathbb{T}$, *is set of time series, T, that record different attributes.*

$$\mathbb{T} = [T_1, T_2, ..., T_D]. \tag{1.2}$$

where, $\mathbb{T}$ is the multi-variate time series, set of uni-variate time series (T) and the D is dimensionality of $\mathbb{T}$ or the number of uni-variate time series. Figure 1.2 illustrates an example for multi-variate time series.

Definition 1.3. ***Correlated Multi-Variate Time Series,*** $\mathbb{T}_R$, *is defined as the a multi-variate time series where a relationship or dependency exists between different*

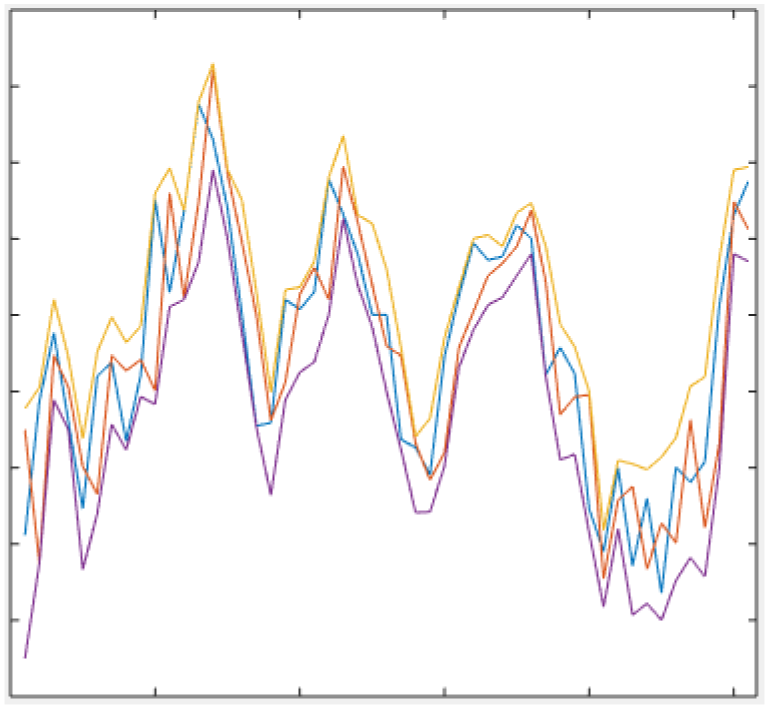


Figure 1.2: Multi-Variate Time Series for GOOG at NASDAQ for Opening Price, Closing Price, Daily Maximum Price and Daily Minimum Price from March 27th 2014 - August 28th, 2015. Source : [4]

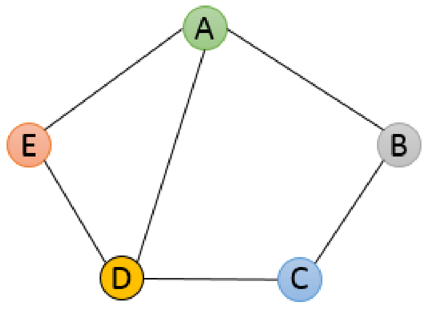
variates of the time series.

$$\mathbb{T}_R = [\mathbb{T}, R] \tag{1.3}$$

where, $\mathbb{T}$ is a multi-variate time series, and R is the relationship matrix / contextual correlation between the variates. For example, this relationship can be the sensor distance or nearness or hierarchy similarity matrix if the multi-variate time series is based on a sensor based networks or the correlation coefficient if there exists a proportionality relationship.

Definition 1.4. ***Contextual Correlation/Relationship Matrix***, *R, defines a correlation model, if any, over the variates of a time series. It could be defined as:*

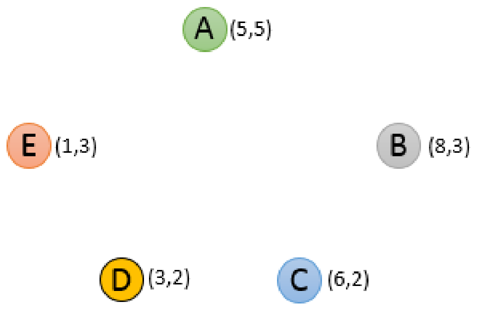
$$R[i, j] = \Theta(T_i, T_j) \tag{1.4}$$



(a)

$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 1 & 0 & 1 & 1 \\ B & 1 & 0 & 1 & 0 & 0 \\ C & 0 & 1 & 0 & 1 & 0 \\ D & 1 & 0 & 1 & 0 & 1 \\ E & 1 & 0 & 0 & 1 & 0 \end{array}$$

(b)



(c)

$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0.00 & 3.00 & 3.16 & 3.60 & 4.47 \\ B & 3.00 & 0.00 & 2.23 & 5.09 & 7.00 \\ C & 3.16 & 2.33 & 0.00 & 3.00 & 5.09 \\ D & 3.60 & 5.09 & 3.00 & 0.00 & 2.23 \\ E & 4.47 & 7.00 & 5.09 & 2.23 & 0.00 \end{array}$$

(d)

Figure 1.3: (a) Physical Connectivity Between Different Sensors (b) Relationship as Graph Connectivity (c) Spatial Position of Sensors (d) Relationship as Sensor Distance (Euclidean Distance)

where $(i, j) \in [1, D]$ *and* Θ *is a application specific function to determine relationship between two variates. This relationship could be computed by using domain knowledge, such as sensor distance or physical connectivity or by examining historical data to determine correlation between the variates.*

With wide spectrum of applications, time series and its varied representations have facilitated involvement of associated metadata with the time series for efficient and effective similarity-based retrieval. The most important concern here is to define the similarity between two time series. For the purpose of quantifying similarity, there is an incumbent need to define a new distance function to leverage the metadata, such as relationship matrix, associated with the time series. This metadata could be used to extract hidden relationships between the variates.

Definition 1.5. ***Variate Hierarchy***, *H_D, is a agglomerative clustering technique that aims to determine hierarchy of variates. The decision to cluster two variates is based on strength of their relationship. Strength of their relationship can be extracted based on how close they are in space (Fig:1.3 (d)) or closely they are connected to each other (Fig:1.3 (b))*

$$MaximumDistanceCluster = max\{R(x, y) \ : \ x \in T, \ y \in T, \ x \neq y\} \tag{1.5}$$

$$MaximumDistanceCluster = min\{R(x, y) \ : \ x \in T, \ y \in T, \ x \neq y\} \tag{1.6}$$

where x_T and y_T are two variates in time series $\mathbb{T}$ and are clusters iff they satisfy the condition (Eq: 1.5 or Eq: 1.6) being used.

Definition 1.6. ***Hierarchy Similarity Matrix***, *H, is defined as the hop similarity between two variates in the variate hierarchy, H_D, created on the basis of the relationship matrix, R.*

$$H(x_T, y_T) = height(H_D) - \frac{edgeDistance(x_T, y_T)}{2} \tag{1.7}$$

where, x_T and y_T are located at the leaf of H_D and $x_T, y_T \in \mathbb{T}$, $height(H_D)$ is defined as number of edges between root and the furthest leaf node in H_D, $edgeDistance(x_T, y_T)$ is the shortest path (number of edges) between x_T and y_T.

To leverage the information extracted from metadata we need an effective distance function to incorporate metadata into similarity measure.

Definition 1.7. *[23] defines* **distance**, *dis, on the data space, $\mathcal{D}$, on time series where $X, Y \in \mathcal{D}$, as follows:*

$$dis : \mathcal{D} \times \mathcal{D} \longrightarrow \mathbb{R} \tag{1.8}$$

where, $\mathbb{R}$ is a real data set. This distance can also be regarded as dissimilarity on $\mathcal{D}$ and following holds true of dis:

1. $dis(X, Y) \geq 0$ - *Non-Negativity*

2. $dis(X, Y) = dis(y, x)$ - *Symmetry*

3. $dis(X, X) = dis(Y, Y) = 0$ - *Reflexivity*

Distance function plays a very significant role in similarity-based retrieval as it is used to quantify similarity of a time series to the query time series. Thus, applications such as classification are directly affected by the choice of distance function and also its robustness to accommodate out-of-phase time series. Out-of-phase does not necessarily mean exact phase difference but also refers to the temporal shift in patterns (local time shift) and wavelength of patterns (acceleration).

Plentiful amount of measures have been proposed to determine similarity between a pair of uni-variate time series like *euclidean distance* [23], *cosine distance*[14, 60] and *dynamic time warping* [12, 40, 61]. Euclidean distance and cosine similarity are well known measures for their ease of computation at the same time, they cannot accommodate temporal changes in patterns, such as, shift and speed. To overcome the limitation of local-time shift, dynamic time warping was proposed. But its robustness to what extent of asynchrony is acceptable is yet be investigated. More details on these measures have been presented in Section 2.1 and Section 2.3.

1.2 Background and Motivation

1.2.1 Similarity Search Query

Similarity-based retrieval has become an universal terminology for mechanisms that determines similarity between a pair of objects. These objects can be multimedia (audio, video, image), text (.doc, .pdf) or a data points. In general, similarity-based retrieval on a time series data space, $\mathcal{D}$, can be achieved by deploying one of the following concepts:

1. *Pattern similarity query* [8, 31]- is defined as a query, on a time series data space, $\mathcal{D}$, with a query file, $\mathcal{Q}$, that returns all the time series contained in $\mathcal{D}$ that are relatively similar to $\mathcal{Q}$ or has some acceleration or deceleration in their pattern.
$$simQuery(\mathcal{Q}, \mathcal{D}) \tag{1.9}$$

2. *Pattern containment query* [64] - is defined as a query, on a time series data space, $\mathcal{D}$, that searches for a specific pattern, $\mathcal{Q}_{\mathcal{P}}$, contained in $\mathcal{Q}$ and returns all the time series that contains $\mathcal{Q}_{\mathcal{P}}$ in them.
$$simPatternQuery(\mathcal{Q}_{\mathcal{P}}, \mathcal{D}) \tag{1.10}$$

These queries can be subtly used to query a time series data space, $\mathcal{D}$, to determine the top-k similar time series in $\mathcal{D}$, *k-Nearest Neighbors Problem*, with respect to $\mathcal{Q}$ or $\mathcal{Q}_{\mathcal{P}}$ [22, 28] and also for range query where the similar files must satisfy a threshold condition.

There are different algorithms proposed for the purpose of similarity search but they can mainly be classified under two headings: first, range query and second, k-nearest neighbor query.

Definition 1.8. ***Range query***, *similarity/distance threshold query [17], Given a distance function or a similarity measure, dis, a data space, $\mathcal{D}$, a query file, $\mathcal{Q}$, and a threshold, ∇, where $\nabla \geq 0$, finds all the matching files in the $\mathcal{D}$ that pass the threshold condition [19].*

$$rangeQuery(\mathcal{Q}, \nabla, dis) = \{\mathbb{S} \in \mathcal{D} \mid dis(\mathcal{Q}, \mathbb{S}) \leq \nabla\} \tag{1.11}$$

where, $\mathbb{S} = \phi$, *if there is no file in* $\mathcal{D}$ *for which* $dis(\mathcal{D}, \mathbb{S}) \nless \nabla$ *[26, 27]*

Definition 1.9. ***k-nearest neighbor query****,* kNN*, Given a distance function or a similarity function,* dis*, a data space,* $\mathcal{D}$*, a query file,* $\mathcal{Q}$*, and top-k, where* $k \geq 1$*, will return k most similar data file to* $\mathcal{Q}$ *in* $\mathcal{D}$*.*

$$knnQuery(\mathcal{Q}, k, dis) = \{\mathbb{S}_i \ \in \ \mathcal{D} \mid \forall \ \mathbb{S} \in \mathcal{D} - \mathcal{D}', dis(\mathbb{S}_i, \mathcal{Q}) \ \leq \ dis(\mathbb{S}, \mathcal{Q})\} \quad (1.12)$$

where, $\mathcal{D}' = \phi, if \ i = 1$ *and* $\mathcal{D}' = \{ \ \mathbb{S}_1, .., \ \mathbb{S}_{i-1}\}, \ if \ 1 < i \leq k$ *[26]*

Similarity queries, such as *Def: 1.8 and Def: 1.9*, can be specialized for querying time series as a pattern similarity or pattern containment queries. These can further be classified as whole-matching query and subsequence matching query where temporal similarity is also taken into account. Formal formulae of these queries will be defined in the view of a range query but they can similarly be written for k-nearest neighbor query as well.

Definition 1.10. *[58] defines* ***whole matching query*** *as, "given a query file, is matched with all the file in the data space to find the ones that are either exactly identical or similar to the query time series". Formally, given a distance function or similarity measure,* dis*, a data space,* $\mathcal{D}$*, a query time series,* $\mathcal{Q}$*, and a threshold,* ∇*, the whole matching query finds all the time series in* $\mathcal{D}$ *that are exactly identical or similar to the* $\mathcal{Q}$*.*

$$wholeMatchQuery(\mathcal{Q}, \nabla, dis) = \{\mathbb{S} \ \in \ \mathcal{D} \mid dis(\mathcal{Q}, \ \mathbb{S}) \leq \nabla\} \quad (1.13)$$

where, $\mathbb{S}$ *and* $|\mathbb{S}_i|$ *is the set of matching time series and length of ith time seriesrespectively.*

Definition 1.11. ***Subsequence matching query*** *[34], can also be regarded as partial match query. The interest of this type of query is to find all the objects that are* *partly similar to the query file. Formally, given a distance function or similarity measure, dis, a data space, $\mathcal{D}$, a query time series, $\mathcal{Q}_{\mathcal{P}}$, with length less than the minimum length of any time series in $\mathcal{D}$ and a threshold, ∇, will return all the time series that contains exact or similar pattern described in $\mathcal{Q}_{\mathcal{P}}$ that satisfy the ∇ condition.*

$$subSequenceMatch(\mathcal{Q}_{\mathcal{P}}, \nabla, dis) = \{\mathbb{S} \in \mathcal{D} \mid dis(\mathbb{S}\,[t_s : (t_s + |\mathcal{Q}| - 1)]) \leq \nabla\} \quad (1.14)$$

where, $|\mathcal{Q}| \leq |\mathbb{S}|$ i.e. the temporal length of $\mathcal{Q}$ must be less than the length of $\mathbb{S}$ and t_s is the start time of the subsequence.

1.2.2 Classification & Clustering

Berkhin [11] refers to clustering as "division of data into groups, called clusters, of similar objects. Each cluster consist of objects that are similar to one another and dissimilar to objects from other clusters." These clusters are the outcome of *patterns* contained the data objects. Wide application of clustering and classification is evident from the amount of work done in multiple domains [6, 11, 24, 25, 30, 48, 56, 76].

Clustering can be used as an exploratory or a confirmatory measure based on the application. Intuitively, a cluster contains similar patterns in it, if not identical, from the patterns contained in other clusters [37]. *Figure 1.4* shows, how the data spread can be categorized into different clusters and how objects that are close in the space are inherently similar to each other.

As discussed, the crux of clustering is to group similar object into a cluster. It is important to make a careful selection of a distance function or similarity measure. As the distance function is used to quantify the similarity or dissimilarity between two objects, therefore its effectiveness is very crucial for efficient clustering. *Euclidean distance* has been the most common choice as a similarity measure because of it's

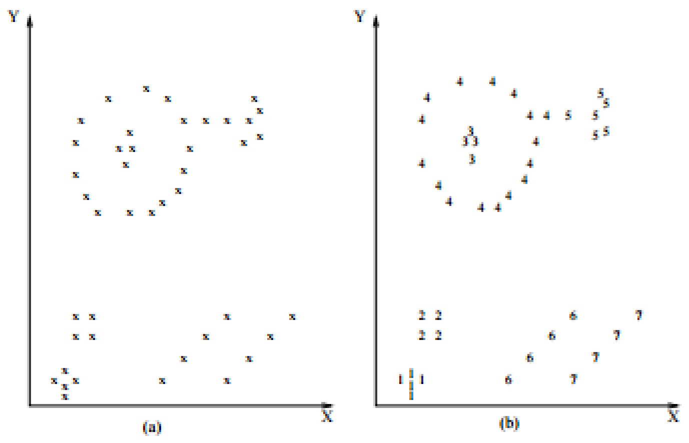


Figure 1.4: (a) Shows The Data Point Spread In The Space, (b) Based On The Separation and Closeness Of Data Point In The Space Different Label Where Assigned That Suggests Similar Data Points Being Grouped Into Different Clusters. Source : [37]

simplicity and linear computational cost.

$$euclidean(X, Y) = \sqrt[2]{\sum_{i=1}^{N}(X_i - Y_i)^2} \tag{1.15}$$

Euclidean distance belongs to the family of $L_p - norm$, Minkowski Distance, where $p = 2$. In Chapter 2, details on the merits and demerits of euclidean distance as a similarity measure for time series data is discussed. More measures like cosine distance, edit distance, dynamic time warping, sub-sequence matching can also be used. Once the distance measure is selected, there are different methodologies that can be used to perform clustering and they are as follows:

1. *Agglomerative and Decisive Clustering* [65] - The two techniques are the two subsets of hierarchical clustering. Agglomerative clustering is a bottom-up clustering technique that considers every object in the data space, $\mathcal{D}$, as individual clusters and iteratively clusters two such object at a given point in time that are

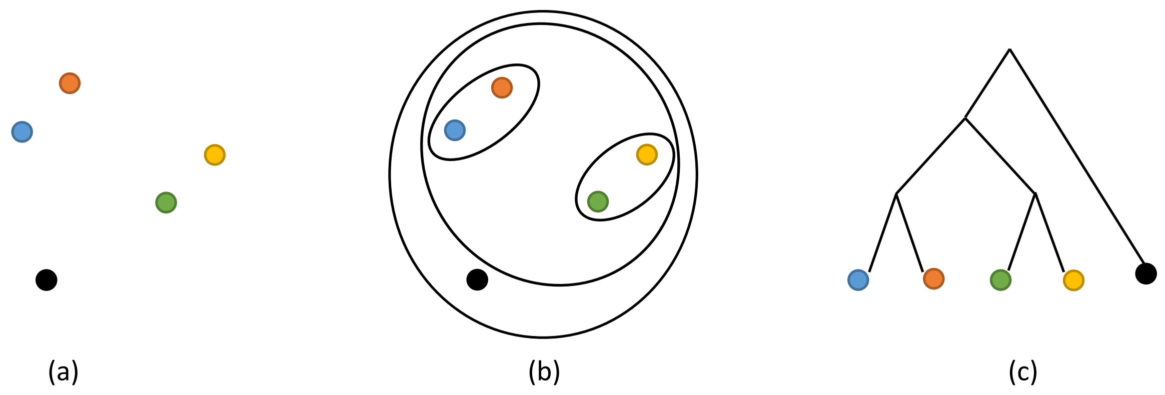


Figure 1.5: (a) Data Space, (b) Hierachical Clustering, (c) Dendogram. Source: [1]

closest to each other until the data space is left with one single big cluster that contains hierarchical clustered objects. On the other hand, decisive clustering, a top-down approach, begins with the assumption that all the objects in the data space are, in fact, a part of one single clusters and hierarchically splits the cluster into two sub cluster until every object belongs to a individual cluster containing itself only. Figure 1.5 illustrates visually the working of hierarchical clustering.

2. *Monolithic and Synthetic Clustering* [10] - These clustering techniques are based on how features associated with the objects are used to perform clustering. When all the features, all dimensions, are used for distance computation the resultant clustering is regarded synthetic clustering, whereas when single feature, one-dimension, is used for clustering, it is regarded as monolithic clustering. Anderberg [10] presents one to the initial monolithic algorithm where features were sequentially considered, i.e. first feature is used to make the first division on data space into two subsets followed can used of second feature to subdivide the subsets into two subsets each.

3. *Deterministic and Stochastic Clustering* [9]- In deterministic clustering exactly

same clusters are achieved on the data space irrespective of when the clustering algorithm is executed whereas in stochastic clustering, different clusters are returned on different iterations of the algorithm which results in good clusters.

4. *Hard and Fuzzy Clustering* [33] - In hard clustering, each object is contained in only one cluster which results in crisp clusters whereas in fuzzy clustering an object can be associated with more than one cluster.

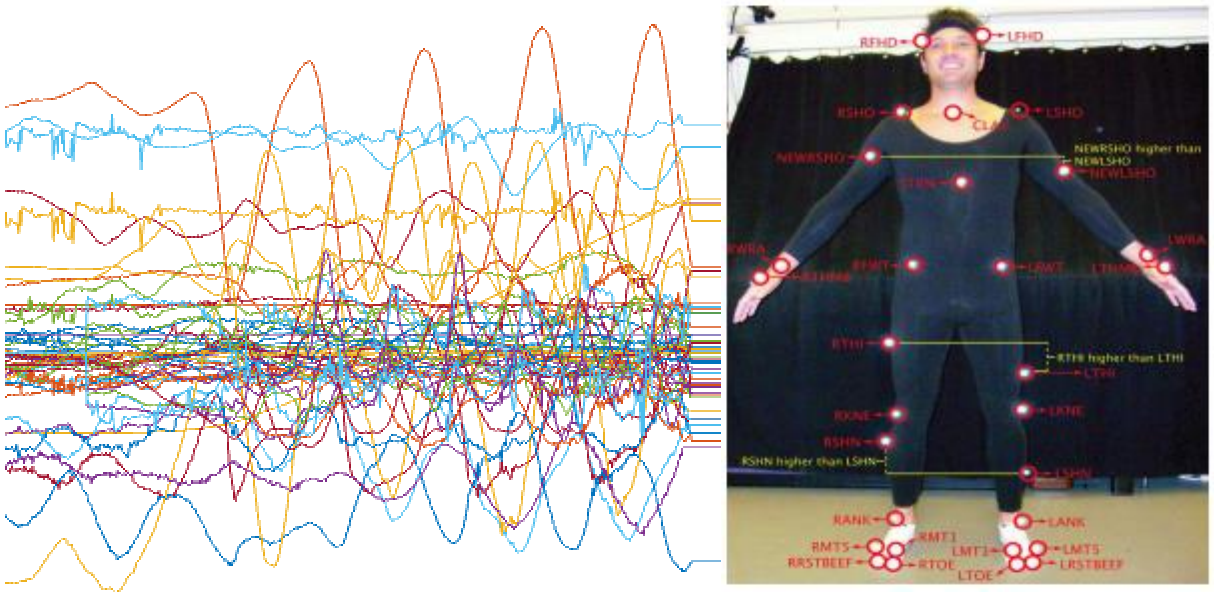
This thesis addresses the problem of increasing the accuracy of *pattern similarity query*, k-nearest neighbor search, and improving effective cluster separation of influence of temporal asynchrony in time series, as the complexity and performance of a similarity measure is affected when the dimensionality and asynchrony in data increases.

1.3 Limitations in Existing Work

With wide application of time series, such as epidemic spread analysis [50], has led to an active spatio-temporal research area that have posed some important questions again and again. Cai in [15] and Wang in [70] proposed algorithms that leverage variate relationships for time series comparison. These challenges are:

1. *High-dimensionality* [2] - With the need to understand the effects of different attributes in various applications; uni-variate time series is no longer sufficient enough to describe and record all the information. As the number of attributes increases, it becomes more difficult to determine similarity between two multi-variate time series as the computational complexity is directly affected by the dimensionality.

[2]which is called multi-variate time series



(a) (b)

Figure 1.6: (a) High-Dimensionality Of The Time Series Can Be Seen, 62-Dimensional (b) Sensor Arrangement On human Body To Record Time Series

2. *Contextual Correlation* - Multi-variate time series representing real world applications are often accompanied by associated information that establishes a relationship across variates, such as sensor network or connectivity information. In this thesis, it is argued that structurally correlated variates are more similar than the variates that are structurally far or less correlated variates.

Figure: 1.6, aptly shows the dimensionality of a multi-variate time series in (a) and (b) shows how the different sensors are placed and connected that establishes the correlation among the variates in a time series. More importantly correlation between the time series helps to establish the dependency on behavior of the sensors based on the analysis of another correlated/connected sensor.

It is important to know that the contemporary *state-of-the-art* measures can effectively determine similarity between a pair of uni-variate time series and multi-variate time series [3] but without taking into account the contextual correlation across the variates. This relationship is crucial to consider while determining the similarity. Measure(s) presented in this thesis leverage the correlation between the variates. Use of contextual correlation helps to effectively overcome the asynchrony in patterns contained in the time series.

1.4 Research Contribution

Contributions presented in this thesis can be categorized into four categories -

1. *Multi-scale-hierarchical time series* representation that leverages the metadata describing the properties associated with the time series. The metadata could be relationship defined between a pair of variates or correlation or distance between sensors. This relationship could be defined in terms of correlation between the variates etc.

2. *WDTW : Weighted dynamic time warping and different weighing strategies* based on variate relationship. This approach leverages the relationship between the variates while determining the similarity across the multi-variate time series. We carry out the eigen-decomposition of the relationship matrix so that the corresponding eigenvalues could be used to determine the weighted similarity across the variates.

3. *PDTW : Projected dynamic time warping* - This approach projected the data space onto to a vector basis extracted from the metadata. This approach enables similarity computation on the orthogonal projection of data.

4. *SIDTW : Structurally inherent dynamic time warping* - We deploy hierarchical clustering on variates from multi-variate time series. This hierarchical clus-

[3] considering each variate independently

tering is based on the relationship between variates such that similar variates are clustered. Variate based clustering improves the performance of similarity computation.

1.5 Organization of Thesis

Thesis thesis is organized as follows:

- In Chapter 2, background and related work have been reviewed
- In Chapter 3, multiple weighing strategies have been proposed for weighted dynamic time warping
- In Chapter 4, de-correlated dynamic time warping is introduced based on Eigen decomposition of hierarchical relationship matrix defined between the variates
- In Chapter 5, structurally inherent dynamic time warping is presented which is based on hierarchical clustering of variates by using the variate distance or connectivity relationship.
- In Chapter 6, details on experimental setup, evaluation criteria, dataset and analysis methods are discussed.

Chapter 7 concludes the thesis and describes the direction for future work.

Chapter 2

RELATED WORK

As discussed in Chapter 1, the main issue with time series analysis is the need for an efficient and effective retrieval mechanism, be it similarity search or performing cluster analysis over the time series. This chapter presents an overview of the existing work in time series analysis and how various algorithms have evolved. In Section 2.1 various linear distance measures are discussed that are used to compute similarity between a pair of time series. Following Section 2.2 presents feature based distance measures for the features extracted from time series data. Section 2.3 discusses the distance functions that allows non-linear mapping between time series. In remainder of this thesis, the symbol, $\mathbb{T}$, will be used to represent both multi-variate time series, $\mathbb{T}$, and correlation-multi-variate time series, $\mathbb{T}_R$.

Section 1.1 discussed multiple data representations that can be used to represent a time series. It is important to define an informative and scalable data representation that facilitates data analysis but at the same time an appropriate definition of distance function is also very important aspect of time series analytics. Various distance measures have been proposed for the said purpose starting from Euclidean distance to extracting SIFT features [16, 51, 70].

2.1 Linear Distance Function

As mentioned in Chapter 1, time series is a sequence of real numbers recorded at an instance in time that can be separated by uniform or non-uniform increments [23]. Leveraging the data representation defined in Equation: 1.1 and 1.2, Euclidean distance was the first ever distance measure used to determine the similarity between two time series. It determines the summation of $L_p - norm$ of values at similar timestamps (Eq : 1.15 : Uni-Variate time series) and for multi-variate time series as $euclidean(\mathbb{X}, \mathbb{Y}) = \sum_{j=1}^{D} \sqrt[p]{\sum_{i=1}^{N}(\mathbb{X}(j)_i - \mathbb{Y}(j)_i)^p}$, where $p = 2$. Euclidean distance was an indisputable distance measure because of its linear computational cost and ease of computation. Mao [52] states "Euclidean distance works well when data space has compact and isolated clusters." Euclidean distance requires the time series, being compared, be of *same temporal length* making it a non-elastic distance measure, one-to-one mapping across the temporal axis, which might not be the case with time series anymore. It is not always possible to get accurate results from Euclidean distance, specially when patterns contained in the time series are not synchronous along the temporal axis (*Figure: 2.1*), that is, *local time shift* [19]. Another major drawback of using Euclidean distance is it's brittleness to data values [40], as it is data-driven not shape-driven. With an unbounded distance space $[0, \infty)$ and its invariance to changes over time had led to development of similarity measure based on Pearson's Correlation Coefficient. This measure accounts for the changes occurring over time, that is, this is a correlation-driven similarity measure:

$$pearsonSimilarity(\vec{X}, \vec{Y}) = 1 - \frac{(\vec{X} - \overline{\vec{X}})(\vec{Y} - \overline{\vec{Y}})'}{\sqrt{(\vec{X} - \overline{\vec{X}})(\vec{X} - \overline{\vec{X}})'}\sqrt{(\vec{Y} - \overline{\vec{Y}})(\vec{Y} - \overline{\vec{Y}})'}} \tag{2.1}$$

At the same time, this measure is not robust against temporal asynchrony in the time series. To overcome the brittleness of Euclidean distance and account for pattern in data, *cosine similarity* was proposed [14, 60]. Although they share the common background definition of *one-to-one* mapping along the temporal axis, cosine similarity gives more importance to shape-based similarity being robust against the amplitude of data values.

Definition 2.1. ***Cosine Similarity***, *C_{sim}, is the measure of similarity that determines the cosine of angle between two vectors. Similarity is the ratio of dot product*

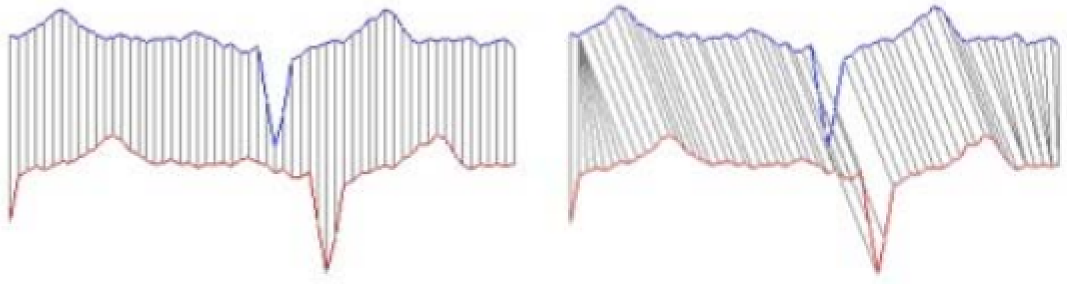


Figure 2.1: Temporal Alignment Between Two Sequences With Local Pattern Shift For Euclidean Distance (left) and Dynamic Time Warping (right) . Source : [49]

of two vector to their magnitude.

$$C_{sim} = \frac{\sum_{t=1}^{N} X(t).Y(t)}{\sqrt{\sum_{t=1}^{N} X(t)^2}\sqrt{\sum_{t=1}^{N} Y(t)^2}} \tag{2.2}$$

$$C_{dis} = 1 - C_{sim} \tag{2.3}$$

Cosine similarity is strictly bound to the range $[-1, 1]$ and the corresponding distance (C_{dis}) is bounded in the range $[0, 2]$. Cosine similarity also requires the time series, being compared, to be of equal length. Various algorithms like *PCA-Similarity Factor* [44] and EROS(*Extended Forbenius norm*) [77] have been proposed that use matrix factorization like singular vector decomposition(SVD) [32] and principle component analysis [73], to transform the time series into equal length and then apply cosine similarity over them.

2.2 Feature-Based Distance Function

Feature-based measures have been in existence in computer vision community since a very long time. Classification algorithms, such as Decision Tree are based on feature space by transforming the sequential data space into a set of features[75]. Laxman in [45] described feature as a local structures that reflects the characteristic point in the sequences. Ji in [39] proposed a feature-extraction algorithm, minimal distinguishing subsequences, that extracts subsequences that can be used as features. Addition feature based measures are proposed that are based on frequency of occurrence of data values. Morchen in [54] proposed use of DFT (Discrete Fourier Transform) and DWT (Discrete Wavelet Transform) for feature extraction, specifically because in DWT both temporal properties and frequencies are preserved whereas in DFT only frequency-based aspects are preserved [47]. Higher frequency feature generated from DWT illustrates global features whereas lower frequency features denotes local features on temporal axis [7]. Wang in [70] and Candan in [16] used extended SIFT-based features for determining robust multi-variate time series features to determine similarity between multi-variate time series.

2.3 Warping Distance Function

As seen in Section 2.1, a distance measure for time series analytics needs to take into account the temporal alignment between time series. Therefore, there is a need for a distance measure that is robust against *local time shift* and also accounts for acceleration and deceleration in patterns contained in time series. One of the most preliminary warping distance was given for determining minimum the number of operations required to make two strings similar. Edit distance [43, 46, 57], a metric measure, determines minimum number of sequence of operations that are required to achieve string similarity. Edit distance considers unit cost of each operation that needs to be performed, as the goal here is to determine minimum number of operations required. This measure has been extended to determine the similarity between two time series as it accounts for *local time shift* in the time series. Using unity cost for every operation is valid for string similarity as the comparison is between symbols contained in the string but not an appropriate cost definition for time series [20]. Since

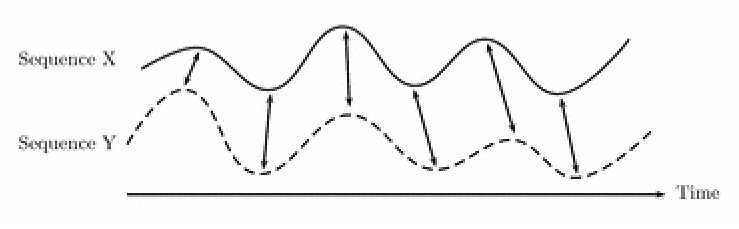


Figure 2.2: Temporal Alignment Between *one*-Dimensional time series. The Arrows Indicates Temporal Mapping [55]

time series is a real sequence that contains real numbers as symbols such cost definition penalizes the similarity between two time series and can result in misclassification.

Berndt [12] proposed dynamic time warping in 1996 as an algorithm that accounts for the limitation of Euclidean distance. Muller states in [55]: "Dynamic time warping (DTW) is a well-known technique to find an optimal alignment between two given (time-dependent) sequences under certain restrictions (Fig. 2.2). Intuitively, the sequences are warped in a nonlinear fashion to match each other." The wide acceptance of DTW is the result of its flexibility in handling different sized time series and ability to account for acceleration and deceleration in patterns contained in the series.

Let us consider two time series, X and Y, of length M and N respectively, where M and N may or may not be equal.

$$DTW(X,Y) = c_p(X,Y) = \begin{cases} Infinity & if M = 0 \ or \ N = 0 \\ min(c_p(X,Y),\ p \ \in P^{M\times N}) & \forall M \neq 0 \ and \ N \neq 0 \end{cases} \tag{2.4}$$

With the use of dynamic programming, computational cost of minimum DTW path can be done in quadratic time, $O(MN)(M \ and \ N)$. At the same time, DTW does not follow *triangular inequality.* Chen in [20] proposed, ERP, an extended version of DTW that satisfies triangular inequality by adding a gap when there is a difference in values during sequence comparison.

DTW was initially given for *one*-dimensional time series [12]. Multiple extensions of DTW have been proposed for *multi*-dimensional time series [62, 68]. The most prevalent ones are vectorized and independent. In vectorized DTW, a multi-variate time series is considered as a *one*-dimensional vector time series, where the length of vector is equal to the dimensionality of time series. On the other hand, independent DTW is applied on each dimension independently and then DTW cost from all the dimensions are added independently. They have been known to perform better than various *multi*-dimensional distance measures like euclidean distance.

Algorithm 1 Vectorized Dynamic Time Warping

```
double dtwV($\mathbb{X}[1..D, 1..M]$,$\mathbb{Y}[1..D, 1..N]$){
  costMatrix $\leftarrow$ array[1..M,1..N]
  for i $\leftarrow$ 1 to M
      costMatrix[i,0] $\leftarrow \infty$
  for i $\leftarrow$ 1 to N
      costMatrix[0,i] $\leftarrow \infty$
  for i $\leftarrow$ 1 to M
    for j $\leftarrow$ 1 to N
      cost $\leftarrow$ | norm2($\mathbb{X}[1..D, i]$ - $\mathbb{Y}[1..D, i]$) |
      DTW[i,j] $\leftarrow$ cost + minimum (DTW[i-1, j], DTW[i, j-1], DTW[i-1, j-1])
  return DTW[M,N]
}
```

DTW-vectorized and DTW-Independent are non-weighted distance measures that are indifferent to dimensions while similarity is computed. This non-weighted distance measures may lead to misclassification [18, 38]. EDR[21] was proposed a gap based weighing. This weighing was a constant penalized the DTW score whenever two time series had different values at an instance in time. In many cases, not all dimensions

Algorithm 2 Independent Dynamic Time Warping

```
totalCost ← 0
for d ← 1 to D{
  totalCost ← totalCost + dtwU($\mathbb{X}[d, 1 : M]$, $\mathbb{X}[d, 1 : N]$)
}
double dtwU($\mathbb{X}[1..M]$,$\mathbb{Y}[1..N]$){
  costMatrix ← array[1..M,1..N]
  for i ← 1 to M
        costMatrix[i,0] ← ∞
  for i ← 1 to N
        costMatrix[0,i] ← ∞
  for i ← 1 to M
     for j ← 1 to N
        cost ← | $\mathbb{X}[i]$ - $\mathbb{Y}[i]$) |
        DTW[i,j] ← cost + minimum (DTW[i-1, j], DTW[i, j-1], DTW[i-1, j-1])
  return DTW[M,N]
}
```

have equal contribution while determining similarity in a multi-dimensional space. We propose **WDTW**: Weighted Dynamic Time Warping and multiple weighing strategies that defines the contribution of each dimension while similarity between time series is determined. We propose both data-driven and metadata-driven weighted measures. We also propose a **PDTW** : Project Dynamic Time Warping to project a time series to a vector basis based on the relationship defined between the variates. Motivated from the variate relationship, we investigate the variate behavior in presence of correlation by generating a multi-scale hierarchical data structure to determine DTW over the hierarchy, SIDTW.

Chapter 3

WDTW : WEIGHTED DYNAMIC TIME WARPING AND WEIGHING STRATEGIES

Use of DTW to align two sequences on temporal axis with local acceleration and deceleration results in higher retrieval accuracy [41]. This has been possible because of its capability to perform non-linear temporal alignment between two time series called optimal warping path. This warping path, is a non-weighted measure that gives equal importance to all the dimensions irrespective of any phase difference or temporal asynchrony between the two time series, which may in turn led to misclassification [18, 38]. False positive outcomes can prove to be costly for application where shape based retrieval is of crucial importance.

All existing distance measure discussed in this thesis, until now, share one particular property in common, are non-weighted towards different dimensions of the time series. This holds true for both linear (Section 2.1) and warping (Section 2.3) distance measure, that is,

$$dis(\mathbb{X}, \mathbb{Y}) = \sum_{i=1}^{D} distanceFunction(\mathbb{X}^i, \mathbb{Y}^i).\mathcal{W}^i \tag{3.1}$$

where, $\mathcal{W}^i = 1$, ith variate, $\mathcal{W}$ are the weights assigned to the variates. The assumption made here is that both time series must have same number of dimensions, D, but can be of different temporal lengths, N. At the same time, presence of temporal asynchrony between different variates of a time series adversely affects their discriminatory behavior while determining similarity between the two time series. Weighing aims to increase the contribution of more discriminating variates to achieve better

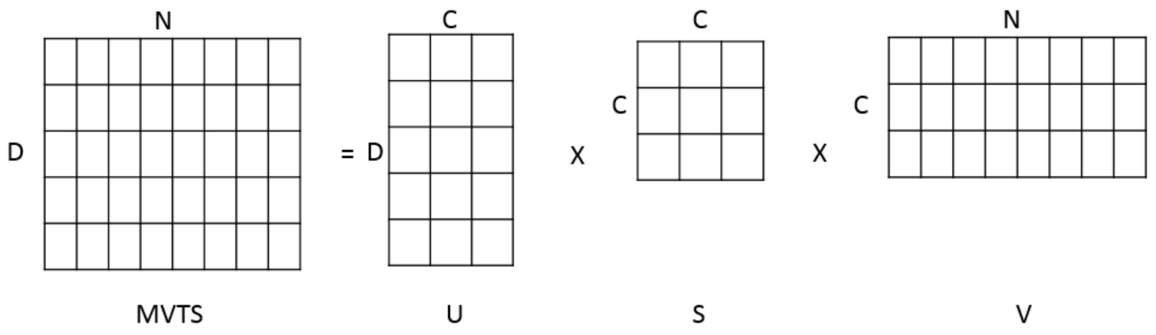


Figure 3.1: Singular Values Decomposition Of Multi-Variate Time Series

separation between the two time series. What is proposed in this thesis is $\mathcal{W}^i \in \mathbb{R}$ and $1 \leq i \leq D$.

This chapter discusses different weighing mechanisms that can be used to quantify importance of each variate to facilitate similarity calculation between two time series. As each multi-variate time series is independently generated, it is highly likely to encounter different sets of discriminating variates for different time series. It becomes a crucial task to extract one common set of weights that accounts for the importance of variates in both time series. In the following section, six different weighing strategies are proposed and discussed. Detailed experimental analysis has been discussed in Chapter 6.

3.1 Data Driven Weights In Data Space

With decomposition, important patterns contained in the time series are extracted. The importance of each extracted pattern in the time series and for each dimension is defined in terms of eigenvalues and eigenvectors. Let us consider *two* multi-variate time series, $\mathbb{X}$ and $\mathbb{Y}$ of size $D \times N_X$ and $D \times N_Y$ respectively, and we will apply singular value decomposition (SVD) on them. *Figure: 3.1* shows, how a time series is decomposed using SVD [32]. In general, SVD is used to transform a

D-dimensional space to C-dimensional vector basis.

$$[U_f, S_f, V_f] = svd(\mathbb{X}) \tag{3.2}$$

where f is the time series file index.

It is nearly impossible to select C common subset of features in reduced space as both time series are independently generated. Therefore, all the dimensions are considered. Intuition behind the use of eigenvalues is to determine the total contribution of each variate based on the patterns contained in them as eigenvalues represents the importance of each pattern.

$$\mathcal{W}_f = U_f \times diag(S_f) \tag{3.3}$$

where $diag()$ transform the diagonal of input matrix into a column matrix. From the Eq: 3.3 we will get two set of weights, $\mathcal{W}_1$ and $\mathcal{W}_2$, one set for each time series. *Figure: 3.2* shows how weights are extracted. As each variate can have different importance across different time series it is critical to formalize a strategy to find common set of weights and this could be achieved in two ways:

1. **Aggregated Weights** - In this particular type of weighing mechanism, element wise average is taken for the total contribution of each variate.

$$\mathcal{WA} = \frac{\mathcal{W}_1 + \mathcal{W}_2}{2} \tag{3.4}$$

2. **Productive Weights** - As seen in aggregated weights, difference in total contribution of two corresponding dimension penalizes the dimension with the higher contribution due to average. This step of penalization is to make two dimensions with different contributions comparable. Productive weight on the other hand attempts to amplify the contribution by taking element-wise product of weights.

$$\mathcal{WP} = \mathcal{W}_1 \times \mathcal{W}_2 \tag{3.5}$$

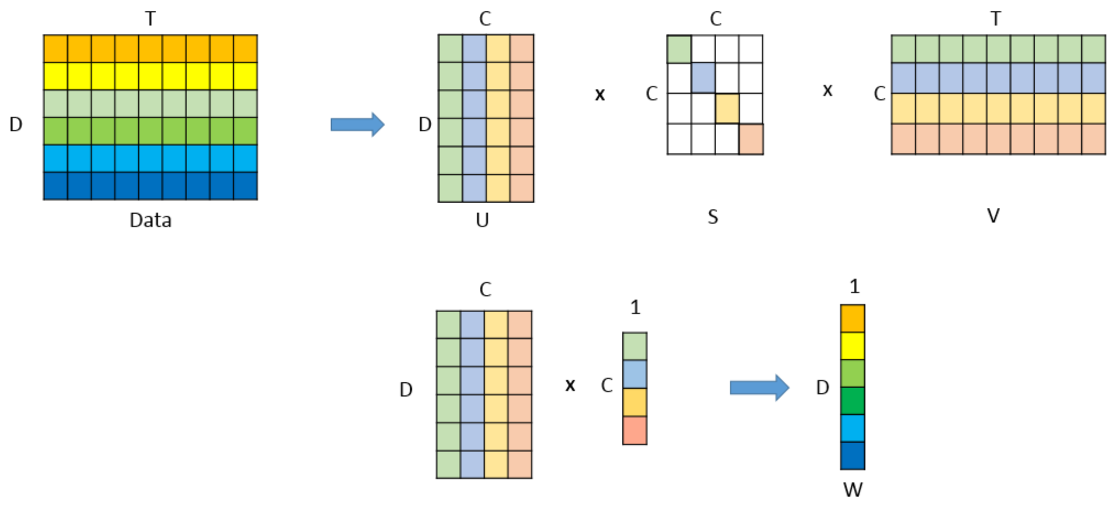


Figure 3.2: Demonstrates Data Decomposition into Eigenvectors and Eigenvalues; Followed By Weight Extraction For Each Variate

Therefore, based on the choice of weights, the similarity between two multi-variate time series can be determined as follows:

1. Aggregated Dynamic Time Warping

$$DTW\_U\_D\_wEDAvg(\mathbb{X}, \mathbb{Y}) = \sum_{i=1}^{D} dtwU(\mathbb{X}^i, \mathbb{Y}^i).\mathcal{WA}^i \tag{3.6}$$

2. Productive Dynamic Time Warping

$$DTW\_U\_D\_wEDProd(\mathbb{X}, \mathbb{Y}) = \sum_{i=1}^{D} dtwU(\mathbb{X}^i, \mathbb{Y}^i).\mathcal{WP}^i \tag{3.7}$$

Use of SVD on data space facilitates the extraction of high variant dimensions. The crux of using this methodology is to leverage high variance in the variates of a time series. Thus, use of eigenvalues makes the dominant dimensions more visible while determining the similarity.

3.2 Data Driven Weights In Feature Space

In Section 3.1, critical patterns contained in time series were extracted with their importance. Based on there importance and contribution in each variate, total weight for each variate was determined. These weights can be used on the features representing the variates in the time series. Therefore, we define the feature component based similarity as follows:

Weighted-Euclidean on Feature Components:

$$weightedEDAvg(\mathbb{X}, \mathbb{Y}) = \sum_{i=1}^{C} ED(U_X^i, U_Y^i).\mathcal{WA}^i \tag{3.8}$$

where, $\mathcal{WA}$ is derived from Eq : 3.4 with S_1 as input. With the use of euclidean distance on feature components we determine the difference in the importance of the identified patterns.

3.3 Contextually-Driven Weights

Section 3.1 and 3.2 discussed different ways to identify critical patterns in the time series and using there contribution to each variate to extract weights thus prevent misclassification. It is important to note that, data-driven weights are solely extracted from the data itself. Thus they are not robust against temporal asynchrony. As asynchrony like shift and stretching alters the pattern by modifying their temporal length and presence in the time series. As we discussed in Section 1.3, the contemporary time series has an associated structural information with them called contextual correlation. This information can provide us with an insight on how variates in a time series are related to each other irrespective of any temporal asynchrony between them.

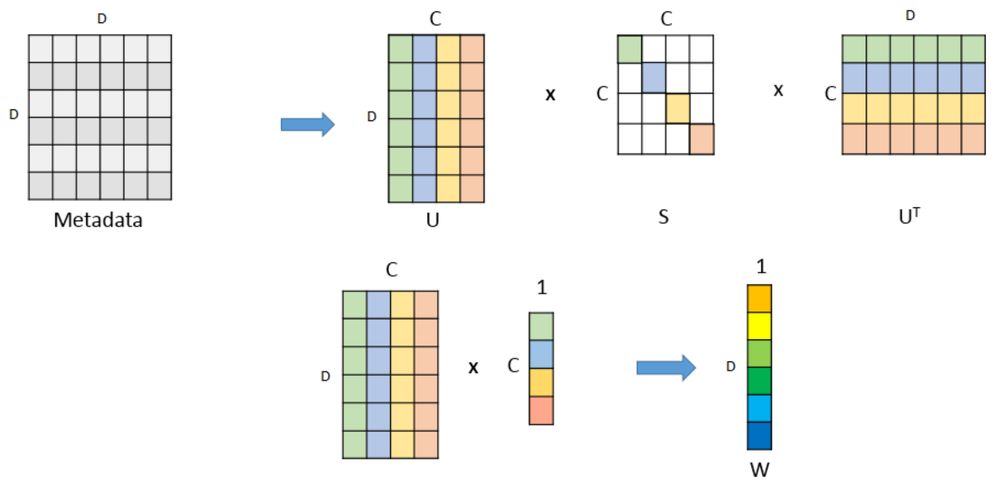


Figure 3.3: Demonstrates Metadata Decomposition into Eigenvectors and Eigenvalues; Followed By Weight Extraction For Each Variate

3.3.1 Contextual Weights

Contextual correlation defines the relationships between the variates contained in the time series. With the use of relationship, variates could be clustered in terms of their closeness with each other. Let us consider a relationship matrix or contextual correlation, R, that defines the relationship between different variates of the time series. This relationship can be extracted from the graph connectivity associated with the time series if the variates denotes sensor or relationship could be sensor distance or correlation coefficient between the variates. The goal here is determine important clusters of variates in time series. The use of a graph for weight extraction highlights the node influence and how the cluster of nodes behaves. To achieve this, we carry out eigendecomposition on the relationship matrix, R.

$$[U_R, S_R, V_R] = eigen(R) \tag{3.9}$$

The total contribution of a variate can be extracted as follows:

$$\mathcal{WR} = U_R \times diag(S_R) \tag{3.10}$$

where, $diag()$ transforms the diagonal of the matrix S_R to the column matrix. Thus the new cost function based on weights extracted from relationship matrix is defined as follows:

$$DTW\_U\_D\_wER(\mathbb{X}, \mathbb{Y}) = \sum_{i=1}^{D} dtwU(\mathbb{X}^i, \mathbb{Y}^i).\mathcal{WR}^i \tag{3.11}$$

Here, the discriminatory behavior of a particular variate in the time series is determined from the spatial properties associated with it. It is argued that more influential the node is in the graph, the more dominance it has, thus the associated weight would be higher.

Use of contextual correlation negates the weighted penalization, as a result of matching the most contributing to least contributing dimensions, that was encountered in the weighing mechanisms proposed in section 3.1 and 3.2. It is critical to understand the kind of relationship being used. A relationship can provide distance or nearness between variates.

$$R_{distance}(i, j) = dis(i, j)\ \forall\ i, j \in T \tag{3.12}$$

$$R_{nearness} = max(R_{distance}) - R_{distance} \tag{3.13}$$

3.3.2 Hierarchical Weights

Use of contextual correlation can give more in-depth information on time series. As discussed in Chapter 1, correlation matrix could be used to determine hierarchical similarity between two multi-variate time series. Hierarchical similarity is used to identify critical cluster of variates to identify contribution of each variate to the clusters. This can be achieved in the following proposed weighing mechanism:

$$H_D = hierarchy(R) \tag{3.14}$$

where, H_D is the hierarchy created on the variates of time series based on the associated R with it (*Def: 1.5*). This results in a tree structure, with the variates located at its leaves. Once the hierarchy is created, hop distance from each leaf node to all the other nodes is determined (*Def: 1.6*) and is called Hierarchy Similarity Matrix, H.

Eq :3.9 and Eq: 3.10 are reused with H as input instead of R.

$$[U_H, S_H, V_H] = eigen(H) \tag{3.15}$$

The total contribution of a variate can be extracted as follows :

$$\mathcal{WH} = U_H \times diag(S_H) \tag{3.16}$$

Therefore the new cost function is,

$$DTW\_U\_D\_wEH(\mathbb{X}, \mathbb{Y}) = \sum_{i=1}^{D} dtwU(\mathbb{X}^i, \mathbb{Y}^i).\mathcal{WH}^i \tag{3.17}$$

With the use of contextual correlation between the variates, variates could be clustered in terms of their closeness to each other. We investigate the resemblance in pattern observed between the variates with close relationship.

3.4 Conclusion

Weighing gives higher importance to highly correlated variates. By assigning high importance in terms of their contribution to important patterns or the cluster of variates in time series help increase the retrieval accuracy.

Experiments are discussed in Chapter 6.

Chapter 4

PDTW : PROJECTED DYNAMIC TIME WARPING

Linear dependence and linear independence are the key concepts associated with the vector space, namely dimensionality. Consider a set of vectors, $\mathbb{V} = \{v_1, v_2 ... v_D\}$, . $\mathbb{V}$ is a set of linearly independent vectors iff any vector v_i cannot be represented in a linear combination of other vectors in $\mathbb{V}$.

$$\alpha_1 v_1 + \alpha_2 v_2 + ... + \alpha_D v_D = 0 \tag{4.1}$$

is only true when $\alpha_i = 0$, where $1 \leq i \leq D$. If the Eq: 4.1 is true for any value of $\alpha \neq 0$, then $\mathbb{V}$ is a set of linearly dependent vectors. In simple terms, linear independence suggests that vectors are orthonormal to each other and they cannot be represented in linear combination of each other. It is well known that eigendecomposition of a square matrix results in a linearly independent eigenvector space and a set of eigenvalues [71] denoting importance of each vector.

4.1 Motivation

Correlation is the main reason for observing pattern similarities across multiple variates. In case of multi-variate time series the correlation is defined in the metadata, relationship matrix. This relationship matrix defines how two variates in a time series are correlated. The goal is to observe the independent behavior of an attribute contained in the variates by projecting them on a vector basis.

4.2 Algorithm

With the increase in dimensionality of time series, associated information have found their way into the data structures used to represent time series. One such information is the relationship defined between the variates. This relationship defines the closeness or correlation between the variates. In this approach we propose projecting a time series to a vector basis extracted from contextual correlation associated with the time series.

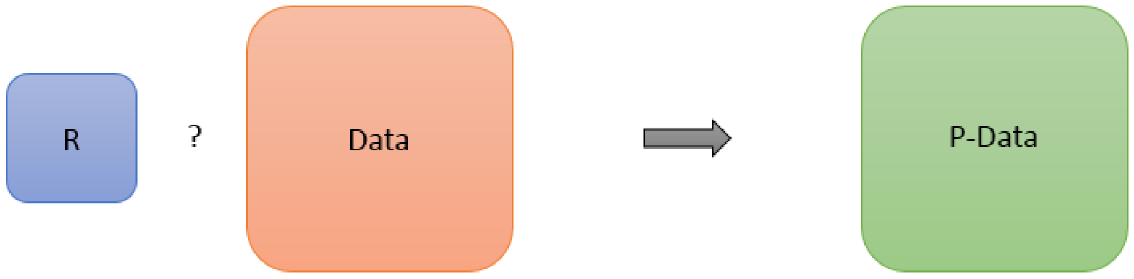


Figure 4.1: Projecting Time Series to Vector Basis

where, ? is defined as the projection operator that extracts the vector basis from contextual correlation, R, and projects the data on to it. Projection on to a basis presents the behavior of a time series when there is minimal effect of any external factor on their behavior.

For the said purpose we use hierarchy similarity matrix, H, to define relationship between the variates, which in turn could be interpreted as correlation between two variate is proportional to the similarity between the variates. With this transformation highly similar variates are projected on to the same vector in the basis. This projection is defined as a two step process:

1. **Vector Basis Extraction** : In this step we extract the vector basis from the

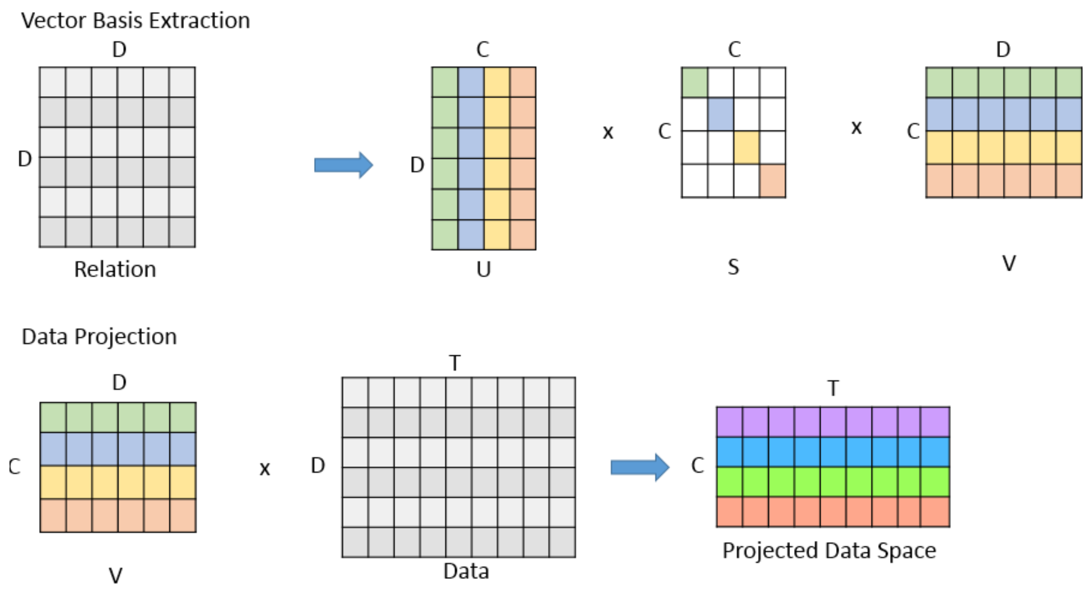


Figure 4.2: Projecting Time Series to Vector Basis

hierarchy similarity matrix (Def: 1.6), H of size $D \times D$.

$$[U_H,\ S_H,\ V_H] = eigen(H) \tag{4.2}$$

this gives us two sets eigenvectors and one set of eigenvalues. U_H, S_H, and V_H are of size $D \times C$, $C \times C$ and $C \times D$ respectively where C is the number of components or critical clusters of variates.

2. **Data Projection** : In this step we project the time series on to the extracted vector basis.

$$iX = V_H \times X \tag{4.3}$$

where iX is the projection of X on V_H and is of size $C \times T$.

Now the projected time series can be used for any operation related to search and analysis. As we saw in Chapter 3, weighing helps in effective similarity calculation, therefore we extend this approach further as Non-weighted and Weighted PDTW.

4.2.1 Non-Weighted PDTW

Since the variates have been projected to a vector basis, investigating the search and analysis with non-weighted measure is one option to explore.

1. DTW_V_HD - Application of vectorized dynamic time warping on $i\mathbb{T}$ highlights the temporal similarity across all the dimensions in the time series.

$$DTW\_V\_HD(i\mathbb{X}, i\mathbb{Y}) = dtwV(i\mathbb{X}, i\mathbb{Y}) \tag{4.4}$$

2. Uni-Variate PDTW (DTW_U_HD) - As the variates are projected to a vector basis, uni-variate dynamic time warping can identify similar individual projected variates.

$$DTW\_U\_HD(i\mathbb{X}, i\mathbb{Y}) = \sum_{j=1}^{D} dtwU(i\mathbb{X}^j, i\mathbb{Y}^j) \tag{4.5}$$

4.2.2 Weighted PDTW

As stated earlier, not all component have similar contribution to the data representation. Therefore, weighted PDTW gives different importance to different dimensions based on the weights determined from the decomposition of the hierarchy similarity matrix.

$$DTW\_U\_HD\_wEH(i\mathbb{X}, i\mathbb{Y}) = \sum_{j=1}^{D} DTW_{uni}(i\mathbb{X}^j, i\mathbb{Y}^j).\mathcal{S_H}^j \tag{4.6}$$

4.3 Conclusion

Projecting data to a vector basis uncovers the relationships between the variates by projecting them on to same vector in the basis. Therefore closely related variates are now represent by a more unifiy representation on the project space.

Experiments are discussed in chapter 6.

Chapter 5

SI-DTW : STRUCTURALLY INHERENT DYNAMIC TIME WARPING

5.1 Overview

It is a well known and proven fact that hidden patterns can be extracted from the time series by generating their multi-scale representation by using the multi-scale representation technique given by Witkin [72], of embedding the data into a family of representation called *scale-space* pattern extraction. This type of data representation was first given for image to facilitate intelligent analysis. Multi-scale representation highlights the pattern that cannot be observed on a particular scale.

A multi-scale representation, *pyramid representation*, is obtained by smoothing and sub-sampling the data iteratively on both variate and temporal axis simultaneously. Iterative smoothing uncovers hidden patterns that can be used to discriminate between the time series unlike images where these patterns are regarded as features and are used as image-representative.

In contrast to the computer vision community where adjacent image pixels are smoothed and sub-sampled together, whereas in time series analysis, adjacency of the variates is defined by the metadata associated with it. This metadata, such as relationship matrix, generates a hierarchy of variates (*Defn:* 1.5).

Correlation based variate clustering discover more coherent patterns across different scales. For experiments and dataset represented in thesis, correlation is defined in terms of how close two variates are? This close proximity insinuates that the patterns contained in the adjacent variates would be more similar, if not identical, than the patterns contained in other variates.

5.2 Motivation

The correlation defined between the sensors impose a structure between them. This structure defines shared properties and behaviors between the time series generated by these sensors (variates).

The multi-scale hierarchical representation creates a hierarchical clustering of these sensors based on how closely they are related. A strong relationship establishes more coherent patterns between them. This coherence suggests that the patterns contained in the two variates would be similar if not identical.

As seen in Section 1.3, high dimensionality makes the similarity computation a computationally costly task. Pyramid structure based representation reduces the dimensionality by a factor of two when moving from finer to coarser scale.

5.3 Algorithm

SIDTW is a *bottom-top-bottom* approach, where it moves bottom-up to generate the multi-scale hierarchical data structure and top-down to determine the similarity by inheriting DTW path from coarser scales. It can also be referred to as a two-pass algorithm.

Pass 1 : Multi-Scale Hierarchical Data Structure

In this step, hierarchical clustering of variates is performed. The decision to cluster two variates is based on the relationship defined between them. This relationship could be defined in terms of distance, variates with minimum distance between them are clustered together and if relationship is defined in terms of correlation between the variates then the variate pair with highest correlation is clustered.

Once the deciding factor is determined, agglomerative hierarchical clustering is performed which results in a hierarchical tree structure with variates positioned on

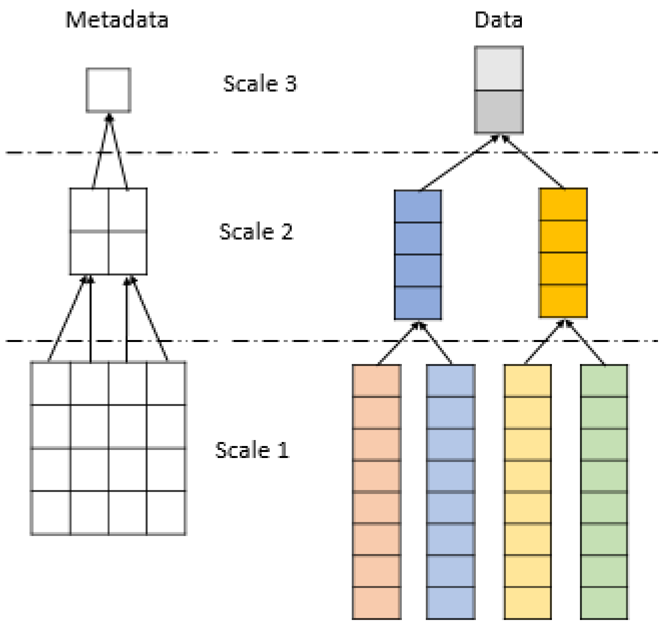


Figure 5.1: Multi-Scale Hierarchical Data Structure

the leaf of the tree.

Step 1: Determine the pair of variates to cluster(e.g. distance)

$$[v_1, v_2] = minimum(Relation) \tag{5.1}$$

Step 2: Determine the next scale variate

$$f = \frac{\mathbb{X}^{v1} + \mathbb{X}^{v2}}{2} \tag{5.2}$$

Step 3: Smoothening the new variate

$$L(.,t) = g(.,t) * f \tag{5.3}$$

where, g is the Gaussian kernel, t is the smoothing factor and f is the signal to be smoothed.

Step 4: Sub-sampling the Signal

$$subSample(i) = \frac{L(2*i-1) + L(2*i)}{2} \tag{5.4}$$

where i ∈ [1,N/2] where N is length of time series. These four steps are iteratively done until root of the tree is reached.

Pass 2 : Inheriting Warping Path

Hierarchical clustering of the variates represents similarity in patterns between them. Therefore, determining independent warping path between the variates does not account for similarity between patterns and relationship between the variates. With a hierarchical data-structure we preserve the relationships between the variates and by proposed inheritance of warping path, pattern similarities between the variates are preserved. In this step, DTW warping path is only computed between the variates at the root or the most coarser scale (*Figure : 5.2*). After the path is determined at the most coarser scale, we translate this path into allowable search space on next finer scale. This bounded space is represented by green colored boxes and region from where path cannot pass through is represented by white boxes in Figure : 5.2. To determine the allowable search space, let us consider the warping path on scale 3. The top left corner of the warping path maps index 2 of one time series to index 1 of another time series, $< 2, 1 >$. Therefore the corresponding allowable search space will be translated from the coarser point, $< 2, 1 >$, to the finer points (*finerP*), $< 3, 1 >, < 3, 2 >, < 4, 1 >, < 4, 2 >$.

$$finerP(x, y) = \begin{cases} (x+1, y), (x+1, y+1), (x+2, y), (x+2, y+1) & if x > y \\ (x, y+1), (x+1, y+1), (x, y+2), (x+1, y+2) & if x < y \\ (x+1, y+1), (x+1, y+2), (x+2, y+1), (x+2, y+2) & if x = y \end{cases} \tag{5.5}$$

where x and y are the x-coordinate and y-coordinate of the warping point on a coarser scale respectively. Eq:5.5 presents a generalized equation for translating a warping point on a coarser scale to candidate warping points on a finer scale. This translation

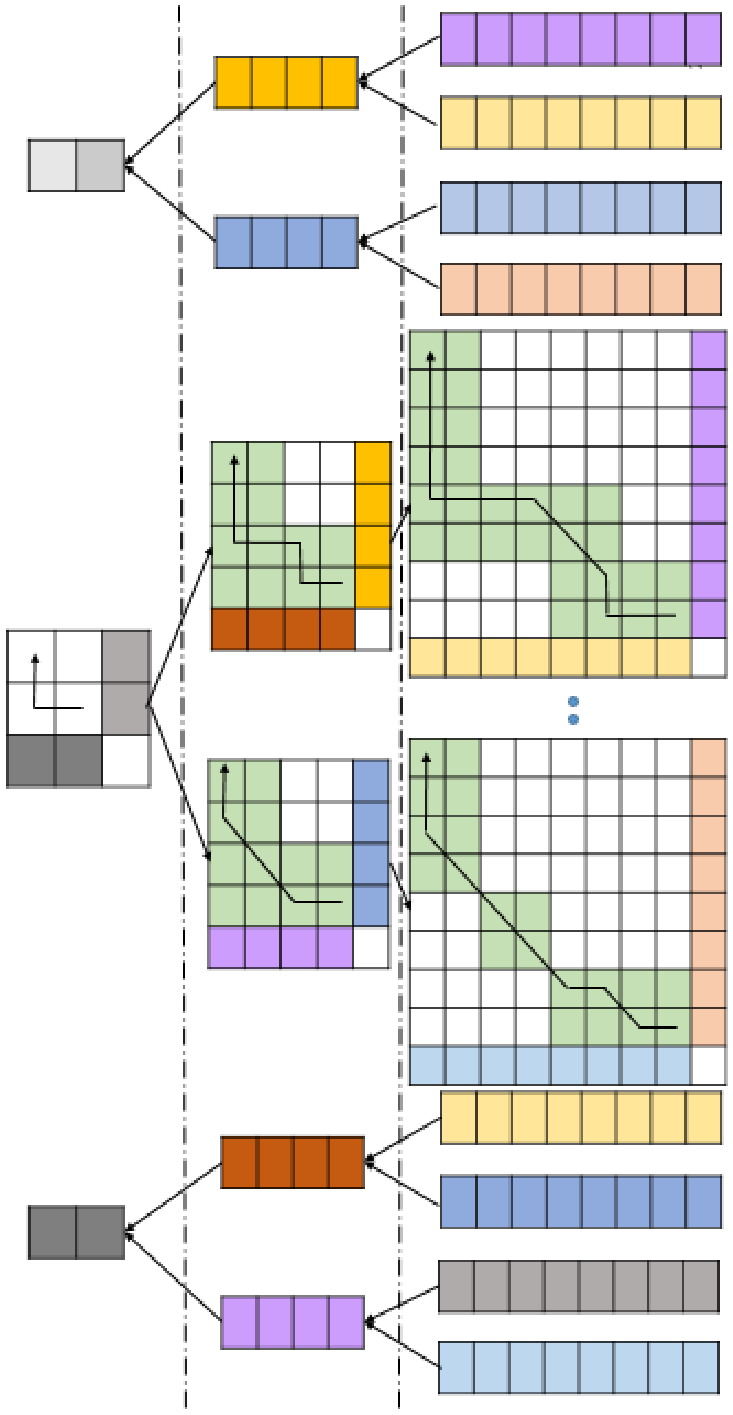


Figure 5.2: Path Inheritance from Coarser Scale to Finer Scale in Multi-Scale Hierarchical Data Structure for Time Series

bounds the search to a list of candidate warping points, that is four times the coarser path's length. This bounding search space also satisfies all the DTW path conditions of monotonicity, continuity and boundary.

5.4 Conclusion

Bounding the search space for path inheritance reflects the correlation between the variates. Path inheritance minimizes the shift in path across different variates when warping path is independently determined but adds sensitivity to sudden change in pattern. Bounding search space to a 2×2 region is a very strict bound. Therefore, this bounding emphasizes on variate clustering, that is, more asynchronous the variates are with each other the better performance is observed. Experiments shows how variates clustering augments the performance by minimizing the error incurred because of bounded mapping.

Experiments are discussed in chapter 6.

Chapter 6

EXPERIMENTAL SETUP AND EVALUATION

In this chapter, evaluation results for *fourteen* similarity measures using *five* gesture recognition datasets (Kaggle Gesture Recognition (W, X, Y, Z)[5] and CMU Motion Capture[2] are presented and evaluated. This thesis focuses on evaluating the retrieval accuracies of various similarity measures and their robustness against various type of asynchrony and to different degree of asynchrony. Robustness is evaluated against different types of temporal asynchrony in the data such as shifts and distortion.

6.1 Datasets

This section discusses the datasets used in the experimental evaluation.

6.1.1 CMU Motion Capture Dataset

Human motions, ranging from walking to dribbling, were recorded in Mocap lab at CMU[2] where subject were asked to perform different activities in a rectangular area of size $3m \times 8m$. 12 infrared MX-40 camera Vicon motion capture system was used to record the gestures. Vicon system requires skeleton information to operate. This input was defined using 41 markers placed on human body at different position. These markers recorded values for multiple attributes, a total of 62 values were recorded at any given instance in time. Therefore the dimensionality of data is 62. These 62 values are the angular information of human joints movements. The relative position (euclidean distance) of these joints is used as contextual correlation. Mocap dataset contains 184 data files with varying temporal length from 150 to 1000 instances. It contains 8 gestures in the dataset.

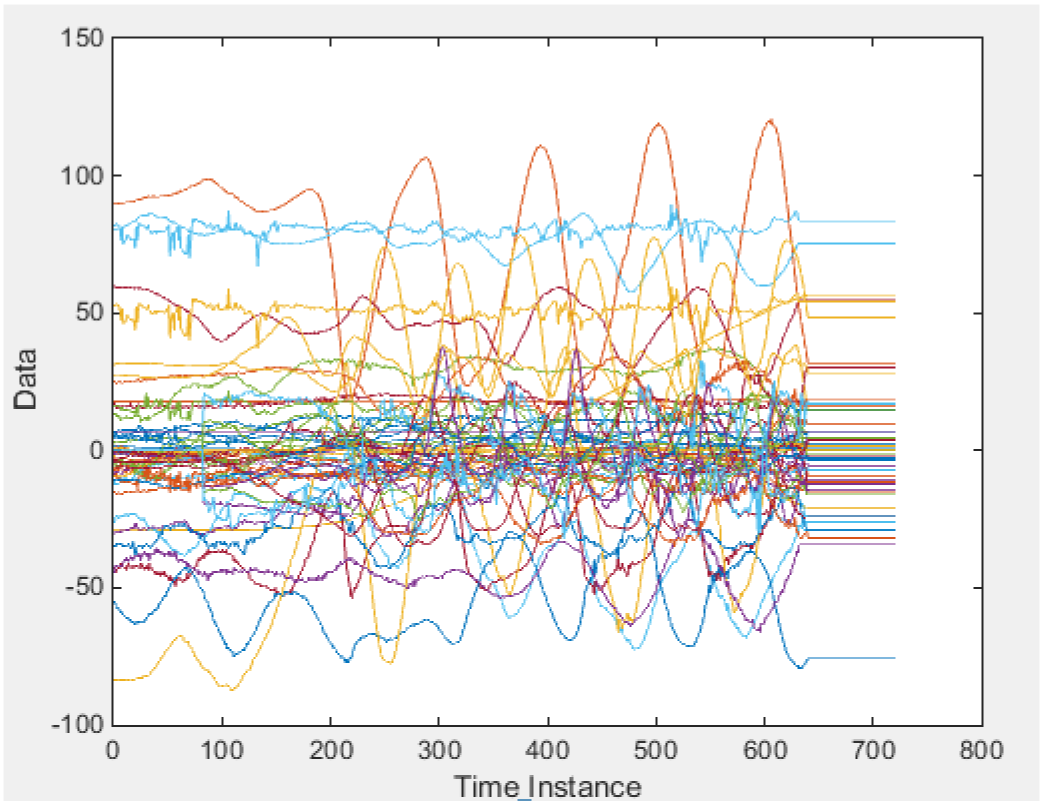


Figure 6.1: Sample Multi-Variate Time Series from Mocap Dataset Each Line Plot Denotes a Variate in Time Series. Data Source: [2]

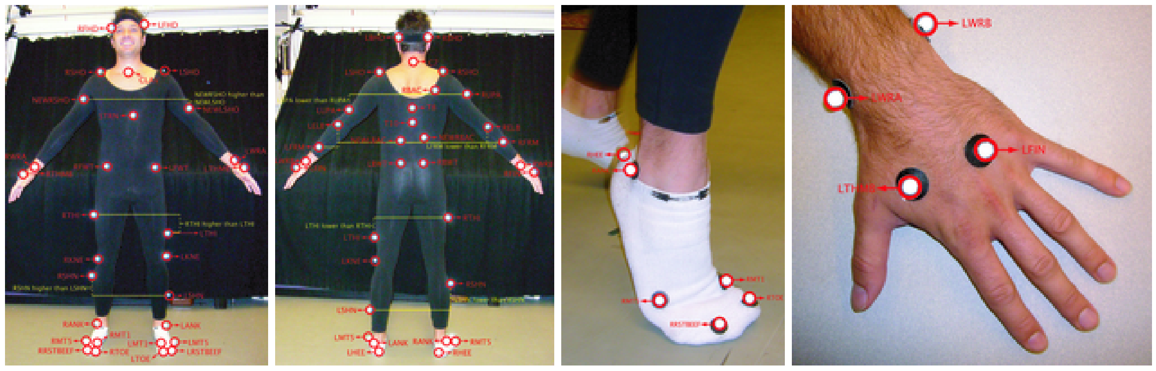


Figure 6.2: Arrangement of Markers on Human Body (From Left-to-Right: Anterior Body, Posterior Body, Inferior Body and Forelimb Palm) Source : [2]

Gesture	**# of series**
Climb	18
Dribbling	14
Jumping	30
Running	19
Salsa	30
Soccer	6
Walk	36
Walk Uneven Terrain	31
Total	**184**

Table 6.1: List of Gestures Contained in Mocap Dataset. Data Source: [2]

6.1.2 Kaggle Gesture Dataset

This data was made available for Kaggle 2013 Multi modal gesture recognition competition in collaboration with Kinect and ICMI 2013[5]. Data was generated using 20 markers place on different human joints. It contains 20 Italian gestures with 31 time series for each gestures. These 20 markers recorded the angular rotation

Gestures: # of series = 31 each			
'vattene'	'seipazza'	'prendere'	'cheduepalle'
'vieniqui'	'combinato'	'noncenepiu'	'tantotempo'
'perfetto'	'freganiente'	'fame'	'buonissimo'
'furbo'	'ok'	'basta'	'messidaccordo'
'chevuoi'	'cosatifarei'	'daccorda'	'sonostufo'

Table 6.2: List of Gestures Contained in Kaggle Dataset

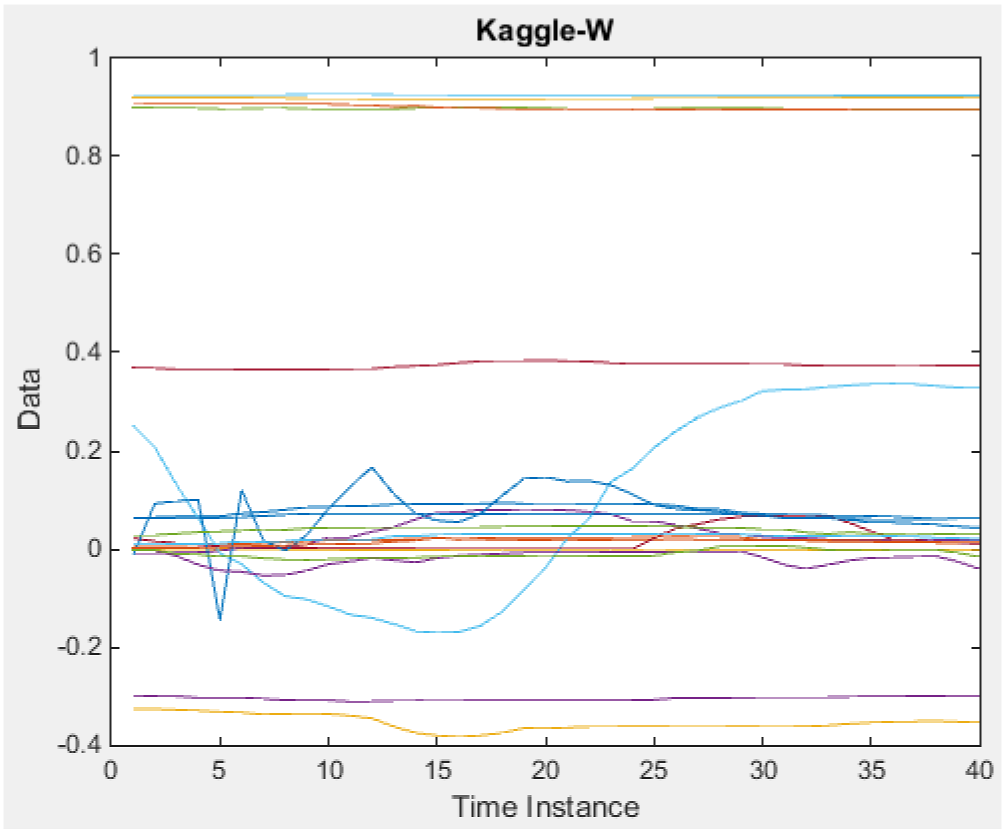


Figure 6.3: Sample Multi-Variate Time Series from Kaggle-W Dataset. Each Line Plot Denotes a Variate in Time Series. Data Source: [5]

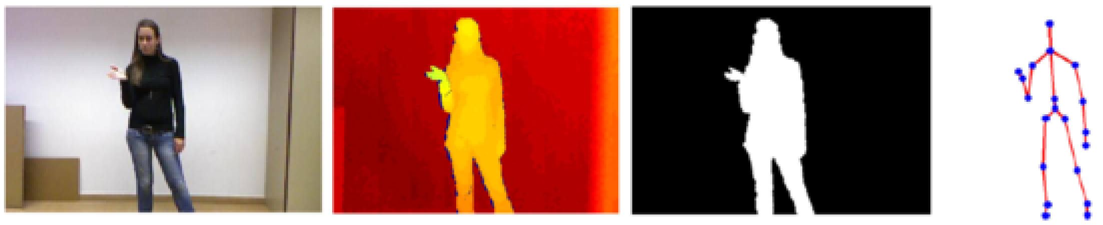


Figure 6.4: Kaggle Gesture (From Left-to-right: Frame of Gesture Video, RGB Rendering of gesture, Grayscale or Depth of Gesture, Marker Position) Source : [3]

of skeletal bones at any given time instance. These orientations were extracted in a three-dimensional camera space, thus rendering 4 different multi-variate time series for W, X, Y and Z component of rotation of skeletal bones.

6.2 List of Algorithms

Algorithm Name	Description	Type
EROS	Cosine of eigenvector of covariance matrix of data	C
PCA	Cosine of all pair combination of eigenvector of data	C
DTW_V	Vectorized dynamic time warping	D
DTW_U	Univariate dynamic time warping	D
DTW_D_wEDAvg	SVD on data, avg eigenvalues DTW	D
DTW_D_wEDProd	SVD on data, multiple eigenvalues DTW	D
Euclidean_ED_wED	SVD on data, Weighted Euclidean on feature components and weights from eigenvalues	D
DTW_D_wEN	weights from decomposition of nearness matrix	M
DTW_D_wER	weights from decomposition of relationship matrix	M
DTW_D_wEH	weights from decomposition of hierarchy matrix	M
DTW_V_PD	VDTW on Project Data on Vector basis from hierarchy	M
DTW_U_PD	UDTW on projected data on vector basis from hierarchy	M
DTW_U_PD_wEH	Weights from decomposition from hierarchy and DTW on projected data	M
SIDTW	Create hierarchy based on relation, force dtw on each variate	M

Table 6.3: List of Algorithms Being Evaluated in the Following Section

6.3 Alternative Algorithms

In this experimental evaluation proposed measures are compared against linear and non-linear distance measures: EROS (Extended Frobenius norm) and PCA Similarity Factor and naive DTW (Dynamic Time Warping).

6.3.1 Dynamic Time Warping

Dynamic time warping (DTW) [59] has been widely accepted as a full-length time series comparison measure. It determines a minimum cost warping path between two uni-variate time series and can be easily used on multi-variate time series by individually using DTW on each variate. An important assumption here is the order of variates. Identical order of the variates is assumed in both time series. Senin [63] formalizes DTW as

$$DTW(X,Y) = c_{p*}(X,Y) = min\{c_p(X,Y), p \in P^{N\times M}\} \tag{6.1}$$

where, $P^{N\times M}$ is the set of all possible warping path defined in cost matrix CM, c_p is the candidate warping path, c_{p*} is the minimum cost warping path.

1. $CM[1,j] = \sum_{k=1}^{j} dis(X(1), Y(j))$
2. $CM[i,1] = \sum_{k=1}^{i} dis(X(i), Y(1))$
3. $CM[i,j] = min(CM[i,j-1], CM[i-1,j], CM[i-1,j-1]) + dis(X(i), Y(j))$

where, M is length of X and N is length of Y.

6.3.2 EROS

EROS (Extended Frobenius Norm), [77] described it as "let us consider two multi-variate time series, $X_{D\times M}$ and $Y_{D\times N}$. Let U_X and U_Y be the two right eigenvector matrices by applying SVD to the covariance matrices, coX and coY, respectively. Let $U_X = [x_1, ..., x_D]$ and $U_Y = [y_1, ..., y_D]$." Formally, EROS defined as

$$EROS(X, Y, w) = \sum_{i=1}^{D} w_i.cosine(U_X^i, U_Y^i) \tag{6.2}$$

where w, is the aggregated weight determined by taking the average of eigenvalues of the two time series.

6.3.3 PCA

PCA similarity factor [44] measures similarity between two groups of principal components.

$$S_{PCA}(X, Y) = \sum_{i=1}^{k1} \sum_{j=1}^{k2} cosine^2(X^i, Y^j) \tag{6.3}$$

where, k1 and k2 are the number of principal components in X and Y respectively that preserve 95% of the variance in data.

6.4 Evaluation Task

For uniform evaluation of algorithms, experiments are based on the following *two* tasks: first, classification accuracy and second, effective separation between clusters of time series. It is important for a similarity measure to have high retrieval accuracy but robustness to temporal asynchrony is crucial too. To measure robustness, gains and losses in accuracy and effective cluster separation is evaluated.

6.4.1 Retrieval Accuracy

To measure the retrieval accuracy, experiments use the true labels assigned to each time series at the start of experiments. Top-k queries are executed on the dataset and accuracy is determined based on the number of time series returns with the same label to that of the query time series's label. Percentage accuracy is the ratio of number of time series returned with query file label to total number of time series returned.

$$\%\ accuracy = \frac{\#\ of\ time\ series\ with\ query\ label}{\#\ of\ time\ series\ returned} \tag{6.4}$$

6.4.2 Cluster Separation

Cluster separation is defined in terms of how far two clusters are. The aim is to keep different clusters at a sufficiently large distance to prevent misclassification. Thus preventing overlaps or outliers between them. In sum, experiments try to identify a similarity measures that creates compact clusters with sufficient separation between different clusters. Separation can be measured in 3 ways:

1. **Cluster Strength** (CS): is the ratio of average inter class distance to average intra class distance. Intuitively, more compact or dense the cluster is, the higher the cluster strength would be. A dense cluster represents closely related time series which means that the time series are highly similar to query time series. Also, the higher value of inter class average distance denoted the time series outside clusters are distanced effectively away to avoid misclassification or overlap.

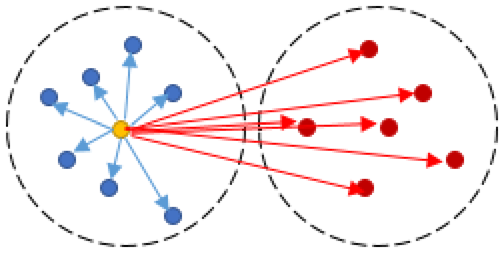


Figure 6.5: Cluster Strength : Blue Arrow Represents Distance Between Query Time Series and Time Series with Same Label as Query; Red Arrow Represents Distance Between Query Time Series and Time Series with Label Other Than Query Time Series Label.

$$CS_C = \frac{Inter\ Class\ Average\ Distance}{Intra\ Class\ Average\ Distance} \tag{6.5}$$

where,

$Inter\ Class\ Average\ Distance\ = \sum dis(Q,T)\ \forall`T_{Label} \neq Q_{Label}$

$Intra\ Class\ Average\ Distance\ = \sum dis(Q,T)\ \forall`T_{Label} = Q_{Label}$

and C is the cluster of interest.

Higher values of CS denotes that the clusters have minimal overlapping with respect to the query time series and there is sufficient distance between the clusters for effective separation.

2. **Border Separation** (BS): Cluster strength is an overall measure that accounts for all the time series in the data space, whereas, border strength is more strict measure that evaluates the boundary of a cluster. The question that it answers is: *How well the cluster boundary is defined such that time series close to the cluster boundary are effectively separated without misclassification?*

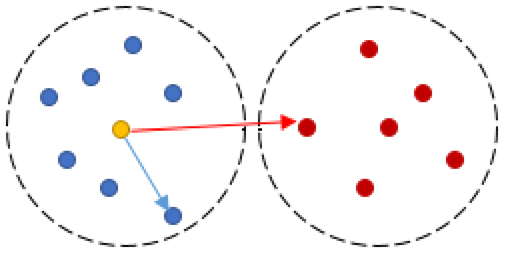


Figure 6.6: Border Separation : Blue Arrow Represents Distance Between Query Time Series and Most Similar Time Series with Label Other Than Query Time Series Label; Red Arrow Represents Distance Between Query Time Series and Least Similar Time Series with Label Same As Query Time Series Label.

$$BS_C = \frac{Most\ similar\ inter\ cluster\ time\ series}{Least\ similar\ intra\ cluster\ time\ series} \tag{6.6}$$

where,

$Most\ similar\ inter\ cluster\ time\ series = min(\sum dis(Q,T)\ \forall`T_{Label} \neq Q_{Label})$

$Least\ similar\ intra\ cluster\ time\ series = max(\sum dis(Q,T)\ \forall`T_{Label} = Q_{Label})$

and C is the cluster of interest. If value of $BS > 1$ then the cluster border is effectively strong in keeping the two cluster separate.

3. **Deceptor Prevention** (DP) : This is the extreme condition check where the outliers from outside cluster member is more similar to that of most similar inside cluster member. Formally, deceptor prevention is the ratio of most similar inter cluster time series to that of most similar intra cluster time series.

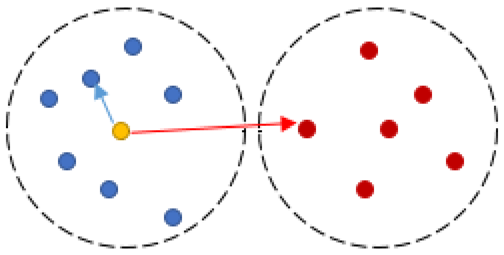


Figure 6.7: Deceptor Prevention : Blue Arrow Represents Distance Between Query Time Series and Most Similar Time Series with Label Other Than Query Time Series Label; Red Arrow Represents Distance Between Query Time Series and Most Similar Time Series with Label Same As Query Time Series Label.

$$DP_C = \frac{Most\ similar\ inter\ cluster\ time\ series}{Most\ similar\ intra\ cluster\ time\ series} \tag{6.7}$$

where,

$Most\ similar\ inter\ cluster\ time\ series = min(\sum dis(Q,T)\ \forall`T_{Label} \neq Q_{Label})$

$Most\ similar\ intra\ cluster\ time\ series = min(\sum dis(Q,T)\ \forall`T_{Label} = Q_{Label})$

and c is the cluster of interest.

If $DP > 1$, this denotes that there are no outliers present for the given query that could be more similar to the query time series than the time series contained in the cluster which the query time series belongs.

6.5 Temporal Asynchrony

We introduce two different type of temporal asynchrony, shift and stretch, which are be further divided in structurally inherent or random, in the time series. These asynchronies are:

1. **Shift-based** : Temporal shift is a type of asynchrony that introduces lag in temporal occurrence of a pattern. Shift creates a step function in the pattern. This step is padded either with the first data value in the time series, right shift, or with the last value in time series, left shift. Right shift suggests delay in occurrence of pattern, whereas left shift denotes advancements in pattern occurrence in time.

   Depending on how shift is introduced in the data representation, multi-scale hierarchical data structure, seen in Chapter 5, shifts can be classified into two categories : Random shift and Inherent Shift. Let us consider, we have to introduce a shift of 25% in time series.

   (a) Random Shift: In case of random shift each variate at each level in the hierarchy is shifted independently. Also, the direction of shift, whether to shift left or right is randomly selected. In case of 25% shift it picks a random integer between 22 and 28 if the length of time series is 100 units. We kept a margin of 3% to keep the decision random.

   (b) Inherent shift: Whereas in case of inherent shift, shift is independently and randomly select at each variate at each level in the hierarchy, H_D, but shift at a particular variate is inherited by the sub-hierarchy underneath it.

2. **Stretch based** : Temporal Stretching is a type of asynchrony that alters the temporal length of occurrence of pattern. Temporal length refers to the duration of time in which a pattern is observed. A random point is picked, between the start and end of time series which divides the time series into two sub-sequences, one of them is compressed and other one is stretched.

    (a) Random Stretching: In case of random stretching, each variate at each level in the hierarchy is stretched independently. Random point, A, is picked on each variate from where the time series will be stretched and compressed. Consider 25% stretching, then point B at a distance of 25% of length of time series is selected at random on either side of the point A.

    (b) Inherent Stretch: Inherent stretching is similar to random but here point A is same for all variates.

Direction of shift or stretch is selected at random. Each asynchrony is introduced at 6 different degree defined in terms of 0% - 50% of the length of time series. For each percentage asynchrony, five iterations are run to eliminate the effect of a particular random choice. The reason for adding percentage based asynchrony is to investigate robustness of a measure to different degree of temporal asynchrony.

6.6 Experimental Setup

Experiments were performed on Windows 7 (6.1.7601) x64 based workstation featuring 3.10 GHz Intel(R) Core(TM) i5-2400, Quadcore CPU with 8GB RAM and MATLAB 2014b 32-bit (8.4.0.150421).

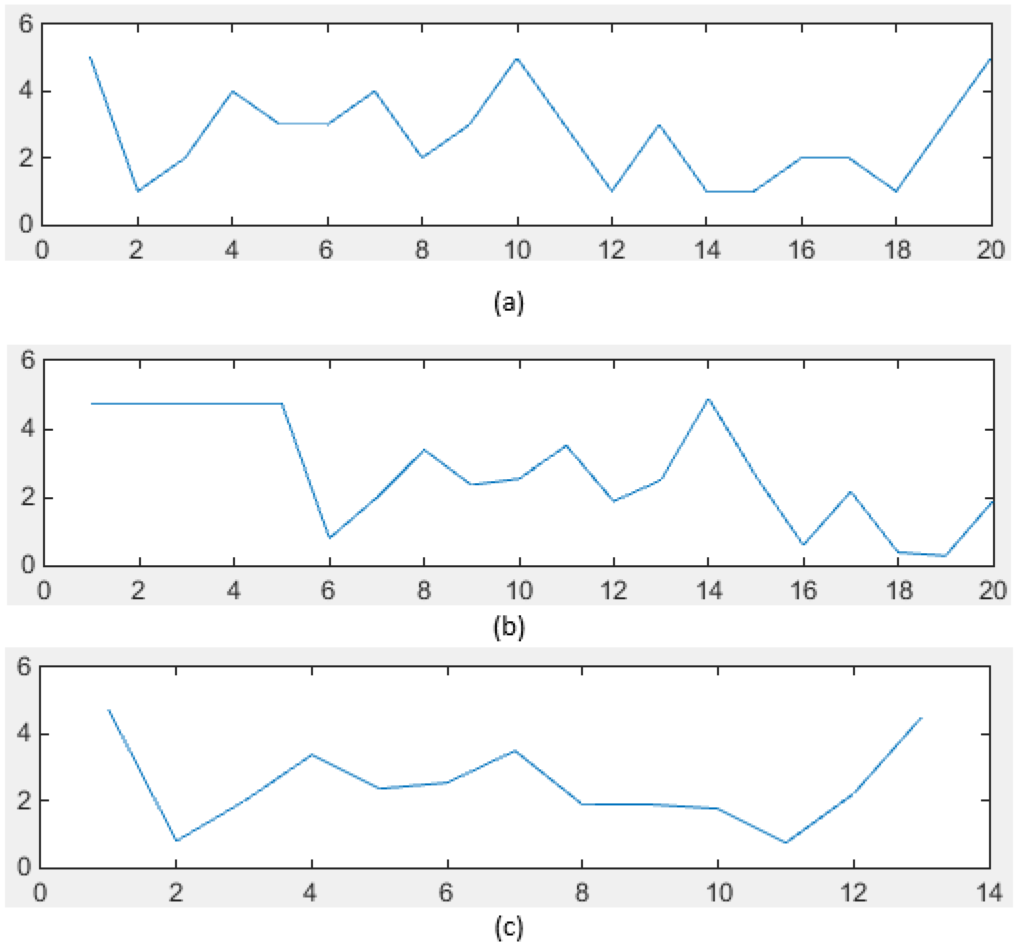


Figure 6.8: Temporal Asynchrony (a)Original Data (b) Right Shift in Data of 25% (c) Stretch of 25%

6.7 Evaluation of Retrieval Accuracy

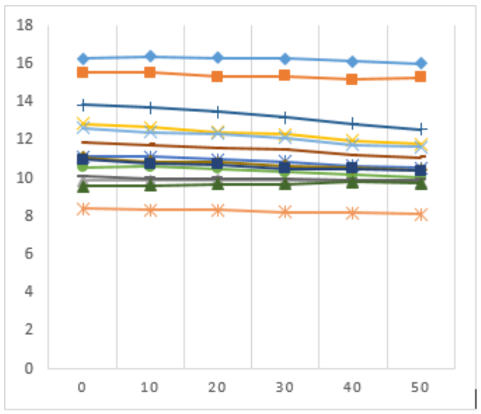
In this evaluation, retrieval accuracy is measured. Three different top-k queries, Top-1, Top-5 and Top-k (k is total number of elements contained in a cluster), are fired on the time series database for both Mocap and Kaggle dataset.

6.7.1 Top-1 Retrieval Accuracy

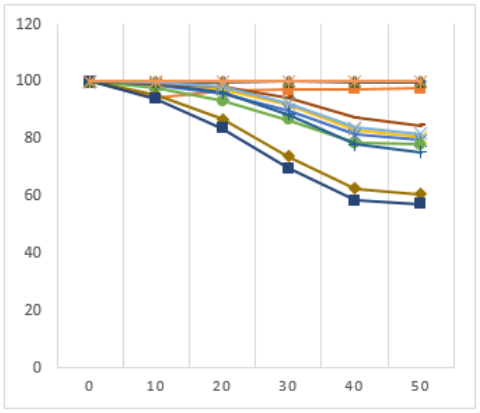
The common observation is the 100% retrieval accuracy for all measures when no asynchrony is observed in time series, with no exception found in the experiments performed for five different datasets. When asynchrony, such as *random stretching* is introduced, a drop in accuracy is observed (*Refer Figure: 6.9*). This drop is significant in the case of data-driven measures, whereas meta-driven measures show insignificant drop in retrieval accuracy for low degree of stretching but in case of higher stretching less drop in retrieval accuracy is observed for metadata-driven measures in comparison to data-driven measures. It is important to note the stable performance of measures operating on feature components extracted from the time series as pattern are preserved in the time series but with different temporal presence (*Refer Figure: 6.9*). Performance of different measures observed in the presence of *inherent stretching* is analogous to the performance observed in the presence of *random stretching* except for the sudden drop in retrieval accuracy for SIDTW at 50% *inherent stretching.* This sudden drop is the outcome of excessive similarity introduced due to the synchronized similarity across all variates in the variate hierarchy, H_D (*Refer Figure : 6.10*).

In the case of *random shift*, the step introduced in the data due to shift adds a pseudo-similarity between the variates in contrast to stretching where the temporal length of pattern is altered. Because of the similarity introduced by the step function, a significant drop in retrieval accuracy is observed (*Refer Figure: 6.10(c)*). But in the presence of *random stretching*, insignificant amount of change in retrieval accuracy is observed (*Refer Figure: 6.10(a)*).

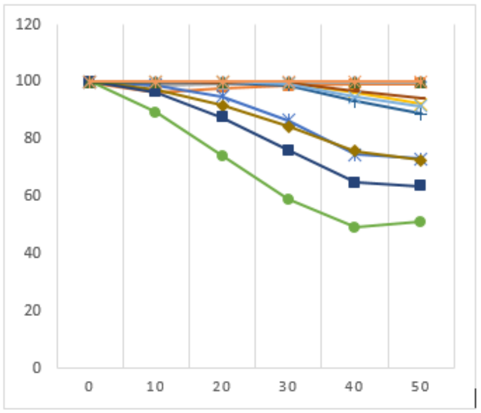
It is important to understand, for Kaggle data (X), different measures behave similarly in the presence of different temporal asynchronies (*Refer Figure: 6.10*). This is observed because of the similar data distribution observed in the variate of



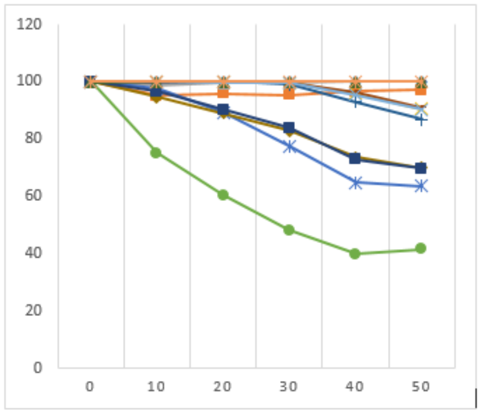
(a) Kaggle - W



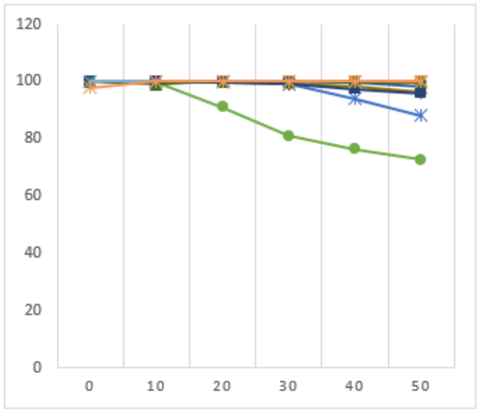
(b) Kaggle - X



(c) Kaggle - Y



(d) Kaggle - Z



(e) Mocap

Figure 6.9: Random Distortion; X-axis: % Asynchrony, Y-axis: % Accuracy for Top-1

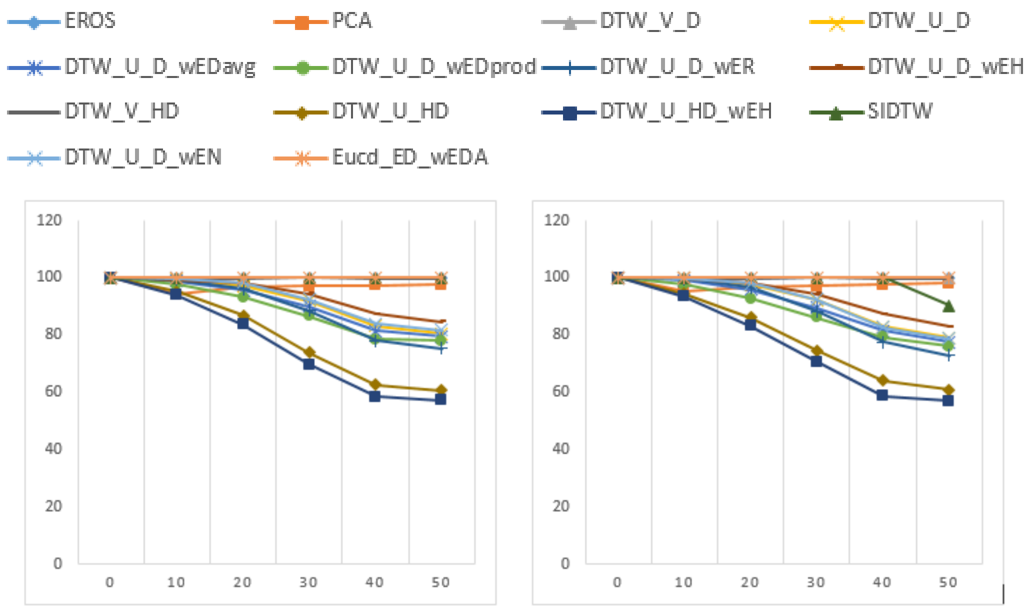
Kaggle-X (*Refer Figure: 6.11*).

6.7.2 Top-5 Retrieval Accuracy

When temporal asynchrony like *random stretching* is observed in data, relatively all measures demonstrate similar behavior across all datasets (*Refer Figure: 6.12*). Measures involving similarity computation over feature components, such as EROS and PCA, show relative robustness or minimal drop in retrieval accuracy in the presence of high degree of *random stretching*, whereas metadata-driven measures show higher accuracy than data-driven measures as metadata is not affected by asynchrony (*Refer Figure: 6.13*). The relative robustness of feature component based measures is due to the use of important patterns extracted from the time series, as patterns are not lost in stretching but their scope on time axis is altered.

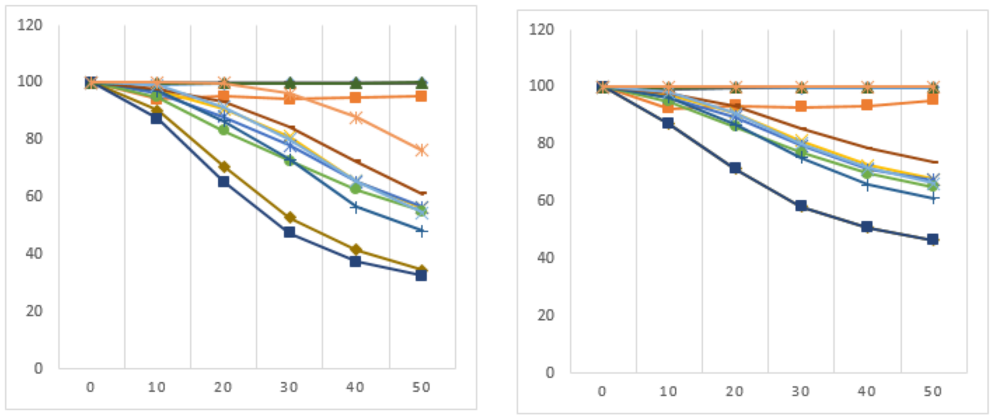
Measures demonstrate a loss in retrieval accuracy when *inherent stretching* is introduced in the time series and have a relatively stable retrieval accuracy for higher degree of *inherent stretching*. This loss in accuracy is observed because of the synchronized stretching that introduces similarity between the variate at the same time instance (*Refer Figure: 6.14*).

In the presence of *temporal shift*, different measures show difference in their robustness for different datasets (*Refer Figure: 6.15*). The reason for this is the difference in data distribution as seen earlier in this section: Mocap and Kaggle has high variance and low variance in data in its variates respectively. But in both cases, *random* and *inherent shift*, a drop in accuracy is observed as a result of step similarity introduced by the shift in pattern. At the same time, measures show relative stability in retrieval accuracy for different degree of distortion (*Refer Figure: 6.16*). Most accuracy loss is observed in mocap data as shift reduced the discrimination power of each variate (*Refer Figure: 6.15 and 6.17(a)*). In *Figure: 6.17(a)*, different variates have different



(a) Random Distortion

(b) Inherent Distortion



(c) Random Shift

(d) Inherent Shift

Figure 6.10: Kaggle - X; X-axis: % Asynchrony, Y-axis: % Accuracy for Top-1

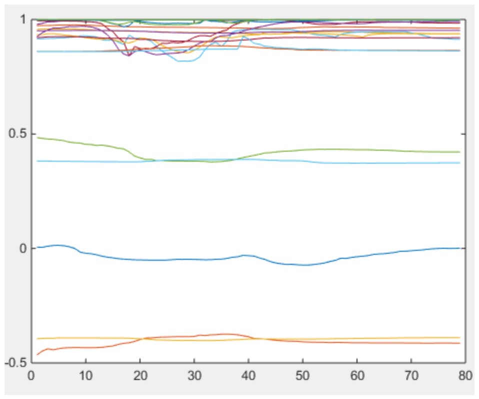
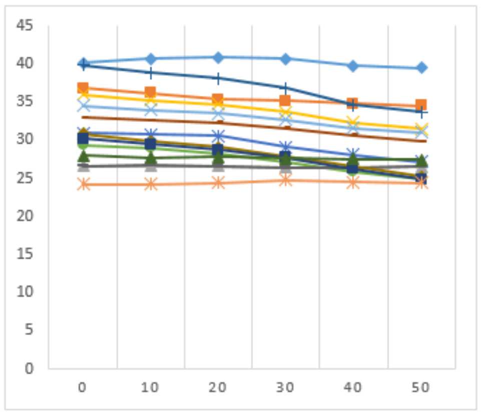


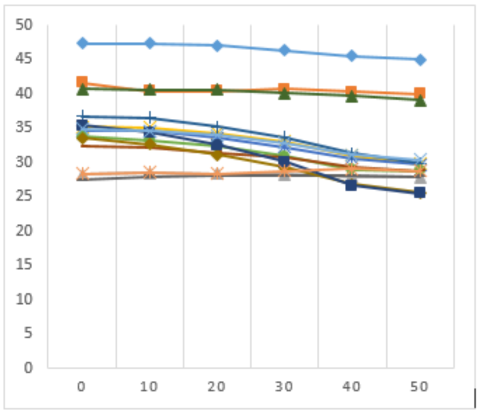
Figure 6.11: Sample Time Series Kaggle-X Dataset

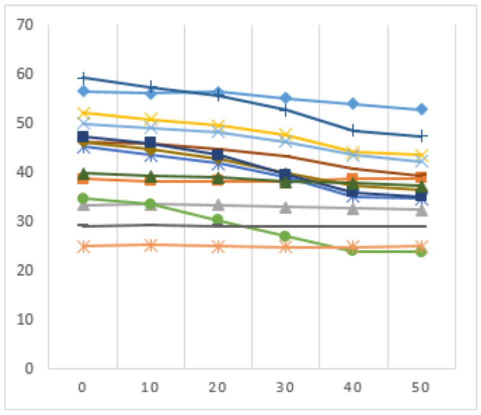
data distribution which is lost because of shift *Figure: 6.17(b) and 6.17(c)*, which introduces a step function that results in a pseudo-similarity between the variates of different time series in contrast to that of kaggle data (X) where variates have similar distributions across variates that do not get affected significantly (*Figure: 6.17(d)*). For *inherent shift* similar performance is observed, whereas measures involving feature components shows relative robustness to shifts and metadata-driven measures demonstrate high accuracy (*Refer Figure: 6.15*).

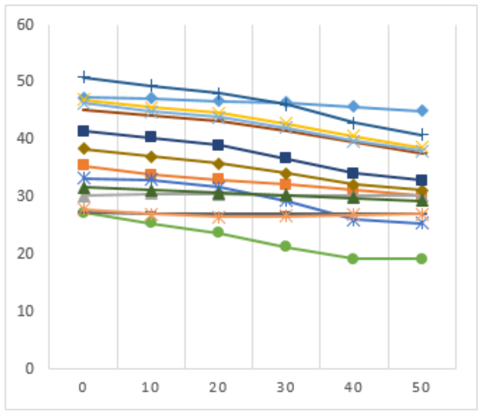
6.7.3 Top-k Retrieval Accuracy

It is clearly observable in *Figure: 6.18* that the performance of metadata-driven measures for top-k retrieval is better than data-driven measures with the exception of EROS (feature component based measure). Specifically, use of distance matrix as metadata gives significantly better retrieval accuracy than naive DTW. Similar retrieval accuracy, with low drop, is observed in the presence of high degree of *random stretching* for mocap dataset as the self-discriminatory power of variates is preserved (*Refer Figure: 6.19*). Again in the case of *inherent stretching* a drop in retrieval accuracy is observed as a result of variate similarity due to stretching at same instance


(a) Kaggle - W


(b) Kaggle - X


(c) Kaggle - Y


(d) Kaggle - Z

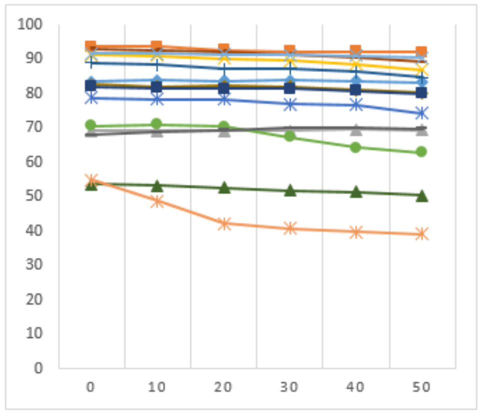
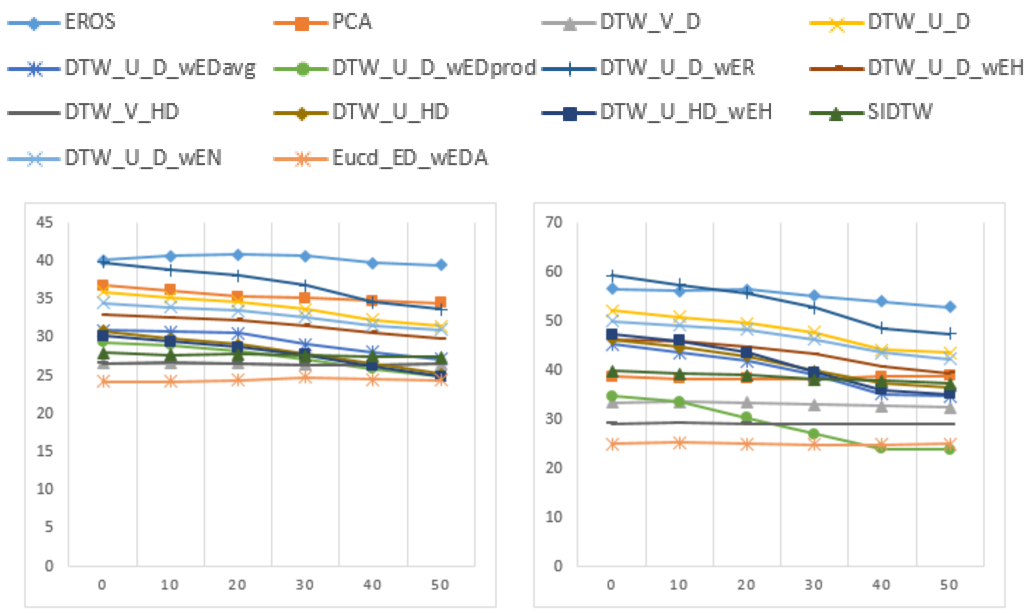

(e) Mocap

Figure 6.12: Random Distortion; X-axis: % Asynchrony, Y-axis: % Accuracy for Top-5



(a) Kaggle - W (b) Kaggle - Y

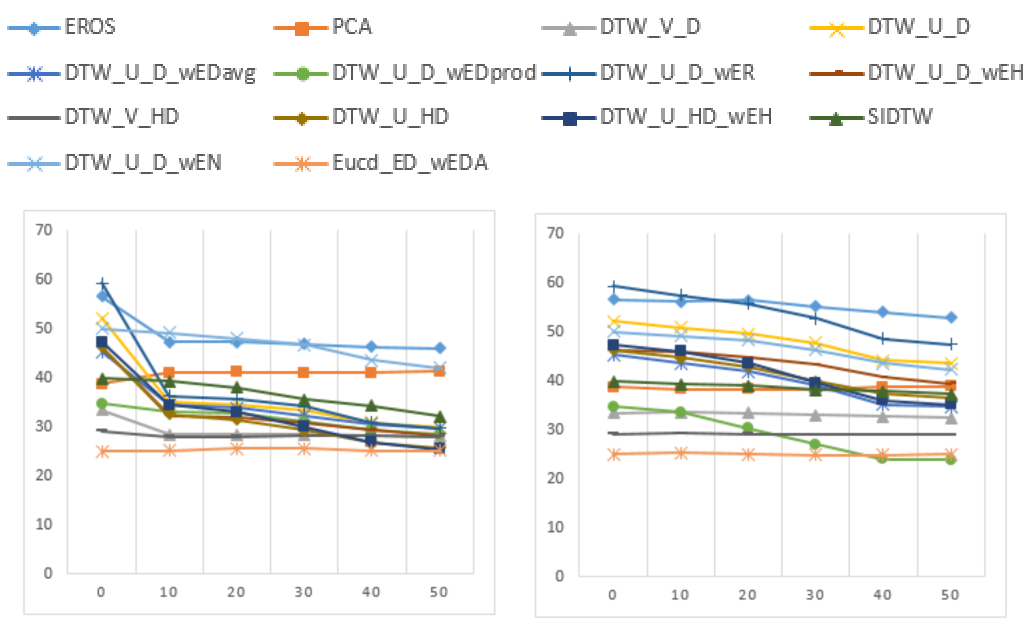
Figure 6.13: Random Distortion; X-axis: % Asynchrony, Y-axis: % Accuracy for Top-5

in time (*Refer Figure: 6.20*).

In case of random shift, significant loss in retrieval accuracy is observed for mo-cap data for all similarity measures except for feature component based measures that have poor retrieval accuracy but are robust against shifts, whereas meta-driven measures still give highest retrieval accuracy (*Refer Figure: 6.21*).

6.8 Evaluation of Effective Cluster Separation

While performing time series classification it is important for a similarity measure to minimize false positive classification. In the following part of this section, cluster separation for different time series similarity measures is evaluated on the following three factors: first, cluster strength (CS), second, border separation (BS), and third, deceptor protection (DP).

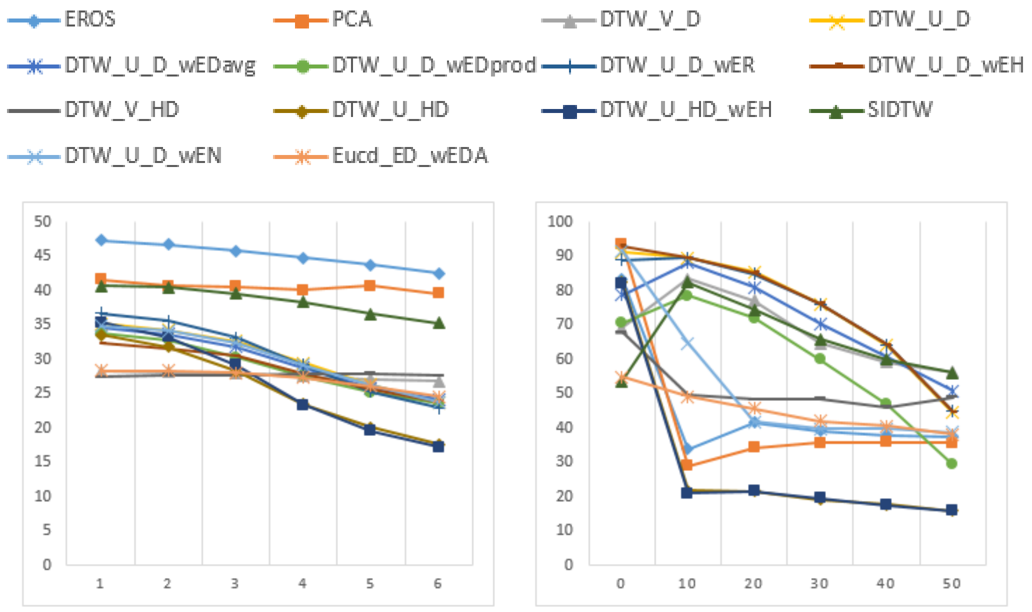


(a) Inherent Distortion (b) Random Distortion

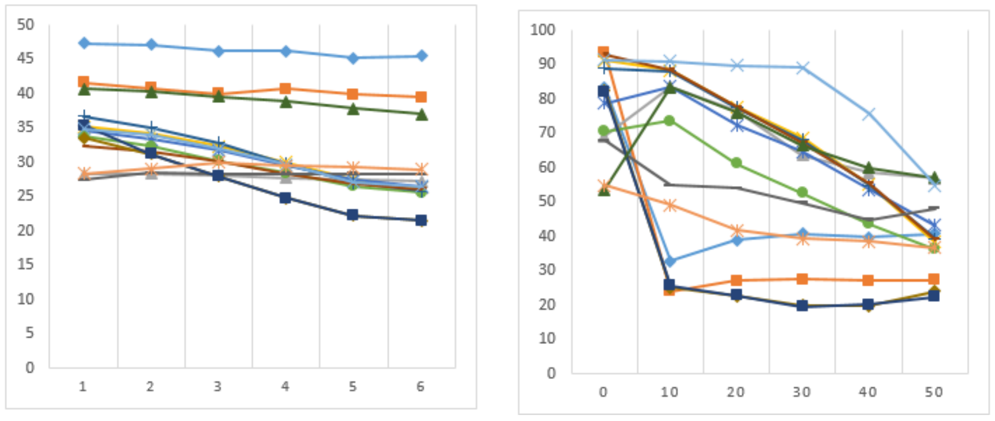
Figure 6.14: Kaggle - Y; X-axis: % Asynchrony, Y-axis: % Accuracy for Top-5

6.8.1 Cluster Strength

When *random stretching* is observed in the data, measures demonstrate near stable separation, that is, they show robustness against the stretching. But if gain and loss in retrieval accuracy in analyzed, metadata-driven measures show the most robust behavior against *random stretching* and data-driven measures show both gain and loss (mostly) in their separation (*Refer Figure: 6.22*). Similar behavior is observed when *inherent stretching* is introduced in the data (*Refer Figure: 6.23(a)*). Robustness of different measures to stretching is the outcome of containment of the entire pattern in the time series and the loss is observed due to the change in temporal occurrence in the pattern. In case of shift in patterns, overall performance of different measures is analogous to the behavior observed in case of presence of stretching in the time series. But in case of shift, a drop in effective separation between clusters of time

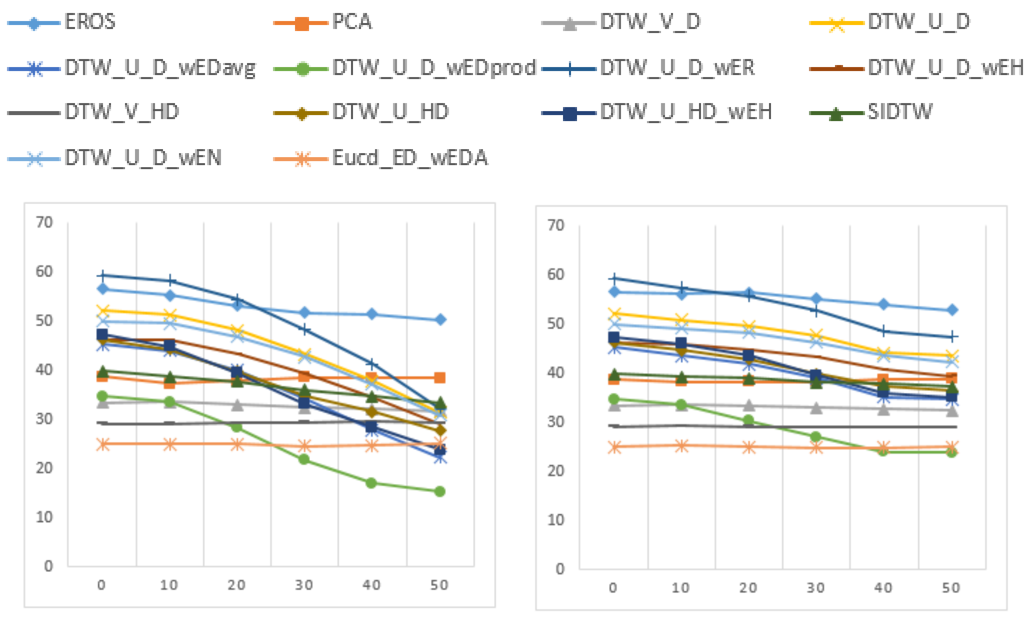


(a) Random Shift on Kaggle-X (b) Random Shift on Mocap



(c) Inherent Shift on Kaggle-X (d) Inherent Shift on Mocap

Figure 6.15: Kaggle - Y; X-axis: % Asynchrony, Y-axis: % Accuracy for Top-5



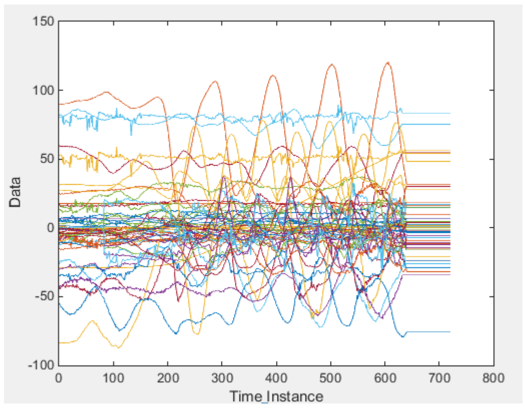
(a) Random Shift (b) Random Distortion

Figure 6.16: Kaggle - Y; X-axis: % Asynchrony, Y-axis: % Accuracy for Top-5

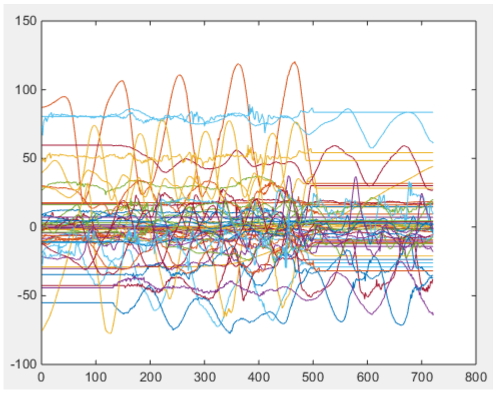
series is observed (*Refer Figure: 6.23(b)*). This observation is due to the similarity introduced by the step function due to temporal shift.

6.8.2 Border Separation

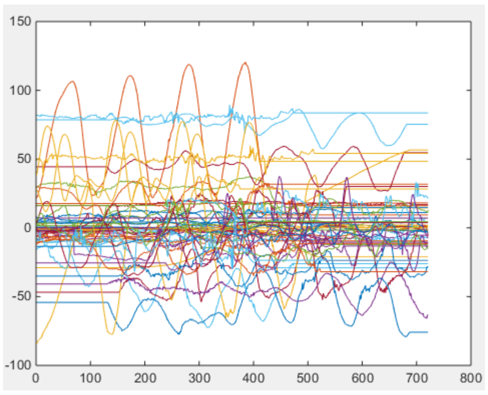
Border separation refers to effective classification of the time series allocated at clusters boundary to minimize misclassification, false positives. Overall, in kaggle dataset, due to the lack of self-discriminatory power amongst the variates (*Refer Figure: 6.17(a)) and 6.17(d)*) a lower separation score is observed (*Refer Figure: 6.24*). In the case of random shifts, lower score for border separation for kaggle is still observed (*Refer Figure: 6.25(b)*). At the same time, mocap dataset shows random behavior because of lose in variate discrimination due to random shift in pattern (*Refer Figure: 6.25(a)*). Upon clear observation, metadata-driven measures perform better border separation than the data-driven measures with an exception to feature



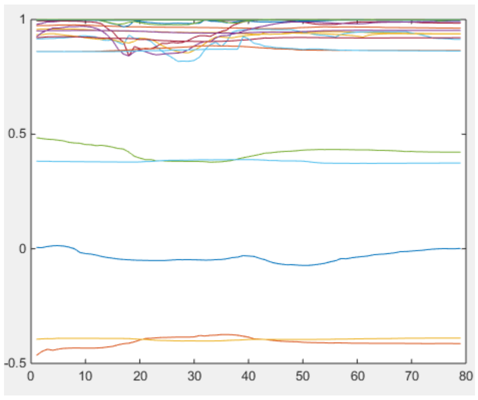
(a) Mocap Dataset



(b) Mocap Dataset



(c) Mocap Dataset



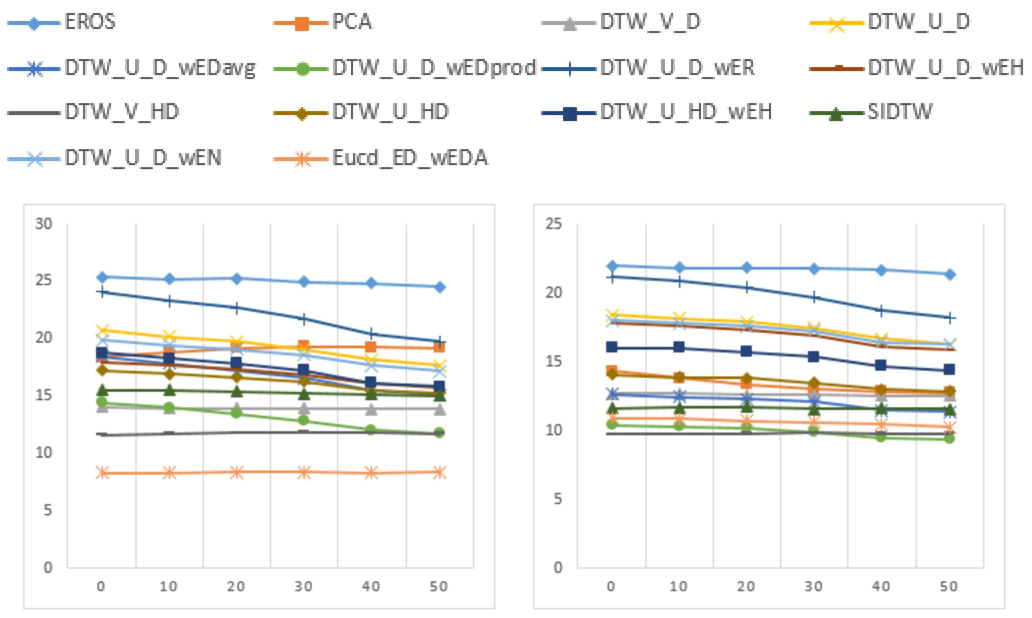
(d) Kaggle - X Dataset

Figure 6.17: Different Type of Temporal Shift on Mocap and Kaggle Data

component based measures.

6.8.3 Deceptor Prevention

There is possibility with every similarity measure for false-positive classification. With deceptor prevention, the classification accuracy of most nearest classification inside and outside cluster of query time series is evaluated.In this view, we can observe that mocap suffers a signification lose in deceptor prevention as variates loose their power of discrimination as the degree of shift based asynchrony is increased, whereas



(a) Kaggle- Y

(b) Kaggle - Z

Figure 6.18: Random Distortion, X-axis: % Asynchrony, Y-axis: % Accuracy for Top-k

in kaggle where variates suffer a insignificant drop in accuracy due to minimal data variance (Refer Figure: 6.26).

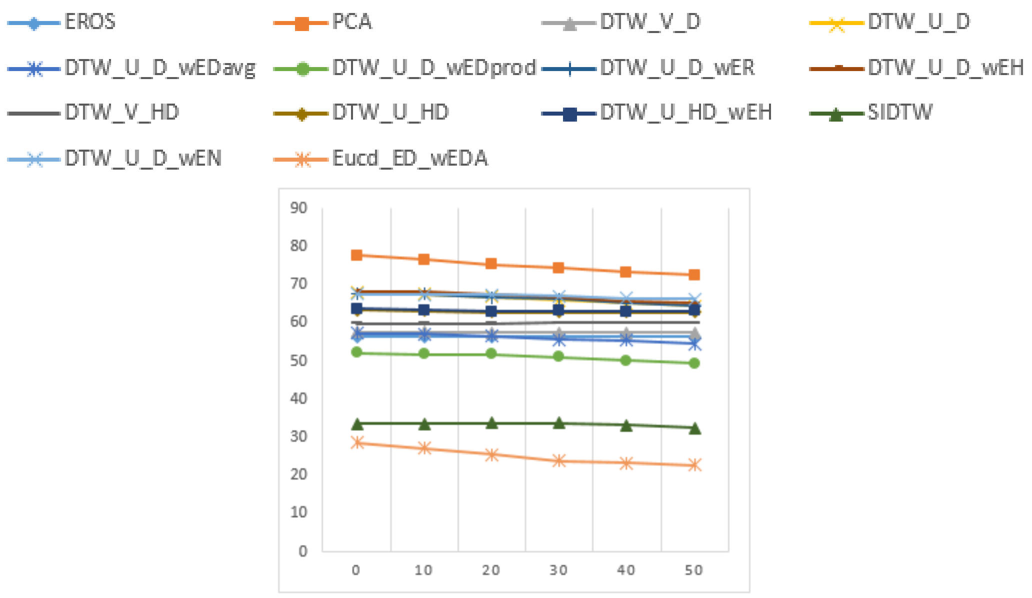


Figure 6.19: Random Distortion Mocap Data, X-axis: % Asynchrony, Y-axis: % Accuracy for Top-k

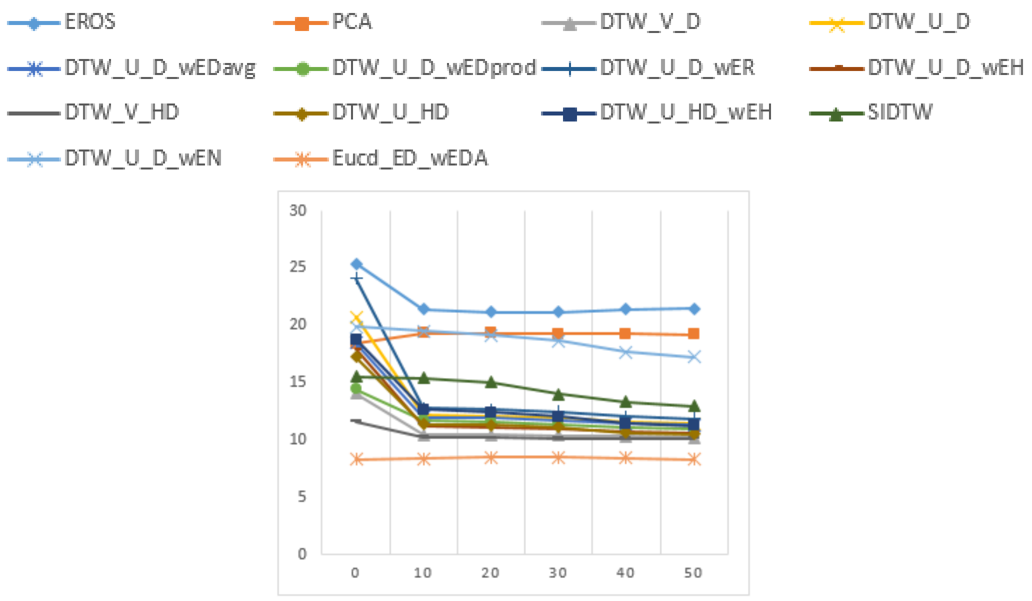


Figure 6.20: Inherent Distortion Kaggle-Y Data, X-axis: % Asynchrony, Y-axis: % Accuracy for Top-k

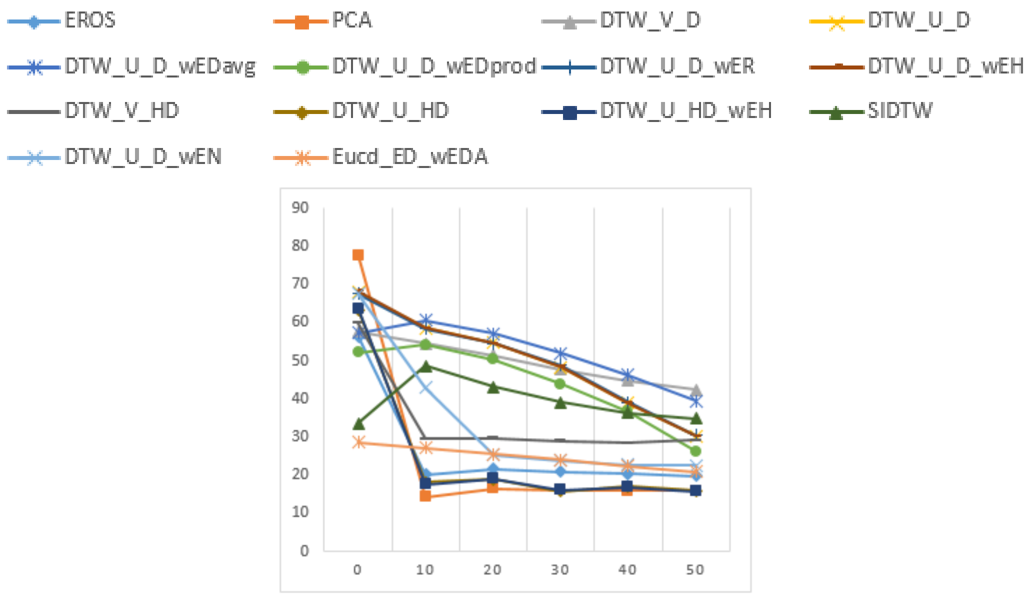
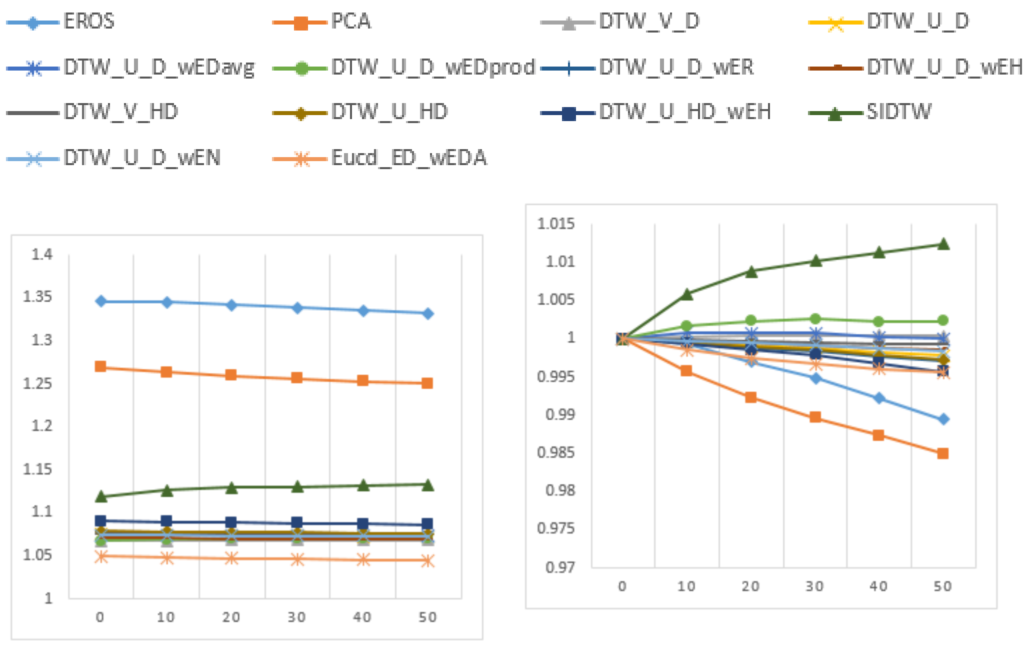
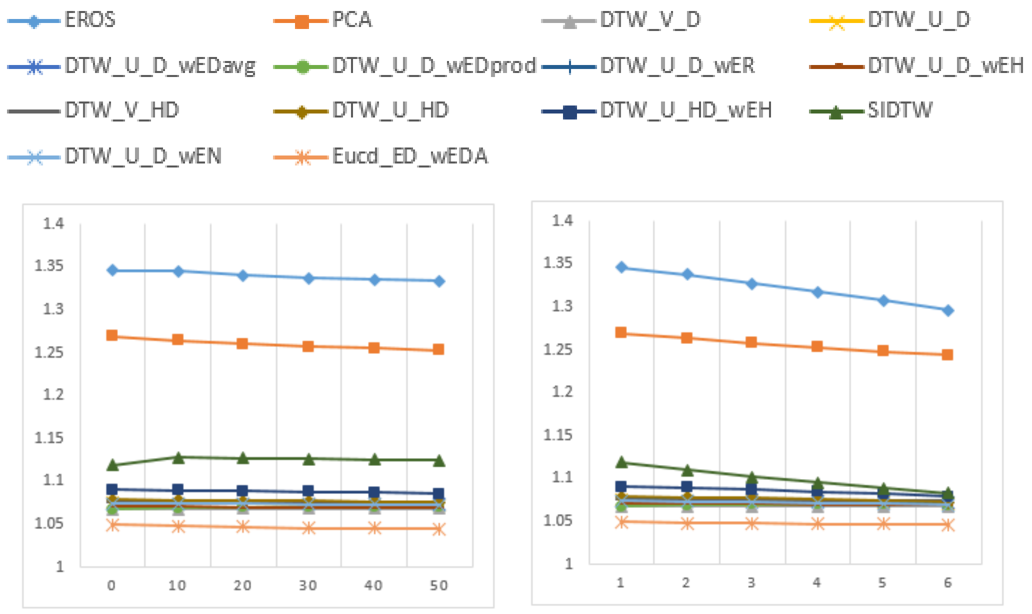


Figure 6.21: Inherent Distortion Kaggle-Y Data, X-axis: % Asynchrony, Y-axis: % Accuracy for Top-k



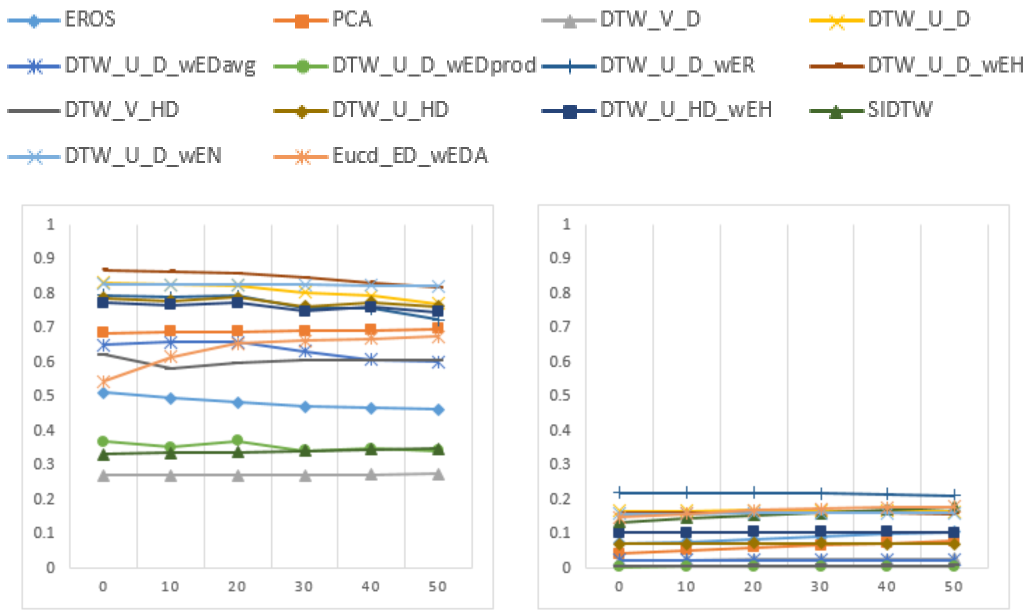
(a) Cluster Separation (b) Separation Gains

Figure 6.22: Random Distortion Kaggle-X, X-axis: % Asynchrony, Y-axis: Separation Ratio



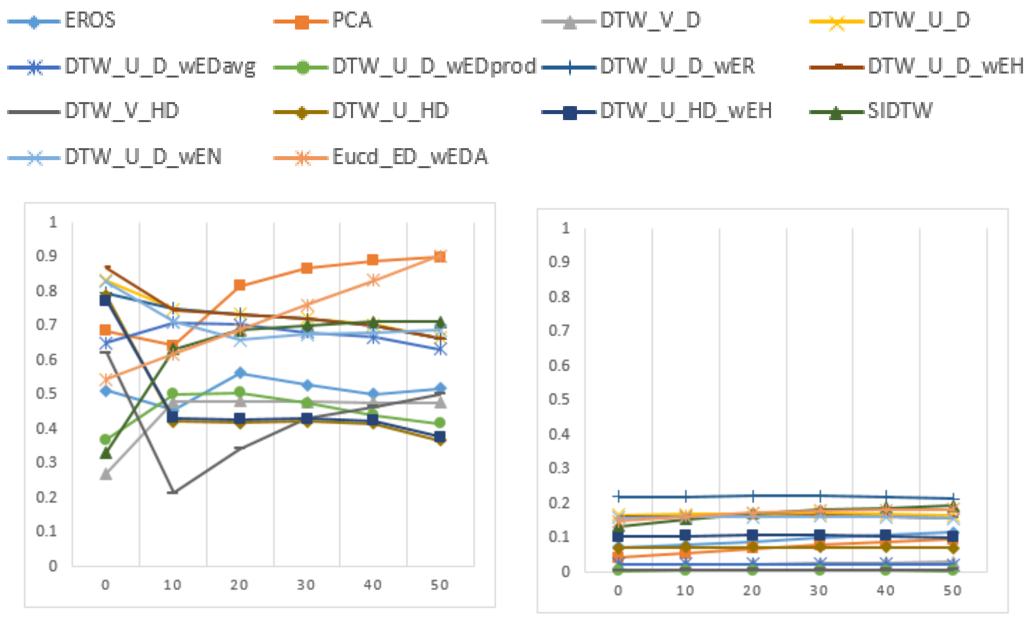
(a) Inherent Distortion (b) Random Shift

Figure 6.23: Kaggle-X, X-axis: % Asynchrony, Y-axis: Separation Ratio



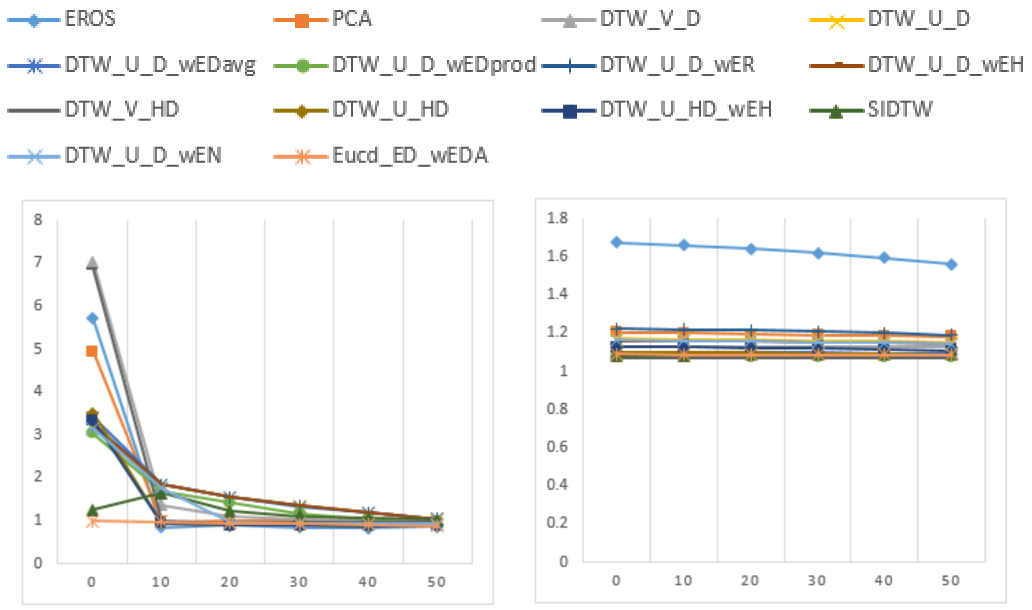
(a) Mocap (b) Kaggle - Z

Figure 6.24: Random Distortion, X-axis: % Asynchrony, Y-axis: Separation Ratio



(a) Mocap (b) Kaggle - Z

Figure 6.25: Random Shift, X-axis: % Asynchrony, Y-axis: Separation Ratio



(a) Mocap (b) Kaggle - Z

Figure 6.26: Random Shift, X-axis: % Asynchrony, Y-axis: Prevention Ratio

Chapter 7

CONCLUSION AND FUTURE WORK

Conclusion

Using weighted measures improved the retrieval accuracy and effective cluster separation, in contrast to the non-weighted measures. As weights add importance to the variates with critical patterns contained in the time series.

In general, measures discussed in this thesis can be broadly classified into three categories, first, feature component based, second, data-driven and third, metadata-driven. Metadata-driven measures demonstrate high retrieval accuracy as relationship between the variates is not affected by the temporal asynchrony. Major loss in accuracy is observed in data-driven measures as temporal asynchrony affect the patterns contained in the time series. Whereas, metadata-driven and feature component based measures either show relative stability to asynchrony or a minor drop in their effectiveness. Also, the behavior of measures dependents on the data distribution in the variates. As seen in kaggle dataset where variates are clustered in terms of their data distribution show minimal loss in their effectiveness in the presence of asynchrony, whereas in case of the mocap dataset, significant loss in effectiveness of measures is observed as the variates have high difference in their data distribution.

The use of contextual correlation between the variates highlights critical variates in the time series. Human gestures are the outcome of various attributes varying simultaneously. DTW_U_D_wER, DTW_U_D_wEH and DTW_U_D_wEN uses different type of relationships between the variates to determine their importance. These algorithms deliver the highest accuracy for similarity-based retrieval. Moreover, by projecting the data space onto a vector basis by using metadata leads to a more robust similarity measure against temporal asynchrony.

Future Work

There are many aspects on which this research could be extended, such as in SI-DTW, determining an optimal window width or an adaptive window. Also, more work could be done on devising more effective time series representation for effective time series analysis.

REFERENCES

[1] "Clustering", URL `http://iss.ices.utexas.edu/images/galois/clustering.png` (2015).

[2] "Cmu graphics lab motion capture database", URL `http://mocap.cs.cmu.edu/` (2015).

[3] "Gesture image", URL `https://kaggle2.blob.core.windows.net/competitions/kaggle/3386/media/banner_chalearn.png` (2015).

[4] "Goog historical prices — google inc. stock - yahoo! finance", URL `http://finance.yahoo.com/q/hp?s=GOOG` (2015).

[5] "Multi modal gesture recognition", URL `https://www.kaggle.com/c/multi-modal-gesture-recognition` (2015).

[6] Abbasi, A. A. and M. Younis, "A survey on clustering algorithms for wireless sensor networks", Computer communications **30**, 14, 2826–2841 (2007).

[7] Aggarwal, C. C., "On effective classification of strings with wavelets", in "Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 163–172 (ACM, 2002).

[8] Agrawal, R., C. Faloutsos and A. Swami, *Efficient similarity search in sequence databases* (Springer, 1993).

[9] Aldous, D. J., "Deterministic and stochastic models for coalescence (aggregation and coagulation): a review of the mean-field theory for probabilists", Bernoulli pp. 3–48 (1999).

[10] Anderberg, M. R., *Cluster Analysis for Applications: Probability and Mathematical Statistics: A Series of Monographs and Textbooks*, vol. 19 (Academic press, 2014).

[11] Berkhin, P., "A survey of clustering data mining techniques", in "Grouping multidimensional data", pp. 25–71 (Springer, 2006).

[12] Berndt, D. J. and J. Clifford, "Using dynamic time warping to find patterns in time series.", in "KDD workshop", vol. 10, pp. 359–370 (Seattle, WA, 1994).

[13] Bhatia, N. *et al.*, "Survey of nearest neighbor techniques", arXiv preprint arXiv:1007.0085 (2010).

[14] Bhattacharyya, A., "On a measure of divergence between two multinomial populations", Sankhyā: The Indian Journal of Statistics pp. 401–406 (1946).

[15] Cai, Y., H. Tong, W. Fan, P. Ji and Q. He, "Facets: Fast comprehensive mining of coevolving high-order time series", in "Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining", pp. 79–88 (ACM, 2015).

[16] Candan, K. S., R. Rossini, X. Wang and M. L. Sapino, “sdtw: computing dtw distances using locally relevant constraints based on salient feature alignments”, Proceedings of the VLDB Endowment **5**, 11, 1519–1530 (2012).

[17] Candan, K. S. and M. L. Sapino, *Data management for multimedia retrieval* (Cambridge University Press, 2010).

[18] Celebi, S., A. S. Aydin, T. T. Temiz and T. Arici, “Gesture recognition using skeleton data with weighted dynamic time warping.”, in “VISAPP (1)”, pp. 620–625 (2013).

[19] Chen, L., *Similarity search over time series and trajectory data*, Ph.D. thesis, University of Waterloo (2005).

[20] Chen, L. 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”, pp. 792–803 (VLDB Endowment, 2004).

[21] Chen, L., M. T. Özsu and V. Oria, “Robust and fast similarity search for moving object trajectories”, in “Proceedings of the 2005 ACM SIGMOD international conference on Management of data”, pp. 491–502 (ACM, 2005).

[22] Cover, T. M. and P. E. Hart, “Nearest neighbor pattern classification”, Information Theory, IEEE Transactions on **13**, 1, 21–27 (1967).

[23] Deza, M. M. and E. Deza, *Encyclopedia of distances* (Springer, 2009).

[24] Dubes, R. C., “Cluster analysis and related issues”, in “Handbook of pattern recognition & computer vision”, pp. 3–32 (World Scientific Publishing Co., Inc., 1993).

[25] Fasulo, D., “An analysis of recent work on clustering algorithms”, Department of Computer Science & Engineering, University of Washington (1999).

[26] Ferreira, M. R. P., L. F. D. Santos, A. J. M. Traina, I. Dias, R. Chbeir and C. Traina Jr, “Algebraic properties to optimize knn queries”, Journal of Information and Data Management **2**, 3, 385–400 (2011).

[27] Ferreira, M. R. P., A. J. Traina, I. Dias, R. Chbeir and C. Traina Jr, “Identifying algebraic properties to support optimization of unary similarity queries.”, in “AMW”, (2009).

[28] Fix, E. and J. L. Hodges Jr, “Discriminatory analysis-nonparametric discrimination: consistency properties”, Tech. rep., DTIC Document (1951).

[29] Garrett, M. W. and W. Willinger, “Analysis, modeling and generation of self-similar vbr video traffic”, in “ACM SIGCOMM Computer Communication Review”, vol. 24, pp. 269–280 (ACM, 1994).

[30] Gibson, D., J. Kleinberg and P. Raghavan, “Clustering categorical data: An approach based on dynamical systems”, Databases **1** (1998).

[31] Goldin, D. Q. and P. C. Kanellakis, "On similarity queries for time-series data: constraint specification and implementation", in "Principles and Practice of Constraint Programming—CP'95", pp. 137–153 (Springer, 1995).

[32] Golub, G. H. and C. Reinsch, "Singular value decomposition and least squares solutions", Numerische mathematik **14**, 5, 403–420 (1970).

[33] Gustafson, D. and W. Kessel, "Fuzzy clustering with a fuzzy covariance matrix", in "1978 IEEE conference on decision and control including the 17th symposium on adaptive processes", No. 17, pp. 761–766 (1978).

[34] Hunt, J. W. and T. G. Szymanski, "A fast algorithm for computing longest common subsequences", Communications of the ACM **20**, 5, 350–353 (1977).

[35] Hwang, J.-R., S.-M. Chen and C.-H. Lee, "Handling forecasting problems using fuzzy time series", Fuzzy sets and systems **100**, 1, 217–228 (1998).

[36] Ilyas, I. F., G. Beskales and M. A. Soliman, "A survey of top-k query processing techniques in relational database systems", ACM Computing Surveys (CSUR) **40**, 4, 11 (2008).

[37] Jain, A. K., M. N. Murty and P. J. Flynn, "Data clustering: a review", ACM computing surveys (CSUR) **31**, 3, 264–323 (1999).

[38] Jeong, Y.-S., M. K. Jeong and O. A. Omitaomu, "Weighted dynamic time warping for time series classification", Pattern Recognition **44**, 9, 2231–2240 (2011).

[39] Ji, X., J. Bailey and G. Dong, "Mining minimal distinguishing subsequence patterns with gap constraints", Knowledge and Information Systems **11**, 3, 259–286 (2007).

[40] Keogh, E. and C. A. Ratanamahatana, "Exact indexing of dynamic time warping", Knowledge and information systems **7**, 3, 358–386 (2005).

[41] Keogh, E. J. and M. J. Pazzani, "Derivative dynamic time warping.", (SIAM, ????).

[42] Korn, F., N. Sidiropoulos and C. Faloutsos, "Fast nearest neighbor search in medical image databases", (1996).

[43] Kruskal, J. B., "An overview of sequence comparison: Time warps, string edits, and macromolecules", SIAM review **25**, 2, 201–237 (1983).

[44] Krzanowski, W., "Between-groups comparison of principal components", Journal of the American Statistical Association **74**, 367, 703–707 (1979).

[45] Laxman, S. and P. S. Sastry, "A survey of temporal data mining", Sadhana **31**, 2, 173–198 (2006).

[46] Levenshtein, V. I., "Binary codes capable of correcting deletions, insertions, and reversals", in "Soviet physics doklady", vol. 10, pp. 707–710 (1966).

[47] Li, T., S. Ma and M. Ogihara, "Wavelet methods in data mining", in "Data Mining and Knowledge Discovery Handbook", pp. 603–626 (Springer, 2005).

[48] Liao, T. W., "Clustering of time series data—a survey", Pattern recognition **38**, 11, 1857–1874 (2005).

[49] Lin, J. and Y. Li, "Finding structural similarity in time series data using bag-of-patterns representation", in "Scientific and Statistical Database Management", pp. 461–477 (Springer, 2009).

[50] Liu, S., Y. Garg, K. S. Candan, M. L. Sapino and G. Chowell-Puente, "Notes2: Networks-of-traces for epidemic spread simulations", in "Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence", (2015).

[51] Lowe, D. G., "Distinctive image features from scale-invariant keypoints", International journal of computer vision **60**, 2, 91–110 (2004).

[52] Mao, J. and A. K. Jain, "A self-organizing network for hyperellipsoidal clustering (hec)", Neural Networks, IEEE Transactions on **7**, 1, 16–29 (1996).

[53] Martin, S., G. Kelly, W. G. Kernohan, B. McCreight and C. Nugent, "Smart home technologies for health and social care support", Cochrane Database Syst Rev **4** (2008).

[54] Mörchen, F., "Time series feature extraction for data mining using dwt and dft", (2003).

[55] Müller, M., "Dynamic time warping", Information retrieval for music and motion pp. 69–84 (2007).

[56] Murtagh, F., "A survey of recent advances in hierarchical clustering algorithms", The Computer Journal **26**, 4, 354–359 (1983).

[57] Navarro, G., "A guided tour to approximate string matching", ACM computing surveys (CSUR) **33**, 1, 31–88 (2001).

[58] Safar, M. H. and C. Shahabi, *Shape analysis and retrieval of multimedia objects* (World Scientific, 2003).

[59] Sakoe, H. and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition", Acoustics, Speech and Signal Processing, IEEE Transactions on **26**, 1, 43–49 (1978).

[60] Salton, G. and M. J. McGill, "Introduction to modern information retrieval", (1983).

[61] Salvador, S. and P. Chan, "Fastdtw: Toward accurate dynamic time warping in linear time and space", in "KDD workshop on mining temporal and sequential data", (Citeseer, 2004).

[62] Sanguansat, P., “Multiple multidimensional sequence alignment using generalized dynamic time warping”, WSEAS Transaction on Mathematics **11**, 8, 684–694 (2012).

[63] Senin, P., “Dynamic time warping algorithm review”, (2008).

[64] Shatkay, H. and S. B. Zdonik, “Approximate queries and representations for large data sequences”, in “Data Engineering, 1996. Proceedings of the Twelfth International Conference on”, pp. 536–545 (IEEE, ????).

[65] Steinbach, M., G. Karypis, V. Kumar *et al.*, “A comparison of document clustering techniques”, (2000).

[66] Tay, F. E. and L. Cao, “Application of support vector machines in financial time series forecasting”, Omega **29**, 4, 309–317 (2001).

[67] Taylor, S. J., “Modelling financial time series”, (2007).

[68] Ten Holt, G., M. Reinders and E. Hendriks, “Multi-dimensional dynamic time warping for gesture recognition”, (2007).

[69] Tsay, R. S., *Analysis of financial time series*, vol. 543 (John Wiley & Sons, 2005).

[70] Wang, X., K. S. Candan and M. L. Sapino, “Leveraging metadata for identifying local, robust multi-variate temporal (rmt) features”, in “Data Engineering (ICDE), 2014 IEEE 30th International Conference on”, pp. 388–399 (IEEE, 2014).

[71] Weisstein, E. W., “Matrix decomposition”, (2000).

[72] Witkin, A. P., “Scale-space filtering: A new approach to multi-scale description”, in “Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP’84.”, vol. 9, pp. 150–153 (IEEE, 1984).

[73] Wold, S., K. Esbensen and P. Geladi, “Principal component analysis”, Chemometrics and intelligent laboratory systems **2**, 1, 37–52 (1987).

[74] Wolski, R., “Dynamically forecasting network performance using the network weather service”, Cluster Computing **1**, 1, 119–132 (1998).

[75] Xing, Z., J. Pei and E. Keogh, “A brief survey on sequence classification”, ACM SIGKDD Explorations Newsletter **12**, 1, 40–48 (2010).

[76] Xu, R., D. Wunsch *et al.*, “Survey of clustering algorithms”, Neural Networks, IEEE Transactions on **16**, 3, 645–678 (2005).

[77] Yang, K. and C. Shahabi, “A pca-based similarity measure for multivariate time series”, in “Proceedings of the 2nd ACM international workshop on Multimedia databases”, pp. 65–74 (ACM, 2004).

APPENDIX A

LIST OF NOTATIONS AND ABBREVIATIONS

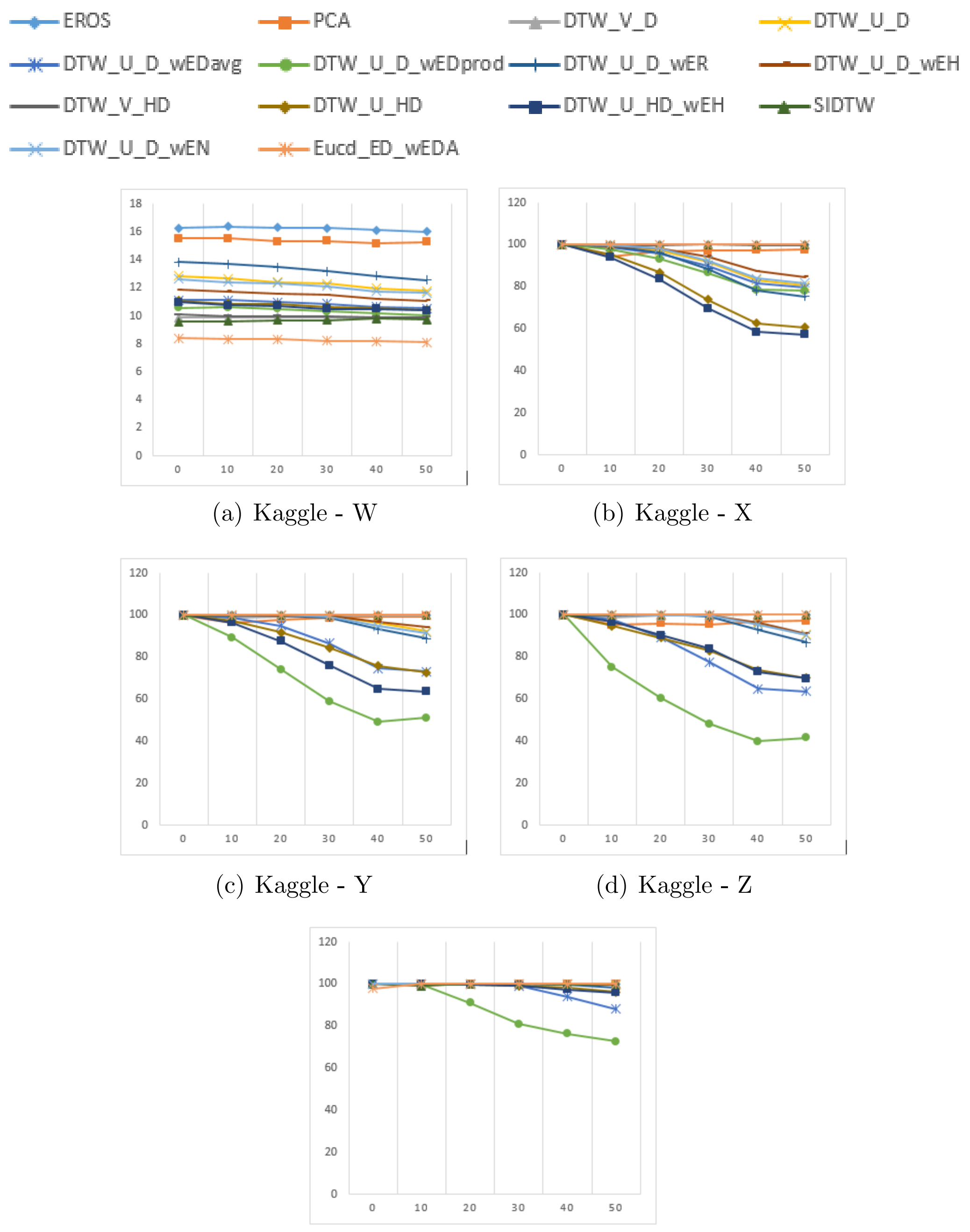
N	Length of time series	2
T	Uni-variate time series	2
d	Value of a variable at the given time instance	2
t	Time instance	2
D	Dimension of the data space	3
$\mathbb{T}$	Multi-variate time series	3
R	Relationship matrix	4
Θ	Correlation function	4
$\mathbb{T}_R$	Correlated multi-variate time series	4
H	Hierarchy Similarity Matrix	6
H_D	Variate Hierarchy	6
X	Sample time series 1	7
Y	Sample time series 2	7
$\mathbb{R}$	Set of real numbers	7
$\mathcal{D}$	Data Space	7
dis	Distance function	7
$\mathbb{S}$	Set of time series	9
$\mathcal{Q}_{\mathcal{P}}$	Query pattern contained in the query time series	8
$\mathcal{Q}$	Query time series	8
∇	Threshold condition for range query	8
p	Type of $L_P - norm$ or Minkowski Distance	11
C_{sim}	Cosine similarity	19
C_{dis}	Cosine distance	19
$\mathcal{W}$	List of weights	24
C	Reduced C-dimensional space	26

S	Set of Eigenvalues	26
U	Left Eigenvectors	26
V	Right Eigenvectors	26
f	Time series file index	26
H	Hierarhical similarity matrix	30
$\mathbb{V}$	Set of D vectors[1] or vector basis	32
x	x-coordinate of warping Point	39
y	y-coordinate of warping Point	39
CS	Cluster strength	49
C	Cluster label or cluster id	50
BS	Border separation of a cluster	50
DP	Deceptor Prevention	51

[1]also called variates

APPENDIX B

COMPREHENSIVE RESULTS



(a) Kaggle - W

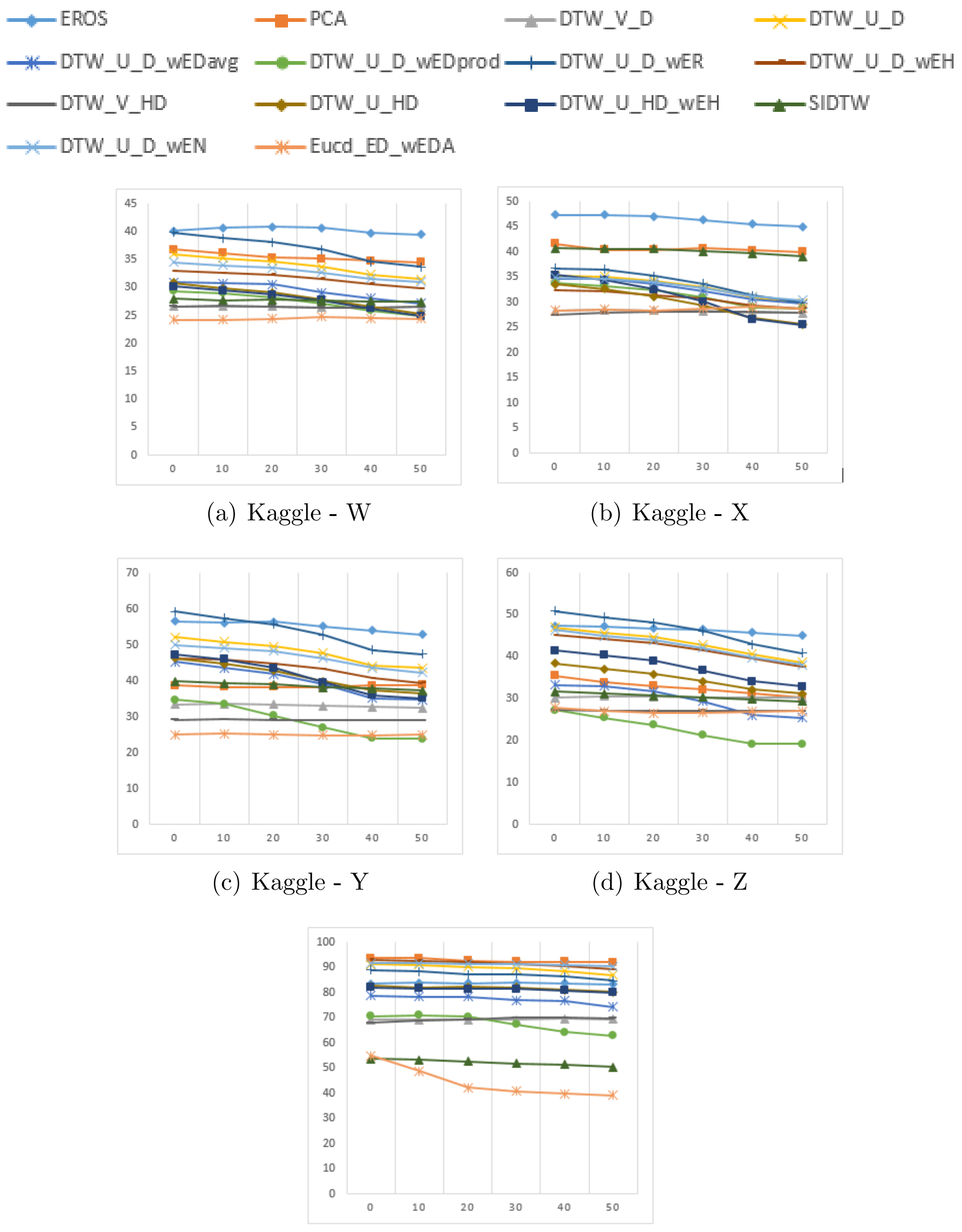
(b) Kaggle - X

(c) Kaggle - Y

(d) Kaggle - Z

(e) Mocap

Figure B.1: Random Distortion; X-axis: % Asynchrony, Y-axis: % Accuracy for Top-1



(a) Kaggle - W

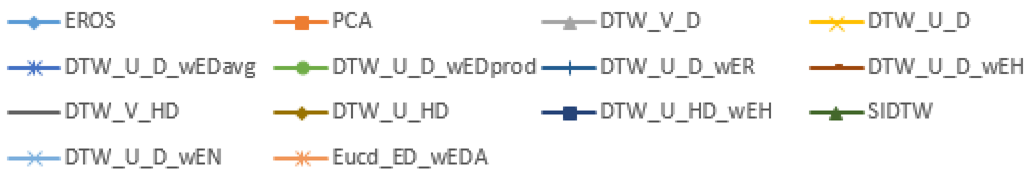
(b) Kaggle - X

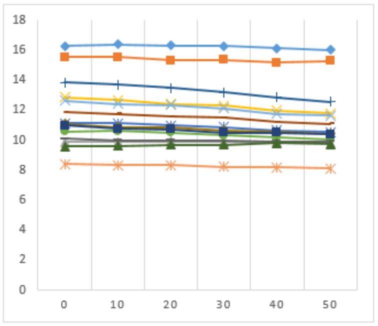
(c) Kaggle - Y

(d) Kaggle - Z

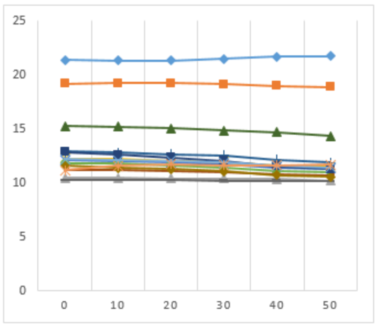
(e) Mocap

Figure B.2: Random Distortion; X-axis: % Asynchrony, Y-axis: % Accuracy for Top-5

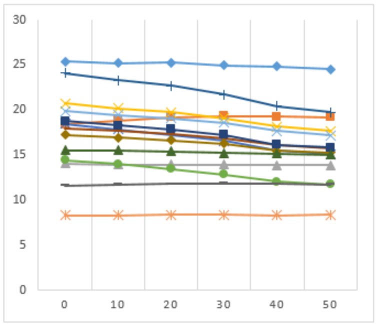




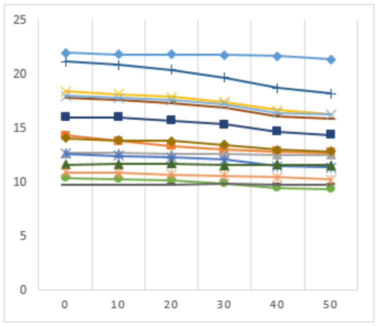
(a) Kaggle - W



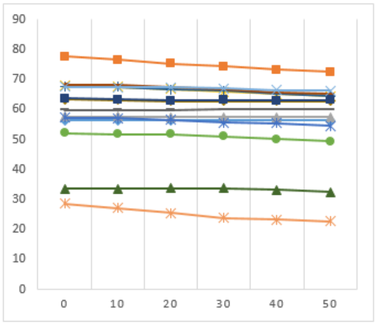
(b) Kaggle - X



(c) Kaggle - Y

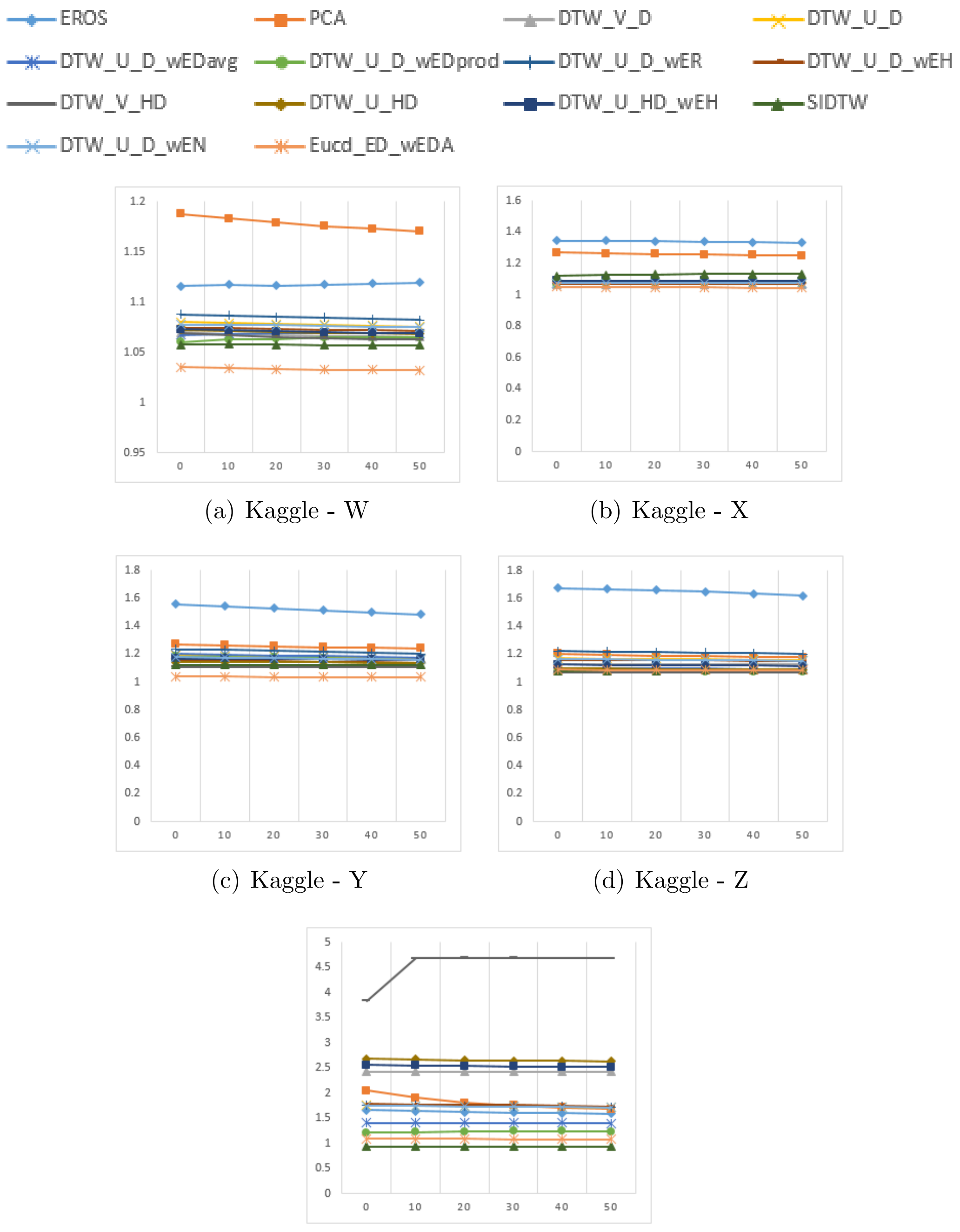


(d) Kaggle - Z



(e) Mocap

Figure B.3: Random Distortion; X-axis: % Asynchrony, Y-axis: % Accuracy for Top-k

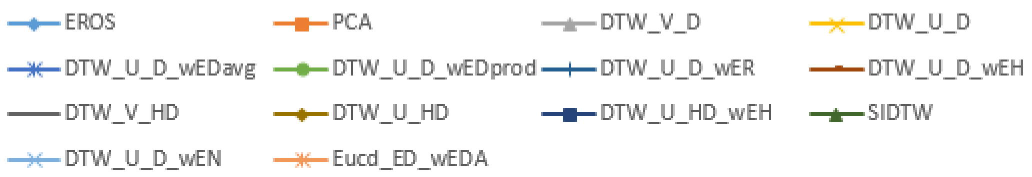


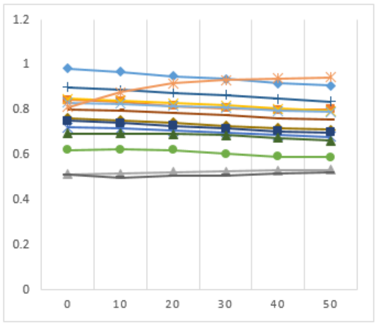
(a) Kaggle - W (b) Kaggle - X

(c) Kaggle - Y (d) Kaggle - Z

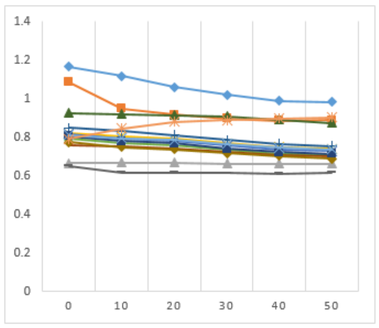
(e) Mocap

Figure B.4: Random Distortion; X-axis: % Asynchrony, Y-axis: Cluster Separation Ratio

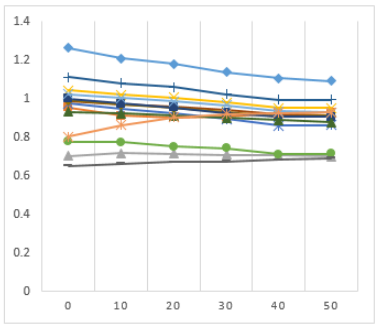




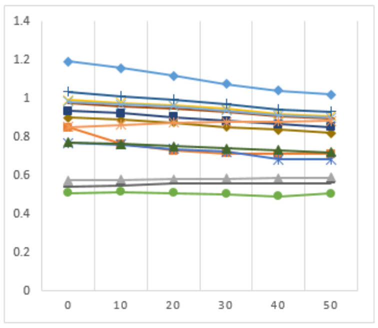
(a) Kaggle - W



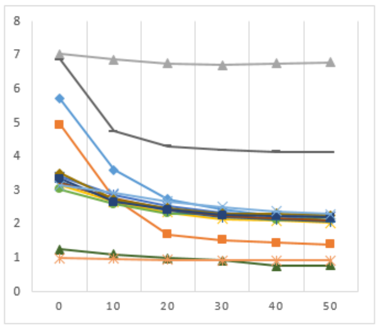
(b) Kaggle - X



(c) Kaggle - Y

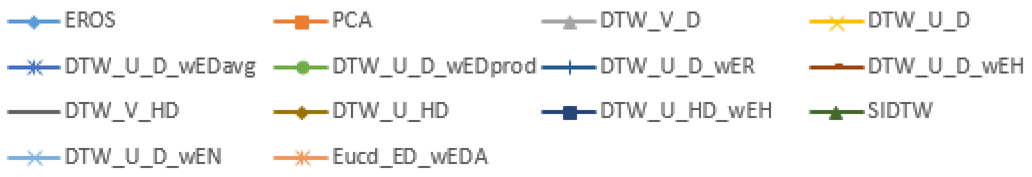


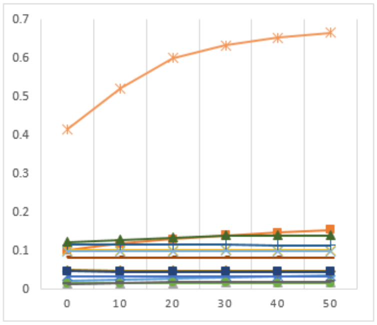
(d) Kaggle - Z



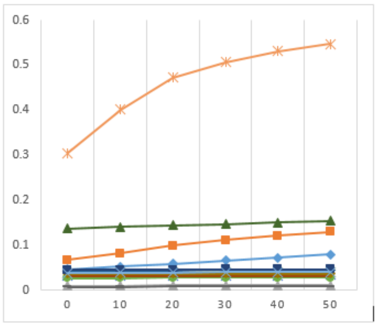
(e) Mocap

Figure B.5: Random Distortion; X-axis: % Asynchrony, Y-axis: Anti-Deception Prevention

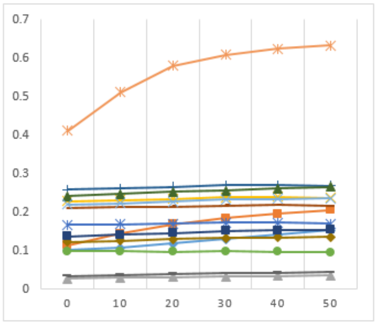




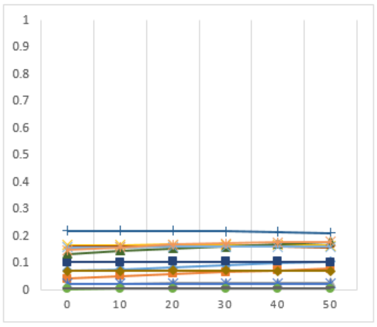
(a) Kaggle - W



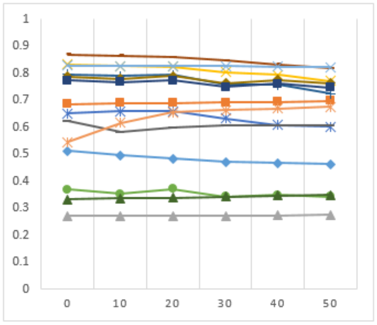
(b) Kaggle - X



(c) Kaggle - Y

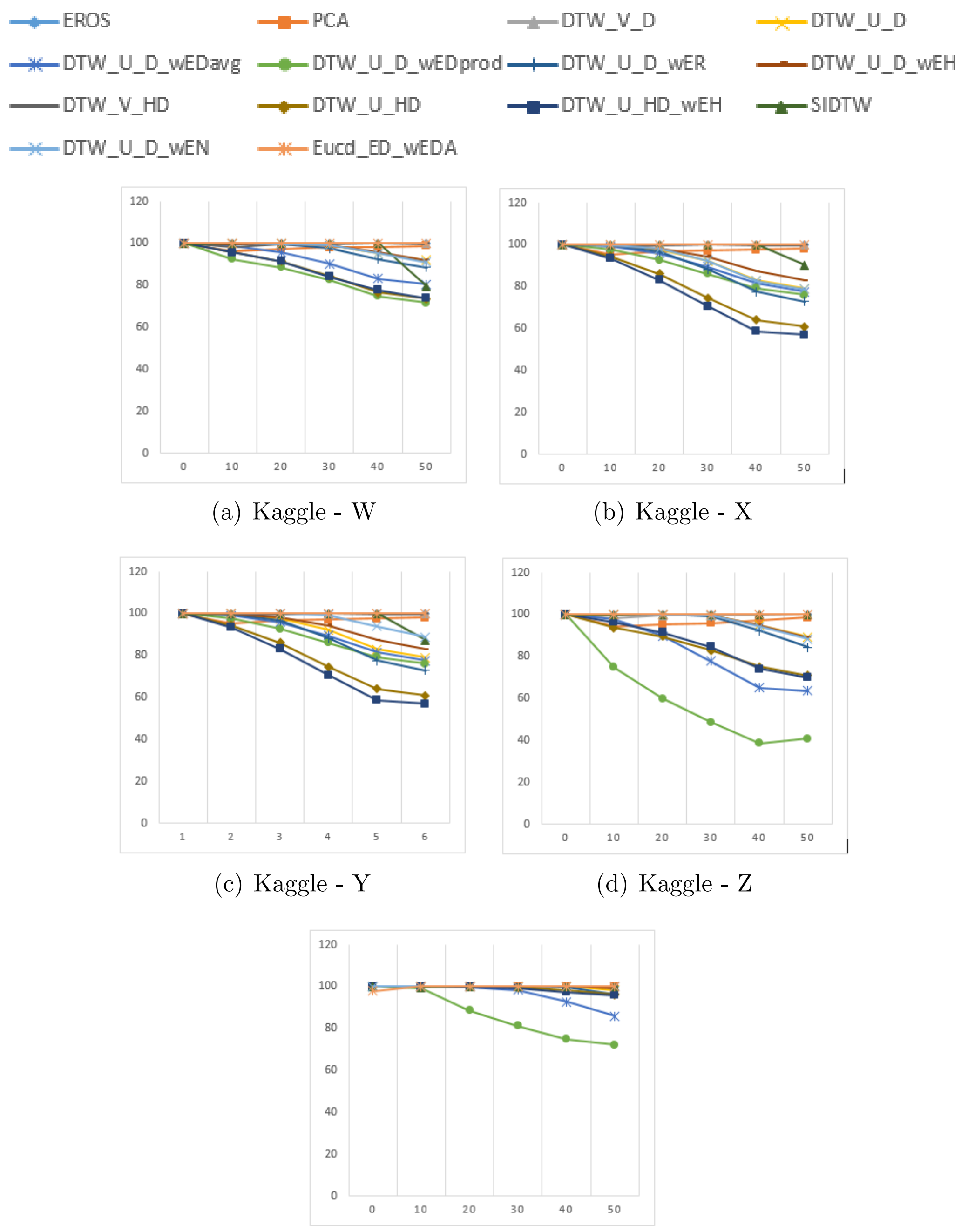


(d) Kaggle - Z



(e) Mocap

Figure B.6: Random Distortion; X-axis: % Asynchrony, Y-axis: Border Separation Ratio



(a) Kaggle - W

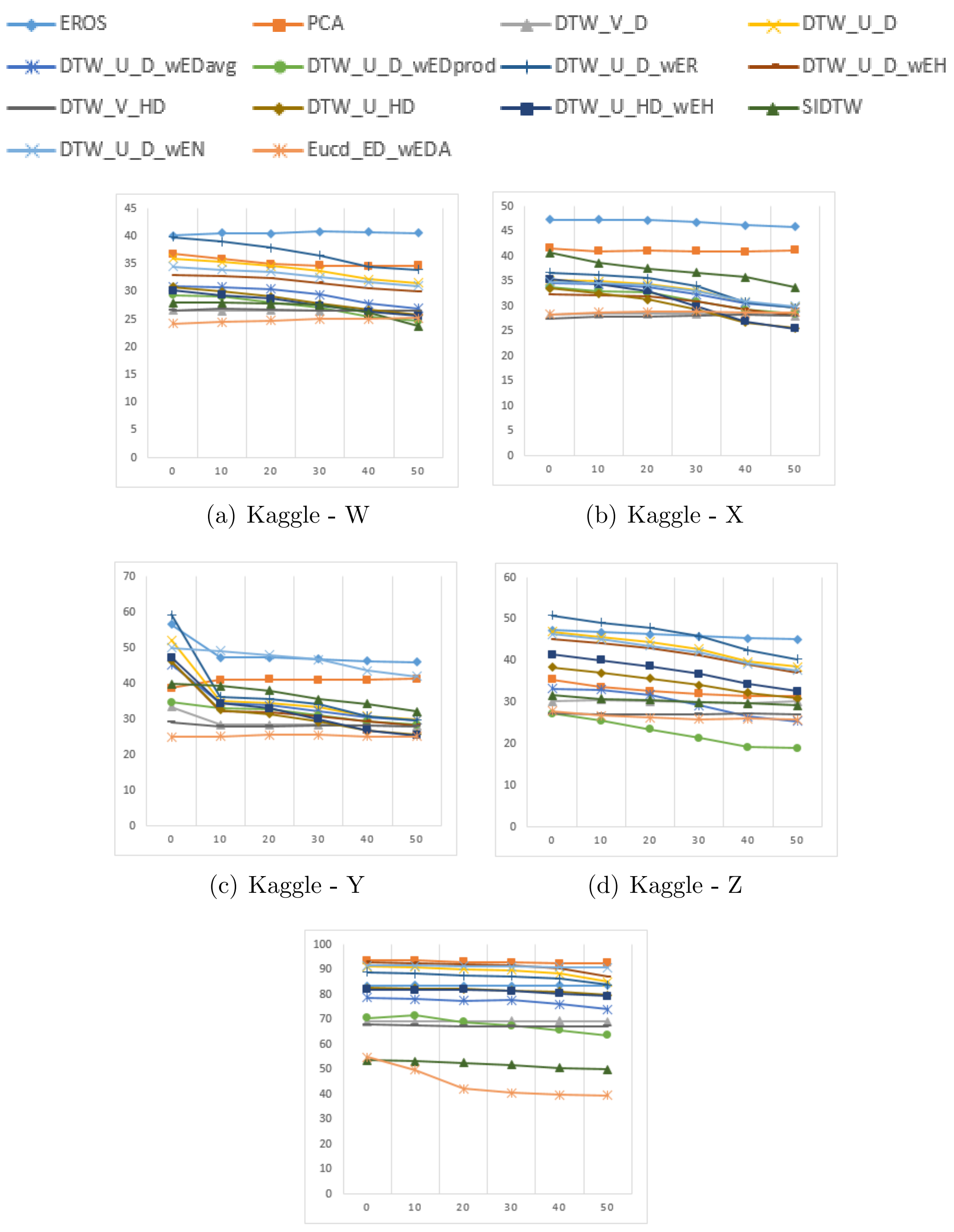
(b) Kaggle - X

(c) Kaggle - Y

(d) Kaggle - Z

(e) Mocap

Figure B.7: Inherent Distortion; X-axis: % Asynchrony, Y-axis: % Accuracy for Top-1

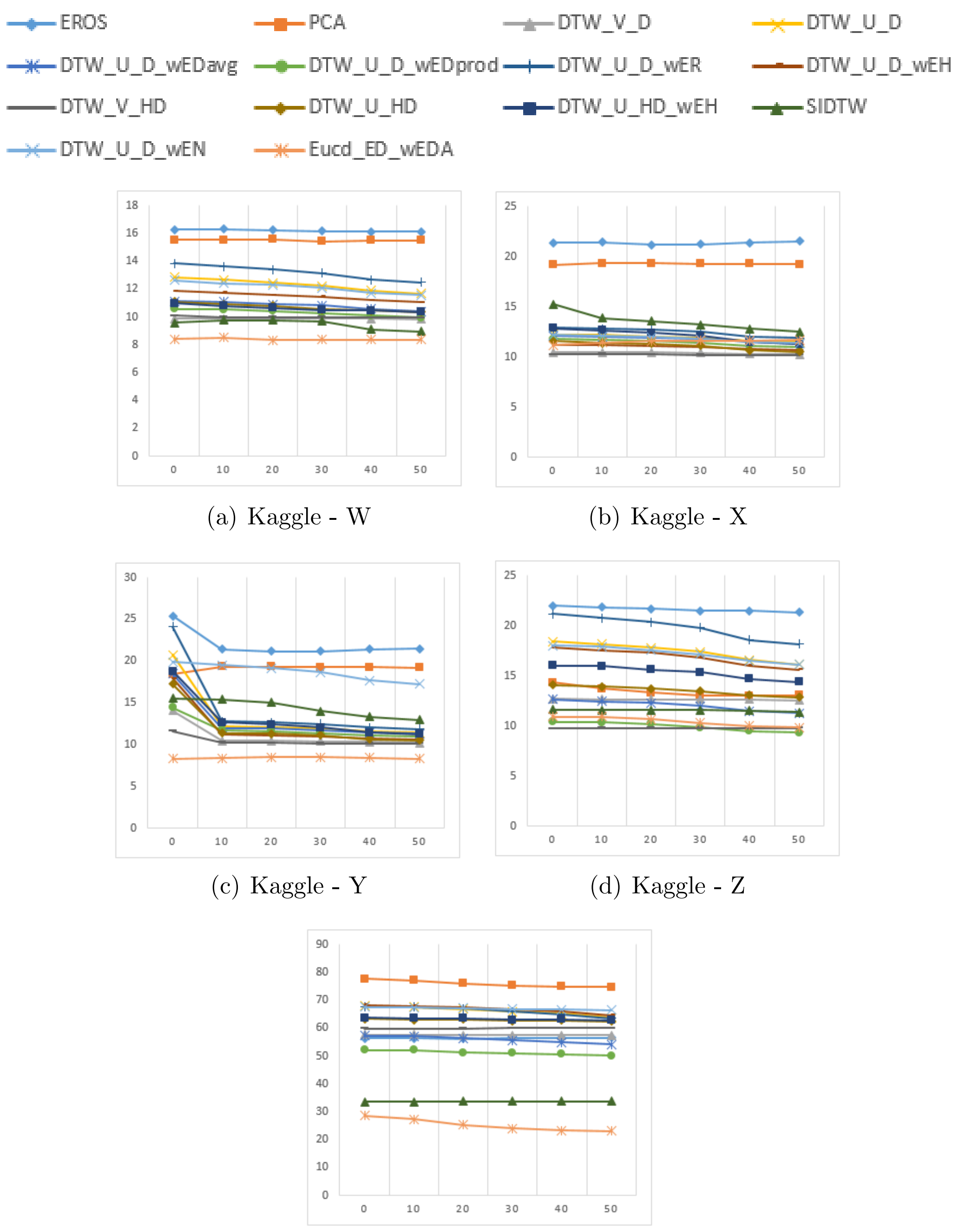


(a) Kaggle - W (b) Kaggle - X

(c) Kaggle - Y (d) Kaggle - Z

(e) Mocap

Figure B.8: Inherent Distortion; X-axis: % Asynchrony, Y-axis: % Accuracy for Top-5



(a) Kaggle - W

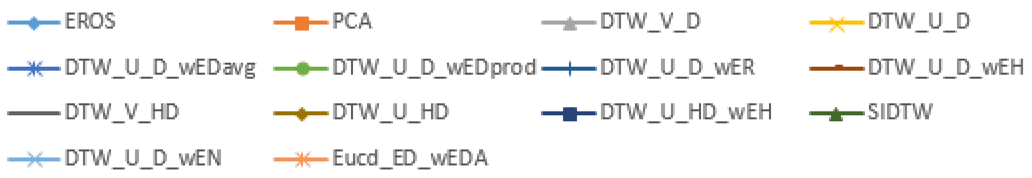
(b) Kaggle - X

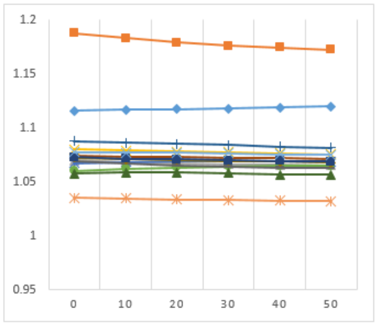
(c) Kaggle - Y

(d) Kaggle - Z

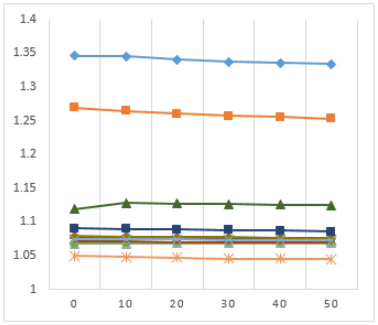
(e) Mocap

Figure B.9: Inherent Distortion; X-axis: % Asynchrony, Y-axis: % Accuracy for Top-k

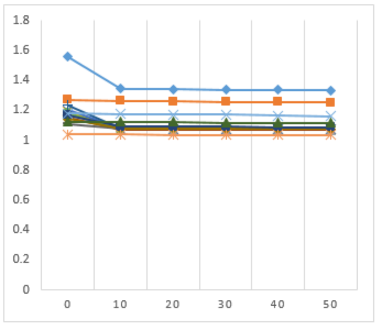




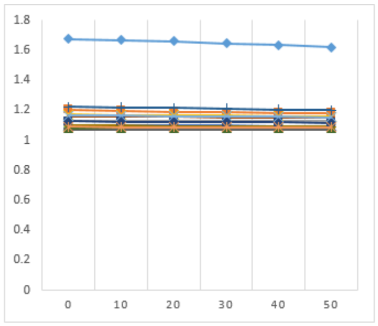
(a) Kaggle - W



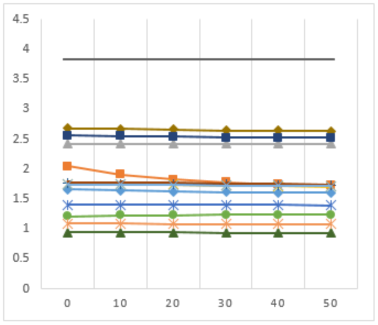
(b) Kaggle - X



(c) Kaggle - Y

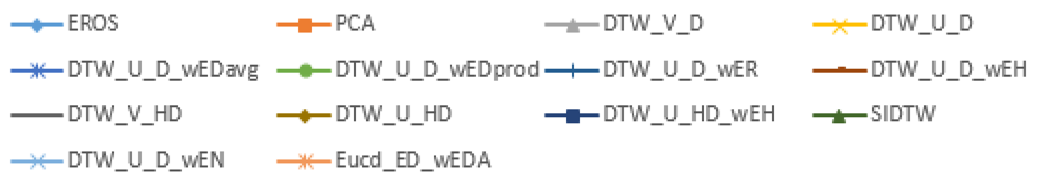


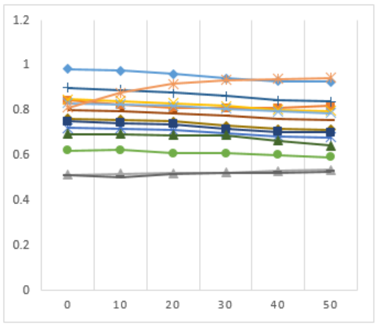
(d) Kaggle - Z



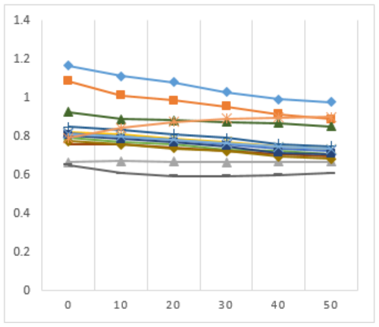
(e) Mocap

Figure B.10: Inherent Distortion; X-axis: % Asynchrony, Y-axis: Cluster Separation Ratio

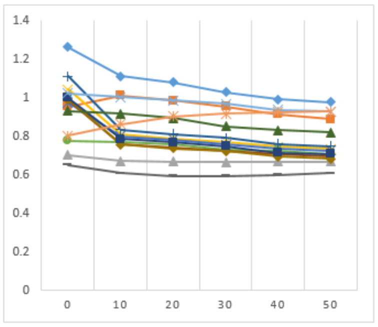




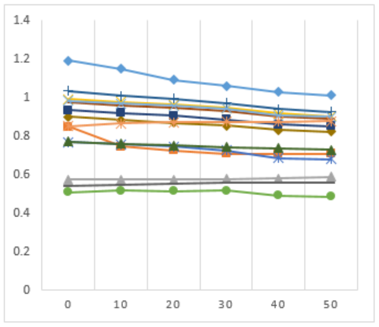
(a) Kaggle - W



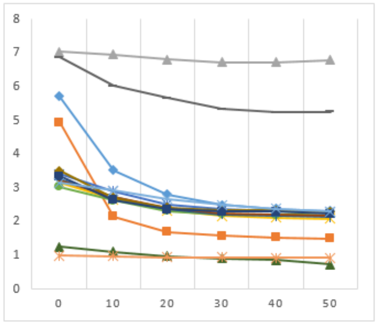
(b) Kaggle - X



(c) Kaggle - Y

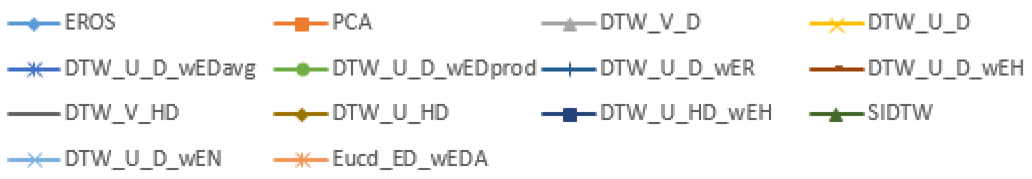


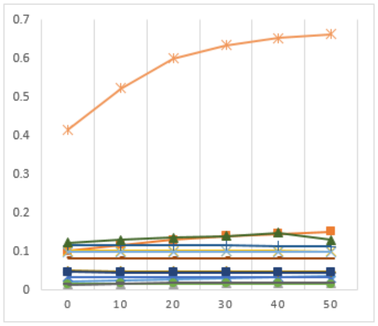
(d) Kaggle - Z



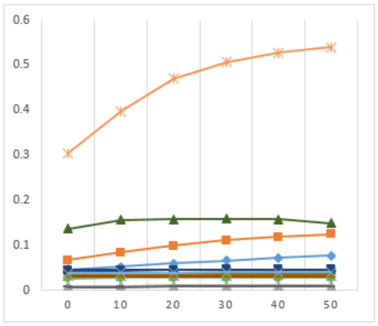
(e) Mocap

Figure B.11: Inherent Distortion; X-axis: % Asynchrony, Y-axis: Anti-Deception Prevention

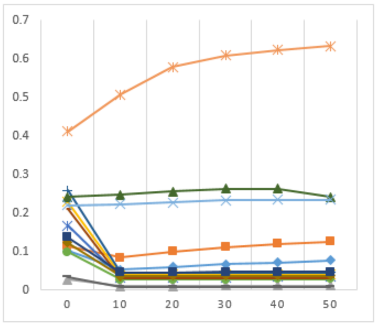




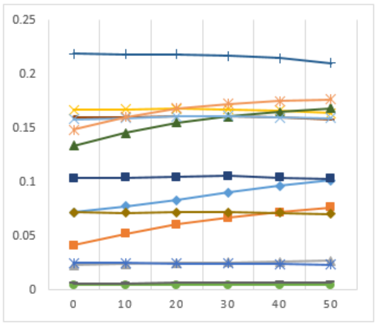
(a) Kaggle - W



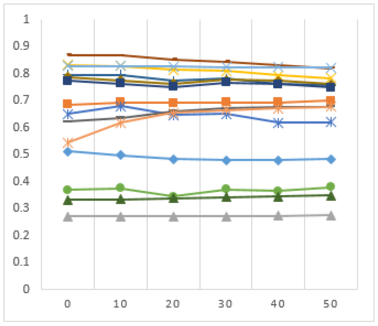
(b) Kaggle - X



(c) Kaggle - Y

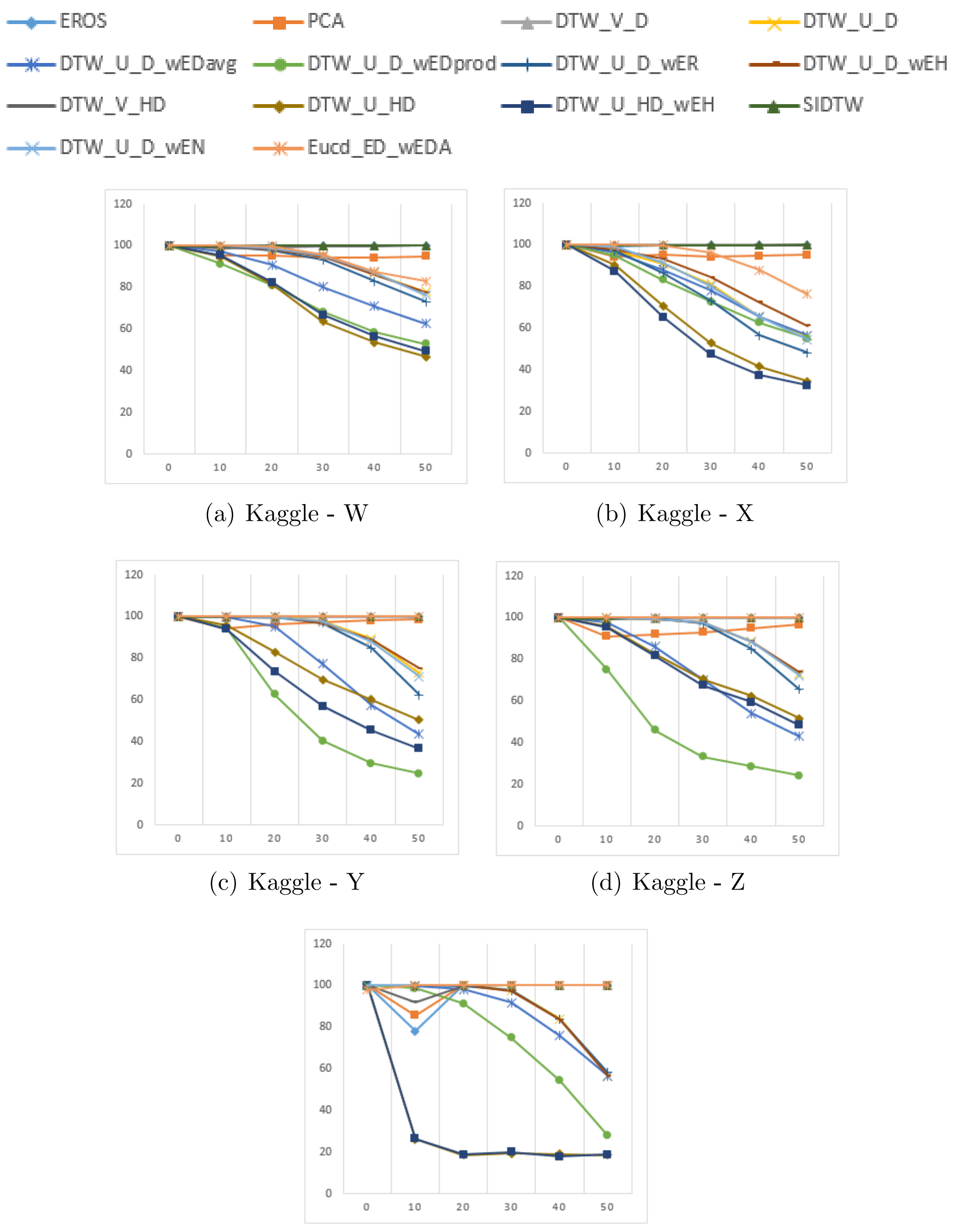


(d) Kaggle - Z



(e) Mocap

Figure B.12: Inherent Distortion; X-axis: % Asynchrony, Y-axis: Border Separation Ratio



(a) Kaggle - W

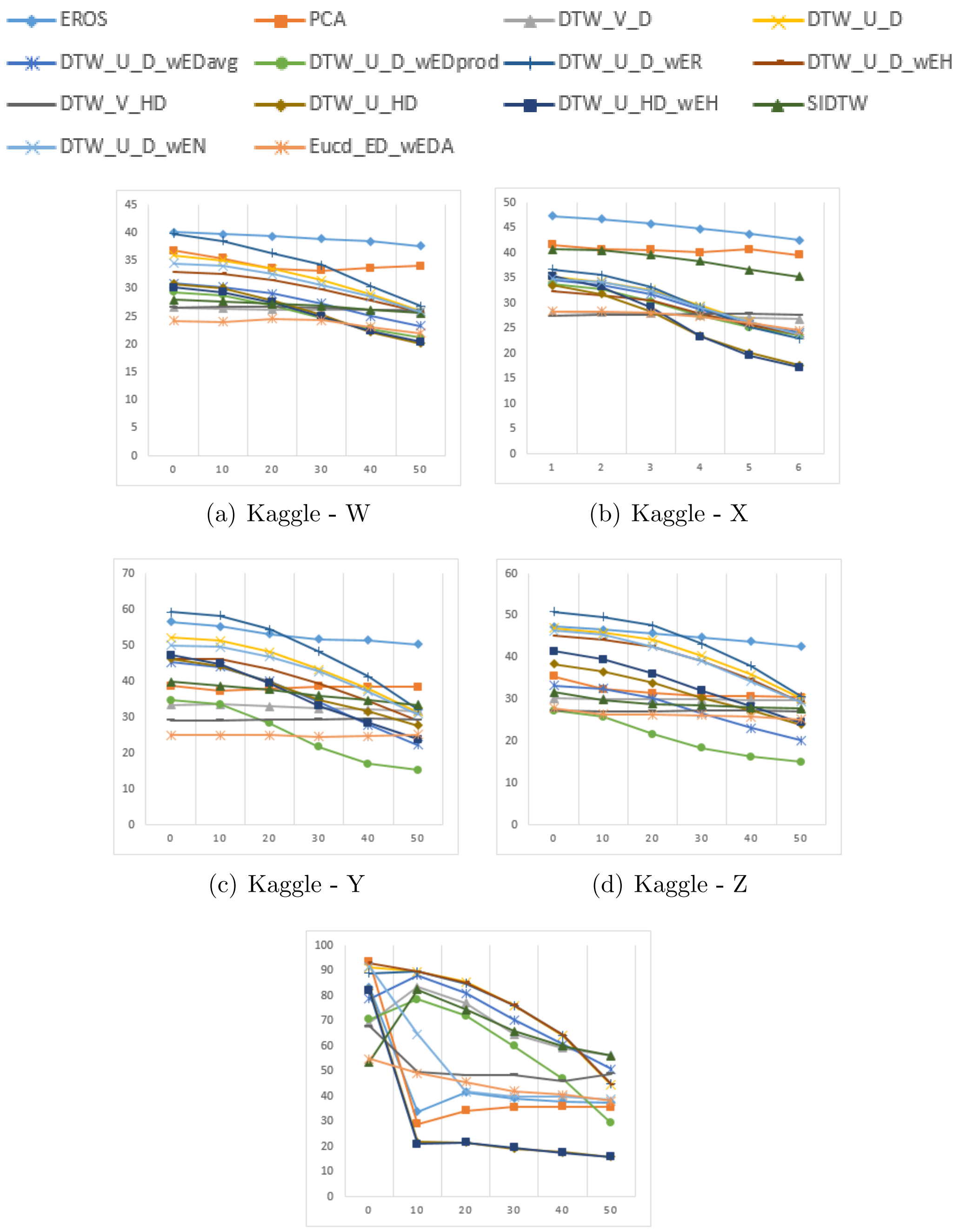
(b) Kaggle - X

(c) Kaggle - Y

(d) Kaggle - Z

(e) Mocap

Figure B.13: Random Shift; X-axis: % Asynchrony, Y-axis: % Accuracy for Top-1



(a) Kaggle - W

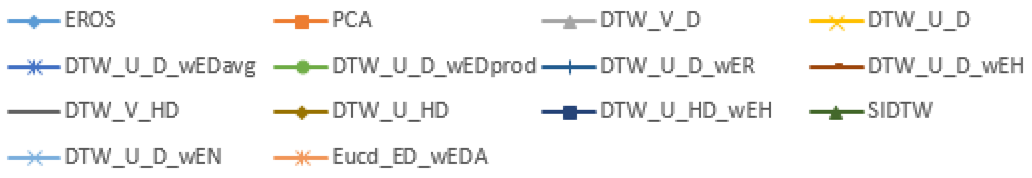
(b) Kaggle - X

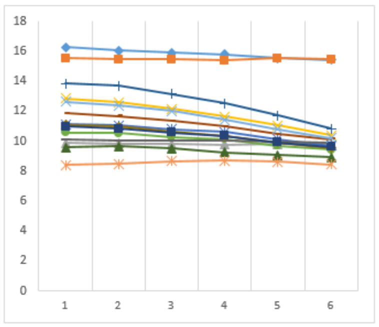
(c) Kaggle - Y

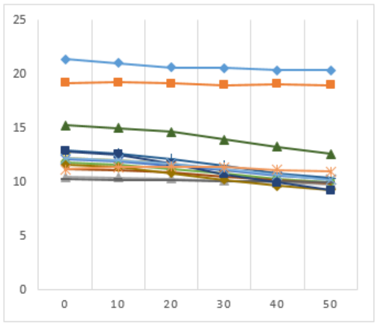
(d) Kaggle - Z

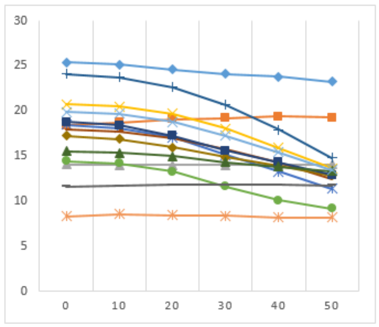
(e) Mocap

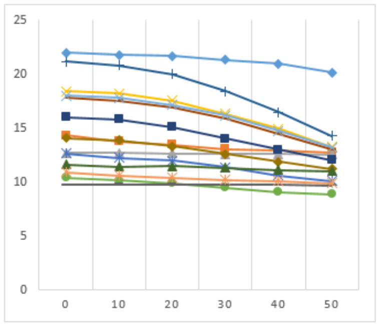
Figure B.14: Random Shift; X-axis: % Asynchrony, Y-axis: % Accuracy for Top-5




(a) Kaggle - W


(b) Kaggle - X


(c) Kaggle - Y


(d) Kaggle - Z

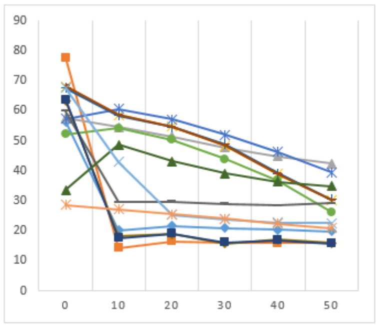
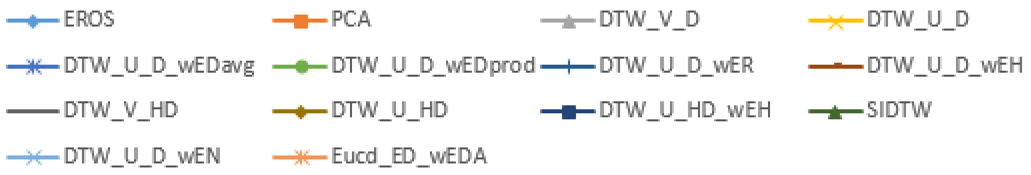
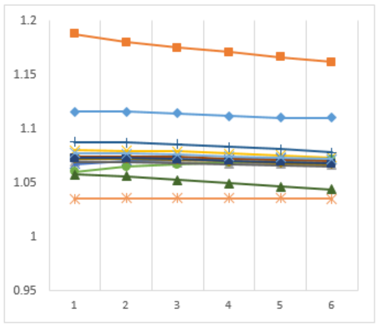
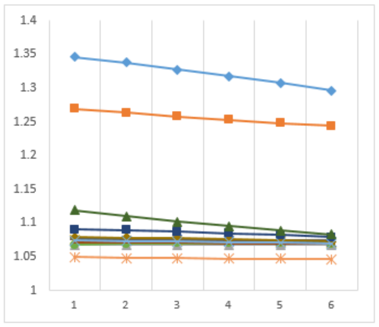

(e) Mocap

Figure B.15: Random Shift; X-axis: % Asynchrony, Y-axis: % Accuracy for Top-k

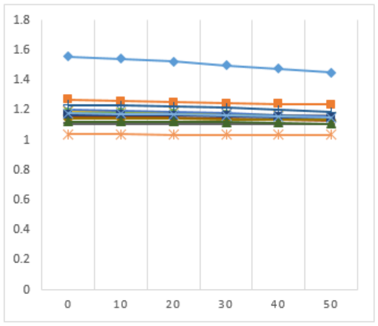




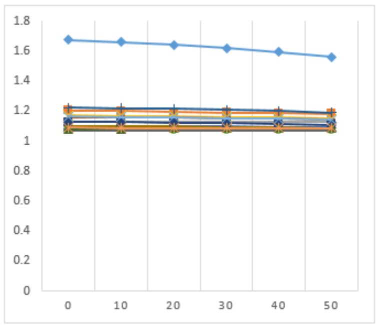
(a) Kaggle - W



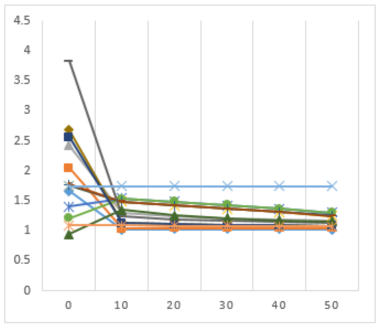
(b) Kaggle - X



(c) Kaggle - Y

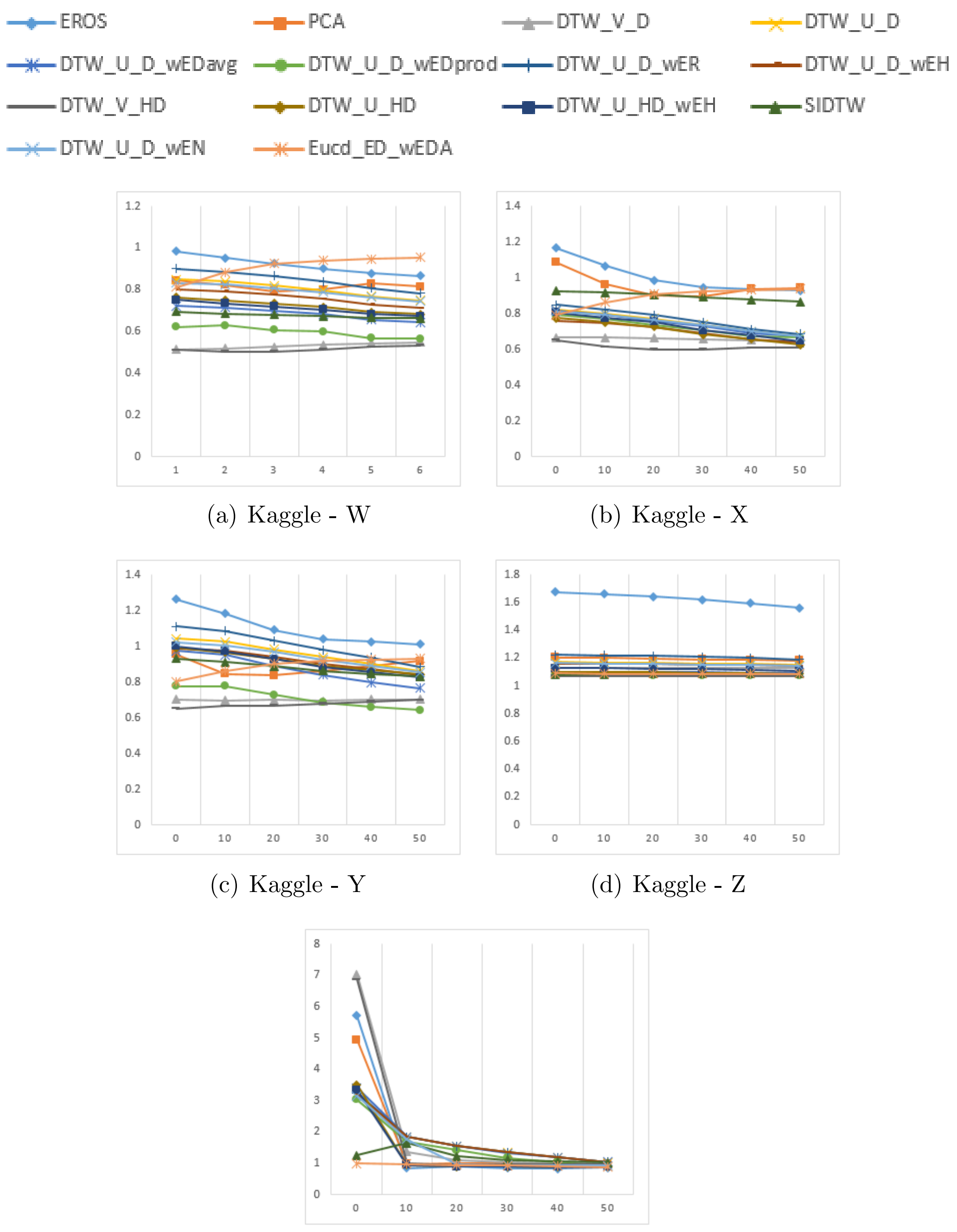


(d) Kaggle - Z



(e) Mocap

Figure B.16: Random Shift; X-axis: % Asynchrony, Y-axis: Cluster Separation Ratio



(a) Kaggle - W

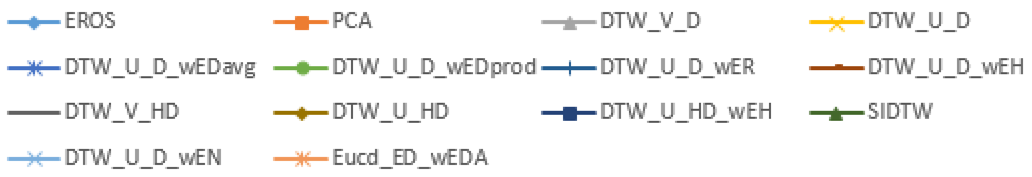
(b) Kaggle - X

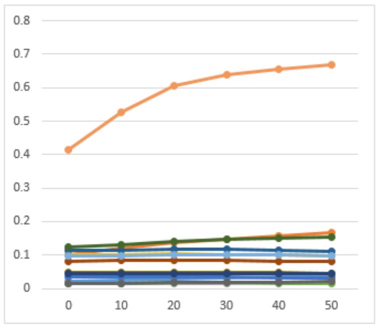
(c) Kaggle - Y

(d) Kaggle - Z

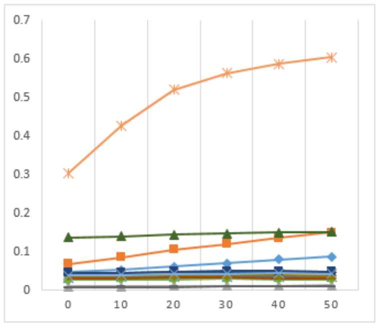
(e) Mocap

Figure B.17: Random Shift; X-axis: % Asynchrony, Y-axis: Anti-Deception Prevention

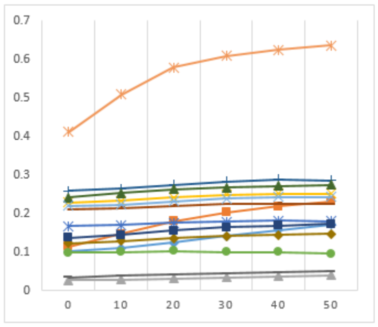




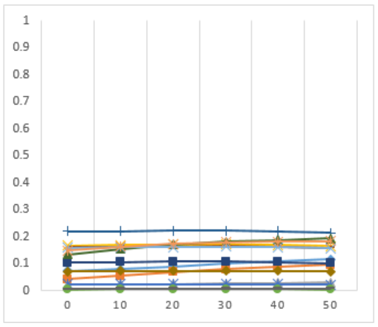
(a) Kaggle - W



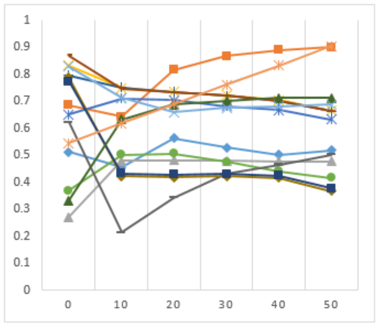
(b) Kaggle - X



(c) Kaggle - Y

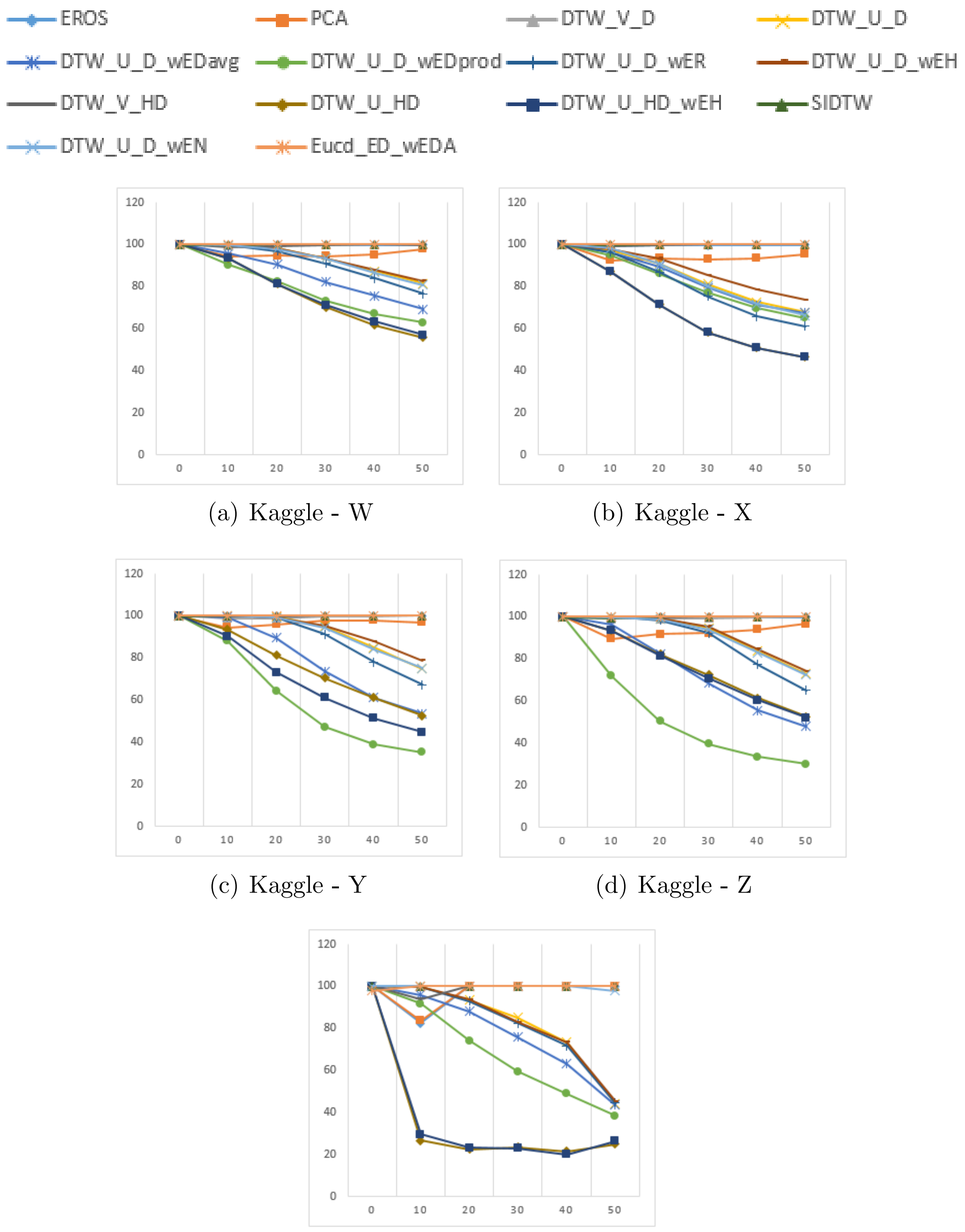


(d) Kaggle - Z



(e) Mocap

Figure B.18: Random Shift; X-axis: % Asynchrony, Y-axis: Border Separation Ratio



(a) Kaggle - W

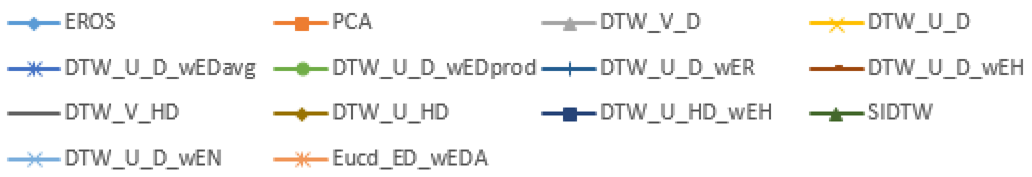
(b) Kaggle - X

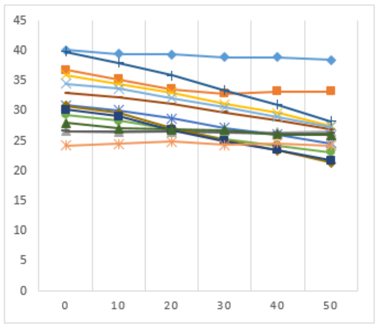
(c) Kaggle - Y

(d) Kaggle - Z

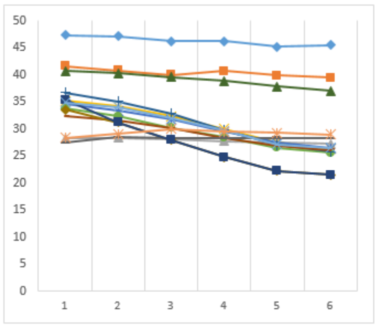
(e) Mocap

Figure B.19: Inherent Shift; X-axis: % Asynchrony, Y-axis: % Accuracy for Top-1

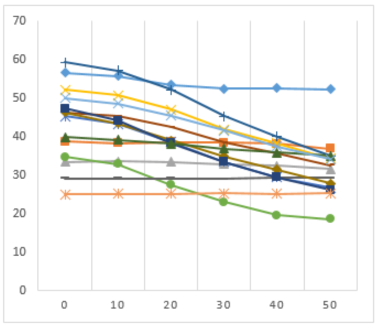




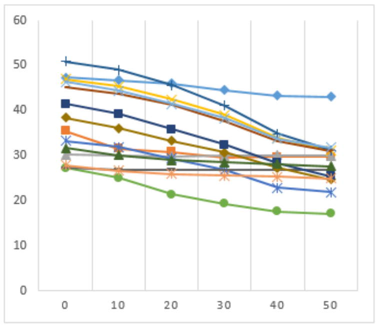
(a) Kaggle - W



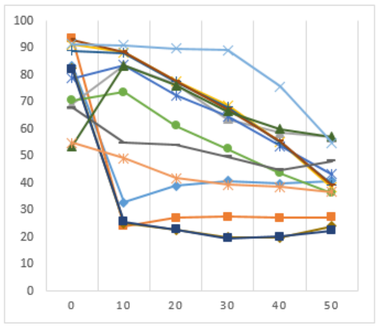
(b) Kaggle - X



(c) Kaggle - Y

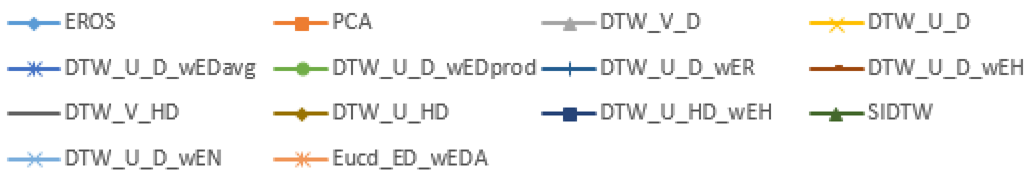


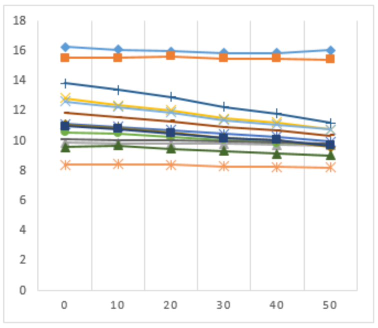
(d) Kaggle - Z

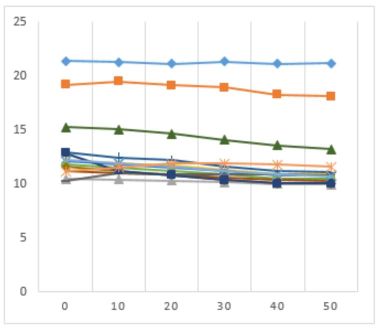


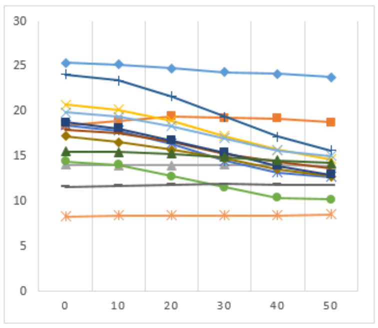
(e) Mocap

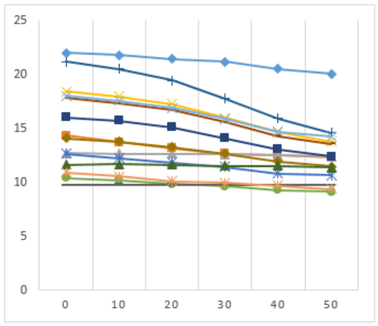
Figure B.20: Inherent Shift; X-axis: % Asynchrony, Y-axis: % Accuracy for Top-5




(a) Kaggle - W


(b) Kaggle - X


(c) Kaggle - Y


(d) Kaggle - Z

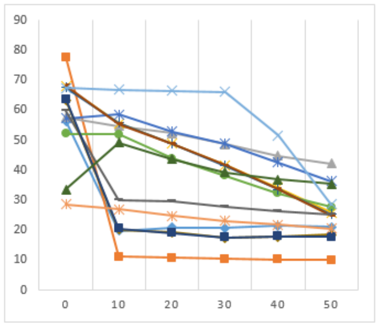
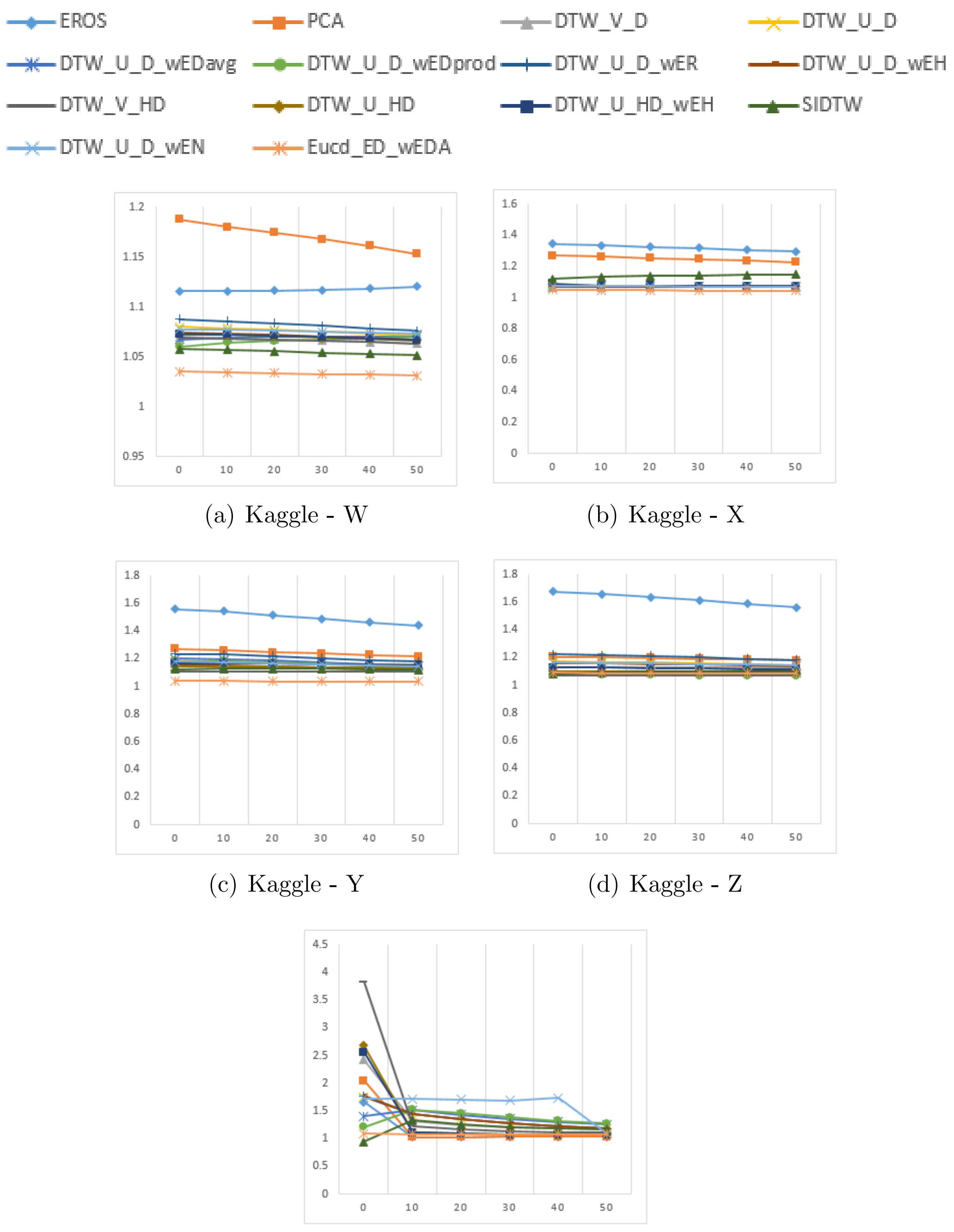

(e) Mocap

Figure B.21: Inherent Shift; X-axis: % Asynchrony, Y-axis: % Accuracy for Top-k



(a) Kaggle - W

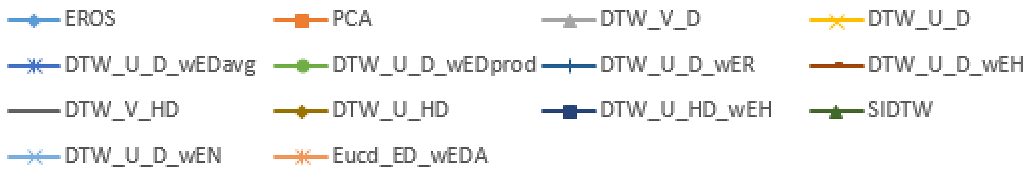
(b) Kaggle - X

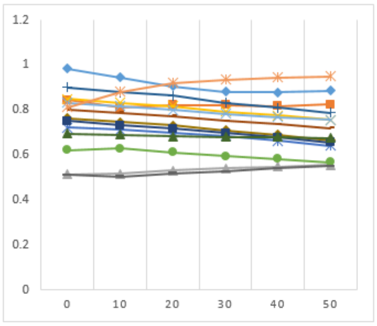
(c) Kaggle - Y

(d) Kaggle - Z

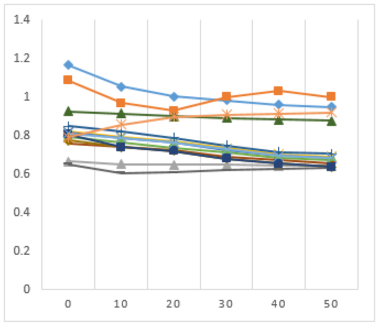
(e) Mocap

Figure B.22: Inherent Shift; X-axis: % Asynchrony, Y-axis: Cluster Separation Ratio

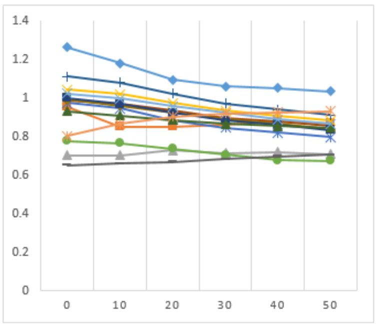




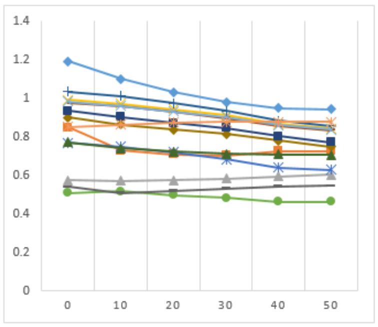
(a) Kaggle - W



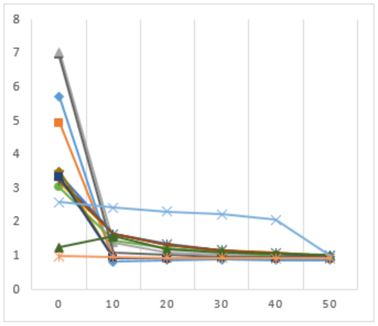
(b) Kaggle - X



(c) Kaggle - Y

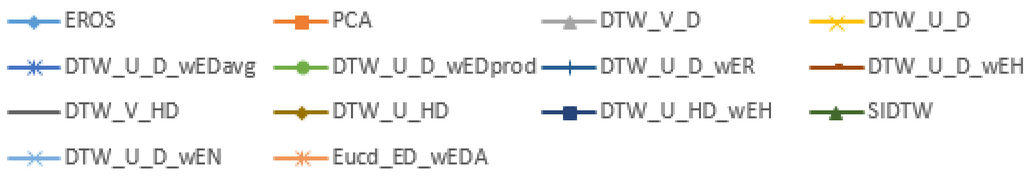


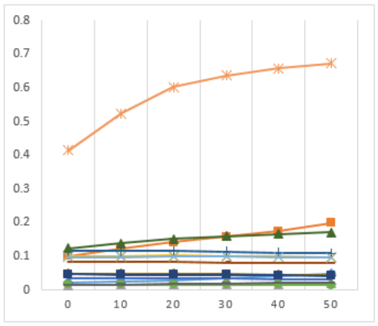
(d) Kaggle - Z



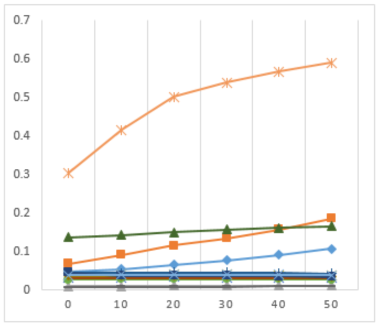
(e) Mocap

Figure B.23: Inherent Shift; X-axis: % Asynchrony, Y-axis: Anti-Deception Prevention

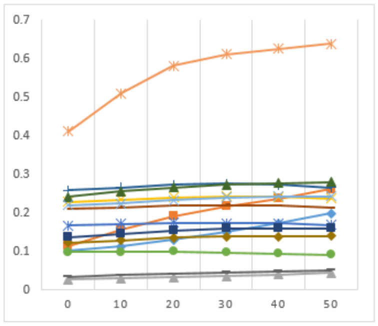




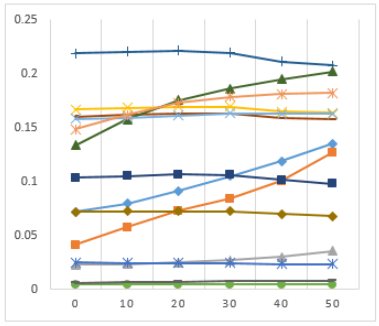
(a) Kaggle - W



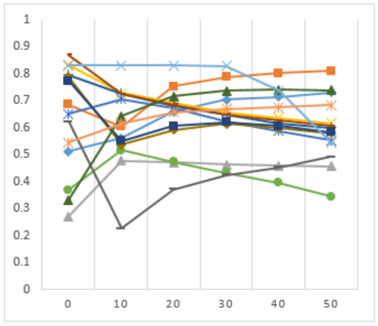
(b) Kaggle - X



(c) Kaggle - Y



(d) Kaggle - Z



(e) Mocap

Figure B.24: Inherent Shift; X-axis: % Asynchrony, Y-axis: Border Separation Ratio