# Efficient Subsequence Alignment of Time Series

A Massively Parallel Approach

## Dissertation

zur Erlangung des Grades
'Doktor der Naturwissenschaften'
am Fachbereich Physik, Mathematik und Informatik
der Johannes Gutenberg-Universität in Mainz


JOHANNES GUTENBERG
UNIVERSITÄT MAINZ

am 11. Juni 2015 vorgelegt von

**Christian Hundt**
geboren am 06. Juli 1983 in Blankenburg
Mainz, den 28. September 2015

**Berichterstatter:**

_____
[left out in the digital version]

_____
[left out in the digital version]

**Datum der mündlichen Prüfung:**

23. September 2015

# Abstract

Time series occur almost everywhere. A large variety of domains deal with the recording and processing of continuously measured data streams including all areas of natural science, medicine and finance. The vast growth of recorded data produced by automated monitoring systems or integrated sensors establishes the need for exceedingly fast algorithms in theory and practice. Hence, the focus of this thesis is the efficient computation of subsequence alignments in streams of time series. Many high-level algorithms amongst other anomaly detection, motif retrieval or the unsupervised extraction of major buildings blocks of time series make excessive use of subsequence alignments and thus there is a urgent need for fast implementations. This thesis consists of three frameworks to address this challenge. This includes four subsequence alignment algorithms and their parallelizations on CUDA-enabled devices, a stream segmentation algorithm and a unified framework for the treatment of Lie group-valued time series:

Firstly, we provide a complete port of the fastest single-threaded implementation of one-nearest-neighbour search, namely the UCR-Suite, to CUDA-enabled accelerators. The proposed implementation includes a novel computation scheme for the calculation of local alignment scores under z-normalized Euclidean distance that can be applied on any concurrent hardware that features a Fast Fourier Transform. Further, we provide an SIMT-compliant mapping of UCR-Suite's lower bound cascade for the efficient computation of local alignment scores under the Dynamic Time Warping (DTW) similarity measure. Both CUDA-parallelizations outperform the state-of-the art by one to two orders-of-magnitude.

Secondly, we investigate two linear time approximations for the elastic alignment of subsequence candidates. On the one hand, we provide an SIMT-compliant relaxation scheme for greedily relaxed DTW and its efficient parallelization on massively parallel devices. On the other hand, we introduce a novel subsequence measure, the Gliding Elastic Match (GEM), that can be computed with the same asymptotic complexity as greedily relaxed DTW but in contrast offers a full relaxation of the penalty matrix. Further advantages of the GEM approach include invariance against time-dependent trends on the measurement-axis and invariance against uniform scaling of the time-axis. Moreover, an extension of GEM to supervised multi-shape segmentation of streams is provided and evaluated on motion-tracking data. Both CUDA-parallelizations feature speedups of up to two orders-of-magnitude in comparison to sequential code.

Thirdly, the treatment of time series is usually limited to real-valued data in the literature. We provide a unified approach to handle Lie group-valued time series. Building upon this, notions of similarity in the group of rotations $SO(3,\mathbb{R})$ and rigid transformations $SE(3,\mathbb{R})$ are established. Furthermore, memory-efficient representations and extensions of elastic measures to handle group elements are discussed. Finally, we verify our theoretical ideas on motion sensor data.

# Zusammenfassung

Zeitreihen sind allgegenwärtig. Die Erfassung und Verarbeitung kontinuierlich gemessener Daten ist in allen Bereichen der Naturwissenschaften, Medizin und Finanzwelt vertreten. Das enorme Anwachsen aufgezeichneter Datenmengen, sei es durch automatisierte Monitoring-Systeme oder integrierte Sensoren, bedarf außerordentlich schneller Algorithmen in Theorie und Praxis. Infolgedessen beschäftigt sich diese Arbeit mit der effizienten Berechnung von Teilsequenzalignments. Komplexe Algorithmen wie z. B. Anomaliedetektion, Motivfabfrage oder die unüberwachte Extraktion von prototypischen Bausteinen in Zeitreihen machen exzessiven Gebrauch von diesen Alignments. Darin begründet sich der Bedarf nach schnellen Implementierungen. Diese Arbeit untergliedert sich in drei Ansätze, die sich dieser Herausforderung widmen. Das umfasst vier Alignierungsalgorithmen und ihre Parallelisierung auf CUDA-fähiger Hardware, einen Algorithmus zur Segmentierung von Datenströmen und eine einheitliche Behandlung von Liegruppen-wertigen Zeitreihen.

Der erste Beitrag ist eine vollständige CUDA-Portierung der UCR-Suite, die weltführende Implementierung von Teilsequenzalignierung. Das umfasst ein neues Berechnungsschema zur Ermittlung lokaler Alignierungsgüten unter Verwendung z-normierten euklidischen Abstands, welches auf jeder parallelen Hardware mit Unterstützung für schnelle Fouriertransformation einsetzbar ist. Des Weiteren geben wir eine SIMT-verträgliche Umsetzung der Lower-Bound-Kaskade der UCR-Suite zur effizienten Berechnung lokaler Alignierungsgüten unter Dynamic Time Warping an. Beide CUDA-Implementierungen ermöglichen eine um ein bis zwei Größenordnungen schnellere Berechnung als etablierte Methoden.

Als zweites untersuchen wir zwei Linearzeit-Approximierungen für das elastische Alignment von Teilsequenzen. Auf der einen Seite behandeln wir ein SIMT-verträgliches Relaxierungschema für Greedy DTW und seine effiziente CUDA-Parallelisierung. Auf der anderen Seite führen wir ein neues lokales Abstandsmaß ein, den Gliding Elastic Match (GEM), welches mit der gleichen asymptotischen Zeitkomplexität wie Greedy DTW berechnet werden kann, jedoch eine vollständige Relaxierung der Penalty-Matrix bietet. Weitere Verbesserungen umfassen Invarianz gegen Trends auf der Messachse und uniforme Skalierung auf der Zeitachse. Des Weiteren wird eine Erweiterung von GEM zur Multi-Shape-Segmentierung diskutiert und auf Bewegungsdaten evaluiert. Beide CUDA-Parallelisierung verzeichnen Laufzeitverbesserungen um bis zu zwei Größenordnungen.

Die Behandlung von Zeitreihen beschränkt sich in der Literatur in der Regel auf reellwertige Messdaten. Der dritte Beitrag umfasst eine einheitliche Methode zur Behandlung von Liegruppen-wertigen Zeitreihen. Darauf aufbauend werden Distanzmaße auf der Rotationsgruppe $SO(3,\mathbb{R})$ und auf der euklidischen Gruppe $SE(3,\mathbb{R})$ behandelt. Des Weiteren werden speichereffiziente Darstellungen und gruppenkompatible Erweiterungen elastischer Maße diskutiert.

# CONTENTS

CHAPTER 1

# INTRODUCTION

*'Time series are a ubiquitous and increasingly prevalent type of data.'* — Eamonn Keogh, *UC Riverside*

Time series occur almost everywhere. A large variety of domains deal with the recording and processing of continuously measured data streams including all areas of natural science, medicine and finance. Trajectories, electrocardiograms (ECG) or stock prices are specific examples in real-world applications. The widespread adoption of automated monitoring systems in particular mobile devices and their integrated sensors has resulted in a vast growth of recorded data. Figure 1.1 depicts four further examples for time series in daily life.

Mathematically speaking, time series can be characterized as follows:

**Definition 1** ($M$-valued time series on real domain)**.** *Let $M$ be a set, $C : \mathbb{R} \to M$ be a function on real domain and $T = \{t_0, t_1, \ldots, t_i, \ldots\}$ a countable subset of $\mathbb{R}$ then the graph $\Gamma := \{(t_0, C(t_0)), (t_1, C(t_1)), \ldots, (t_i, C(t_i)), \ldots\}$ of the restriction $C|_T : T \to M$ is called an **$M$-valued time series**.*

The subset $T$ canonically inherits the total order property from $\mathbb{R}$ and thus we can fix the convention $t_0 < t_1 < \cdots < t_i < \ldots$ for the involved time stamps $t_i$ of the individual observations $C(t_i)$. Roughly speaking, a time series is a measurement of values over a predefined time grid. The measurements $C(t_i) \in \mathrm{im}(C|_T)$ can take values from any mathematical set $M$. Usually, $M$ is a subset of the real numbers e.g. when recording scalar physical quantities like temperatures or

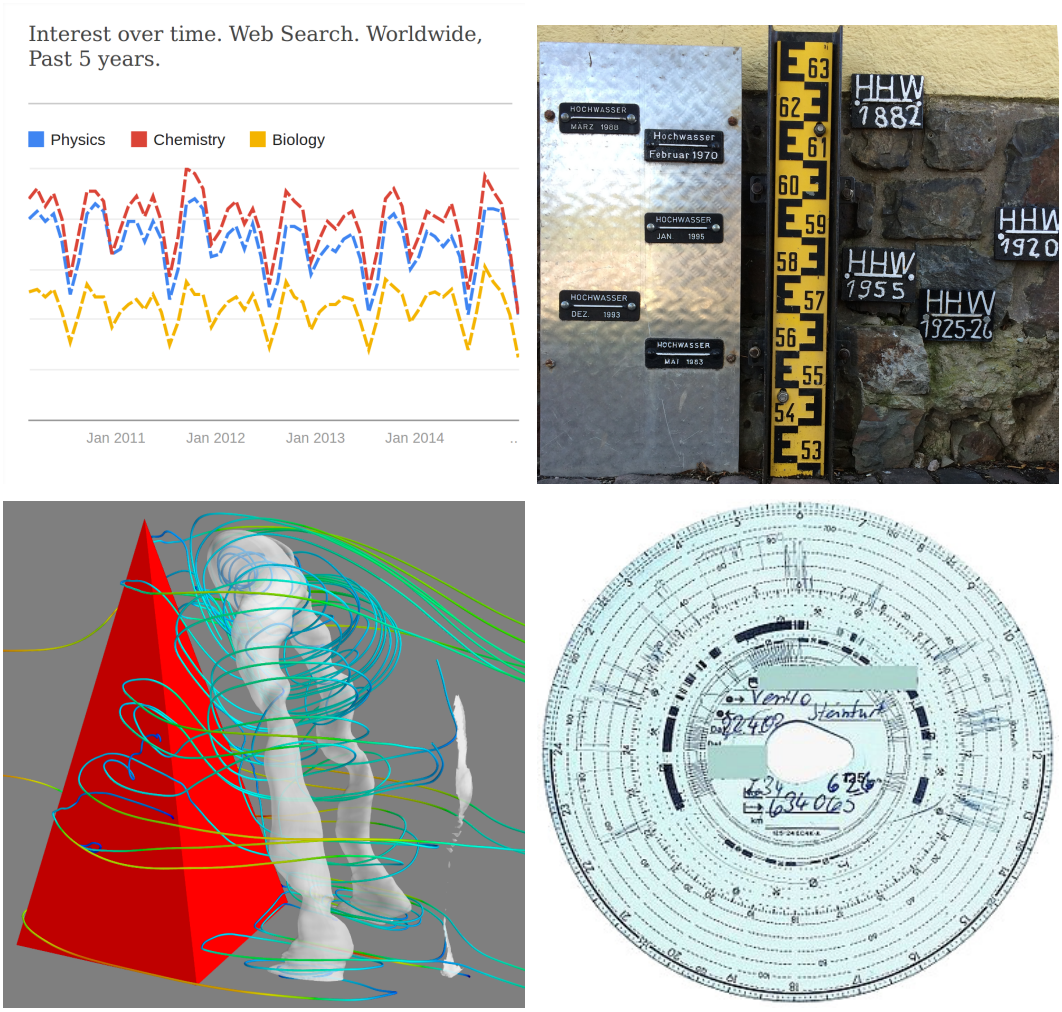


Figure 1.1: Four examples for the occurrence of time series in daily life. The upper left panel depicts the search interest for the three scientific fields 'physics', 'chemistry' and 'biology' as stated by the google trends engine [1]. Similar graphs can be obtained for 'math' and 'computer science'. Obviously, the three graphs are strongly correlated and could be used as an indicator function for lecture periods at universities. The upper right panel illustrates historic signs for the registration of high tides of the river Rhine in Bingen (Germany). Each sign corresponds to one entry in a time series of water levels. Trajectories in any dimension can be described as a mapping of events on the time-axis onto spatial coordinates and thus can be reinterpreted as time series. An example for trajectories gained during the integration of the wind field for a pyramidal orography is given in the lower left panel. The circular orbits of particles in the lee of a mountain are an important indicator for potential banner cloud formation. The last panel depicts a tachograph disk (image gratefully provided by [2]) used in trucks on German roads for the monitoring of the vehicles' velocity. The records are used for security reasons by law enforcement officials to ensure proper rest periods of the drivers.

voltages. However, $M$ could also be chosen to be a high-dimensional vector space (e.g. video data where $M = \mathbb{R}^{\text{width}\times\text{height}}$), a differentiable manifold (e.g. orientations on the unit sphere $M = \mathscr{S}^2$), a Lie-group (e.g. rigid transformation in Euclidean space $M = SE(3,\mathbb{R})$) or a wild mix of nominal and continuous features (e.g. state space of a car). In practice, one can even find time series of time series e.g. seasons of a TV show as time series of episodes which themselves are time series of images – yet one could define a time series of seasons.

All mentioned examples for time series exhibit a one-dimensional domain. Depending on the use case, extensions to higher-dimensional domains are conceivable but will not be covered by this thesis. Nevertheless, time series can be produced from higher-dimensional data during the extraction of feature representations e.g. fish shapes [3] or histogram generation from manuscripts [4]. To achieve that, one-dimensional submanifolds embedded in high-dimensional data are parametrized by an external parameter $t$. Exemplary, one could extract an iris from an image of a face by applying a Hough transformation [5] right after the processing of an edge detection algorithm. Consequently, the external parameter $t$ is given by the spherical angle and the values could be determined by the intensity of the image at the computed contour (see Figure 1.2). These time series are sometimes called 'pseudo time series' but can be handled in the same manner as conventional time series since they obey Definition 1.

Recalling the iris data example, new questions arise such as: 'Can we use the time series representation to distinguish different individuals by just taking an image of their iris?' and 'Given a newly measured 3-tuple of time series, can we efficiently determine the corresponding entry in a database of iris data in order to design an authentication mechanism?' or 'Assume we can efficiently compute a correspondence between time series of iris data, how robust is the procedure? Can we offer quality scores or certainty measures?' Time series data mining is all about answering these questions. Generalizing the workflow of a typical time series data mining application, we can roughly separate it into three stages:

**Stage 1: Find a Proper Representation**

Dealing with streamed sensor data, the format of the time series is usually given by sequences of scalar values annotated with time stamps. Depending on the task, this raw representation may be more or less suitable. On the one hand, the characteristics of the measured event could only be contained by a subset of the time stamps. Considering the whole data stream including potentially meaningless features could mitigate the quality of our model. Recent research demonstrated that ignoring reasonable amounts of the measured data may be beneficial for the data mining task [6]. On the other hand, preprocessing the time series amongst others offset removal, equalization of amplitudes, smoothing, quantization of the measurement-axis [7], transformation to other coordinate

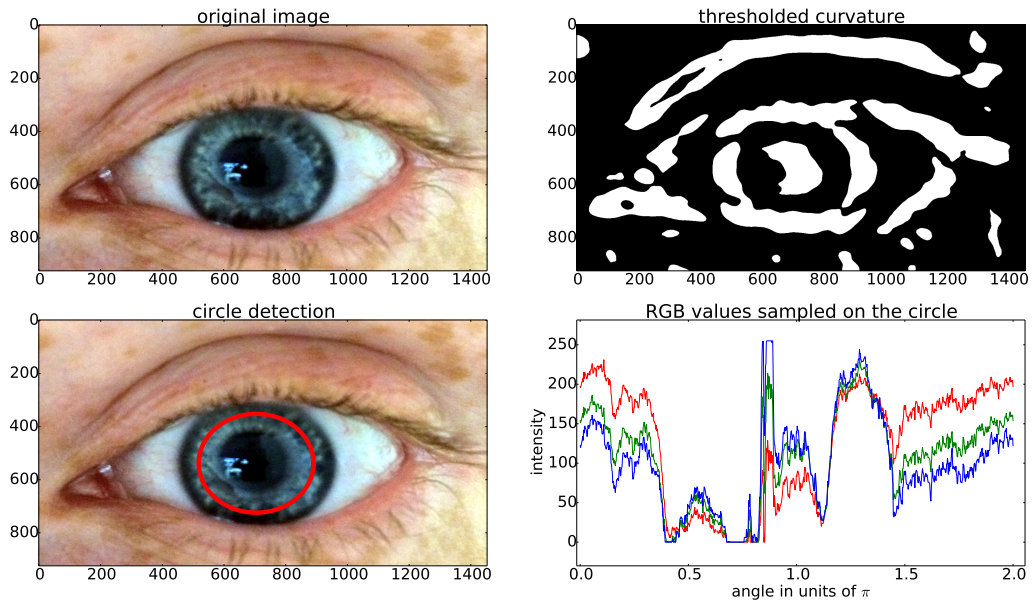


Figure 1.2: Exemplary workflow for the computation of three pseudo time series from two-dimensional image data (upper left panel). Firstly, regions of high curvature are isolated by thresholding. The curvature is determined by a convolution of the original image's gray-scale version with the second derivative of a Gaussian kernel with appropriately chosen width (upper right panel). Secondly, the center and radius of the dominating circle can be processed using a Hough transformation. A parametrization of the resulting circle has been plotted directly onto the original image (lower left panel). Finally, three time series can be extracted by sampling the red, green and blue intensities of the RGB image at the coordinates of the obtained one-dimensional submanifold $\mathscr{S}^1$ (last panel).

systems (e.g. polar coordinates) and the calculation of intermediate quantities (e.g. angular momentum from radius and momentum) may increase the quality of the model. As a result, a major portion of the time is spent on finding a suitable representation of the data.

### Stage 2: Determine a Notion of Similarity

When measuring scalar values on metric manifolds e.g. displacements on the x-axis one can directly assign a notion of similarity between two values $C(t_i)$ and $C'(t_i)$ by simply computing the time-local distance $d\big(C(t_i), C'(t_i)\big)$. In contrast, when dealing with multiple channels there is no canonical method to achieve that e.g. the distance between two measurements of a pair of momentum and voltage

$C(t_i) = \big(p(t_i), U(t_i)\big)$ and $C(t_i) = \big(p'(t_i), U'(t_i)\big)$. One could use weighted averages of time-local distances for each coordinate or even more complex assignments exploiting dependencies between the physical quantities. Although multivariate time series can be interpreted as time series taking values in $\mathbb{R}^m$, a suitable metric $d$ on the underlying space $M \neq \mathbb{R}^m$ may be difficult to access or even unknown. As an example, normed quaternions can be embedded in $\mathbb{R}^4$ but should not be compared by summing up squared residues of the coordinates since they live on the curved manifold $SU(2,\mathbb{C})$.[1] Even worse, when tracking the positions and orientations of fingers over time, we might measure over twenty coordinates at each time step – hence a proper distance between a pair of hand poses is not obvious.

Assume one can handle the above-mentioned challenges, we still have to provide a global distance measure $d(C, C')$ between two time series considering their time domains $T$ and $T'$. Depending on the task, certain invariances may be exploited to design an appropriate notion of distance. As an example, when collecting iris data analogous to Figure 1.2 we cannot ensure, that all images are properly aligned and as a result the angular domains exactly map onto each other. A rotation-invariant assignment of time can drastically improve the quality of the used measure as shown in [8]. Other measures could respect invariances like transforming the time coordinates [9] as well as occlusion and sensor loss during the recording of the time series [10].

**Stage 3: Implementation of Efficient and Robust Algorithms**

Once a proper representation and a suitable notion of distance have been determined, the corresponding data structures and algorithms have to be implemented and evaluated. On the one hand, efficiency in terms of runtime is a major challenge: Assume we have sampled the time series at $n$ time steps, an efficient algorithm for the computation of a distance value between two instances should ideally need no more than $\mathcal{O}(n)$ operations. Superlinear dependencies in the length of the time series may be acceptable if reasonable improvements in terms of quality can be reached but are often considered prohibitive. Even more, due to the rise of Big Data, e.g. Walmart stores more than 2.5 petabytes of sales transactions [11], a future direction could be the development of sublinear algorithms that can produce reliable statements while only having seen a tiny fraction of the whole data. As a result, time series data mining aims for simple algorithms with low memory requirements, ideally constant or linear, and reasonable dependencies in the number of time steps. On the other hand, the quality of the used distance measure should be sufficiently high to distinguish between different classes of time series. Depending on the characteristics of the collected data, more

[1] Note, Euclidean distance measures the length of the shortest path between two points in flat Cartesian space $\mathbb{R}^m$ i.e. the geodetic paths are straight lines.

or less complex algorithms may be utilized to guarantee the desired quality and robustness. In 2001, Banko and Brill [12] demonstrated that they can mutually outperform natural language disambiguation algorithms by simply using more data. A very similar observation was made by Ding et al. [13]. The authors proved that simple measures like Euclidean distance can beat complex ones in terms of quality by expanding the training set. As a result, the runtime and quality of time series data mining algorithms have to be evaluated carefully to determine a proper trade-off.

These three stages can be interleaved or iterated several times until the desired accuracy is achieved. During the next chapter of this thesis, we will investigate common data mining techniques for the analysis of time series data using two publicly available data sets, namely the Cylinder-Bell-Funnel [14] (CBF) and the Gun-Point [15] dataset. Both exhibit different characteristics which can be exploited for the demonstration of data representations and similarity measures.

**Toy Dataset 1: Cylinder-Bell-Funnel**

The CBF dataset is a synthetically created database consisting of three characteristic classes. The generating functions for the classes are given as follows [14]:

$$\begin{aligned} C(t) &:= (6+\eta)\cdot\chi_{[a,b]}(t)+\epsilon(t) &&, \\ B(t) &:= (6+\eta)\cdot\chi_{[a,b]}(t)\cdot\tfrac{t-a}{b-a}+\epsilon(t) &&, \\ F(t) &:= (6+\eta)\cdot\chi_{[a,b]}(t)\cdot\tfrac{b-t}{b-a}+\epsilon(t) &&, \end{aligned}$$

where $t \in \{0,\ldots,127\}$, $\chi_{[a,b]}$ the indicator function on $[a,b]$, $\eta$ as well as $\epsilon(t)$ drawn from a standard normal distribution $\mathcal{N}(\mu=0,\sigma=1)$, $a$ an integer uniformly drawn from the interval $[16,32]$ and $b-a$ an integer uniformly drawn from $[32,96]$. As a result, CBF exhibits a reasonable variability in the amplitudes ($\eta$) and time-dependent noise on the measurement-axis ($\epsilon(t)$) as well as variable length and position of the shape's support on the time-axis ($[a,b]$). Figure 1.3 depicts a family of time series for each of the three classes. Concluding, CBF has the following properties:

- **(infinite number of samples)** Since CBF is synthetic we can virtually create any desired number of labelled samples for the evaluation of similarity measures. Although CBF is heavily used in time series data mining research, the lack of application limits its credibility. Hence, real-world datasets should additionally be taken into account to back-up the results gained on CBF.

- **(variable amplitudes)** The amplitudes $(6+\eta)$ vary for distinct samples and thus CBF can be used to demonstrate invariance or robustness against amplitude alteration.

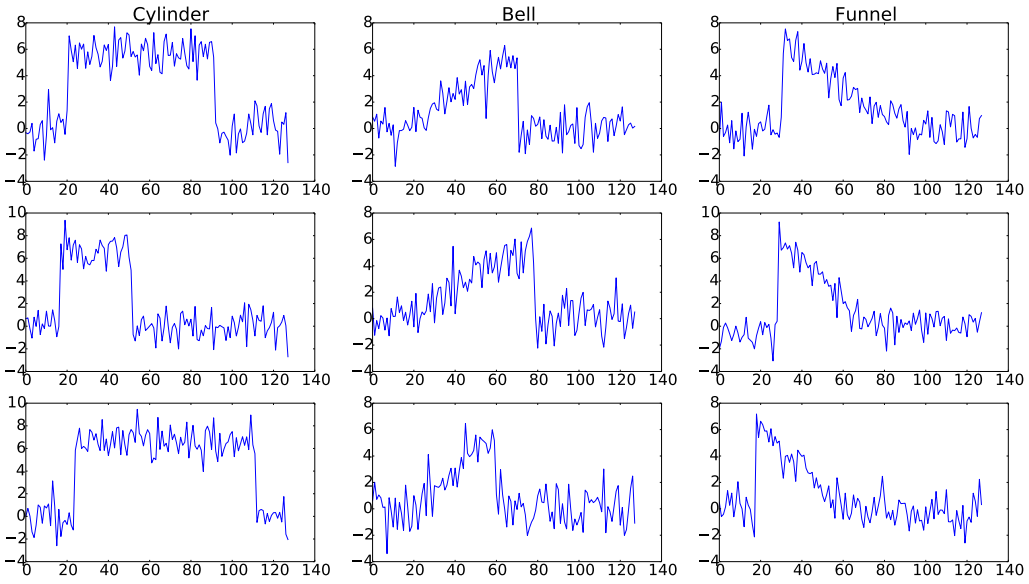


Figure 1.3: An example of three time series for each class of the CBF datasets. Note, the non-negligible variability in amplitudes, temporal noise and support.

- **(time-dependent noise)** The random variable $\epsilon(t)$ introduces a non-negligible noise-to-signal-ratio of roughly $\frac{1}{6}$ and thus CBF can be used to demonstrate robustness of similarity measures against temporal distortions from sensors.

- **(variable support)** The support $\chi_{[a,b]}$ of the perceivable shape varies for different samples such that CBF is ideally suited to demonstrate the benefits of elastic similarity measures which feature invariance against local transformations of the time domain in contrast to classical lock-step measures.

- **(approximate global alignment)** Omitting the temporal noise $\epsilon(t)$, CBF can be considered to be zero on the complimentary set $[0,a) \cup (b,127]$ of the support. Hence, each instance is embedded into a zero vector of length 128 which relieves us of the burden to determine the beginning and end of the measured event. However, this approximate global alignment can be considered to be a major weakness of CBF since many real-world datasets lack an implicit segmentation of the measured signal.

Concluding, CBF covers common artefacts which may occur during the recording of time series and thus can be used as a straw man during early experiments with novel algorithmic approaches. Nevertheless, CBF does not consider frequently occurring effects such as time-dependent offsets, global scaling of the time-axis

Figure 1.4: Six sample images of Chotirat Ann Ratanamahatana taken from the motion-capturing of hand positions while pulling a gun. For later analysis, the coordinates of the red glove are tracked over time. Analogous, the same experiment is repeated without a gun. The images and the measured stream have been gratefully donated by Eamonn Keogh. Manually labelled instances are published as the Gun-Point dataset in the UCR Repository [16].

or signal loss due to sensor failure. Hence, we introduce another comprehensible dataset from a real-world experiment.

#### Toy Dataset 2: Gun-Point

The Gun-Point dataset consists of 200 time series each representing the tracked spatial coordinates of a hand while pulling a gun. The coordinates are determined by the center of the red glove used during the recording in front of a green screen and afterwards sampled with 150 time ticks (see Figure 1.4). The dataset can be split into two classes: 100 instances with a gun and another 100 instances pretending to hold a gun – hence the two class labels 'Gun-(Draw)' and 'Point'. Figure 1.5 depicts instances for each of the two classes. The properties of the Gun-Point dataset as provided by the UCR Repository can be described as follows:

- **(real-world data)** Originating from a real experiment, benchmarks of similarity measures provide more credibility in terms of portability to other motion-capturing tasks. The manual segmentation and labelling of the stream is cumbersome resulting in a small number of instances.

- **(slowly-varying)** The amount of noise is negligible and the dominant shape exhibits only low frequency contributions (half a period of a sine wave). As

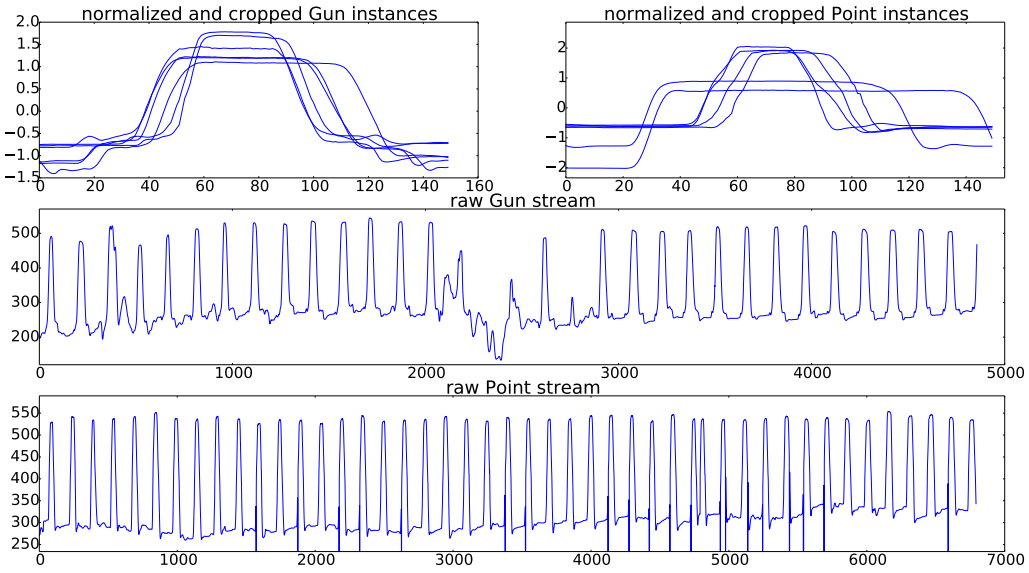


Figure 1.5: An example of six instances for each of the two classes of the Gun-Point dataset as provided by the UCR Repository [16]. Although the raw streams exhibit coordinates for both spatial dimensions, Keogh et al. decided to only consider the x-axis. The illustrated instances were preprocessed to have zero mean and a standard deviation of one. Moreover, the corresponding intervals for each measured event were determined manually (both panels on top). The raw streams exhibit additional artefacts such as variable amplitude, baseline-wandering, sensor failure and missing segmentation (both panels at the bottom).

a result, Gun-Point is ideally suited to demonstrate dimensional reduction techniques which often omit high frequency contributions of the signal.

- **(approximate global alignment)** As mentioned above, the stream has been labelled and segmented manually. Hence when using Gun-Point as provided by the UCR Repository, we cannot benchmark segmentation tasks unless we artificially concatenate the individually normalized instances.

During the theoretical sections of this thesis, we will either use CBF or Gun-Point as toy dataset to demonstrate the strength or weaknesses of certain representations and similarity measures.

### Organization of this Thesis

This thesis is divided into five further chapters. Chapter 2 provides a brief introduction to classical techniques used in the field of time series data mining. This includes a discussion of predominant similarity measures, namely Euclidean

distance (ED) and Dynamic Time Warping (DTW). Moreover, we discuss several invariances of time series data that have to be taken care of during the construction of high quality measures amongst others invariances against offset displacement, amplitude variation or local deformation of the time-axis. For each invariance we provide comprehensive examples that underline their benefit in clustering tasks. Finally, an exhaustive discussion of lower bounding techniques in terms of unitary symmetries is provided. Frequently used dimensional reduction techniques such as discrete Fourier and wavelet coefficients as well as principal component analysis are presented in a unified framework. An experienced reader may completely skip this section and continue with Chapter 3.

An extension of global similarity measures to local subsequence alignments is given in Chapter 3. This includes a detailed discussion of one-nearest-neighbour (1NN) search of subsequence candidates in a huge stream of time series data under ED and DTW. Traditional lower bounding techniques for DTW are analysed theoretically and investigated for their parallelization potential. Several improvements of the state-of-the-art implementation of 1NN search, namely the UCR-Suite, are proposed and ported to CUDA-enabled accelerators. As an example, we propose an analytic expression for the computation of local alignment scores under z-normalized Euclidean distance that can easily be parallelized on any concurrent device that features a Fast Fourier Transform. Contrary to UCR-ED's linear dependency, our algorithm exhibits an immeasurable dependency on the query length and thus can be used to match queries of arbitrary size without a perceivable mitigation in performance. The gained speedups exceed two orders-of-magnitude. Analogous, we provide an efficient CUDA-parallelization of UCR-Suite's lower bound cascaded DTW portion. Speedups of more than one order-of-magnitude are achieved. In summary, we provide highly efficient drop-in replacements for both UCR-ED and UCR-DTW which allow for the faster processing of even bigger streams at exactly the same accuracy.

Chapter 4 deals with linear time approximations of DTW and their use in subsequence alignment tasks. We provide a CUDA-parallelization of greedily relaxed subsequence DTW. A runtime improvement of two-order-of-magnitude can be achieved in comparison to sequential code. Moreover, we propose a novel subsequence measure, the Gliding Elastic Match (GEM), that can be computed with the same asymptotic runtime as greedily relaxed DTW but in contrast offers a full relaxation of the penalty matrix. Further advantages of GEM include invariance against time-dependent trends on the measurement axis and invariance against uniform scaling of the time-axis. The CUDA-parallelization of GEM outperforms UCR-DTW by up to two orders-of-magnitude. Finally, an extension of GEM to supervised multi-shape segmentation of streams is provided and evaluated on motion-tracking data.

The treatment of time series is limited to real-valued measurements in the aforementioned chapters. However, when dealing with time series from motion

sensors we have to process orientations in the angular domain and displacements of the reference frame. Hence, we provide a unified framework for the alignment of Lie group-valued time series in Chapter 5. This includes a brief introduction to smooth manifolds and group theory. Building upon this, notions of similarity in the group of rotations and rigid transformations are established. Furthermore, memory-efficient representations and extensions of arbitrary elastic measures to handle group elements are discussed in detail. Finally, we demonstrate the applicability of our theoretical ideas on motion sensor data. The final chapter concludes the thesis and discusses further directions of future research.

CHAPTER 2

# TIME SERIES DATA MINING

*'It is a capital mistake to theorize before one has data.'* — Sherlock Holmes *(Arthur Conan Doyle)*

As we have seen, huge amounts of time series data are collected in a broad variety of applications from different domains. Once the database is acquired, new tasks such as the structural analysis of the measured data, its interpretation and the derivation of rules arise. One may ask questions like: 'What is the most unusual time series in my database?' (anomaly detection [7]) or 'Given this new measurement of time series data – have I seen this before ?' (retrieval [17]). Further considerations may include the automated partitioning of the database into distinct groups of time series that share the same characteristics (clustering [6]) or the classification of new measurements into outliers and valid entities (classification and outlier detection [13]).

## 2.1 Metrics and Dissimilarities

In order to accomplish these tasks, one has to determine whether two time series are correlated i.e. they share similar features or not. For this reason, we require quantifiable measures to assign values of similarity to a pair of time series. In most cases, there exists a natural embedding of time series into the Cartesian space $\mathbb{R}^n$ of $n$ real-valued coordinates since time series are represented as sequences of consecutive measurements on the time-axis. However, in some situations e.g. on curved manifolds or even Lie groups one might measure quantities with a less

obvious notion of distance. Thus, for the time being we proceed with a general definition and narrow it down to the given situation in each case.

**Definition 2** (metric space)**.** *Let $M$ be a non-empty set and $d : M \times M \to \mathbb{R}$ a real-valued function, then the ordered tuple $(M,d)$ is called a* ***metric space*** *if and only if*

1. $d(C,C') = 0 \iff C = C' \quad \forall C,C' \in M$ ***(coincidence)***
2. $d(C,C') = d(C',C) \quad \forall C,C' \in M$ ***(symmetry)***
3. $d(C,C') + d(C',C'') \geq d(C,C'') \quad \forall C,C',C'' \in M$ ***(triangle inequality)*** .

From all three conditions we can deduce that metrics are positive definite maps. This condition is often imposed in the definition of metric spaces but apparently it is redundant.

**Proposition 1** (non-negativity)**.** *Let $(M,d)$ be a metric space then*

$$d(C,C') \geq 0 \quad \forall C,C' \in M.$$

*Proof.* Starting with the triangle inequality we observe $d(C,C') + d(C',C) \geq d(C,C)$. Using symmetry and coincidence we deduce that $2 \cdot d(C,C') \geq d(C,C) = 0$ for all $C,C' \in M$. □

Another inequality statement can be deduced directly from the triangle inequality. As we will see in the next subsection, the reversed triangle inequality can be used to lower-bound the family of induced metrics on normed spaces.

**Proposition 2** (reversed triangle inequality)**.** *Let $(M,d)$ be a metric space then*

$$\left| d(C,C'') - d(C',C'') \right| \leq d(C,C') \quad \forall C,C',C'' \in M \quad .$$

*Proof.* Beginning with the triangle inequality $d(C,C') \geq d(C,C'') - d(C',C'')$ we assume $d(C,C'') - d(C',C'') \geq 0$ and therefore $d(C,C') \geq \left| d(C,C'') - d(C',C'') \right|$. Otherwise, when $d(C,C'') - d(C',C'') < 0$ we interchange the roles of $C,C'$ and $C''$. Starting once again with the triangle inequality $d(C',C) + d(C,C'') \geq d(C',C'')$ we subtract the second term on the left hand side and obtain the final result $d(C',C) \geq \left| d(C',C'') - d(C,C'') \right|$. □

### 2.1.1 The $L_p$-Norm Family

For the sake of simplicity, assume the measurement of $N$ equally long and real-valued time series $C^{(k)}$ on a uniformly spaced time domain $T := \{t_0, \ldots, t_i, \ldots, t_{n-1}\}$ such that $t_{i+1} - t_i$ is a constant for all $i \in \{0, \ldots, n-2\}$. Keeping this constant in mind, the time ticks $t_i$ can be mapped onto the index set $\mathscr{I} := \{0, \ldots, n-1\}$ without any loss of information. Hence, each of the time series candidates $C^{(k)}$ for $k \in \{0, \ldots, N-1\}$ is represented by an element in the vector space $\mathbb{R}^n$. The components of a time series $C^{(k)}$ shall be denoted as $C_i^{(k)} \in \mathbb{R}$ where $i$ belongs to the index set $\mathscr{I}$. Extensions to higher-dimensional values $C_i^{(k)} \in \mathbb{R}^m$ where $m > 1$ or even group-valued measurements are conceivable but will be omitted in this chapter. We will pursue these approaches later in this thesis. The vector space $\mathbb{R}^n$ can be equipped canonically with a norm.

**Definition 3** (normed vector space)**.** *Let $M$ be a vector space over the field $\mathbb{F}$ and $\|\cdot\| : M \to \mathbb{R}$ a real-valued function, then the ordered tuple $(M, \|\cdot\|)$ is called* ***normed vector space*** *if and only if*

1. $\|C\| = 0 \iff C = 0$ ***(definiteness)***
2. $\|\alpha C\| = |\alpha| \|C\| \quad \forall \alpha \in \mathbb{F},\ \forall C \in M$ ***(homogeneity)***
3. $\|C\| + \|C'\| \geq \|C + C'\| \quad \forall C, C' \in M$ ***(triangle inequality)*** .

Every norm $\|\cdot\|$ induces a metric $d_{\|\cdot\|}$ on the corresponding vector space by the mapping $d_{\|\cdot\|}(C, C') = \|C - C'\|$ for all $C, C' \in M$. However, not every metric induces a norm. In the following, the $L_p$-norm family and its induced metrics can be used to equip $M = \mathbb{R}^n$ with a notion of distance:

$$d_{L_p}(C, C') := ||C - C'||_p = \left( \sum_{i=0}^{n-1} \left| C_i - C'_i \right|^p \right)^{\frac{1}{p}} \quad \text{where} \quad p > 0 \quad .$$

The most common choice for the parameter is $p = 2$ since $d_{L_2}$ captures the 'usual' concept of distance in terms of the Pythagorean theorem. During the rest of this thesis, we will refer to $d_{L_2}$ as Euclidean distance (ED). Additionally, the algebraic structure of $L_2$ and its dual space $L_2^*$ can be exploited to speed-up calculations as we will see in Chapter 3. Nevertheless, the square of residues $(C_i - C'_i)^2$ can disproportionally bias the metric towards outliers, which may lead to not robust behaviour on noisy datasets. Figure 2.1 illustrates an example where the $L_2$-norm is unsuitable. In this case, the Manhattan Distance is often used where $p = 1$ to avoid unstable performance. In contrast, if the bias towards outliers is desired the uniform norm for $p \to \infty$ and its induced metric $d_{L_\infty}$ are infrequently used. Other values of $p$ are conceivable but almost never realized.[1]

[1] A counterexample can be found in [18], where the authors construct bagging classifiers over an ensemble of different values of $p$.

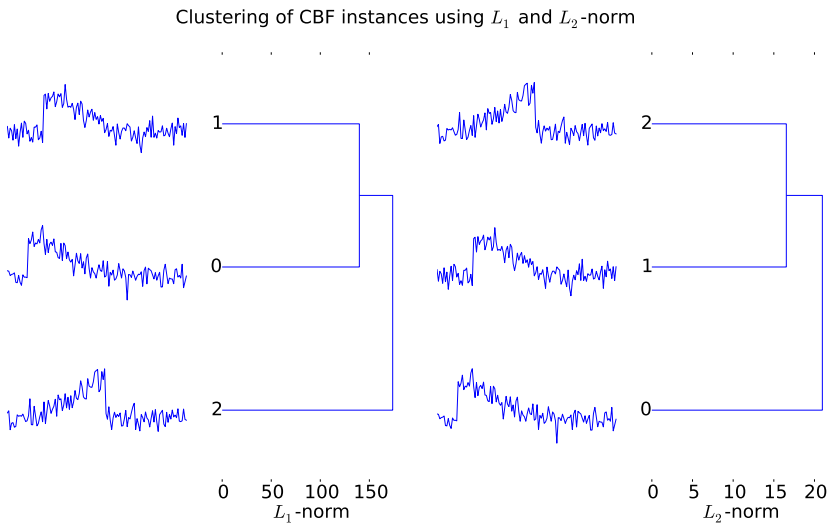


Figure 2.1: Agglomerative clustering of three instances from the popular Cylinder-Bell-Funnel dataset [16] using complete linkage. On the left hand side, two Funnel instances are correctly clustered together using the $L_1$-norm. The remaining instance from the Bell class is excluded from the cluster. In contrast, the $L_2$-norm fails to predict the correct clustering on the right hand side.

Let us recall the reversed triangle inequality from Proposition 2. Rewriting it in terms of induced metrics implies the following inequality for all $C, C'$ and $C''$:

$$\Big|d\big(C,C''\big)-d\big(C',C''\big)\Big| \leq d\big(C,C'\big) \quad \Longrightarrow \quad \Big|\|C-C''\|-\|C'-C''\|\Big| \leq \|C-C'\| \quad .$$

Setting $C''$ to be the zero-vector we observe $\mathrm{LB}\big(C,C'\big) := |\|C\|-\|C'\|| \leq \|C-C'\|$. Assuming we have precomputed $\|C^{(k)}\|$ for all $C^{(k)}$ in the database which accounts for $\mathcal{O}(N \cdot n)$ time we can evaluate $\mathrm{LB}\big(C,C'\big)$ in constant time. As a result, the reversed triangle inequality can be used to speed-up nearest neighbour search on every normed vector space. Other lower bounds can be deduced directly from the triangle inequality in order to speed-up data mining algorithms e.g. a fast computation scheme for the popular k-Means clustering algorithm [19].

In summary, metrics and norms are appropriate theoretical constructs to capture similarity between time series. Moreover, the triangle inequality may be the most versatile part among the definition properties.

### 2.1.2 Pearson's Correlation Coefficient

Consider a time series $C = (C_0, \ldots, C_{n-1})$ of length $n$ and a globally shifted and amplitude-scaled variant $C' = a \cdot C + b := (a \cdot C_0 + b, \ldots, a \cdot C_{n-1} + b)$ for $a \in \mathbb{R} \setminus \{0\}$ and $b \in \mathbb{R}$. This situation may occur if $C$ and $C'$ are recorded with different devices

i.e. with different calibration or in different physical units e.g. Fahrenheit and Celsius. Although $C$ and $C'$ share the same characteristics their $L_p$-distance does not vanish in general. Assume $C$ has at least one non-vanishing value, then:

$$d_{L_p}(C,C')^p = \sum_{i=0}^{n-1} \left|C_i - C'_i\right|^p = \sum_{i=0}^{n-1} \left|C_i - a\cdot C_i - b\right|^p \neq 0 \qquad \forall\, a \neq 1 \;\vee\; \forall\, b \neq 0 \quad .$$

However, we can capture their similarity with Pearson's empirical correlation coefficient (PCC) $\rho : \mathbb{R}^n \times \mathbb{R}^n \to [-1,1]$ as follows:

$$\rho(C,C') := \frac{\frac{1}{n}\sum_{i=0}^{n-1}\left(C_i - \mu_C\right)\cdot\left(C'_i - \mu_{C'}\right)}{\sigma_C \cdot \sigma'_C} \quad ,$$

where $\mu_{(\cdot)}$ is the average and $\sigma_{(\cdot)}$ the empirical standard deviation of $C$ or $C'$, respectively. Since $\mu_{C'} = a\cdot\mu_C + b$ and $\sigma_{C'} = |a|\cdot\sigma_C$ the correlation coefficient evaluates to $\rho(C,C') = \frac{a}{|a|}\frac{1}{n}\sum_{i=0}^{n-1}\left(C_i - \mu_C\right)^2/\sigma_C^2 = \operatorname{sgn}(a)$. In general, $\rho$ takes values near 1 for strongly correlated time series and $-1$ for strongly anti-correlated ones i.e. $\operatorname{sgn}(a) = -1$. Values near 0 indicate that both time series are uncorrelated. Note that $\rho$ can only capture affine dependencies between a pair of time series (straight lines) and therefore will fail to predict non-linear correlation.[2]

In order to assign a notion of distance to a pair of time series we can define the Pearson distance as $d_\rho : \mathbb{R}^n \times \mathbb{R}^n \to [0,2]\,,\ (C,C') \mapsto d_\rho(C,C') := 1 - \rho(C,C')$. As a result, $d_\rho$ is blind for global shifts and amplitude-scaling on the measurement-axis since both time series have vanishing distance $d_\rho(C,C') = 0$ for all $a > 0$. Unfortunately, $d_\rho$ does not fulfil the triangle inequality[3] and also violates the coincidence property since $d_\rho(C,C') = 0$ but $C \neq C'$. Hence, the Pearson distance is not a metric. Nevertheless, a close relationship between PCC and ED can be shown.

**Definition 4** (z-normalization)**.** *Let $C \in \mathbb{R}^n$ be a time series of length $n$ then*

$$z : \mathbb{R}^n \to \mathbb{R}^n\ , (C_0,\ldots,C_{n-1}) \mapsto z(C) = \frac{C-\mu}{\sigma} := \left(\frac{C_0-\mu}{\sigma},\ldots,\frac{C_{n-1}-\mu}{\sigma}\right)$$

*where $\mu$ is the average and $\sigma$ the standard deviation is called **z-normalization**.*

The nomenclature originates in z-scores of standardized random variables. Analogous, a z-normalized time series $z(C)$ has zero mean and a standard deviation of one. This procedure corresponds to the manual removal of the shift and an equalization of amplitudes on the measurement-axis in a preprocessing

[2] The interested reader may refer to the Maximal Information Coefficient [20] – a sophisticated correlation measure to capture non-linear dependencies.

[3] Choose $C = (1,0,0)$, $C' = (1,0,1)$ and $C'' = (0,0,1)$ for a counterexample since $\frac{1}{2} + \frac{1}{2} < \frac{3}{2}$.

phase. Moreover, z-normalization defines an equivalence relation $\sim_z$ on the space of time series due to the equivalence kernel $\sim_z$: $C \sim C' :\Longleftrightarrow z(C) = z(C')$. The equivalence classes $\{z(C) \mid C \in \mathbb{R}^n\}$ consist of time series which only differ in global shifts and amplitudes. Employing ED on the equivalence classes in the quotient space $\mathbb{R}^n / \sim_z$ uncovers the relationship between ED and the Pearson distance [6].

**Proposition 3** (z-normalized ED)**.** *Let $C, C' \in \mathbb{R}^n$ be two time series of length $n$ then*

$$\mathrm{zED}\big(C, C'\big) := \Big\| z\big(C\big) - z\big(C'\big) \Big\|_2 = \sqrt{2 \cdot n \cdot \big(1 - \rho\big(C, C'\big)\big)} \quad .$$

*Proof.* Starting on the left hand side taking the square since $\|C\| \geq 0$ for all $C$:

$$\Big\| z\big(C\big) - z\big(C'\big) \Big\|_2^2 = \sum_{i=0}^{n-1} \big(z\big(C\big)_i - z\big(C'\big)_i\big)^2 = \sum_{i=0}^{n-1} \Big(z\big(C\big)_i^2 - 2 \cdot z\big(C\big)_i \cdot z\big(C'\big)_i + z\big(C'\big)_i^2\Big) \quad .$$

Since $\mu_C = \frac{1}{n} \sum_{i=0}^{n-1} z\big(C\big)_i = 0$ and $\sigma_C = \frac{1}{n} \sum_{i=0}^{n-1} z\big(C\big)_i^2 = 1$ for all $C \in \mathbb{R}^n$ we deduce:

$$= 2 \cdot n - \sum_{i=0}^{n-1} 2 \cdot z\big(C\big)_i \cdot z\big(C'\big)_i = 2 \cdot n \cdot \left(1 - \frac{1}{n} \sum_{i=0}^{n-1} \frac{C_i - \mu_C}{\sigma_C} \cdot \frac{C'_i - \mu_{C'}}{\sigma_{C'}}\right) .$$

Finally, the definition of PCC can be substituted and the proof is finished. □

Hence, z-normalized ED is proportional to the square root of the Pearson distance. Note that in contrast to ED on the quotient space $\mathbb{R}^n / \sim_z$, the mapping of representatives

$$d_z : \mathbb{R}^n \times \mathbb{R}^n, \big(C, C'\big) \mapsto \sqrt{d_\rho\big(C, C'\big)}$$

still violates the coincidence property and therefore is **not** a metric on $\mathbb{R}^n$. The use of z-normalization can significantly boost the quality of the used distance measure. Figure 2.2 illustrates how zED outperforms ED on slightly rescaled and shifted instances of the popular Cylinder-Bell-Funnel dataset. However, z-normalization is not always advisable. Counterexamples where z-normalization can even be harmful will be discussed later in Chapter 4. Furthermore, the reversed triangle inequality cannot be used as an effective lower bound since it degenerates to the trivial zero lower bound:

$$\mathrm{LB}\big(z(C), z(C')\big) := |\|z(C)\| - \|z(C')\|| = \sqrt{n} - \sqrt{n} = 0 \quad .$$

Nevertheless, distinct lower bounds exploiting Parseval's theorem are still conceivable. A detailed discussion of orthogonal basis systems and their use for the construction of lower bounds can be found in Section 2.2.

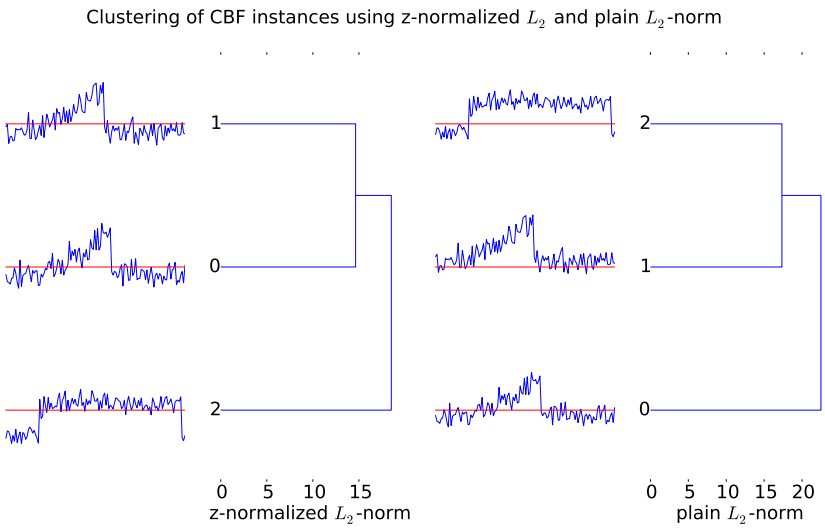


Figure 2.2: Agglomerative clustering of three instances from the popular Cylinder-Bell-Funnel dataset [16] using complete linkage. On the left hand side, two Bell instances are correctly clustered together using the z-normalized $L_2$-norm. The remaining instance from the Cylinder class is excluded from the cluster. In contrast, the plain $L_2$-norm fails to predict the correct clustering on the right hand side. The slightly shifted mean of the time series strongly influences the final result. The time-axis (zero vector) has been plotted as a red line for comparison.

### 2.1.3 Dynamic Time Warping

We have seen, that global shift and amplitude invariance can improve the quality of time series clustering. Hence, the reader may ask: 'Are there other invariance that can be exploited to improve the situation?' Both ED and its normalized variant zED calculate the final distance value by summing up index-wise differences. As a result, the time ticks of both time series are bijectively mapped onto each other in a strict one-to-one manner. This lock-step property is desirable from a theoretical point of view but may be too restrictive for real-world datasets.

Consider two voice recordings of the word 'exact'. The online service of Oxford Learner's Dictionaries [21] specifies the pronunciation as 'ɪgˈzækt' for both American and British English. However, the provided recordings differ in the length of syllables. The American speaker spends more time on the 'a' than his British counterpart. Figure 2.3 illustrates the z-normalized signals for both voice records and a direct comparison of both time series. Obviously, both recordings have similar shape but occasionally tend to be out of phase. As a result, a rigid one-to-one mapping of indices is apparently not advisable. The simplest approach to fix the phase shifts would be the local adjustment of the time-axis in such a manner that

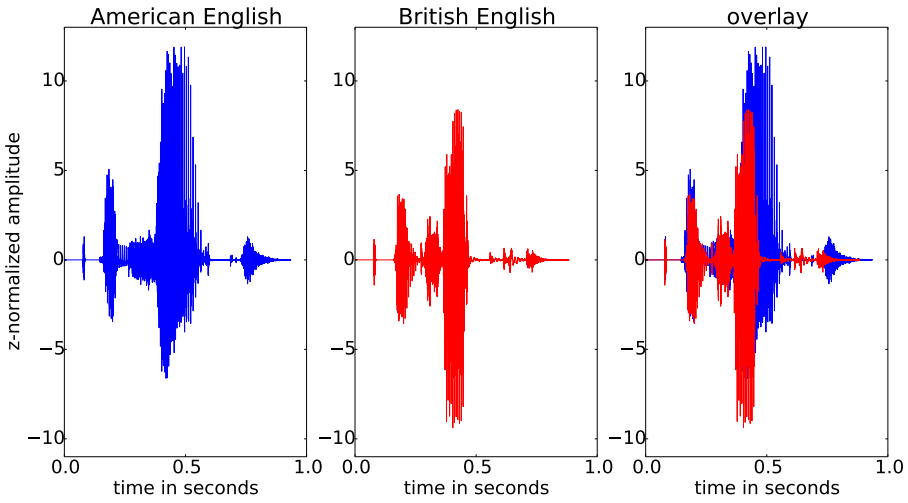


Figure 2.3: An example for two voice records of the word 'exact' in American (blue) and British (red) English pronunciation taken from the online service of Oxford Learner's Dictionaries. The time series were sampled at 40100 Hz for roughly one second. Both time series are aligned at the initial spike. The overlay of both signals emphasizes the difference in local phases.

both speakers spend the same time on each syllable. In 1994, Berndt and Clifford introduced such a time-warping mechanism to the data mining community [22] which was developed before by Sakoe and Chiba in 1978 [9] in the field of speech recognition. Figure 2.4 illustrates the idea of local shrinking and stretching of the time-axis for the two given voice recordings. Thus, we have to give a formal description for the dynamic deformation of the time-axis. However, we do not want to allow arbitrary mappings between the index sets of $C$ and $C'$ and therefore impose the following restriction:

**Definition 5.** *Let* $\mathcal{I} := \mathrm{dom}(C)$ *and* $\mathcal{J} := \mathrm{dom}(C')$ *be the index sets of the time series* $C$ *and* $C'$*, then the sequence of tuples* $\gamma := \big((i_l, j_l) \in \mathcal{I} \times \mathcal{J}\big)_l$ *is a* ***monotone, continuous and bounded warping path*** *if and only if*

$$\begin{aligned} &\min\left(i_{l+1} - i_l, j_{l+1} - j_l\right) \geq 0 \quad \wedge \\ &\max(i_{l+1} - i_l, j_{l+1} - j_l) = 1 \quad \forall l \in \{0, \ldots, |\gamma| - 2\}\,, \end{aligned}$$

*where* $(i_0, j_0) = (0,0)$ *and* $(i_{|\gamma|-1}, j_{|\gamma|-1}) = (|C| - 1, |C'| - 1)$.

Three major properties of the Dynamic Time Warping similarity measure can be directly derived from the definition.

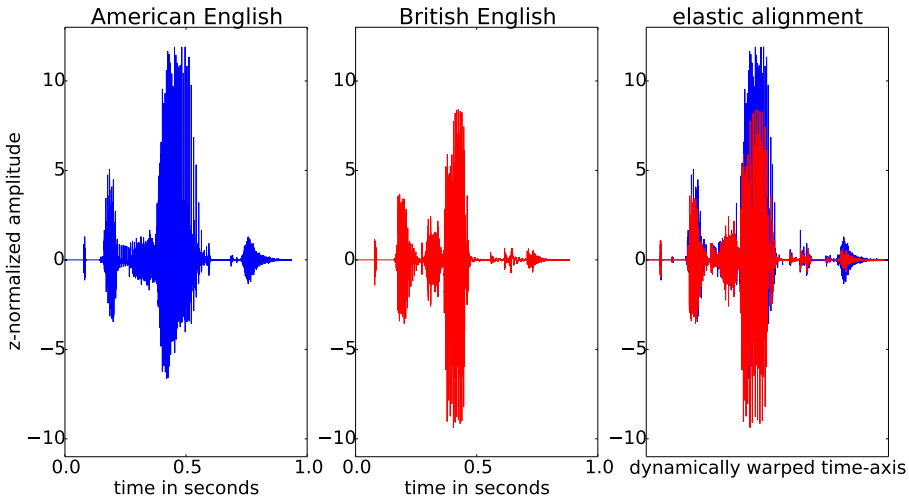


Figure 2.4: The same example for two voice records of the word 'exact' in American (blue) and British (red) English pronunciation taken from the online service of Oxford Learner's Dictionaries. Here, we locally dilated and contracted the time-axis to find an optimal alignment in terms of least square error. As a result, both voice recordings have an intuitive alignment in comparison to the original setting.

- (**monotony**) Each segment of the warping path $\gamma$ has to increment at least one index of $C$ or $C'$. As a result, we are not allowed to map an index tuple $(i_l, j_l)$ several times.
- (**continuity**) Consecutive entries in $\gamma$ must be reached by horizontal, vertical or diagonal steps of length 1. Hence, every index of $C$ and $C'$ is matched without gaps.
- (**bounding**) The warping path starts at the first index of $C$ and $C'$. Analogously, $\gamma$ ends at the last index. Therefore, the final mapping is a global alignment of indices.

For each combination of indices $(i_l, j_l)$ in the warping path $\gamma$ we can locally assign non-negative weights $w : \mathscr{I} \times \mathscr{J} \to \mathbb{R}_0^+$, $(i_l, j_l) \mapsto w(i_l, j_l)$ that describe the similarity between the corresponding values $C_{i_l}$ and $C'_{j_l}$ of both time series. The local weights are usually determined by the $L_p$-norm on $\mathbb{R}$. Finally, the DTW distance calculates the optimal warping path in terms of the following definition.

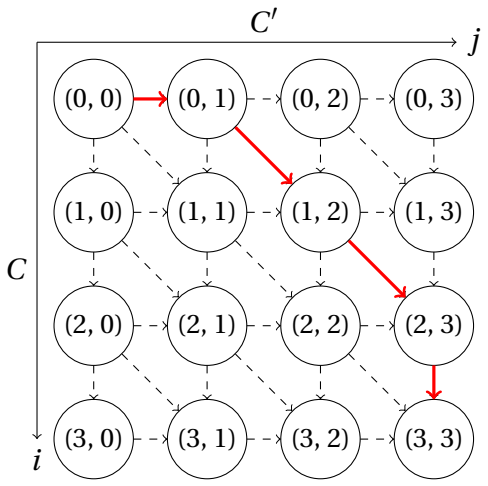


Figure 2.5: An example for the DAG representation of the optimization problem for the DTW distance measure between two time series of length four. The nodes are connected by horizontal, vertical and diagonal edges of step size one (continuity). Each edge increments at least on index (monotony). Due to the bounding property of DTW the optimal warping path (red) starts in the upper left and ends in the lower right cell.

**Definition 6** (DTW)**.** *Let* $\Gamma$ *be the set of all monotone, continuous and bounded warping paths. The* ***optimal warping path*** $\hat{\gamma}$ *and its associated* ***measure*** $\hat{d}$ *with respect to a given weighting function* $w : \mathcal{I} \times \mathcal{J} \to \mathbb{R}_0^+$ *are defined as:*

$$\hat{\gamma} := \underset{\gamma \in \Gamma}{\operatorname{argmin}} \sum_{(i_l, j_l) \in \gamma} w(i_l, j_l) \quad \textit{and} \quad \hat{d} := \min_{\gamma \in \Gamma} \sum_{(i_l, j_l) \in \gamma} w(i_l, j_l) \quad .$$

This optimization problem is equivalent to the calculation of a shortest path within a directed acyclic graph (DAG). Each cell $(i_l, j_l)$ is represented by a node and has maximal three incoming edges with associated weights $w(i_l, j_l)$. The Single-Source Shortest Path problem on DAGs can be solved in linear time $\mathcal{O}(|V| + |E|)$ by the relaxation of nodes in topological order [23]. Fortunately, we do not need to compute a topological sorting since it is implicitly given by the relaxation scheme if we interpret the cell indices $(i, j)$ as integers in $b$-adic representation $i \cdot b + j$ where $b = |C'|$. As a result, we can consecutively relax the cells in row-major order without violating node dependencies (see Figure 2.5). In practice the computation is achieved by dynamic programming using a matrix of size $\big(|C| + 1\big) \times \big(|C'| + 1\big)$. The corresponding relaxation scheme can be written recursively

$$M[i, j] = w(i-1, j-1) + \min \begin{cases} M[i-1, j] \\ M[i, j-1], \\ M[i-1, j-1] \end{cases} \quad \text{where} \quad \begin{array}{l} M[0,0] = 0 \\ M[0, j] = \infty \; \forall \, j \geq 1 \\ M[i, 0] = \infty \; \forall \, i \geq 1 \end{array} \quad .$$

As a result, the final DTW measure $M\big[|C|,|C'|\big]^{\frac{1}{p}}$ can be computed in $\mathcal{O}\big(|C|\cdot|C'|\big)$ time.[4] The optimal warping path $\hat{\gamma}$ is determined by back-tracing the predecessor information stored in a separate matrix. The $L_p$-norm is a special case where $\gamma$ consists only of nodes on the main diagonal. Hence, the Euclidean-flavoured DTW measure fulfils the inequality $\mathrm{DTW}\big(C,C'\big) \leq \mathrm{ED}\big(C,C'\big)$ for all time series $C, C'$ of the same length. The resulting map $\mathrm{DTW} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_0^+$ has the following properties:

- The assigned measure is non-negative. This can easily be proved since the final value is a finite sum of non-negative edge weights.
- DTW is a symmetric map, i.e. $\mathrm{DTW}\big(C,C'\big) = \mathrm{DTW}\big(C',C\big)$ for all $C, C' \in \mathbb{R}^n$ since both variants result in the same DAG representation.
- The measure $\mathrm{DTW}\big(C,C'\big)$ vanishes whenever $C$ is a time-warped variant of $C'$. This is true especially for $C = C'$. However, the reversed statement $\mathrm{DTW}\big(C,C'\big) = 0 \Rightarrow C = C'$ does not hold in general. As a result, the coincidence property of metrics is violated.
- The triangle inequality is violated as well. The special case $C = (1,1,1)$ $C' = (1,1,0)$ and $C'' = (1,0,0)$ provides a counterexample where

$$\mathrm{DTW}\big(C,C'\big) + \mathrm{DTW}\big(C',C''\big) = 1 + 0 < \sqrt{2} = \mathrm{DTW}\big(C,C''\big) \quad .$$

In summary, despite its desirable properties DTW is **not** a metric. In the following, we will refer to non-negative and symmetric functions as (dis-)similarity or distance measures, respectively.

A comparison between DTW and ED for two Cylinder instances of the Cylinder-Bell-Funnel dataset is illustrated in Figure 2.6 and 2.7. The assumptions about z-normalization also apply to DTW since warping only effects the alignment on the time-axis. DTW's invariance against local phase shifts can drastically boost the classification quality on strongly warped data. The Cylinder-Bell-Funnel dataset is an example where the relative classification error of a one-nearest-neighbour classifier (1NN) decreases from $12.21\% \pm 3.66\%$ for ED to almost vanishing $0.47\% \pm 0.46\%$ for DTW. Another example is given by the Two-Pattern dataset [16] where we can improve from $9.48\% \pm 0.68\%$ to an immeasurable error of $0.00\% \pm 0.00\%$ over 50 shuffled splits of the dataset.[5] In summary, DTW should be favoured on the majority of datasets over ED despite its increased computational complexity.

[4] Note, the shortest path algorithm is indeed linear in the number of nodes and edges of the graph. However, this number is proportional to $|C| \cdot |C'|$ which results in quadratic runtime in terms of lengths.

[5] See Chapter 4 for a detailed description of the experiments and the used protocols.

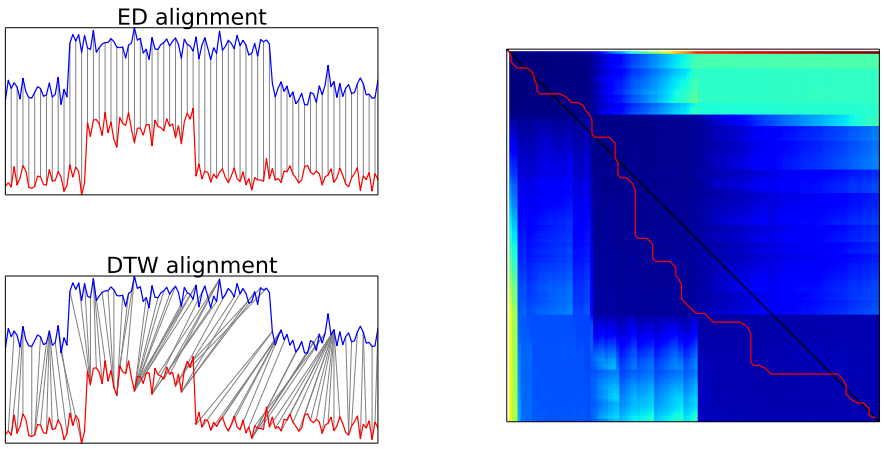


Figure 2.6: An example of two Cylinder instances from the Cylinder-Bell-Funnel dataset aligned with ED and DTW. In contrast to ED, DTW can capture the local phase shifts and therefore correctly maps the edges onto each other. The relaxed penalty matrix $M$ and the optimal warping path $\hat{\gamma}$ of DTW (red) and the diagonal path of ED (black) are illustrated on the right hand side. The used colormap (jet) ranges from blue (small values) over yellow (medium values) to red (high values).

### 2.1.4 Sakoe-Chiba-Constrained DTW

The proposed relaxation scheme for DTW determines the optimal warping path on the whole graph. Empirical studies [15] suggest that DTW's quality (in terms of 1NN-classification error) can be increased by restricting warping paths to the neighbourhood of the main diagonal. Common variants are the Sakoe-Chiba band and the less used Itakura parallelogram [24] which both exclude nodes on the upper right and lower left region of the penalty matrix (see Figure 2.8). As a result, pathological alignments are excluded which may increase classification quality.

The Sakoe-Chiba band restricts the indices of a candidate $C$ with a positive window $W \in \mathbb{N}_0$ such that $\max(i-W,0) \leq j \leq \min(i+W,|C'|-1)$ for all $j \in \mathrm{dom}(C')$. Historically, the global constraints were introduced to reduce the computational complexity of DTW from $\mathcal{O}(|C| \cdot |C'|)$ to $\mathcal{O}(W \cdot |C'|)$ [25]. Unfortunately, the asymptotic complexity remains the same since $W \propto |C|$. This can easily be seen if one considers a resampling of the time-axis of the involved time series by an arbitrary positive factor. The window $W$ has to be scaled by the same factor to ensure comparability. Furthermore, the external parameter $W$ has to be learned on a training set to provide optimal results. Nevertheless, in most cases the constrained variants of DTW outperform the unconstrained (full) DTW. As an example, the

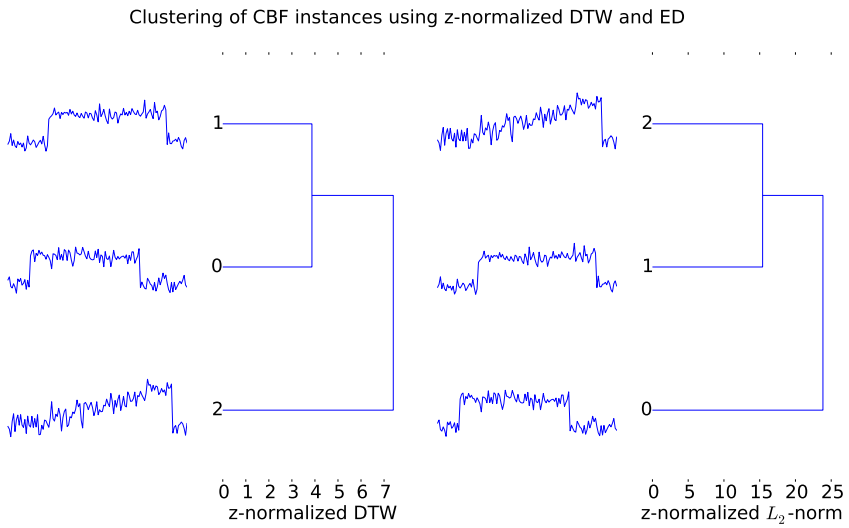


Figure 2.7: Agglomerative clustering of three instances from the popular Cylinder-Bell- Funnel dataset using complete linkage. On the left hand side, two Cylinder instances are correctly clustered together using z-normalized Euclidean-flavoured DTW. The remaining instance from the Bell class is excluded from the cluster. In contrast, z-normalized ED fails to predict the correct clustering on the right hand side due to the approximate global alignment of the Cylinder (1) and Bell (2) instance. The global shift on the time-axis between both Cylinder instances can only be captured by DTW.

relative 1NN-classification error of unconstrained DTW drops from $31.60\% \pm 1.59\%$ to $22.58\% \pm 1.61\%$ for Sakoe-Chiba-constrained DTW on the 50Words dataset in the UCR-Repository [16]. Another example for an impressive improvement can be observed during the 1NN-classification of the CinC_ECG_torso dataset (UCR-Repository) where constrained DTW performs with $4.80\% \pm 2.56\%$ in contrast to unconstrained DTW with far worse $29.76\% \pm 3.79\%$.[6] The literature even provides approaches how to learn the shape of global constraints [15] which is challenging due to the problem of overfitting to the training set. The pseudo-code for the Sakoe-Chiba-constrained DTW measure (CDTW) is listed in Figure 2.9.

### 2.1.5 Other Similarity Measures

A plethora of similarity measures for time series has been proposed during the last decades. The majority of the proposed algorithms for the global alignment of two time series are variations of Levenshtein's edit distance for the comparison of strings [26] (including DTW).

[6] See Chapter 4 for a detailed description of the experiments and the used protocols.

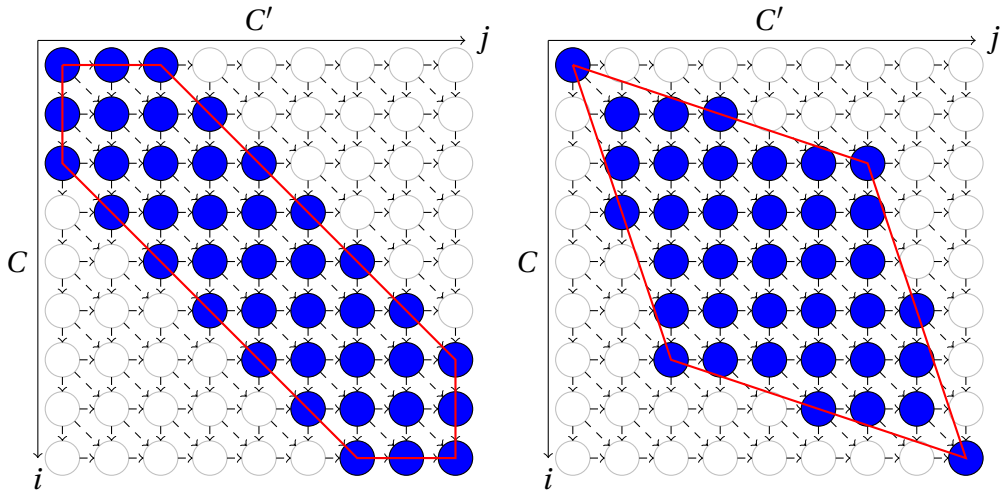


Figure 2.8: An example for the Sakoe-Chiba band constraint (left) and the Itakura parallelogram constraint (right). Both global constraints restrict the warping path to the neighbourhood of the main diagonal.

```
1:  function CDTW²(C, C', W)
2:      M ← array[|C| + 1, |C'| + 1]                          ▷ penalty matrix
3:
4:      for i = 0 → |C| do                                    ▷ initialize penalty matrix
5:          for j = max(i − W − 1, 0) → min(i + W + 1, |C'|) do
6:              M[i, j] ← ∞
7:          end for
8:      end for
9:      M[0, 0] ← 0
10:
11:     for i = 1 → |C| do                                    ▷ relax all nodes in the band
12:         for j = max(i − W, 1) → min(i + W, |C'|) do
13:             P ← min(M[i − 1, j − 1], M[i − 1, j], M[i, j − 1])
14:             M[i, j] ← P + (C_{i−1} − C'_{j−1})²
15:         end for
16:     end for
17:     return M[|C|, |C'|]
18: end function
```

Figure 2.9: Pseudo-code for the computation of the constrained DTW measure with Sakoe-Chiba restriction. The penalty matrix $M$ is only relaxed within the given Sakoe-Chiba band. The unconstrained DTW measure can be seen as a special case of CDTW where $W \to \infty$.

Edit distance with Real Penalty (ERP) is an example where Chen et al. equipped an elastic (time-warped) distance measure similar to DTW with a triangle inequality [27]. The same author introduced Edit Distance on Real sequence (EDR) [28] which provides an effective computation of the proposed similarity measure at the expense of accuracy. The compared time series are reinterpreted as strings where two values match if and only if they lie within a predefined $\epsilon$-neighbourhood. Longest Common Subsequence (LCSS) [29] and Minimum Variance Matching (MVM) [10] exhibit the relaxation of DAGs. The same is true for the symbolic matching algorithms of Needleman-Wunsch [30] and Smith-Waterman [31].

Furthermore, the literature provides a vast variety of similarity measures for time series utilizing geometric complexity [32], information theoretic approaches (approximation of Kolmogorov complexity via compression) [33] as well as variations of Dynamic Time Warping amongst others Derivative DTW [34], incomplete DTW relaxation [35], greedy DTW relaxation [36] and Δ-DTW (quotient of DTW and ED) [4].

## 2.2 Representations of Time Series

The introduced similarity measures in this chapter are based on the index-wise comparison of residues of the involved time series. However, other representations of the time series which do not use the actual values assigned to the time ticks are conceivable. For this purpose, consider distinct sequences of values $\hat{C}$ and $\hat{C}'$ closely related to the original time series $C$ and $C'$. An equivalent notion of distance $\hat{d}$ between $\hat{C}$ and $\hat{C}'$ can be defined by $\hat{d}(\hat{C},\hat{C}') := d(C,C')$. However, this theoretical construction lacks applicability unless we can provide explicit expressions for the calculation of the representation $\hat{C}$ and the involved similarity measure $\hat{d}$. From the computational view, one may ask if it is beneficial to compute a new representation and the associated distance measure to gain the same result. As we will see, there exist a class of representations that can be utilized to significantly speedup the calculation.

### 2.2.1 Unitary Bases of $\mathbb{C}^n$

The canonical representation of time series $C = (C_0,\ldots,C_{n-1})$ in $\mathbb{R}^n$ is equivalent to the decomposition of $C$ into axis-aligned (and thus real-valued) unit vectors:

$$C = \sum_{i=0}^{n-1} C_i \cdot \boldsymbol{e}_i \quad \text{where} \quad (\boldsymbol{e}_i)_j := \delta_{ij} \quad \text{and} \quad \delta_{ij} := \begin{cases} 1 & \text{if} \quad i = j \\ 0 & \text{if} \quad i \neq j \end{cases} .$$

The coefficients $C_i$ map exactly onto the entries of the given time series. The (standard) basis vectors $E := \{\boldsymbol{e}_i\}$ are mutually orthogonal and normalized to unit

length i.e. $\langle \boldsymbol{e}_i, \boldsymbol{e}_j \rangle = \delta_{ij}$ for all $i$ and $j$. Consider now another normed orthogonal basis $B := \{\boldsymbol{b}_j := \sum_{i=0}^{n-1} U_{ij} \cdot \boldsymbol{e}_i\}$ which can be obtained by a unitary transformation

$$U \in U(n, \mathbb{C}) := \{U \in \mathbb{C}^{n \times n} \mid U^\dagger \cdot U = \mathbb{1}\}$$

of the coordinate system i.e. length and angle preserving isomorphism on $\mathbb{C}^n$. As we will see, it is not mandatory that the transformation is fully complex such that in some cases a real-valued orthogonal transformation $O(n, \mathbb{R}) \subset U(n, \mathbb{C})$ is sufficient. Nevertheless, we discuss the general case to avoid redundancy in the following subsections. Consequently, we can calculate the new coefficient representation of $C$ in the basis $B$ by a simple matrix-vector multiplication:

$$C \stackrel{!}{=} \sum_{j=0}^{n-1} \hat{C}_j \cdot \boldsymbol{b}_j = \sum_{j=0}^{n-1} \hat{C}_j \cdot \big( \sum_{i=0}^{n-1} U_{ij} \cdot \boldsymbol{e}_i \big) = \sum_{i=0}^{n-1} \big( \sum_{j=0}^{n-1} U_{ij} \cdot \hat{C}_j \big) \cdot \boldsymbol{e}_i \quad \Rightarrow \quad C = U \cdot \hat{C} \quad .$$

This accounts for $\mathcal{O}(n^2)$ operations for each of the $N$ time series candidates $C^{(k)}$. Later in this section we will introduce special transformations that can be computed in $\mathcal{O}(n \cdot \log n)$ or $\mathcal{O}(n)$ time, respectively.

The Euclidean distance between two time series candidates $C$ and $C'$ is the same independent of the choice of the used unitary basis.

**Proposition 4** (isometries of complex finite vector spaces)**.** *Let $C \in \mathbb{C}^n$ be a time series of length n and E as well as B two unitary bases of $\mathbb{C}^n$. Further let $C_i$ and $\hat{C}_j$ be the coefficient representations of C according to E and B then*

$$\sum_{i=0}^{n-1} |C_i|^2 = \langle C, C \rangle = \langle \hat{C}, \hat{C} \rangle = \sum_{j=0}^{n-1} |\hat{C}_j|^2 \quad .$$

*Proof.* Exploiting the isometry property of $U(n, \mathbb{C})$ by a straight-forward substitution of $C = U \cdot \hat{C}$ and the duality property of the scalar product $\langle U \cdot \hat{C}, U \cdot \hat{C} \rangle = \langle \hat{C}, (U^\dagger \cdot U) \cdot \hat{C} \rangle$ on $\mathbb{C}^n$. The same is true for differences of time series candidates $C - C'$ since $U$ is a linear map. □

Thus, independent of the chosen unitary basis we can define a simple lower bound by restricting the index set to a subset.

**Proposition 5** (index set constrained lower bound)**.** *Let $C, C' \in \mathbb{C}^n$ be two time series of length n and $\mathcal{I} = \mathrm{dom}(C) = \mathrm{dom}(C')$ the index set of C and $C'$. ED constrained to any subset $\mathcal{J} \subseteq \mathcal{I}$ of indices is a lower bound of ED over the whole index set $\mathcal{I}$ i.e.*

$$\mathrm{LB}^2\big(C, C'\big) := \sum_{j \in \mathcal{J}} |C_j - C'_j|^2 \leq \sum_{i \in \mathcal{I}} |C_i - C'_i|^2 = \mathrm{ED}^2\big(C, C'\big) \quad .$$

*Proof.* Since the residues $|C_i - C'_i|$ are non-negative for all $i \in \mathscr{I}$ any partial sum of residues over $\mathscr{J}$ is smaller or equal than the full sum over all indices. □

Consequently, **any** partial sum of residues represented in **any** unitary basis is a lower bound of ED. Nevertheless, we have to provide a suitable basis and an appropriate constraint of the index set for the construction of tight lower bounds.

### 2.2.2 Principal Component Analysis

Consider an $N \times n$ data matrix $D$ with entries $D_{ki} := C_i^{(k)}$ representing $N$ real-valued time series candidates each of length $n$. The associated $n \times n$-shaped covariance matrix of $D$ can be calculated as square of the centred data matrix:

$$\Sigma := (D-\mu)^T \cdot (D-\mu) \quad \text{where} \quad (D-\mu)_{kj} := D_{kj} - \mu_j = D_{kj} - \frac{1}{N}\sum_{k=0}^{N-1} C_j^{(k)} \quad .$$

The average time series $\mu$ is subtracted from each time series candidate $C^{(k)}$ in every row. Thus, the centred time series $\tilde{C}_j^{(k)} := C_j^{(k)} - \mu_j$ add up to the zero-vector of length $n$. An eigenvalue decomposition with real eigenvectors can be computed since $\Sigma$ is a real-valued and normal ($\Sigma^T \cdot \Sigma = \Sigma \cdot \Sigma^T$) square matrix. The associated spectrum $\sigma(\Sigma) = \{\lambda_i \in \mathbb{R}_0^+\}$ is real due to symmetry ($\Sigma^T = \Sigma$) and positive semi-definite since $\langle x, \Sigma x\rangle = \langle (D-\mu)x, (D-\mu)x\rangle \geq 0$ for all $x \in \mathbb{R}^n$. Thus, the eigenvalue decomposition can be written as follows:

$$\Sigma = U^T \cdot \Lambda \cdot U \quad \text{where} \quad \Lambda = \text{diag}(\sigma(\Sigma)) \quad \text{and} \quad U \in O(n,\mathbb{R}) \subset U(n,\mathbb{C}) \quad .$$

As a result, the $n$ eigenvectors $\boldsymbol{b}_j$ (rows of $U$) each of length $n$ represent an orthogonal basis of $\mathbb{R}^n$. Sorting the eigenvectors according to a descending order of the eigenvalues one obtains the principal directions of the dataset. Great eigenvalues indicate that a reasonable amount of the data's variance is captured by the associated eigenvector. Eigenvectors with small eigenvalues can be neglected since they do not contribute significantly to the description of the data (see Figure 2.10). Finally, each (centred) time series can be transformed into the basis of eigenvectors $B = \{\boldsymbol{b}_j\}$. Restricting the orthogonal (and thus unitary) transformation $U$ to the top-$k$ eigenvectors $U_{\text{top-}k}$ by setting the remaining $n-k$ eigenvectors to zero, we obtain a lower bound for ED:

$$\text{ED}^2(U_{\text{top-}k}(C), U_{\text{top-}k}(C')) = \sum_{j=0}^{k-1} (\hat{C}_j - \hat{C}'_j)^2 \leq \text{ED}^2(C, C') \quad .$$

The factorization of the data matrix $\Sigma$ is usually obtained by a singular value decomposition of $(D-\mu)^T$ which accounts for asymptotic $\mathscr{O}(\min(n,N) \cdot n \cdot N)$ operations. The resulting runtime is computationally intractable for the majority

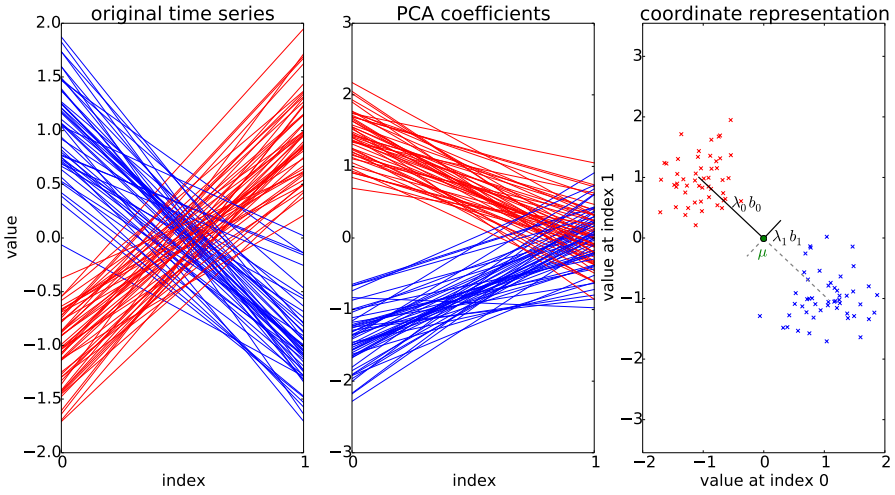


Figure 2.10: An example for short time series of length $n = 2$ which can be grouped into two classes. In the canonical representation (first panel) ED should be calculated on the whole index set since the restriction on the first index might not be distinctive. At least one time series of the blue class has a smaller distance to the red class than to its own. In contrast, the first coefficient (index 0) of the principal component representation (second panel) is sufficient for a reasonable approximation of ED. The second coefficient (index 1) does not contribute significantly to the measure and therefore can be neglected. The coordinate representation (last panel) emphasizes this observation. Obviously, the projection of the data onto the first basis vector $\boldsymbol{b}_0$ provides sufficient descriptive power.

of online algorithms. Furthermore, the decomposition only addresses the structure of the time series present in $D$. The factorization may not provide a suitable eigenvector when appending a new measurement to the dataset and thus has to be recomputed. Nevertheless, the principal component representation is ideally suited as a preprocessing step for the (unsupervised) clustering of time series due to its close relationship to the $k$-Means algorithm [37].

### 2.2.3 Discrete Wavelet Representations

Consider a slightly noisy time series $C$ with length $n = 2 \cdot m$ for a suitable $m > 0$. To get rid of the noise, one could average nearby values with a low-pass filter:

$$L\colon \mathbb{R}^n \to \mathbb{R}^m \ , C \mapsto L(C) \ := \left(\tfrac{C_0+C_1}{2}, \tfrac{C_2+C_3}{2}, \ldots, \tfrac{C_{n-2}+C_{n-1}}{2}\right) \quad .$$

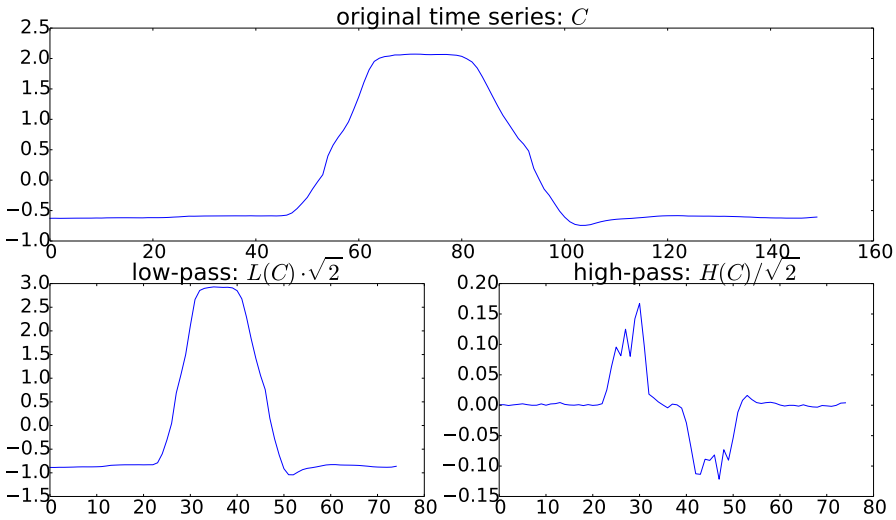


Figure 2.11: A time series candidate $C$ from the Gun-Point dataset of length $n = 150$ (panel on top). The even indices of $U(C)_{\text{even}} = \sqrt{2} \cdot L(C)$ correspond to the $n/2$ coefficients of the low-pass filter (left panel at the bottom). The coefficients at the odd indices represent the high-pass portion $U(C)_{\text{odd}} = \frac{1}{\sqrt{2}} \cdot H(C)$ (right panel at the bottom). The contributions from the high-pass filter are approximately one order-of-magnitude smaller since $C$ is a slowly varying signal. Thus, the restriction of ED on even indices is a tight lower bound of the whole ED measure.

Analogously, one could stress the differences of nearby entries by calculating the discrete derivative. The associated high-pass filter is given by:

$$H\colon \mathbb{R}^n \to \mathbb{R}^m,\, C \mapsto H(C) := \left(\tfrac{C_1 - C_0}{1}, \tfrac{C_3 - C_2}{1}, \ldots, \tfrac{C_{n-1} - C_{n-2}}{1}\right) \quad .$$

Both filters are linear and can be computed with only $\mathcal{O}(n)$ operations. Their matrix representations $L, H \in \mathbb{R}^{m \times n}$ are given as follows:

$$L := \frac{1}{2}\begin{pmatrix} 1 & 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 1 & 1 \end{pmatrix} \quad \text{and} \quad H := \begin{pmatrix} -1 & 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & -1 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & -1 & 1 \end{pmatrix} \quad .$$

The original time series $C$ can be reconstructed from the filtered signals $L(C)$ and $H(C)$. This can easily be proved by the construction of an orthogonal (and

thus unitary) transformation interleaving the rows of $L$ and $H$:

$$U : \mathbb{R}^n \to \mathbb{R}^n, C \mapsto U(C) := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 & \dots & 0 & 0 \\ -1 & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 1 & \dots & 0 & 0 \\ 0 & 0 & -1 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 & 1 \\ 0 & 0 & 0 & 0 & \dots & -1 & 1 \end{pmatrix} . \begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ \vdots \\ C_{n-2} \\ C_{n-1} \end{pmatrix} .$$

Assume that the amount of noise is negligible then the high-pass filtered signal is almost vanishing i.e. $H(C) \approx (0,0,\dots,0)$.[7] Consequently, the major contribution of the time series is located in the low-pass portion (see Fig 2.11). As a result, we can define a tight lower bound of ED by summing only over the even indices of $U(C)$ and $U(C')$. Further, we can append the odd indices at the end to obtain the original measure:

$$\mathrm{ED}(C,C')^2 = \mathrm{ED}^2(U(C),U(C')) = \mathrm{ED}^2_{\mathrm{even}}\big(U(C),U(C')\big) + \underbrace{\mathrm{ED}^2_{\mathrm{odd}}\big(U(C),U(C')\big)}_{\approx 0} .$$

This approach can be extended hierarchically to determine a reasonable approximation of $\mathrm{ED}(C,C')$ with far less than $\frac{n}{2}$ indices. The first published and most popular variant of a multi-resolution transformation is the discrete Haar filter [38]. For $n = 4$ its matrix representation has the following form:

$$U_{\mathrm{Haar}_4} := \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -\sqrt{2} & \sqrt{2} & 0 & 0 \\ 0 & 0 & -\sqrt{2} & \sqrt{2} \end{pmatrix}$$

Row zero corresponds to the low-pass filter and the remaining rows are equivalent to derivatives on different time scales. Again assuming slowly varying signals, the major contribution to the measure will be caused solely by the low-pass filter. Although the matrix is dense, $U(C)$ can be calculated with only $\mathcal{O}(n)$ operations using the lifting-scheme for discrete (second generation) wavelets [39].

Beyond the Haar filter, a plethora of discrete wavelets have been proposed in the last century (see Figure 2.12 for an example) amongst others the Daubechies family (guaranteeing a maximal number of vanishing moments) [40], the Symlet family (a symmetrized variant of the Daubechies family) [41] and biorthogonal wavelets (where $U$ is not necessarily unitary). Similar aggregation techniques that focus solely on the low-pass can be found in the literature. Piecewise Aggregate Approximation (PAA) [42] and Symbolic Aggregate approXimation (SAX) [7] are popular extensions of the discussed low-pass approximation.

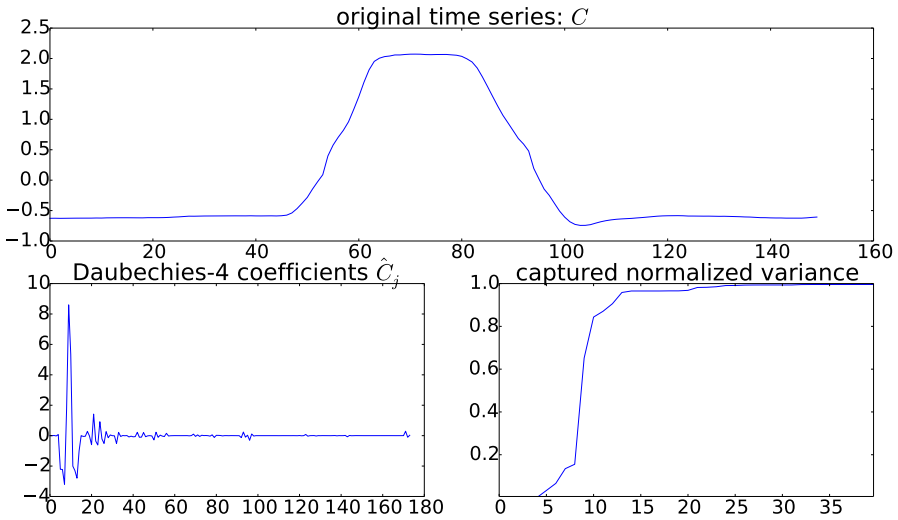


Figure 2.12: A time series candidate $C$ from the Gun-Point dataset of length $n = 150$ (panel on top). The coefficients representation in a Daubechies-4 wavelet basis exhibits an exponential decline in terms of coefficient amplitudes (left panel at the bottom). The decomposition filters are stated explicitly at `http://wavelets.pybytes.com/wavelet/db4/`. The captured normalized variance $\sum_{j=0}^{k-1} |\hat{C}_j|^2 / \sum_{j=0}^{n-1} |\hat{C}_j|^2$ saturates rapidly for a small number of used coefficients $k$ (right panel at the bottom). Thus, with only $k = 15$ coefficients we can describe more than 95% of ED's contribution.

Wavelet theory is not constrained to discrete domains and can also be developed for Hilbert spaces on continuous domains. An interested reader may refer to the comprehensive introductions by Daubechies [40] and Mallat [43].

### 2.2.4 Discrete Fourier Transform

In contrast to previous subsections that introduced real-valued orthogonal transformations $U \in O(n, \mathbb{R})$ for the construction of lower bounds, this section focuses on an actual unitary basis transformation. Thus, any real-valued time series $C \in \mathbb{R}^n$ has to be embedded into the complex vector space $\mathbb{C}^n$ by adding an imaginary part of zero for each time tick. A decomposition of $C$ into sinusoidal basis vectors

$$(\boldsymbol{b}_j)_i := \frac{1}{\sqrt{n}} \exp\left(\mathrm{i} 2\pi \frac{i \cdot j}{n}\right) \quad \text{where} \quad \mathrm{i} := \sqrt{-1}$$

[7] In particular, this is true for all constant time series $C$.

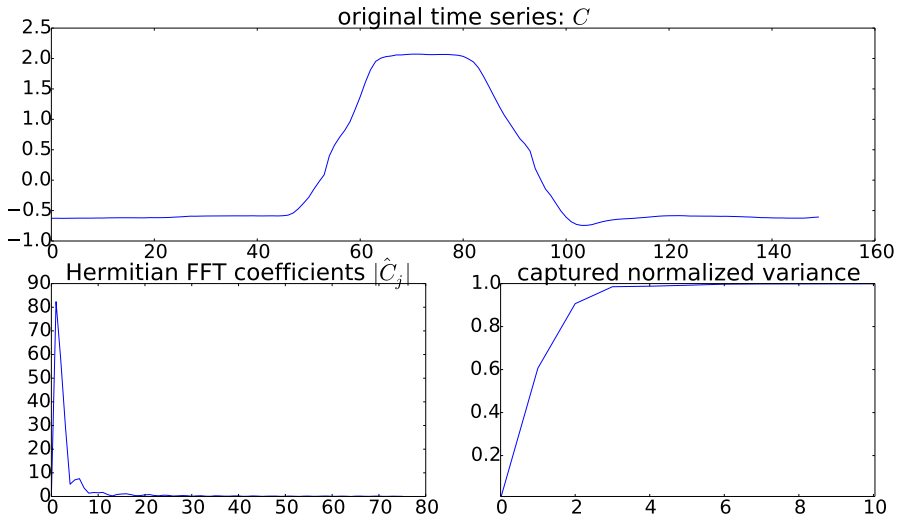


Figure 2.13: A time series candidate $C$ from the Gun-Point dataset of length $n = 150$ (panel on top). The Fourier coefficients representation exhibits an exponential decline in terms of coefficient amplitudes (left panel at the bottom). Since $C$ is z-normalized (zero mean) the first coefficient $C_0$ vanishes. The captured normalized variance $\sum_{j=0}^{k-1} |\hat{C}_j|^2 / \sum_{j=0}^{n-1} |\hat{C}_j|^2$ saturates rapidly for a small number of used coefficients $k$ (right panel at the bottom). Thus, with only $k = 6$ coefficients we can describe more than 99% of ED's contribution.

can be obtained by mapping the canonical coefficients by a unitary matrix $U \in U(n, \mathbb{C})$ with entries $U_{ji} = (\boldsymbol{b}_j)_i^*$ such that $\hat{C}_j = \sum_i U_{ji} C_i$. The common definition of the discrete Fourier transform (up to a normalization factor[8]) is given as follows:

$$\hat{C}_j = \frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} C_i \cdot \exp\left(-\mathrm{i}2\pi \frac{i \cdot j}{n}\right) \quad .$$

The substitution $j \mapsto n - j$ reveals a redundancy in the Fourier representation for real-valued signals: $\hat{C}_{n-j}^* = \hat{C}_j \ \forall j > 0$ (so-called Hermitian symmetry). As a result, it is sufficient to compute the first $\frac{n}{2}$ coefficients. Although $U$ is a dense matrix the matrix-vector product $U(C)$ can be computed with only $\mathcal{O}(n \cdot \log n)$ operations using Cooley et al.'s Fast Fourier Transform (FFT) published in 1965 [44]. This algorithm exploits the structure of $U$ by recursively splitting it into two sub-problems of half the size. Ironically, Carl Friedrich Gauss discovered an equivalent algorithm with the same asymptotic complexity more than 150 years earlier [45].

[8] Here, the normalization factor was chosen such that the basis vectors have unit length.

For a slowly varying signal the projection onto low frequency basis vectors (small $j$s) will be dominant.[9] Thus, we can safely truncate the sequence of Fourier coefficients by removing the almost vanishing contributions from high-frequencies (see Figure 2.13). Analogous to the previous sections, we can define a lower bound of ED by restricting the index set to the top-$k$ basis vectors. This procedure is equivalent to lossy compression techniques used in many audio/video encoders.

### 2.2.5 Indexing with Spatial Access Methods

The dimensional reduction to only a few coefficients allows for the indexing of time series with common spatial access methods amongst others R-trees [46] and kd-trees [47]. In this context, the use of the canonical representation with all $n$ data points is not advisable since the querying process degenerates to linear search within the database due to the so-called curse of dimensionality [48].

Furthermore, this approach assumes that dimensional reduction (PCA, Wavelet, FFT etc.) as well as the construction of the spatial access data structure can be carried out in a preprocessing phase. Similar indexing techniques exist for the use of elastic measures DTW and Longest Common Subsequence [49]. However, recent research on the subsequence alignment of time series in long data streams suggests that lower-bounded linear search is the most appropriate method in terms of memory consumption when providing support for arbitrary query lengths as stated by Keogh et al. [17]:

> *'Moreover, if we are interested in tackling a trillion data objects we clearly cannot fit any index in the main memory, much less all of them, or any two of them.'*

and execution speed during the querying process (same source):

> *'While our work has focused on fast sequential search, we believe that for DTW, our work is faster than all known indexing efforts.'*

Consequently, we focus on the acceleration of lower-bounded linear search during the rest of this thesis and neglect classical indexing techniques despite their superior properties on small data sets with hard-coded query length.

[9] For the special case of constant time series all coefficients except $\hat{C}_0 = \sqrt{n} \cdot \overline{C}$ vanish.

CHAPTER 3

# PARALLELIZATION OF THE UCR-SUITE

*'Making it faster (usually an order of magnitude, or no one cares). '* — Eamonn Keogh, *UC Riverside*

Euclidean Distance (ED) and Dynamic Time Warping (DTW) are cornerstones in the field of time series data mining. Many high-level algorithms like kNN-classification [13], clustering [6] or anomaly detection [7] make excessive use of these distance measures as subroutines. Furthermore, the vast growth of recorded data produced by automated monitoring systems or integrated sensors establishes the need for efficient implementations.

An example for tremendous amounts of time series data is PhysioBank[1] – a repository of over 60 datasets featuring over 40,000 recordings of annotated, digitized physiologic signals and time series [50]. The total quantity of the recorded data exceeds 4TB. Another popular source for time series data is the UCR-Repository [16] which consists of over 40 datasets from different domains including mass spectra, trajectories, power consumption records and leaf shapes.

Mining such amounts of time series data demands for exceedingly fast algorithms in theory and practice. Furthermore, many high-level data mining algorithms among others kNN-classifiers [13], classification with kernel machines [51], Kernel-k-Means as well as spectral clustering [52] rely on the pair-wise distance information between all time series in the database or training set. In these cases, the workload is proportional to the squared number of database entries $N$

[1] `http://www.physionet.org`

such that the processing of even small datasets can be computationally demanding. Let $T_d$ be the time consumption of the used similarity measure then the total time complexity is in $\mathcal{O}(N^2 \cdot T_d)$. Consequently, a moderate amount of only 100 database entries increases the runtime of the used high-level algorithm by a factor of 10,000 in comparison to the subroutine of comparing two time series.

In contrast to algorithms that employ pair-wise distance information between pairs of time series, one encounters algorithms that extract subsequence candidates from individual database instances [6] or an isolated stream [17]. On the one hand, a single time series of length $n$ consists of $\frac{n(n+1)}{2}$ subsequence candidates $C^{(k)}$ of all possible lengths $|C^{(k)}| \in \{1,\ldots,n\}$. Hence, the runtime scales quadratically independent of the choice of the used distance measure. On the other hand, consider a stream of time series data $S$ of length $n$ and a query $Q$ of length $m$ then there exist $n-m+1$ possible locations to align $Q$ in $S$. The resulting time complexity corresponds to $\mathcal{O}(n^2 \cdot T_d)$ in the former and $\mathcal{O}(n \cdot T_d)$ in the latter case. In both situations, we have to consider a significant increasing of the runtime in comparison to the ordinary (global) alignment of two time series.

There exist three major strategies to face the challenge of reducing the runtime for the above mentioned applications:

- **(Parallelization)** The total runtime always depends on the computational cost of the involved similarity measure $d$. Therefore, it is advisable to speedup its computation. This can be achieved with massively parallel algorithms on modern hardware accelerators. For this purpose, suitable computation schemes have to be designed. Nevertheless, the asymptotic runtime will remain the same such that the speedup needs to be considerable i.e. one or two orders-of-magnitude.

- **(Exploit Redundancy)** Redundant information during the computation can be exploited to lower the theoretical runtime of the algorithm. This strategy can be agonizing since there is no straight way to achieve this. Nevertheless, a mitigation in the asymptotic complexity during the processing of streams with millions of data points will be clearly noticeable. In combination with parallelization techniques enormous improvements in the performance are conceivable.

- **(Lower Bounding)** Many algorithms rely only on the smallest distance between a given query $Q$ and the set of candidates $C^{(k)}$ in a database or stream. As a result, we can abandon a computation of $d\big(Q,C^{(k)}\big)$ if it exceeds the value of a precomputed distance measure. This early-exit is possible since $d(Q,C^{(k)})$ is at least as big as the optimal distance $d^* := \min_k d\big(Q,C^{(k)}\big)$. Moreover, if we can find a distinct similarity measure LB which is computational less demanding and satisfies $\mathrm{LB}\big(Q,C^{(k)}\big) \leq d\big(Q,C^{(k)}\big)$ for all queries

$Q$ and all candidates $C^{(k)}$, we can use it to lower-bound the original similarity measure. Initially, one computes the lower bound for a candidate and decides whether the computation of the original similarity measure is still promising or not. Depending on the runtime and the pruning power of the used lower bound this technique can be a powerful tool to speedup alignment algorithms.

In summary, time series data mining is challenging – even modestly sized datasets may demand for vast computational resources. Nevertheless, powerful tools can be utilized to overcome these issues. In the following, we will combine efficient pruning techniques and massively parallel algorithm design to develop novel computation schemes for ED and DTW on modern hardware accelerators.

The rest of this chapter is organized as follows: Section 3.1 repeats the basic normalization and lower-bounding techniques for ED and DTW. Moreover, we give a brief introduction in the alignment of subsequences in huge streams followed by a detailed discussion of the UCR-Suite[17], the current state-of-the-art implementation of time series subsequence alignment. The parallelization of subsequence ED and DTW on CUDA-enabled accelerators is presented in section 3.2. Finally, a performance evaluation of our implementation in comparison to the UCR-Suite is given in Section 3.3.

## 3.1 The UCR-Suite

Among other important tasks in time series data mining, stream monitoring is a crucial application for the inspection of the measured data [53]. Usually, a system of sensors records consecutive values over time with no further information about the signal itself i.e. annotations, regions of interest or changes of state in the underlying process are completely lacking. Nevertheless, a concerned data miner might be interested in the building blocks of the data in order to reveal its latent structure. Questions about the periodicity of the stream, the potential occurrence of anomalies [7] or characteristic segments to be used in clustering or classification arise [6]. Hence, we pursue a high-level description of the measured data.

A popular example is given by the long-term measurement of electrocardiograms (ECG). The recording of the signal is often accomplished with only little or no supervision by a domain expert. Usually, a cardiologist annotates some heartbeats right after the calibration but will leave the remaining measurement unsupervised. As a result, a predominant part of the measured stream holds no meta-data at all. Analogously to the above mentioned problems, one would like to determine the exact location of each individual heartbeat or the most unusual heartbeat during the whole measurement. Figure 3.1 illustrates this idea on a small excerpt of an ECG stream. Further tasks may be the automated grouping

into healthy and abnormal beats for the construction of a classifier to enable future investigations of the patient's recovery.

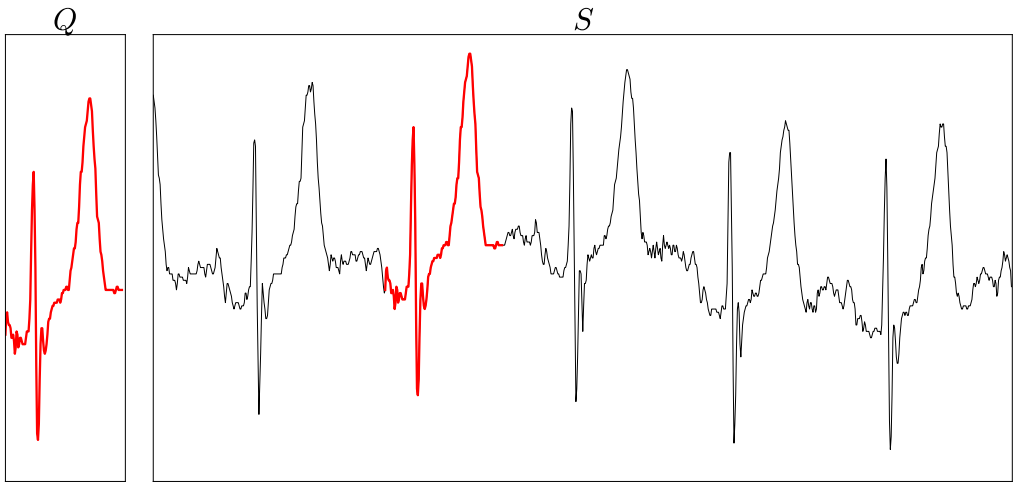


Figure 3.1: Local alignment of a heartbeat query (red/bold) in a stream of ECG data (grey/thin) using locally z-normalized ED.

Another example is given by the recording of thickness profiles of produced metal bands during the hot-rolling of steal. The metal is deformed in several rolling steps to obtain the final product. For this purpose, the primal steal ingots are heated in an oven to achieve a proper temperature for the deformation. Small variations in the spatial distribution of the temperature may cause irregularities in the thickness which becomes noticeable in characteristic spikes. These occasionally occurring skidmarks can drastically reduce the quality of the product which may cause severe economical losses. Given their shape one may ask if these spikes are present and for the corresponding location in the measured stream. Further considerations may involve the construction of automated skid-mark detectors and the design of robust quality measures (see Figure 3.2).

Consequently, the alignment of characteristic shapes (heart beats, skidmarks etc.) in long data streams is the fundamental task to be accomplished for the construction of high-level representations of the involved time series. The UCR-Suite [17], a recent approach by Rakthanmanon et al., is to our best knowledge the fastest implementation of locally z-normalized subsequence alignment under ED and DTW and thus needs further investigation. It uses sophisticated techniques to speedup the computation amongst others lower bound cascades and on-the-fly z-normalization. During the following sections, the used techniques and their asymptotic complexities are discussed in detail.

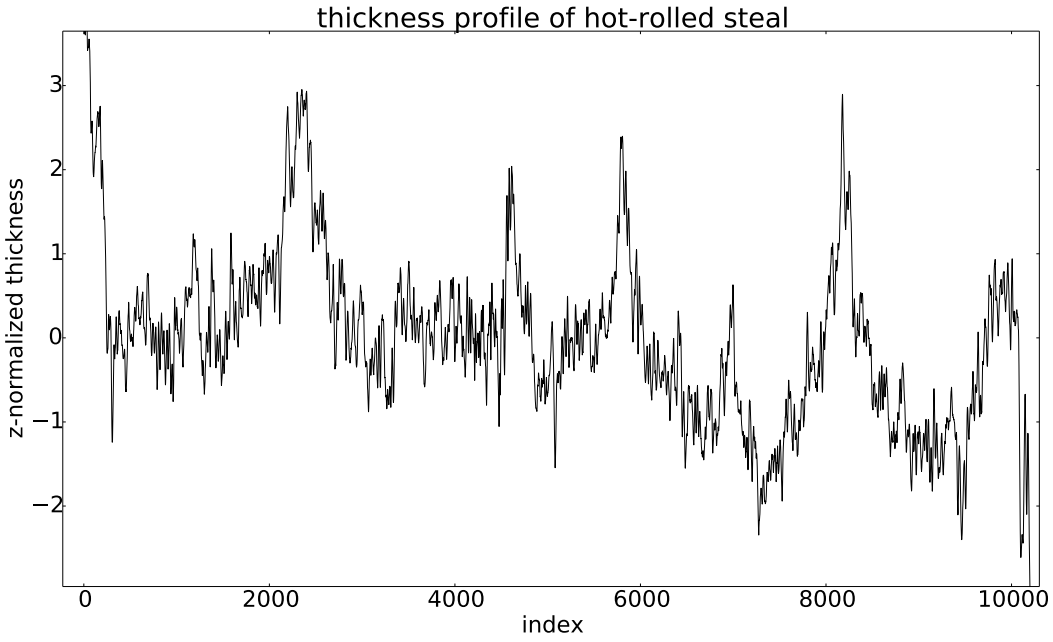


Figure 3.2: An example for a z-normalized thickness measurement of a metal coil during the hot-rolling of steal. At least five obvious sharp spikes (skidmarks) can be found in the measurement that may decrease the quality of the final product.

### 3.1.1 Subsequence Alignment in Streams

A straight-forward approach for the calculation of the best alignment position of a query $Q$ in a long stream $S$ is calculating the distances for each of the subsequence candidates $C^{(k)}$ by shifting an extraction window of fixed size $|Q|$ over the whole stream. As we have seen in the previous chapter, z-normalization is a crucial preprocessing step and thus each of the $|S|-|Q|+1$ possible subsequences have to be normalized independently[2] before they can be compared to the query. The optimal alignment position is determined by the smallest distance measure amongst all possible candidates. Figure 3.3 states the pseudo-code of the proposed algorithm.

Let $T_z$ be the cost of the normalization step and $T_d$ the amount of time for the calculation of the involved distance measure then the overall computational cost has an asymptotic time complexity of $\mathcal{O}\big(|S| \cdot (T_z + T_d)\big)$. ED and z-normalization can be processed in $\mathcal{O}(|Q|)$ time, yet DTW accounts for $\mathcal{O}(|Q|^2)$ operations. Thus the runtime is at least in $\mathcal{O}(|S| \cdot |Q|)$ and upper-bounded by $\mathcal{O}(|S| \cdot |Q|^2)$. The resulting runtime may be computationally intractable even for moderately sized databases. The naïve approach has another drawback: Imagine we introduced

[2] Please note, it is not sufficient to simply z-normalize the whole stream at once.

```
1: function SUBSEQUENCE_ALIGN(Q, S)
2:     Q ← normalize(Q)                          ▷ Q normalized once in preprocessing
3:     distances ← array[|S| − |Q| + 1]                  ▷ distance for each position
4:
5:     for k = 0 → |S| − |Q| do                          ▷ for each alignment position
6:         C ← normalize(S[k : k + |Q|])                 ▷ extract alignment candidate
7:         distances[k] = distance_measure(Q, C)                 ▷ compute distance
8:     end for
9:
10:    k̂ ← argmin(distances)                            ▷ pick best alignment position
11:    d̂ ← distances[k̂]                                ▷ pick best alignment distance
12:    return (k̂, d̂)
13: end function
```

Figure 3.3: Pseudo-Code of a general subsequence alignment algorithm.

an extraordinarily fast similarity measure $d_{\text{magic}}$ that can be computed in zero time, the normalization step would still need $\mathcal{O}(|S| \cdot |Q|)$ operations. As a result, a separate computation of the stream's local statistics is advisable.

### 3.1.2 Windowed z-Normalization

The empirical mean $\mu_k$ and standard deviation $\sigma_k$ of a time series candidate $C^{(k)}$ are defined as:

$$\mu_k := \sum_{i=0}^{|Q|-1} p_i \, C_i^{(k)} \quad \text{and} \quad \sigma_k^2 := \sum_{i=0}^{|Q|-1} p_i \, (C_i^{(k)} - \mu_k)^2 \quad \text{where} \quad p_i := \frac{1}{|Q|}.$$

Rewriting the equations as expectations of a uniformly distributed random variable i.e. $\mu_k = E\big(C^{(k)}\big)$ and $\sigma_k^2 = E\big((C^{(k)} - E(C^{(k)})^2\big)$ we can derive an alternative formula for $\sigma_k$ since $E$ is an idempotent and linear map:

$$\sigma_k^2 = E\big((C^{(k)})^2\big) - E\big(C^{(k)}\big)^2 = \frac{1}{|Q|} \sum_{i=0}^{|Q|-1} \big(C_i^{(k)}\big)^2 - \mu_k^2 \quad .$$

On the first sight, there is no benefit: The mean $\mu_k$ and standard deviation $\sigma_k$ still account for $\mathcal{O}(|Q|)$ operations. However, we can extend the formulas to the whole stream by substituting $C_i^{(k)}$ by $S_{i+k}$. Using a discrete version of the fundamental theorem of calculus we can reinterpret the sums over $S_j$ and $S_j^2$ as differences of

the integral functions $F$ and $G$ evaluated at the borders:

$$\frac{1}{|Q|}\sum_{i=0}^{|Q|-1} S_{i+k} = \frac{1}{|Q|}\left(\sum_{j=0}^{|Q|-1+k} S_j - \sum_{j=0}^{k-1} S_j\right) = \frac{F(k+|Q|)-F(k)}{|Q|} \quad ,$$
$$\frac{1}{|Q|}\sum_{i=0}^{|Q|-1} S_{i+k}^2 = \frac{1}{|Q|}\left(\sum_{j=0}^{|Q|-1+k} S_j^2 - \sum_{j=0}^{k-1} S_j^2\right) = \frac{G(k+|Q|)-G(k)}{|Q|} \quad .$$

Both, $F$ and $G$, can be computed with exclusive prefix sums in $\mathcal{O}(|S|)$ time. Rakthanmanon et al. [17] and Chan et al. [54] mention the risk of numerical instabilities during the computation of the proposed calculation scheme for the standard deviation if $\sigma$ is small and the number of time ticks is large. In cases where double precision cannot guarantee the desired accuracy for the whole stream $S$, a partitioned calculation of the prefix sums $F$ and $G$ is conceivable. Thus, we process the windowed standard deviation and mean independently on batches of a fixed size. Neglecting the small overlaps between the individual batches, the computational effort is the same due to the associativity of the sum operation.

Thus, when moving the computation of the stream statistics to the preprocessing phase, the asymptotic time complexity of the general subsequence alignment algorithm can be modified to $\mathcal{O}(|S| \cdot T_d + |S|) = \mathcal{O}\big(|S| \cdot (T_d + 1)\big)$. Assuming again, $d_{\text{magic}}$ to perform distance calculations with zero operations, the runtime would be reduced to the $\mathcal{O}(|S|)$ contribution of the z-normalization. Although $d_{\text{magic}}$ cannot possibly exist, we will see in subsection 3.2.2, that it is indeed possible to calculate ED with only logarithmic[3] effort per time tick.

### 3.1.3 Warping Envelopes

The (constrained) DTW similarity measure with its $\mathcal{O}\big(|Q|^2\big)$ contribution per alignment position is considered to be superior to ED in means of quality on the majority of data sets. The overall asymptotic runtime of $\mathcal{O}\big(|S| \cdot |Q|^2\big)$ may be too demanding, especially for long queries. Thus, we need to construct efficient lower bounds analogous to the unitary transformation in Chapter 2. However, the elastic mapping of indices prohibits the same approach unless we can rewrite DTW's warping property in terms of a lock-step assignment of indices. A naïve approach would be the following [55]:

1. Calculate the global maximum and minimum $Q_-$ and $Q_+$ of the query.
2. During the comparison of a candidate sequence $C$ with $Q$, assume that each value in $C$ can be ideally assigned to a value in $Q$ as long as $C$ exclusively takes values in $[Q_-, Q_+]$. In this case, we have constructed the trivial zero lower bound, since the resulting measure does not contribute at any point.

[3] Note, $\log_2 10^9 < 30$ – that is almost as fast as $d_{\text{magic}}$.

3. Assume now, that at least one value $C_i$ is greater than $Q_+$. Since warping only affects the time-axis, there must be a non-vanishing contribution, namely $(C_i - Q_+)^2$. It is smaller than the actual DTW measure because we over-optimistically assumed ideal warping for values in $[Q_-, Q_+]$.

4. Extend this approach to all values $C_i$ that are either greater than $Q_+$ or smaller than $Q_-$. By the same argument, the sum over the non-vanishing contributions is a lower bound of DTW.

Recall DTW's bounding property i.e. the first and last indices have to be matched onto each other. As a result, we can define $\mathrm{LB}_{\mathrm{Kim}}$ [56] (see Figure 3.4) as:

$$\mathrm{LB}^2_{\mathrm{Kim}}(Q,C) := (Q_0 - C_0)^2 + (Q_{|Q|-1} - C_{|Q|-1})^2 + \sum_{i=1}^{|Q|-2} \begin{cases} (C_i - Q_+)^2 & \text{if } C_i > Q_+ \\ (C_i - Q_-)^2 & \text{if } C_i < Q_- \\ 0 & \text{else} \end{cases} .$$

Please note, although $\mathrm{LB}_{\mathrm{Kim}}$ is not symmetric in the arguments $Q$ and $C$, their roles may be reversed while still obtaining a lower bound for DTW in $\mathcal{O}(|Q|)$ time.

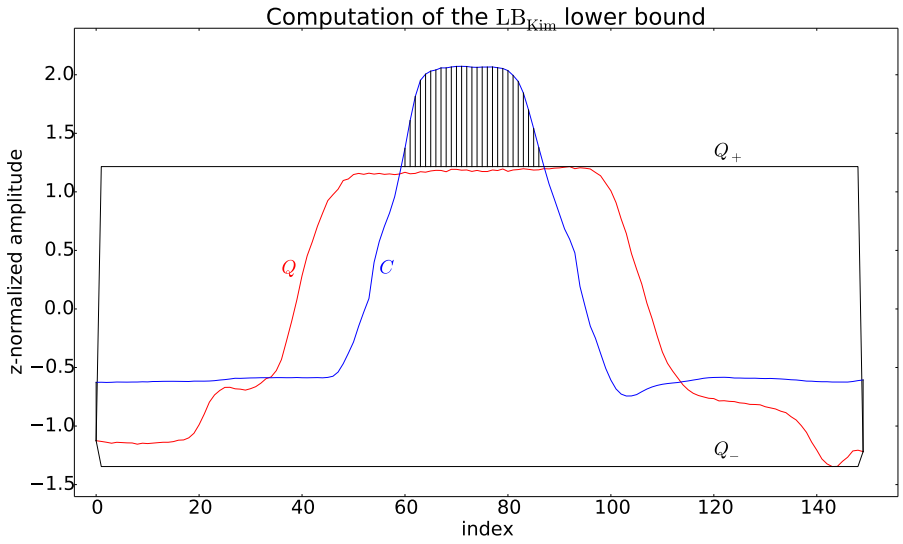


Figure 3.4: An example for two z-normalized time series from the Gun-Point dataset. The classic $\mathrm{LB}_{\mathrm{Kim}}$ lower bound calculates the differences between the candidate sequence $C$ and the so-called warping-envelope of the query $Q$. Here, the measure only contributes non-vanishing residues in the middle and at the endpoints of both time series.

Furthermore, one may utilize the proposed lower bound for indexing with spatial access methods by restricting $\mathrm{LB}_{\mathrm{Kim}}$ to the 4−tuple $(C_0, C_{|Q|-1}, C_-, C_+)$ for

each subsequence candidate $C$. Unfortunately, the pruning power of this low-dimensional representation will most likely be insufficient on z-normalized data due to the following observation:[4] Each z-normalized subsequence candidate $C$ has zero mean and a standard deviation of one. Assuming similar statistical distributions for the values of different alignment candidates, the global extrema $C_+$ and $C_-$ are approximately constant as mentioned in [17].

> '*The original definition of* $\mathrm{LB}_{\mathrm{Kim}}$ *also uses the distances between the maximum values from both time series and the minimum values between both time series in the lower bound, making it* $\mathcal{O}(n)$*. However, for normalized time series these two extra values tend to be tiny and it does not pay to compute them, and ignoring them allows the bound to be* $\mathcal{O}(1)$*, a fact we will exploit below.*'

For that reason, only the differences in the first and last index contribute reasonably. This tiny fraction of the whole index set renders the measure useless, especially for long queries.

In order to improve the situation, the UCR-Suite [17] uses a modified variant[5] of $\mathrm{LB}_{\mathrm{Kim}}$ that accumulates the warped contributions of the first and last $k$ indices. To achieve this, the penalty matrix of residues $M[i,j] = (Q_i - C_j)^2$ is filled with zeros except for the upper left and lower right $k \times k$ submatrices. Afterwards, the optimal warping path is calculated independently on both remaining squares (see Figure 3.5). The sum of both measures is a lower bound of the unconstrained DTW measure since it neglects the potentially non-vanishing contributions on the remaining part of the penalty matrix. Consequently, the following inequality chain holds for all $Q$ and $C$ and $W \geq 0$:

$$\mathrm{LB}_{\mathrm{Kim}}\big(Q,C\big) \leq \mathrm{DTW}\big(Q,C\big) \leq \mathrm{CDTW}\big(Q,C,W\big) \quad .$$

In practice, $k$ is chosen to be a small constant such that $\mathrm{LB}_{\mathrm{Kim}}$ can be processed in $\mathcal{O}(k^2) = \mathcal{O}(1)$ time. However, this modified lower bound may also perform inefficiently during the pruning of very long time series since it only considers a small fraction $\frac{2k}{|Q|}$ of the whole index set.

### 3.1.4 Lemire's Efficient Streamed Min-Max-Algorithm

It seems to be hard to construct tight lower bounds for the unconstrained DTW measure where we have to consider either global extrema or only small fractions of the index set. This can be explained as follows: Full DTW allows for arbitrarily

[4] Note the similarity to the infeasibility of the reversed triangle inequality for zED (see 2.1.2), where we obtain the trivial zero lower bound.

[5] During the rest of this thesis, we focus on the modified variant and omit the classic $\mathrm{LB}_{\mathrm{Kim}}$ lower bound.

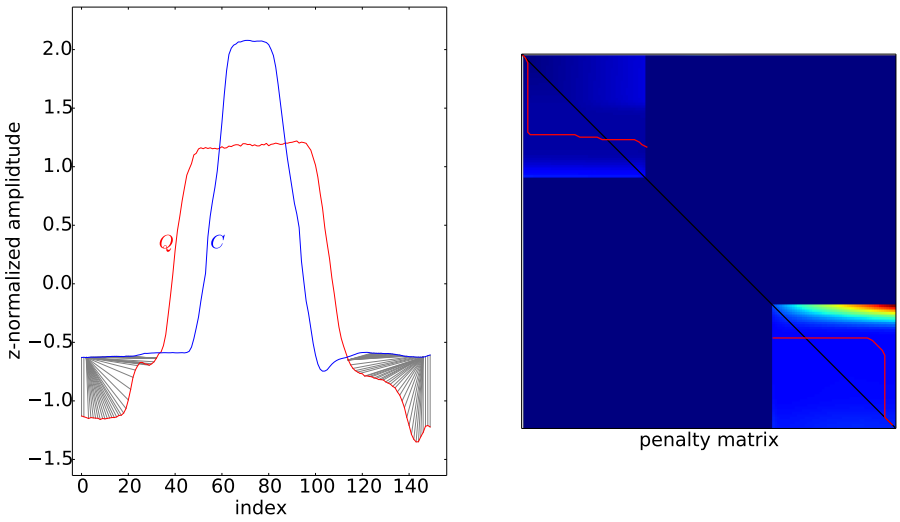


Figure 3.5: An example for two z-normalized time series from the Gun-Point dataset. The modified $\mathrm{LB}_{\mathrm{Kim}}$ lower bound calculates the unconstrained DTW between the candidate sequence $C$ and the query $Q$ only at the upper left and lower right $k \times k$ squares of the penalty matrix. In this illustration, we chose $k = 50$ for a better visualization. However, in practice $k$ is set to small constants e.g. $k = 5$. The used colormap (jet) ranges from blue (small values) over yellow (medium values) to red (high values).

big dilatation and contraction of the time-axis. In the worst case, all indices but one (the first) of a query could be mapped onto the last index of a candidate sequence. As a result, we have either to calculate the actual contribution or assume that the contribution will vanish. The first approach degenerates to a complete DTW relaxation and the latter results in an ineffective lower bound for long queries. However, the situation can be improved significantly for constrained elastic measures.

Recall the Sakoe-Chiba-constrained variant of DTW discussed in Chapter 2. It restricts the warping path to the neighbourhood of the main diagonal. For a given index of the candidate $l$ the warping of indices is limited to a defined region in the interval $\Omega(l) := \left[\max(0, l - W), \min(|Q| - 1, l + W)\right]$ of maximal length $2 \cdot W + 1$. As a result, one could calculate the windowed extrema $L_l$ and $U_l$:

$$(L_l, U_l)_Q := \left(\min_{k \in \Omega(l)} Q_k, \max_{k \in \Omega(l)} Q_k\right) \quad \text{for all } l \in \{0, \ldots, |Q| - 1\} \quad .$$

The constructed time series $L$ and $U$ describe a local (time-dependent) warping envelope in contrast to the global assignment $L = Q_-$ and $U = Q_+$ of the classic

$\mathrm{LB}_{\mathrm{Kim}}$ lower bound. By the same argument, we can define another lower bound since the delocalization of an index $l$ is bound to $\Omega(l)$ during the warping of the time-axis [24] (see Figure 3.6)[6]

$$\mathrm{LB}_{\mathrm{Keogh}}\big((L,U)_Q, C\big) := \sum_{l=0}^{|Q|-1} \begin{cases} (C_l - U_l)^2 & \text{if } C_l > U_l \\ (L_l - C_l)^2 & \text{if } C_l < L_l \\ 0 & \text{else} \end{cases} .$$

Similar to $\mathrm{LB}_{\mathrm{Kim}}$ the roles of $Q$ and $C$ could be interchanged while still obtaining

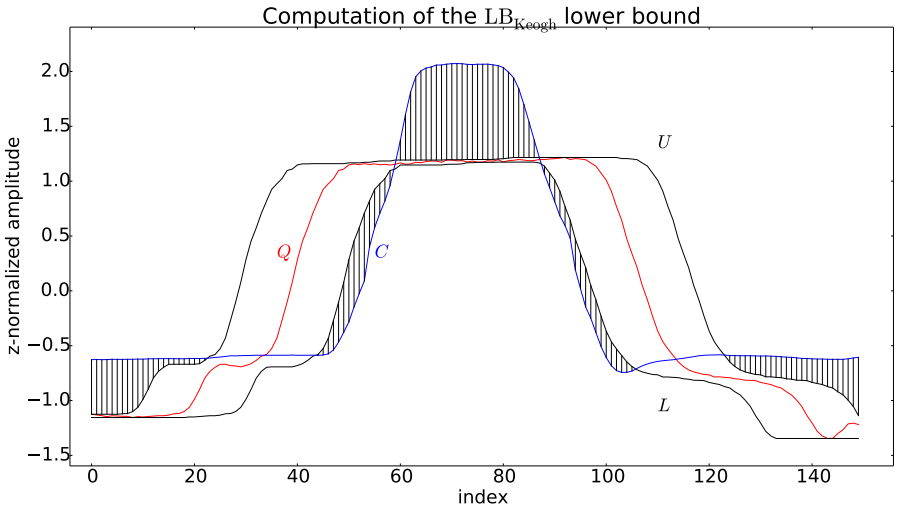


Figure 3.6: An example for two z-normalized time series from the Gun-Point dataset. The $\mathrm{LB}_{\mathrm{Keogh}}$ lower bound calculates the accumulated square of residues between the candidate sequence $C$ and the warping envelope $(L,U)_Q$ of the query. In this illustration, we chose the Sakoe-Chiba constraint to locally restrict the warping path to $W = 10$ indices to the left and right. In contrast to the classic $\mathrm{LB}_{\mathrm{Kim}}$ lower bound, the resulting measure considers a bigger contribution of residues and thus is considered to be tighter (see Figure 3.4).

a lower bound. The computation of the warping envelope $(L,U)_Q$ can naïvely be accomplished with $\mathcal{O}\big(W \cdot |Q|\big) = \mathcal{O}\big(|Q|^2\big)$ operations by the local determination of the extrema in $\Omega(l)$ for all indices $l$. This will not influence the runtime drastically since $(L,U)_Q$ has only to be computed once at the beginning of the alignment task. However, the situation is completely different if we allow to interchange the roles of $Q$ and $C$. The local computation of $(L,U)_C$ would demand for $\mathcal{O}\big(|Q|^2\big)$ at each alignment position resulting in an overall runtime of $\mathcal{O}\big(|S| \cdot |Q|^2\big)$ – as bad as the brute-force computation of DTW itself.

[6] A rigid proof can be found in the corresponding publication.

```
1: function LEMIRE-MIN-MAX(Q, W)
2:     L̂, Û ← [0], [0]                          ▷ double-ended queues for the indices
3:     L, U ← array[|Q|], array[|Q|]              ▷ resulting warping envelope
4:
5:     for i = 1 → |Q| − 1 do
6:         if i > W then                ▷ reporting the extrema after leaving the window
7:             U[i − W − 1] ← Q_front(Û)
8:             L[i − W − 1] ← Q_front(L̂)
9:         end if
10:        if Q_i > Q_{i−1} then                    ▷ readjustment of monotonicity
11:            Û.pop_back()
12:            while Q_i > back(Û) do
13:                Û.pop_back()
14:            end while
15:        else
16:            L̂.pop_back()
17:            while Q_i < back(L̂) do
18:                L̂.pop_back()
19:            end while
20:        end if
21:
22:        L̂.push_back(i)            ▷ finally add the new index at the right position
23:        Û.push_back(i)
24:
25:        if i = 2 · W + 1 + front(Û) then   ▷ remove indices when not present in window
26:            Û.pop_front()
27:        else if i = 2 · W + 1 + front(L̂) then
28:            L̂.pop_front()
29:        end if
30:    end for
31:
32:    for i = |Q| → |Q| + W do                ▷ write out the remaining W indices
33:        U[i − W − 1] ← Q_front(Û)
34:        L[i − W − 1] ← Q_front(L̂)
35:
36:        if i − front(Û) ≥ 2 · W + 1 then
37:            Û.pop_front()
38:        end if
39:        if i − front(L̂) ≥ 2 · W + 1 then
40:            L̂.pop_front()
41:        end if
42:    end for
43:    return (L, U)_Q
44: end function
```

Figure 3.7: Pseudo-code for the efficient computation of the warping envelope.

Fortunately, Lemire [57] proposed a streamed min-max algorithm which can carry out the same task with no more than $3 \cdot |Q|$ operations. The basic idea behind this approach can be explained as follows: One maintains two separate lists of indices $\hat{L}$ and $\hat{U}$ that describe monotonic (not necessarily consecutive) subsequences of the values $Q_l$ from a certain position. Exemplary, $\hat{L}_0$ indicates that $Q_{\hat{L}_0}$ is the global minimum in the interval $[0, W-1]$. Furthermore, $\hat{L}_1$ encodes recursively the index of the global minimum in $[\hat{L}_0, W-1]$ and so on. As a result, $Q_{\hat{L}_m} \leq Q_{\hat{L}_{m+1}}$ for all $m$ such that $(Q_{\hat{L}_0}, Q_{\hat{L}_1}, \ldots)$ is a monotonic non-decreasing sequence. The same is carried out for the global maxima i.e. $(Q_{\hat{U}_0}, Q_{\hat{U}_1}, \ldots)$ is a monotonic non-increasing sequence. Hence, the windowed extrema on $[0, W-1]$ can be reported as $L_0 = Q_{\hat{L}_0}$ and $U_0 = Q_{\hat{U}_0}$. When shifting the alignment window the lists $\hat{L}$ and $\hat{U}$ have to be updated accordingly by popping indices from the front (if $\hat{L}_0$ or $\hat{U}_0$ are no more present in the window) and back (if the new value $Q_W$ violates the monotonicity). Figure 3.7 states the pseudo-code.

Exploiting the monotonicity of the sequences $(Q_{\hat{L}_m})_m$ and $(Q_{\hat{U}_m})_m$, every value $Q_l$ is compared amortized three times. As a result, the reversed variant $\mathrm{LB}_{\mathrm{Keogh}}\big((L,U)_C, Q\big)$ can be calculated in only linear time per alignment position independent of the chosen window $W$. Moreover, we can reduce the asymptotic cost for the calculation of all warping envelopes from $\mathcal{O}\big(|S| \cdot |Q|\big)$ to $\mathcal{O}\big(|S|\big)$ with only minor losses in the tightness of the lower bound.[7] Assume we calculate the warping envelope $(L,U)_S$ on the whole stream without applying z-normalization instead of doing the same for each z-normalized subsequence candidate $z\big(C^{(k)}\big)$. How can we relate the subsequences of the global envelope $(L,U)_S$ restricted to the interval $[k, k+|Q|-1]$ to the individual local envelopes $(L,U)_{z(C^{(k)})}$? Since the maximum and minimum operations commute index-wise with z-normalization

$$\max\Big(\frac{x-\mu}{\sigma}\Big) = \frac{\max(x)-\mu}{\sigma} \quad \text{for all} \quad x, \mu \in \mathbb{R}, \sigma > 0 \quad ,$$

it does not matter which is applied first. Furthermore, the readjusted global envelope encloses all local envelopes due to the following observation:

- Consider the indices of $(L,U)_{z(C^{(k)})}$ that lie apart from the borders of the subsequence candidate by at least $W$ positions, then $(L,U)_{z(C^{(k)})}$ and $(L,U)_S$ have exactly the same values after z-normalization since they determine the windowed extrema on that portion of the stream and consequently agree.

- Values at indices near the borders of the subsequence candidate (i.e. less than $W$ positions apart) may differ slightly since $(L,U)_{z(C^{(k)})}$ does not consider the overlap during the shifting of the window. However, the maximum can only increase when considering additional values from the halo and analogously the minimum can only decrease.

[7] To our best knowledge, this idea has not been published so far.

Thus, the local envelope of $C^{(k)}$ must be enclosed index-wise by the global envelope of $S$ (see Figure 3.8). The tightness of the lower bound is almost the same, since the envelope may differ only on $2 \cdot W$ indices.[8] Similar to z-normalization with prefix sums, we can substitute the computation of the local envelopes by a global computation in a preprocessing phase with only $3 \cdot |S|$ comparisons.

As we have seen, the envelopes are compared in a lock-step manner during the evaluation of $\mathrm{LB}_{\mathrm{Keogh}}$. Since the windowed extrema $(L,U)_C$ are locally smoothed variants of the candidate sequence $C$, dimensional reduction techniques can be utilized using e.g. piecewise constant approximations of the envelope [7, 42] or spectral approximations like PCA and Wavelet/Fourier coefficients as seen in Chapter 2. Unfortunately, for each possible length of $Q$ an indexing structure has to be constructed. Thus, this approach renders intractable in terms of memory when providing support for queries of arbitrary length.

### 3.1.5 Assembly of the Building Blocks

We have seen in the previous subsections, that the local statistics and warping envelopes can be computed with only linear cost in a preprocessing phase right before the actual alignment task (see Table 3.1). Since we do not expect a subsequence similarity measure to find an optimal alignment in less than $\mathcal{O}(|S|)$ time,[9] z-normalization $(\mu_k, \sigma_k)$ and the computation of the envelopes $(L,U)_Q, (L,U)_S$ are not discussed in the following but can be considered to be given implicitly whenever needed. The UCR-Suite consists of two major time series alignment

| preprocessing step | naïve runtime | final runtime | technique |
|---|---|---|---|
| z-normalization of $Q$ | $\mathcal{O}(\|Q\|)$ | $\mathcal{O}(\|Q\|)$ | – |
| z-normalization of all $C^{(k)}$ | $\mathcal{O}(\|S\| \cdot \|Q\|)$ | $\mathcal{O}(\|S\|)$ | prefix sums |
| warping envelope of $Q$ | $\mathcal{O}(\|Q\|^2)$ | $\mathcal{O}(\|Q\|)$ | Lemire-$3n$ |
| warping envelope of all $C^{(k)}$ | $\mathcal{O}(\|S\| \cdot \|Q\|^2)$ | $\mathcal{O}(\|S\|)$ | commutativity |

Table 3.1: Asymptotic worst case runtimes of preprocessing steps. The mitigation in the asymptotic time complexities is a prime example for the exploitation of redundant information during the processing.

algorithms: A lock-step calculation of z-normalized subsequence ED (sED) and an elastic subsequence alignment under the constrained DTW similarity measure (sCDTW) also using z-normalization. The authors introduced further improvements to speedup sED and sCDTW in comparison to former state-of-the-art

[8] The window $W$ ranges usually between 5% and 10% of the query length. Thus, in the worst case 80% of the envelope is not affected. Moreover, the remaining 20% may only differ slightly.

[9] It should at least know the value of each entry of the stream $S$ to guarantee optimality.

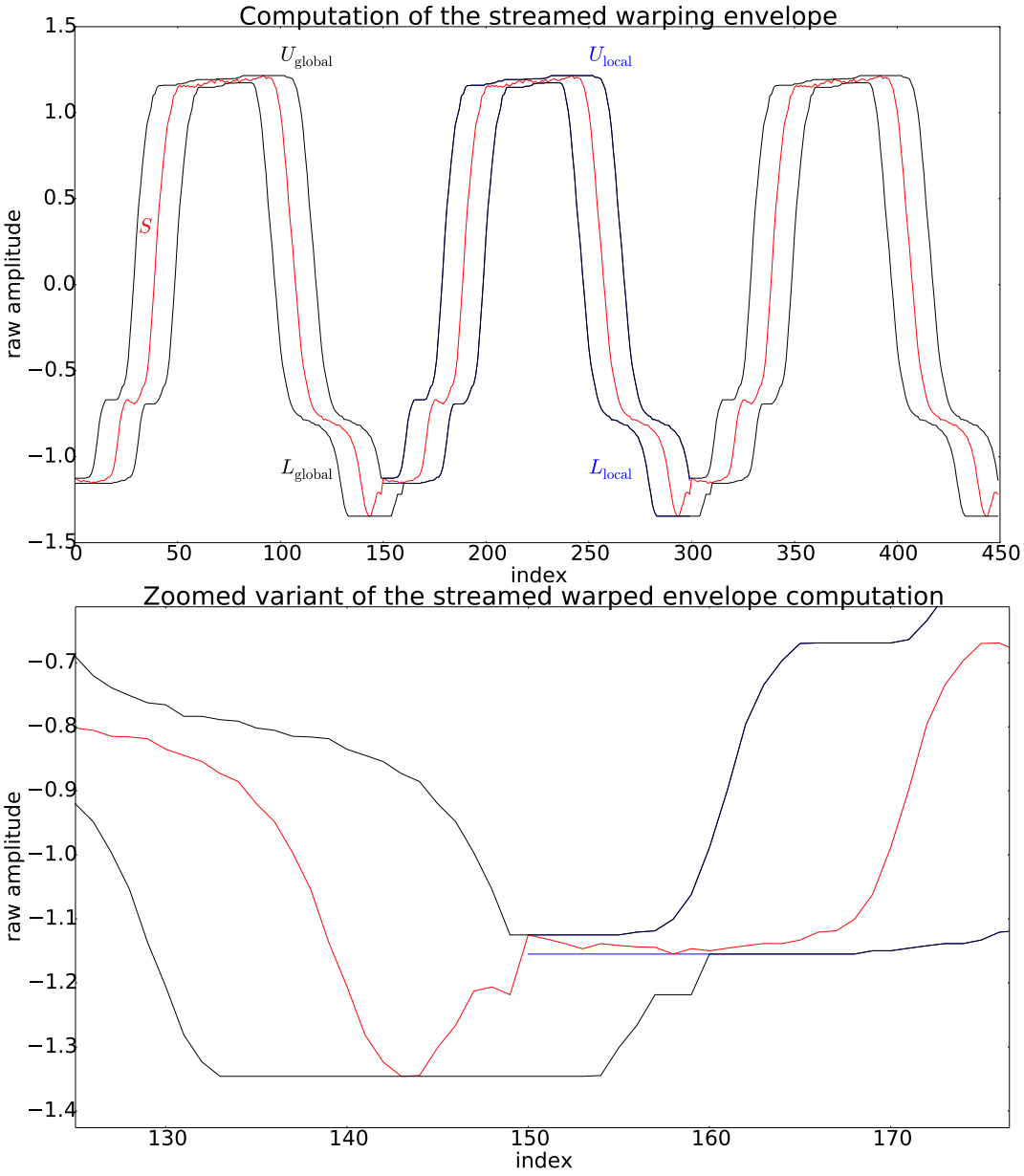


Figure 3.8: An example of a stream built from an instance of the Gun-Point dataset using concatenation (red). Obviously, the global envelope (black) encloses the local envelope (blue) at each index of the subsequence in the middle. Only minor differences in the values are perceivable at the borders caused by the overlap of the window ($W = 10$). The region of interest is magnified in the lower panel. As a result, the streamed envelope can be considered as tight as the local one.

implementations which usually rely on early-abandoning of the measure itself, lower-bounding and avoiding the computation of square roots:

- Since the sum operation + is associative and commutative, we can process ED in an arbitrary order of the indices. Thus, $\mathrm{ED}(Q,C)^2 = \sum_i (Q_{\pi(i)} - C_{\pi(i)})^2$ for any permutation $\pi$ in the symmetric group $\mathscr{S}(|Q|)$. Assume $\mathrm{BSF} \in \mathbb{R}_0^+$ being your best-so-far alignment score, then $\mathrm{ED}(Q,C)^2$ can be abandoned early if any partial sum of squared residues exceeds that value. Amongst all $|Q|! = |\mathscr{S}(|Q|)|$ possible permutations Rakthanmanon et al. propose to sort the values of $Q$ in descending order of their magnitude $|Q_i|$. This heuristic reordering can be applied in a preprocessing phase with only $\mathscr{O}(|Q| \cdot \log|Q|)$ effort and may reasonably decrease the overall runtime.

- Assume there is more than one dedicated lower bound for a similarity measure. In the majority of publications the authors determine the best amongst all known lower bounds or improve a promising candidate e.g. $\mathrm{LB}_{\mathrm{improved}}$ as enhancement of $\mathrm{LB}_{\mathrm{Keogh}}$ [58]. Thus, occasionally loose lower bounds like $\mathrm{LB}_{\mathrm{Kim}}$ are often omitted. In contrast, Rakthanmanon et al. propose to use all these lower bounds in a cascade beginning with the least demanding in means of runtime and then switching to more expensive ones.

Although it seems counter-intuitive to do more work than needed, the latter can be justified as follows: Assume you have calculated a perfect alignment $\mathrm{BSF} = d\big(Q, C^{(k)}\big) = 0$ for some $k$ and all other subsequence candidates differ at least in the first index when comparing them to $Q$.[10] The obviously loose lower bound $\mathrm{LB}^2_{\mathrm{loose}}\big(Q,C\big) := (Q_0 - C_0)^2$ would prune every other candidate in constant time. Thus, the search on the remaining indices could be accomplished with $\mathscr{O}(|S|)$ effort both under ED and DTW. Although a perfect match is very unlikely, BSF will be almost vanishing after the inspection of only a small portion of the stream. It would be crude to use a more expensive lower bound in this situation – even the very efficient linear time $\mathrm{LB}_{\mathrm{Keogh}}$ would be a waste of resources.

During the elastic alignment of time series $\mathrm{LB}_{\mathrm{Kim}}$ is a perfect representative for a cheap but occasionally loose lower bound. In contrast, if $\mathrm{LB}_{\mathrm{Kim}}$ cannot exclude a certain candidate, it is reasonable to invest the linear runtime of $\mathrm{LB}_{\mathrm{Keogh}}$ or its reversed variant. The full relaxation of DTW is performed only if all lower bounds fail. Similar to the abandoning of the ED measure itself, DTW can be equipped with a (sightly more complex) early-exit mechanism.[11] Let $d(\cdot,\cdot)$ be a measure or

[10] The first assumption may be overoptimistic but the second one is almost always fulfilled.

[11] An obvious decision criterion for an early exit can be found when using a wavefront relaxation for DTW . A detailed description of this computation scheme can be found in the next section.

lower bound then its early-exit capable variant shall be denoted as follows:

$$d^{\mathrm{BSF}}(Q,C) := \begin{cases} d(Q,C) & \text{if} \quad \mathrm{BSF} > d(Q,C) \\ \infty & \text{else} \end{cases} .$$

With the help of this definition we can state the simplified pseudo-code of the lower bound cascade utilized in the UCR-Suite under the CDTW measure. Since it is a single-threaded online algorithm the IO of the stream, memory accesses and the calculation of the stream's statistics $(\mu_k, \sigma_k)$ can be considered to be processed in a lazy-evaluation fashion i.e. expensive square roots and divisions during z-normalization are only executed if the measure is smaller than the already obtained BSF-value. Thus, z-normalization is completely suppressed in Figure 3.9 to avoid the introduction of distractive clutter in the control flow. The computation of UCR-Suite's subsequence alignment under ED omits lower bounds and uses only early-exit capable processing of z-normalization and ED combined with the above mentioned reordering of indices.

Concluding, Rakthanmanon et al. exploited redundancy and lower bounds to achieve faster execution times. However, the authors consider parallelization but provide no explicit strategy – in contrast, they underline that UCR-ED and UCR-DTW are faster than existing multi-core implementations:

> *'It is important to note that while we can get essentially linear speedup using multicores, the software improvements we will present in the next section completely dwarf the improvements gained by multicores. As a concrete example, a recent paper shows that a search of a time series of length 421,322 under DTW takes "3 hours and 2 minutes on a single core. The (8-core version) was able to complete the computation in 23 minutes" [34]. However, using our ideas, we can search a dataset of this size in just under one second on a single core. '*

In this certain context, this may be a valid statement where the authors compare their implementation to already existing algorithms. Nevertheless, modern hardware accelerators amongst others CUDA-enabled graphics cards (Nvidia K20x: ~ 1 Tflops/s) and many-core boards like the Xeon Phi (5110p: ~ 1 Tflops/s) effectively[12] provide up to five times more double-precision operations per seconds than a state-of-the-art CPU (Xeon E5-2680v2: ~ 200 Gflops/s) using **all** cores [59]. During the rest of this chapter, we will design computational schemes that allow for the porting of the described (often inherent sequential) techniques to parallel processors. Furthermore, we explicitly implement these schemes on CUDA-enabled accelerators and multi-core CPUs.

[12] We avoid to cite exact numbers for the peak performance provided by the vendors. In most applications, they do not reflect reality.

```
1: function UCR_CASCADED_CDTW(Q, S, W)
2:     (k̂, BSF) ← (−1, ∞)                          ▷ initialize best match with dummy values
3:
4:     for k = 0 → |S| − |Q| do                      ▷ for each alignment position
5:         C ← S[k : k + |Q|]                        ▷ extract alignment candidate
6:
7:         if BSF < LB_Kim^BSF(Q, C) then            ▷ 3-point full DTW ∈ 𝒪(1)
8:             continue
9:         end if
10:        if BSF < LB_Keogh^BSF((L, U)_Q, C) then   ▷ Q-enveloped LB ∈ 𝒪(|Q|)
11:            continue
12:        end if
13:        if BSF < LB_Keogh^BSF((L, U)_C, Q) then   ▷ C-enveloped LB ∈ 𝒪(|Q|)
14:            continue
15:        end if
16:
17:        d ← CDTW^BSF(Q, C, W)                     ▷ complete constrained relaxation ∈ 𝒪(|Q|²)
18:
19:        if d < BSF then                           ▷ check for best alignment
20:            (k̂, BSF) ← (k, d)
21:        end if
22:    end for
23:
24:    return (k̂, BSF)
25: end function
```

Figure 3.9: Pseudo-code for the efficient computation of sCDTW.

## 3.2 Parallelization on CUDA-enabled accelerators

Besides the reasonable increasing in floating point operations modern hardware accelerators operate with a significantly better energy efficiency than common CPUs. Thus, especially in high performance computing (HPC) environments, it can be expected that the ratio of hardware accelerator cards will increase in the near future. Exemplary, under the top 10 entries of the Green 500 benchmark[13] nine out of ten systems explicitly state NVIDIA K20x or K40 cards. For historical reasons, GPUs are specialized on single-precision floating point operations which can be processed up to five times faster than double precision on that dedicated hardware. As a result, they are ideally suited for the processing of big amounts of floating point data e.g. time series streams of enormous lengths.

Unfortunately, GPU cores process the data in a different manner than thread-

[13] The Green 500 list from June 2014 can be found at:
`http://www.green500.org/lists/green201406&green500from=1&green500to=100` .

ing models on common multi-core CPUs. As we will see, a straight-forward port of the stated pseudo-codes is often impossible due to the single-instruction-multiple-threads (SIMT) constraint on graphics hardware. In the following subsection, we give a brief overview of the Compute Unified Device Architecture (CUDA), which provides specialized language bindings (C/C++ and Fortran dialects) for the programming of NVIDIA graphics cards. Although there exist unified approaches for the programming of GPUs from other vendors and multi-core CPUs (e.g. OpenCL and OpenACC), CUDA is nowadays the dominant programming model in scientific applications designed for the execution on graphics hardware. In CUDA, the parallelism has to be expressed explicitly allowing for precise control on a low level which, if done properly, results in efficient performance.

### 3.2.1 A Brief Review of CUDA

This subsection aims for a brief introduction into CUDA and thus cannot be considered to be complete. An interested reader may refer to [60] for an extensive overview. Whenever hardware specifications are cited we assume an NVIDIA Tesla K40 graphics card [61]. In the following, we focus solely on the features of CUDA that are required for the understanding of our algorithmic design and the impact on performance. This includes the organization of parallelism, memory and caches as well as synchronization.

#### Computational Model

The CUDA programming model achieves parallelism with a massive amount of threads which each are executed independently on a separate stream processor (SP). The number of SPs can exceed several thousands e.g. 2880 for an NVIDIA K40 allowing for the simultaneous processing of vast amounts of scalar operations. The SPs themselves are organized in a grid of streaming multiprocessors (SM) each consisting of several hundreds SPs (NVIDIA K40: 192 SPs in each of the 15 SMs). Figure 3.10 depicts the hardware layout for the GK110 architecture which is used by K40 boards. The scheduler can assign more than one thread to a single SP. The corresponding context switches between the execution of threads do not reasonably mitigate the overall performance due to a large register file. Exemplary, an NVIDIA K40 GPU provides 65536 32bit registers per SM which allows for the simultaneous maintenance of thousands of lightweight threads without the need for dumping states. As we will see later, this switching between threads with minimal overhead can be exploited to hide the latency of the attached memory. Threads are organized in thread blocks which loosely map to the above mentioned SMs. Similar to the scheduling of threads the scheduler may maintain more blocks than the number physically existing SMs. Concluding, up to 1024 threads are


Figure 3.10: Layout of the GK110 chip used in NVIDIA K40 accelerator cards [61].

grouped into a grid of up to $2^{32} - 1$ thread blocks. Depending on the user's needs the grids can be configured to map indices in the 1D, 2D or 3D domain.

Roughly speaking, a CUDA-enabled GPU can be utilized as a huge vector machine: A typical use case would be the pixel-wise smoothing of an image where each thread calculates a weighted average over a small region of the neighbouring pixels. In this case, each thread would execute exactly the same sequence of operations similar to the single instruction multiple data (SIMD) computation model of vector units in modern CPUs. Although data can be (re)arranged to lie in consecutive memory for many applications, SIMD puts on us an additional constraint which may be too restricting.[14] The CUDA programming model relaxes the SIMD paradigm to a less restrictive computation scheme – the single instruction multiple thread (SIMT) model:

- Single instruction is only enforced on batches of 32 consecutive threads. However, branch divergence within this so-called warp is still allowed but will be serialized by the hardware. Thus, all threads in a warp should execute the same branch to achieve reasonable performance.

[14] Note, when smoothing an image the pixels on the x-axis are arranged in consecutive memory. In contrast, memory accesses along the y-axis are scattered through the whole memory.

- Consecutive threads in a warp do not necessarily need to access consecutive memory. Let $\mathscr{I} = (i_0, i_1, \ldots)$ be a random sequence of indices then thread $j$ is allowed to access memory at position $i_j$. In contrast, SIMD would enforce thread $j$ to access the memory at position $j + k$ for some constant $k \in \mathbb{N}_0$.

Nevertheless, it is advisable to treat a warp, whenever possible, as if it was an SIMD vector unit of length 32.

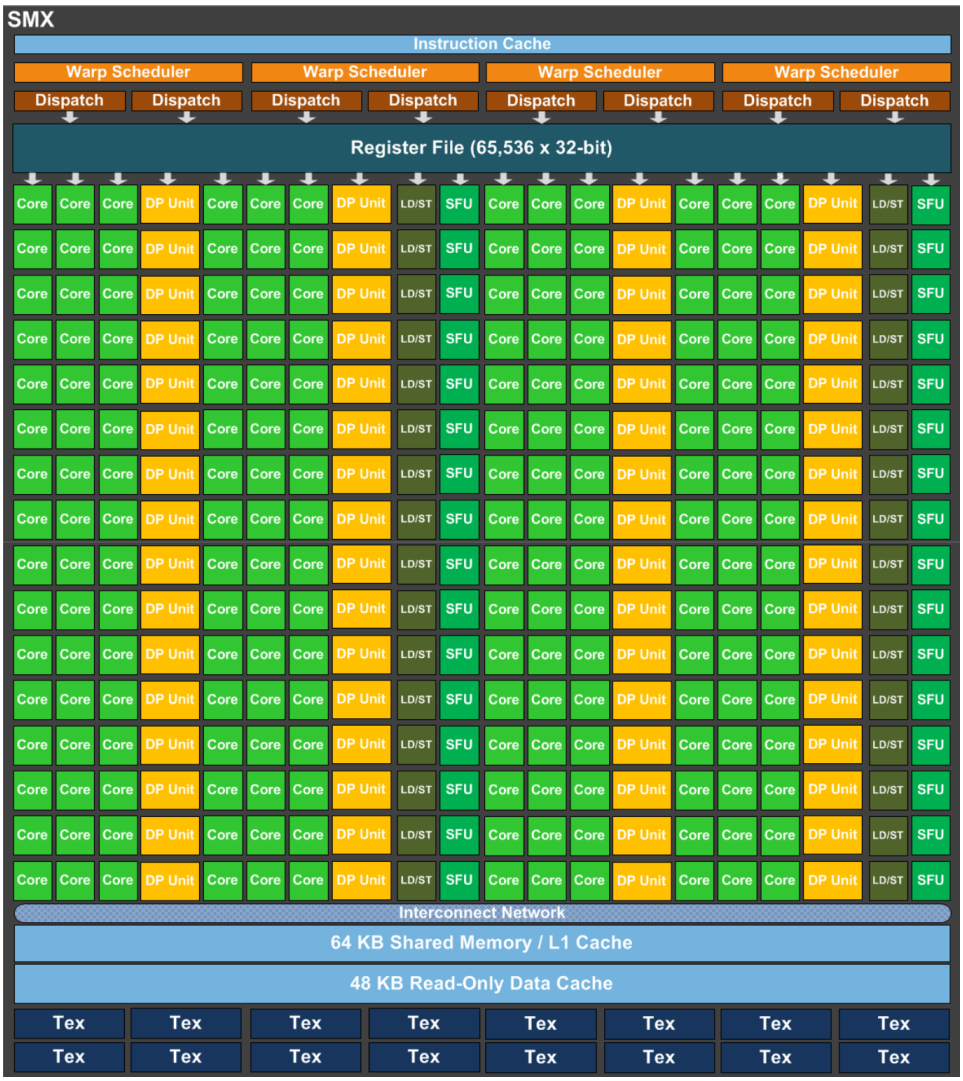


Figure 3.11: Layout of a single streaming multiprocessor unit [61]. Obviously, there are three times more single-precision units (green cores) than double-precision units (yellow cores) which explains the performance gap between both computation modes. All cores in an SM share the same low latency level one cache and constant memory.

### Memory

CUDA-enabled accelerators provide at least three different types of memory. In contrast to CPUs, the level one cache has to be programmed explicitly. Only a minor part of the level one cache and the whole level two cache of the GPU is administered implicitly by the hardware. Similar to Von Neumann architectures memory bandwidth is considered to be the major bottleneck during the computation. Thus, the programmer has to carefully choose a proper memory utilization scheme to ensure execution speed.

- **(Global Memory)** Modern GPUs are equipped with up to 12 GB dedicated GDDR5 RAM. The CPU's RAM is physically detached from the GPU's off-chip memory and connected via a PCIe bus operating at ~ 6 GB/s. As a result, the user must explicitly transfer the data between the host system (host) and the CUDA device (device). The address space of the host and device are strictly separated.[15] Each thread can access the global memory of the GPU with an overall bandwidth of about 300 GB/s. Memory accesses should ideally be coalesced i.e. consecutive threads access consecutive memory locations. Assuming **one** single precision floating point operation per datum and omitting latencies of the memory and the involved floating point operation, we are bound by the global memory bandwidth to process only $\frac{300\text{GB/s}\cdot}{4\text{B/flop}} = 75$ Gflops/s – merely 2% of the theoretical peak performance (NVIDIA K40: 4.29 Tflops/s). In contrast, when reusing data, the utilization of shared memory and (local) registers can drastically improve this situation.

- **(Shared Memory)** Each SM has access to 64 KB on-chip memory split into 48 KB of user-controlled cache and 16 KB level one cache controlled by the hardware. Depending on the user's needs the sizes of the partitions may be reversed to 16 KB/48 KB with an explicit option before the launch of the computation. Access to shared memory is orders-of-magnitude faster than global memory reads but is restricted to the threads in one block. Hence, it can be utilized for inter-thread communication but cannot be used for inter-block communication. Shared memory is partitioned into 32 memory banks with a user-defined word size of either four or eight bytes. During the concurrent execution each memory bank should be read by only one thread at once – otherwise the accesses are serialized by the hardware which causes a notable decreasing in performance. Since host data cannot be transferred directly to the shared memory, it has to be staged in global memory. Thus, the use of shared memory is only advisable when the cached data is accessed several times or in random order.

[15] CUDA 6.0 introduced a unified memory model which automatically administers the data transfers between the host and the device. However, the physical address spaces remain separated.

- **(Registers and Local Memory)** Each thread has access to its own private memory. On the one hand, the user can define fast registers within the register file limitation of $64 \cdot 1024 \cdot 32$ bits per SM and at maximum 255 registers per thread. If this limit is exceeded the register are spilled into slow global memory and thus should be avoided at any cost. Unfortunately, registers cannot be enumerated with for-loops which may result in cluttered code. On the other hand, local arrays can be defined in a thread which allows for a convenient enumeration. However, this so-called local memory resides off-chip and should consequently be called 'thread-local global memory'. Besides minor optimization techniques using interleaved addressing, local memory is almost as slow as accesses to the global memory. Therefore, whenever applicable, shared memory should be utilized instead of local arrays to avoid poor performance. If the whole data fits into registers and no inter-thread communication is needed, it is advisable to omit the use of shared memory and fully rely on the even faster registers.

Furthermore, CUDA offers at least two mechanism for the implicit caching of global memory, namely constant memory and textures. On the one hand, constant memory can be considered as a small portion of 64 KB heavily-cached global memory. It has to be declared in the program header and is read-only during the concurrent execution of threads. If no cache misses occur, the access is almost as fast as register reads. However, if threads access different memory positions at once which are not present in the cache, the reads are serialized by the hardware. On the other hand, texture objects can be bound to the global memory up to a length of $2^{27}$ elements for the $1D$ case. Textures provide faster access than raw reads from global memory if the access patterns exhibit spatial locality. Further hardware-accelerated filter modi like clamping and linear interpolation can be exploited to save some flops during the computation. Unfortunately, the read-only texture memory has to be declared in the header of the program and thus cannot be defined dynamically during runtime. Concluding, the major challenge during the design of efficient CUDA-programs is to find an optimal distribution of the memory and the determination of proper access patterns.

### Latency Hiding

Assume our algorithm uses optimal access patterns resulting in maximum throughput once the data is flowing. However, we still have to consider the initial overhead of memory access (latency): During this time span, the GPU cannot possibly process anything which results in a reasonable waste of computational resources. There exist two popular strategies for latency hiding. Please note, the following techniques cannot reduce latency at all, they aim for a better utilization of the hard-

ware by overlapping computation and memory accesses resulting in an overall shorter runtime of the program:

- **(Over-Provisioning of Threads)** The latency of a global memory read can take up to several hundreds of cycles in contrast to only a few cycles for the actual floating point operation. A SM can maintain up to 2048 threads simultaneously although it only consists of 192 SPs. During the time span a warp has to wait, another warp can process its already received data. Assuming ideal overlap of the computation and memory accesses, the latency can be almost completely hidden. Thus, when dealing with only a few floating point operations per memory access, a huge number of active threads i.e. high occupancy may help to improve the performance. Nevertheless, the performance will be bound by the throughput of the used memory.

- **(Compute to Global Memory Access Ratio)** Over-provisioning of threads is often advised in the literature but may be disadvantageous. In a recent talk, Volkov demonstrated that 'Better Performance at Lower Occupancy' is indeed possible [62]. In the meantime, mainstream libraries e.g. cuBLAS [63] and cuFFT [64] reduced the number of concurrent threads and gained additional performance. This can be achieved by significantly increasing the work done in a thread. Exemplary, Volkov introduced a simple memcpy-kernel where each thread copies up to 14 `float4` structs at once. This kernel saturates at only 4% occupancy and is faster than NVIDIA's native `cudaMemcpy` implementation. Furthermore, complex kernels may benefit from instruction-level parallelism and the possibility to communicate directly via fast registers. The Compute to Global Memory Access Ratio (GCMA) is an important indicator whether an algorithm performs well (high values) or wastes its computational resources while waiting for memory (low values).

Concluding, the more time is spent on computation the less latency matters.

### Synchronization

As mentioned before, warps scheduled in a block are executed concurrently. Unfortunately, the CUDA programming model does not provide a mechanism to control the order of warps during execution. However, it is ensured that all threads in a warp are executed simultaneously and thus there is no need to synchronize them. Many algorithms rely on synchronization barriers e.g. right after the loading of the data to shared memory. Within a block of threads this can be done explicitly with a `__syncthreads()` call. In contrast, there exists no equivalent device-side command to synchronize the program across several thread blocks. Although one

could simulate global barriers by writing state variables to slow global memory, this approach is not advisable due to performance reasons.

Nevertheless, global synchronization can be enforced explicitly and implicitly from the host side. On the one hand, the user can rely on the fact that all threads within a predefined stream finished their work right after the corresponding kernel has terminated. Thus, when relying on global barriers, the algorithm can be split into a sequence of kernel calls which are stacked within a loop. When overlapping CPU and GPU computations, the user can enforce a global barrier across all streams with an explicit `cudaDeviceSynchronize()` call. Moreover, CUDA provides an event mechanism to model dependencies between predefined streams of threads. On the other hand, some commands implicitly synchronize the device amongst others memory allocations (`cudaMalloc[Host]`), non-asynchronous memory operations (`cudaMemcpy` and `cudaMemset`) and the configuration of the L1-cache to shared memory ratio (`cudaDeviceSetCacheConfig`).

### 3.2.2 Euclidean Distance

Let us recall the general subsequence alignment algorithm stated in Figure 3.3. It could be used for the one-nearest-neighbour (1NN) search of a query $Q$ in a stream $S$ under ED and in fact the UCR-Suite [17] effectively uses this computation scheme in combination with the mentioned optimization techniques. The asymptotic worst case time complexity is in $\mathcal{O}(|S| \cdot |Q|)$ due to the linear cost of ED.[16] Although, the proposed reordering of the query's indices can be trivially implemented on a GPU, the early-abandoning of ED cannot be mapped efficiently in an SIMT-compliant way: In a warp of consecutive threads, the alignment position which abandons last determines the overall runtime, even worse, we have to use expensive communication to ensure that all threads can safely exit. On the one hand, the communication could be decreased by checking only every few cycles if all threads finished their computation. On the other hand, we could argue that the runtime is anyway in $\mathcal{O}(|S| \cdot |Q|)$ and thus we aim to reduce the leading constant by simply using the additional flops on the GPU.

In contrast, we propose a completely different approach which can be parallelized easily and has a better worst time complexity than UCR-ED. The algorithm is capable to calculate all of the $|S| - |Q| + 1$ alignment scores without using lower bounds and at the same time being faster than UCR-ED. Moreover, its asymptotic runtime is **independent** of the query length $|Q|$ and thus queries with millions of values can be accomplished with no perceivable mitigation in runtime. The rest of this subsection is based on our publication at the International Conference on Parallel Processing (ICPP 2014) [65].

[16] Please note, early abandoning of ED is still in $\mathcal{O}(|Q|)$ unless we use an even weaker lower bound similar to $LB_{Kim}$. Thus, UCR-ED reduces the constant omitted by the Landau notation.

Assume we omit z-normalization of the involved time series then all distance scores for $k \in \{0,\dots,|S|-|Q|\}$ can be computed as follows:

$$\mathrm{ED}\big(Q, C^{(k)}\big)^2 = \sum_{i=0}^{|Q|-1} \big(Q_i - S_{i+k}\big)^2 = \sum_{i=0}^{|Q|-1} \big(Q_i^2 - 2\cdot Q_i \cdot S_{i+k} + S_{i+k}^2\big) \quad .$$

Although the right-hand-side of the equation looks more complex than the initial expression, we can lower the naïve $\mathcal{O}(|S|\cdot|Q|)$ runtime to a log-linear dependency in the stream length. The first and the third summand can be calculated with the help of prefix sums in $\mathcal{O}(|S|)$ time and memory similar to the windowed computation of the z-normalization. The second term can be written as a convolution which results in $\mathcal{O}(|S|\cdot\log|S|)$ runtime using the Fast Fourier Transform (FFT) [44]. To achieve that, the query has to be embedded into a zero-vector $\iota(Q) := (Q_0,\dots,Q_{|Q|-1},0,\dots,0)$ of length $|S|$ since

$$\rho(k) := \sum_{i=0}^{|Q|-1} Q_i \cdot S_{i+k} = \sum_{l=0}^{|S|-1} \iota(Q)_{l\%|S|} \cdot S_{(l+k)\%|S|} \quad \text{for all } k \in \{0,\dots,|S|-|Q|\} \quad .$$

As a result, the correlation term $\rho(k)$ can be calculated by a convolution of $S$ and the time-reversed embedding $\iota(Q)^r := (Q_0,0,\dots,0,Q_{|Q|-1},Q_{|Q|-2},\dots,Q_1)$.[17] The reversal of $Q$ could be accomplished efficiently in shared memory to avoid backward addressing. However, the same can be achieved in a more elegant way by the complex conjugation of $Q$ in the (dual) frequency space. The modulo operations are suppressed in the following to avoid unnecessary clutter in the formulas.

**Theorem 1** (Correlation Theorem)**.** *Let $\iota : \mathbb{R}^{|Q|} \to \mathbb{R}^{|S|}$, $Q \mapsto (Q_0,\dots,Q_{|Q|-1},0,\dots,0)$ be an embedding of the query $Q$ into a zero-vector of length $|S|$, then*

$$\rho(k) = \sum_{l=0}^{|S|-1} \iota(Q)_l \cdot S_{l+k} = \mathscr{F}^{-1}\Big(\mathscr{F}\big(\iota(Q)\big)^* \odot \mathscr{F}\big(S\big)\Big)_k ,$$

*where $\mathscr{F}$ is the discrete Fourier Transform , $\odot$ the point-wise product of vectors and $*$ the common complex conjugation.*

*Proof.* By a straightforward calculation of the right-hand-side:

$$\begin{aligned}\mathscr{F}\big(\iota(Q)\big)^*_s \mathscr{F}\big(S\big)_s &= \sum_{l,l'=0}^{|S|-1} \iota(Q)^*_l \cdot S_{l'} \cdot \exp\big(\tfrac{-2\pi\mathrm{i}(l'-l)s}{|S|}\big) \\ &= \sum_{l=0}^{|S|-1} \iota(Q)^*_l \cdot \sum_{k=0}^{|S|-1} S_{l+k} \cdot \exp\big(\tfrac{-2\pi\mathrm{i}ks}{|S|}\big) .\end{aligned}$$

In the last term we used the cyclic index substitution $k = l' - l$. The final result is obtained by the inverse transform. Please note, the entries of $\iota(Q)$ are real such that $\iota(Q)^* = \iota(Q)$. □

[17] Note, the sign of $j$ has to be altered to rewrite $\rho(k)$ as convolution.

Thus, sED($Q$, $S$) can be processed with only three FFT calls and an index-wise multiplication in the dual space. Subsequence ED can be equipped with z-normalization afterwards using the precomputed statistics $\mu_k$ and $\sigma_k$. Without loss of generality assume $Q$ already being z-normalized i.e. $Q$ has a vanishing mean $\sum_l Q_l = 0$ and a normalized standard deviation of $\frac{1}{|Q|}\sum_l Q_l^2 = 1$. Furthermore, set $z(S)_{l+k} := \frac{S_{l+k}-\mu_k}{\sigma_k}$ and by the same argument $\frac{1}{|Q|}\sum_l z(S)_{l+k}^2 = 1$ for all $k \in \{0,\dots,|S|-|Q|\}$. It holds:

$$\begin{aligned}
\mathrm{sED(Q,S)}^2 &= \min_k \sum_{l=0}^{|Q|-1} \left(Q_l - z(S)_{l+k}\right)^2 \\
&= \min_k \sum_{l=0}^{|Q|-1} \left(Q_l^2 - 2\cdot Q_l \cdot z(S)_{l+k} + z(S)_{l+k}^2\right) \\
&= \min_k \left(2\cdot|Q| - \frac{2}{\sigma_k}\sum_{l=0}^{|Q|-1}\left(Q_l\cdot S_{l+k} - Q_l\cdot\mu_k\right)\right) \\
&= 2\cdot\left(|Q| - \max_k \frac{1}{\sigma_k}\sum_{l=0}^{|Q|-1} Q_l\cdot S_{l+k}\right) \qquad .
\end{aligned}$$

The final equation can be seen as an extension of Proposition 3 being aware of translations along the time-axis. Besides the constant $|Q|$ and the division by the precomputed statistics $\sigma_k$, z-normalized sED is effectively given by the plain correlation term $\rho(k)$. If one considers data streams with almost constant $\sigma_k$, which can be safely assumed for e.g. ECG signals, then maximizing correlation over all alignment positions is equivalent to minimizing ED while at the same time guaranteeing translation invariance along the measurement-axis. As max is an associative map, one could split the computation of sED into smaller batches if the stream size exceeds the RAM of the GPU. Furthermore, we can reduce the workload of the used FFT algorithm by a factor of two since we can exploit the Hermitian symmetry of Fourier-transformed signals on real-valued domains.

Our implementation utilizes the NVIDIA CUDA FFT library (cuFFT) [64] which is bundled with the NVIDIA CUDA toolkit and Intel's FFT routines from the Math Kernel Library (mklFTT) [66]. Both libraries provide arbitrary length support using 2,3,5,7-radix schemes. Nevertheless, $Q$ and $S$ are embedded in zero-vectors whose sizes are a multiple of $2^4\cdot 5^4 = 10000$ to avoid poor performance when $|S|$ is prime. Finally, the optimal value of sED($Q$, $S$) is obtained using a call to a device-wide reduce primitive from the CUB library. An advantage of our implementation is portability. The major portion of the runtime is spent in cuFFT and CUB routines. Hence, the algorithm can benefit from future performance improvements of the libraries and next-generation GPUs without editing a single line of code.

### 3.2.3 Subsequence Constrained DTW

Convolution is a bilinear map i.e. $(Q+Q')\star(S+S') = Q\star S+Q'\star S+Q\star S'+Q'\star S'$ for all time series $Q, Q', S$ and $S'$. In contrast to lock-step measures, elastic measures allow for the dynamic mapping of indices which does not obey this linear property. Thus, the structure of algorithms for the elastic alignment of subsequences may be similar to the processing of convolutions but unfortunately cannot benefit from FFT-accelerated approaches. As a result, the use of lower bounds is mandatory to ensure reasonable runtime. In the following, the presented work is based on our publication at ICPP 2014 [65]. We consider three different approaches for computing the CDTW measure on CUDA-enabled accelerators.

#### Thread-Level Parallelization of CDTW

This approach assigns one CUDA-thread to one CDTW alignment which can be achieved by a straight-forward port of the pseudo-code in Figure 2.9. Since the CUDA shared memory is limited to 48 KB, a single warp of 32 threads can only address window sizes of less than 100 single-precision floating point values:

$$\frac{48 \cdot 1024\,\text{bytes}}{\text{sizeof(float)} \cdot 2\,\text{rows} \cdot 32\,\text{threads}} > \underbrace{2 \cdot W + 1}_{\text{total window size}} \quad .$$

As a result, the penalty matrix has to be stored in slow global[18] memory even for moderately sized query lengths.

#### Block-Level Parallelization of CDTW

We propose a fine-grained parallelization scheme to avoid the limitations of the shared memory. The CUDA-programming model allows for a block-wide synchronization of threads but does not guarantee a specific execution order. To address this problem, our implementation exhibits a wave-front relaxation scheme, also used by the parallelization of the Smith-Waterman algorithm [67]. It updates the nodes of the DAG in topological order. Each cell on a minor diagonal is updated independently accessing the two previous diagonal lanes of length $W+2$ (see Figure 3.12). All of the $|Q|+|C|+1$ lanes in total can be relaxed using only three lanes by cyclic writes to the shared memory. As a result, the runtime and memory requirements are equivalent to the sequential code. Moreover, the shared memory footprint allows to employ window sizes of up to 4,094 values since:

$$\frac{48 \cdot 1024\,\text{bytes}}{\text{sizeof(float)} \cdot 3\,\text{lanes}} > \underbrace{W+2}_{\text{lane size}} \quad .$$

[18] Please note, even locally allocated arrays within a kernel (known as local memory) reside in global memory.

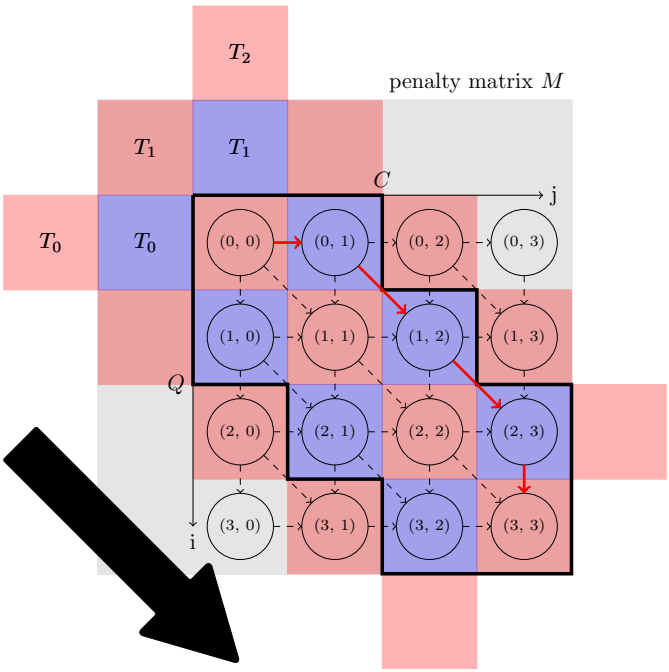


Figure 3.12: An example for the wave-front relaxation scheme of CDTW using a window of $W = 1$ (black thick border). Only three consecutive diagonal lanes (blue and red boxes) of maximum length $W + 2 = 3$ are utilized to ensure $\mathcal{O}(W)$ memory. In each lane, one sentinel cell stores the value $\infty$ in order to avoid comparisons at the borders of the Sakoe-Chiba band. Each thread $T_0, \ldots, T_{W+1}$ can operate independently on the cells in its lane. The thick arrow indicates the relaxation order of the lanes.

During relaxation of a single lane the indices of $Q$ and $C$ are read in random order. Texture memory can be bound to the stream $S$ to gain higher performance when accessing spatially located indices of the subsequence candidate $C$. Additionally, we can use the remaining shared memory to cache the whole query.

### Batch-Level Parallelization of CDTW

The block-level parallelization can be disadvantageous if the window size is too small. Assume $W = 1$ analogous to the example in Figure 3.12 then only three threads are involved in the computation of one CDTW measure. In this case less then ten percent of the threads in a warp (equivalent to 32 threads) are used resulting in noticeable performance losses. Thus, we propose a batch-level approach. The relaxation of a few CDTW computations is executed simultaneously in one block. The lanes of the first CDTW calculation is processed by the first batch of $W+2$ threads and so on. We choose one block to contain four to eight warps since a

further increase of the number of threads would decrease performance. The batch-level parallelization scheme is only used for small lane lengths i.e. $W+2 \leq 256$ (equivalent to eight warps). For greater lane lengths the above-mentioned block-level parallelization is used.

### 3.2.4 Lower Bound Cascades

As mentioned before, $\mathrm{LB}_{\mathrm{Kim}}$ is a partially relaxed DTW on a minor portion of the penalty matrix. Our implementation processes the upper left and lower right submatrices (each of size $5 \times 5$) for the first and last five entries of the query $Q$ and a subsequence candidate $C$. $\mathrm{LB}_{\mathrm{Kim}}$ is a lower bound since it neglects the non-negative contributions on the rest of the penalty matrix. Each CUDA-thread independently calculates one $\mathrm{LB}_{\mathrm{Kim}}$ value. The whole submatrix is stored in only ten registers since the relaxation of cells only depends on the current and previous row indices. Multiple accesses to $Q$ and $C$ can also be cached in registers. Therefore, coalesced reads from global memory can be guaranteed. Both submatrices are calculated independently such that the final value is the sum of the associated measures of both optimal warping paths. As long register spilling does not occur the relaxation of bigger sized submatrices is conceivable.

The computation of $\mathrm{LB}_{\mathrm{Keogh}}$ depends on the precomputed warping envelope $(L,U)_Q$ and the subsequence candidate $C$. $(L,U)_Q$ can be determined using Lemire's streaming min-max-filter [57] with amortized $3 \cdot |Q|$ comparisons. This can be delegated to the CPU because the size of $Q$ is usually much smaller than the length of $S$. In our implementation each CUDA-thread calculates one $\mathrm{LB}_{\mathrm{Keogh}}$ value analogously to the processing of $\mathrm{LB}_{\mathrm{Kim}}$. The computation is very similar to a (non-linear) convolution of a small sized kernel with a huge one-dimensional image.[19] The envelope is written to constant memory such that consecutive threads can simultaneously access the same indices. Shared memory can be utilized as follows: Assume $\mathrm{LB}_{\mathrm{Keogh}}$ is calculated on consecutive candidates in the stream. Each thread in a block loads one of the consecutive values of $S$ to shared memory. Additionally, $|Q|-1$ halo values are appended to ensure the overlap of the kernel with the right boundary. The redundant reads from $S$ during the calculation of the residues can now benefit from the up to 100 times faster shared memory in comparison to global memory. Figure 3.13 illustrates the used memory layout.

Further, we introduce our lower bounding strategy for the incremental computation of the best CDTW alignment of a query in a stream of time series data.

- Firstly, the windowed statistics $\mu_k$ and $\sigma_k$ are calculated for all $|S|-|Q|+1$ alignment positions using the above-mentioned linear time and memory algorithm. The z-normalization of the query and its envelope are precomputed on the CPU and subsequently transferred to constant memory.

[19] Unfortunately, the non-linear structure prohibits a log-linear scheme.

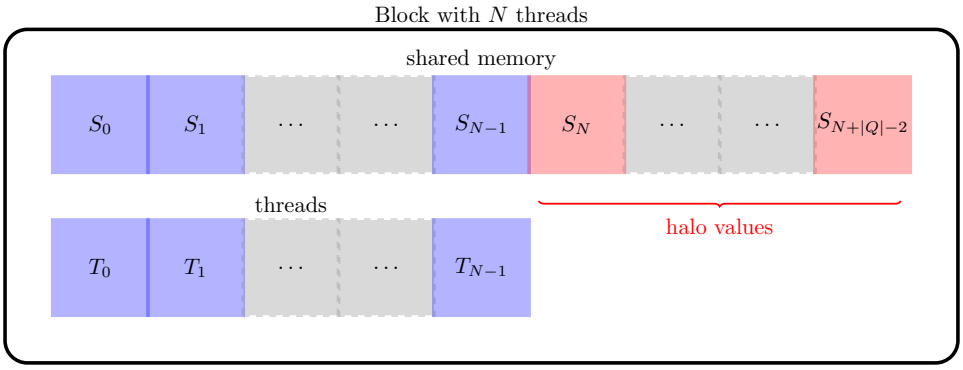


Figure 3.13: Memory layout for $LB_{Keogh}$ within a block of dimension $N$. Each of the $N$ CUDA-threads (blue/left-hand-side) independently operates on one of the $N$ consecutive positions. Furthermore, $|Q|-1$ halo values (red/right-hand-side) are appended to the shared memory to guarantee boundary overlap.

- Secondly, the fast register version of $LB_{Kim}$ calculates the lower bounds for all alignment indices. Furthermore, a call to a device-wide radix-sort primitive from the CUB library rearranges the indices according to ascending values of the lower bound. Since $LB_{Kim}$ is a partially relaxed DTW, we assume its values to be approximately proportional to its corresponding CDTW measure. As a result, it is likely that the optimal subsequence alignment is situated at the beginning of the reordered indices.

- Thirdly, the CDTW measure is processed on a first chunk of indices obtaining a preliminary best-so-far (BSF) value of the CDTW measure using a device-wide min-reduce primitive from the CUB library. If this BSF value is smaller than the first $LB_{Kim}$ value from the next chunk we can abort since every CDTW value of the remaining indices must be greater than our computed CDTW alignment. Otherwise, the next chunk is processed.

- Fourthly, on each of the next chunks we use $LB_{Keogh}$ to filter the candidates from the index list that can safely be pruned since their lower bound value is greater than the current BSF value. This can be achieved with array compaction techniques using prefix sums to avoid the use of lists on the GPU. This procedure is repeated until the BSF value is smaller than the first $LB_{Kim}$ value in the next chunk or there are no chunks left respectively.

Figure 3.14 illustrates the used lower-bound cascade. Similar to the computation of sED, the associativity of the min-map can be exploited to execute this scheme in batches if the stream size exceeds the RAM of the GPU.

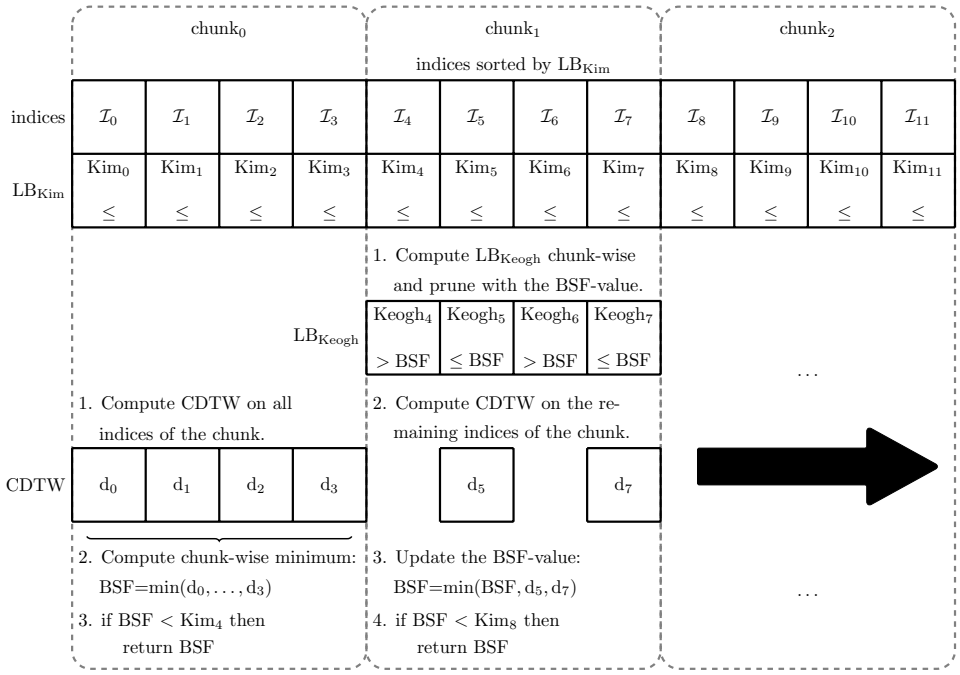


Figure 3.14: Illustration of the used lower bound cascade for a chunk size of four. Initially, all indices are sorted by $LB_{Kim}$ and a preliminary BSF-value is computed on $chunk_0$. In the further procedure, we can early-exit if the smallest $LB_{Kim}$-value in the next chunk is greater than the already obtained BSF-value. The amount of CDTW computations in the following chunks can be reduced using the $LB_{Keogh}$ lower bound. This procedure is repeated for consecutive chunks until there are no chunks left or an early-exit can be achieved, respectively.

## 3.3 Performance Evaluation

The performance evaluation has been carried out on the following platform:

- (**CPU**) Intel Core i7-3970X @ 3.50 GHz (six cores using Hyper-Threading) with 32 GB DDR3 RAM @ 1600 MHz
- (**GPU**) NVIDIA GeForce GTX TITAN @ 836 MHz 6 GB RAM, CUDA version 5.0 (Compute Capability 3.5)
- (**OS**) Ubuntu Linux Release 13.10 (64bit), gcc 4.8.1

To ensure a fair comparison, all tested algorithms have been compiled and executed on the same machine. The CFLAGS for the compilation of CPU-side source code provide -O3 optimization for our implementations as well as for the compared algorithms of other authors. FFT code provided in Intel's Math Kernel Library has been compiled with the Intel C++ Compiler version 14.0.0.

Since the execution times of the compared algorithms may depend on the characteristics of the input data, we provide the performance evaluation for two publicly available datasets from different domains:

- (**ECG**) The dataset consists of approximately 22 hours streamed electrocardiograms (20,140,000 data points) and was used during the evaluation of the UCR-Suite by Rakthanmanon et al.[17]. The full dataset can be downloaded at the supplementary website of the paper [68]. We reuse this dataset to ensure a fair comparison between the UCR-Suite and our implementations. An excerpt of the data stream is visualized in Figure 3.1. The use of windowed z-normalization is advisable due to the time-dependent offset of the stream. A separately measured stream for the extraction of queries is provided, too.

- (**Metal**) The dataset consists of 1,000 pairs of thickness signals measured during the hot rolling of metal bands in steel mills. The obtained time series are used for quality monitoring during the production process. Each time series is measured twice to avoid accidental mix-ups. Hence, we can access two individually recorded streams (both of about 11 million data points) – one for the extraction of the queries and one for the target data stream. The full dataset can be downloaded at [69]. Analogous to the ECG signals, the data exhibits time-dependent offset shifts, too (see Figure 3.2). Hence the use of windowed z-normalization is advisable.

The experimental setup is the same for all algorithms:

1. For each experimental setup i.e. algorithm, dataset and parameter setting (query length, warping window) determine the execution times $T_i$ for the subsequence alignment of ten randomly chosen queries $Q_i$ in the associated stream of time series data. The queries are taken from a separate dataset such that they are not present in the stream $S$. The specific queries are given in the data folder of our supplementary website [69]. The length of the used queries ranges from 128 to 8,192 to cover a wide spectrum of application scenarios.

2. Report the average $\mu$ and standard deviation $\sigma$ of the random variable $T_i$. Afterwards, speedups are determined using the averaged execution times. The given error bars indicate $\pm 1\sigma$ of the determined standard deviation.

### 3.3.1 Euclidean Distance

The GPU and CPU versions of our described log-linear scheme for the computation of locally z-normalized subsequence alignment using ED are compared to the state-of-the-art implementation of the UCR-Suite (UCR-ED) [17]. To our best knowledge, UCR-ED is the fastest implementation of sED so far. Nevertheless, the

Table 3.2: z-Normalized Subsequence Euclidean Distance

sED ECG dataset – averaged runtime in seconds and speedups:

| $\|Q\|$ | 128 | 256 | 512 | 1024 | 2048 | 4096 | 8192 |
|---|---|---|---|---|---|---|---|
| GPU | 0.617 | 0.618 | 0.620 | 0.616 | 0.628 | 0.640 | 0.636 |
| MKL | 8.000 | 7.990 | 7.985 | 7.999 | 8.017 | 7.996 | 8.032 |
| UCR | 0.629 | 0.821 | 1.489 | 5.006 | 17.62 | 60.69 | 235.9 |
| MKL/GPU | 12.97 | 12.94 | 12.87 | 12.99 | 12.76 | 12.49 | 12.64 |
| UCR/GPU | 1.020 | 1.329 | 2.400 | 8.129 | 28.06 | 94.83 | 371.2 |

sED Metal dataset – averaged runtime in seconds and speedups:

| $\|Q\|$ | 128 | 256 | 512 | 1024 | 2048 | 4096 | 8192 |
|---|---|---|---|---|---|---|---|
| GPU | 0.437 | 0.434 | 0.436 | 0.439 | 0.440 | 0.449 | 0.448 |
| MKL | 3.930 | 3.930 | 3.937 | 3.929 | 3.931 | 3.929 | 3.922 |
| UCR | 0.863 | 2.005 | 4.470 | 8.785 | 15.16 | 28.72 | 55.30 |
| MKL/GPU | 8.996 | 9.064 | 9.033 | 8.944 | 8.943 | 8.758 | 8.754 |
| UCR/GPU | 1.976 | 4.624 | 10.25 | 19.99 | 34.50 | 64.02 | 123.4 |

implementation uses plain text files for the input of the queries and stream data which can be exceedingly expensive during the parsing of floating point values. Therefore, we have tuned the UCR code to handle binary inputs such that the whole data can be read once and written to arrays afterwards. As a result, the access times to the query and stream is only limited by the bandwidth of the RAM. The reported execution times for all algorithms exclude file IO to ensure fair competition and prevent influence of the used file system. Note that the binary-tuned versions always perform better than the original code and can be downloaded at our supplementary website [69].

Since single-precision FFT may not provide the needed accuracy for extremely long streams, our implementation features double-precision for floating point values. The obtained execution times are visualized in Figure 3.15 and stated explicitly in Table 3.2. The CUDA-implementation outperforms UCR-ED on all datasets for all tested parameter settings. The CPU code using Intel's Math Kernel Library (MKL-ED) performs worse than UCR-ED for short queries but better for longer ones. The runtime of our log-linear scheme does not depend on the query length resulting in almost constant execution times. In contrast to UCR-ED, arbitrarily long time series ($|Q| \leq |S|$) can be queried without a perceivable increase of the runtime. Furthermore, the log-linear scheme computes **all** alignments at once, UCR-ED only provides **one** optimal alignment candidate. Nevertheless, UCR-ED is a remarkably fast algorithm on short queries.

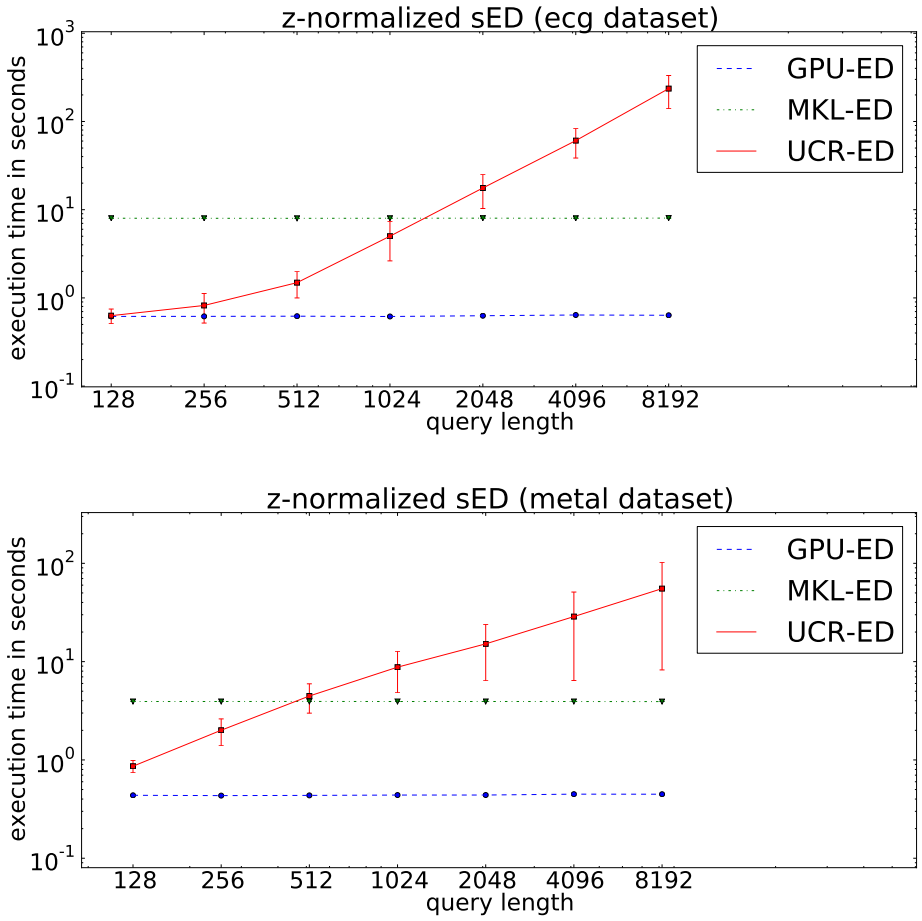


Figure 3.15: Averaged execution times for the subsequence alignment using locally z-normalized Euclidean distance on the two described datasets ECG ($|S| \approx 20$ million) and Metal ($|S| \approx 11$ million). Three different algorithms were investigated: Our CUDA-implementation (GPU-ED) using the cuFFT library (blue circles/dashed line), a CPU-implementation (MKL-ED) using Intel's Math Kernel Library (green triangles/dotted line) and an IO-tuned version of the subsequence Euclidean distance implementation of the UCR-Suite (red rectangles/solid line).

### 3.3.2 Constrained Dynamic Time Warping

We compare the runtimes of the CDTW measure for the described thread-level and block-level variants without using a lower bounding cascade to evaluate the raw CDTW performance. Furthermore, a single-core and an OpenMP (12 threads) implementation according to the pseudo-code in Figure 2.9 are tested. The OpenMP variant parallelizes the for-loop over the different alignment positions. The computation scheme of CDTW does not depend on the dataset, thus we only investigate

the ECG dataset for relative band sizes of 10% to 20% of the query length. Empirical benchmarks [16] of the 1NN-classification quality of CDTW suggest that the size of the Sakoe-Chiba band should be between 0% and 20%. Greater band sizes usually do not increase the classification quality and can even be harmful to the classifier.[20]

The stream size is reduced to 10,000 data points because of the demanding time complexity of $\mathcal{O}(|S| \cdot |Q| \cdot W)$. As stated before, our CDTW implementation uses a mixed-precision approach: The statistics of the stream are calculated in double-precision and the lower bounds as well as the relaxation of the penalty matrix are computed in single-precision. During the experiments, the maximum query length is limited to 4,096 due to limitations of the constant memory size but could easily be removed by transferring the query to global memory.

Execution times are illustrated in Figure 3.16 and explicitly reported in Table 3.3. The combined block/batch-level approach is about two times faster than the naïve thread-level implementation. Furthermore, it is about one order-of-magnitude faster than the OpenMP version and almost two orders-of-magnitude faster than the single-core variant. As a result, we focus on the faster block/batch-level CDTW methods for the remaining benchmarks.

### 3.3.3 Lower-Bounded Dynamic Time Warping

Our CUDA-parallelization (GPU-DTW) is compared to the CDTW portion of the UCR-Suite (UCR-DTW). UCR-DTW features a variety of additional strategies to reduce the runtime during the pruning of unpromising candidates among others an early-exit capable $\mathrm{LB}_{\mathrm{Kim}}$ and CDTW computation including the reordering of the query indices and a reversed $\mathrm{LB}_{\mathrm{Keogh}}$ lower-bound using warping envelopes $(L, U)_C$ of the candidates in the stream. Unfortunately, many of these techniques cannot be easily used on a GPU due to the SIMT restriction within warps. As a result, our described lower-bound cascade has a smaller filter efficiency than the one used by UCR-DTW and we therefore do not expect to gain the same speedups as in the raw CDTW computation. In the following, the stream size $S$ is fixed to 100,000 and the same experimental protocol is repeated for both ECG and Metal since the lower bound cascade is influenced by the characteristics of the time series. Additionally, UCR-DTW was tuned again with binary file IO to ensure fair competition. The execution times are reported in Figure 3.17 and Table 3.4, respectively. GPU-DTW outperforms UCR-DTW on all tested datasets for all tested settings. In most cases speedups of one order-of-magnitude can be achieved. Performance improves for longer queries e.g. on the ECG dataset about two third of the raw CDTW speed can put through the lower bound cascade.

[20] Please, see the CinC_ECG_torso dataset in the UCR repository where even Euclidean distance performs better than the unconstrained DTW measure.

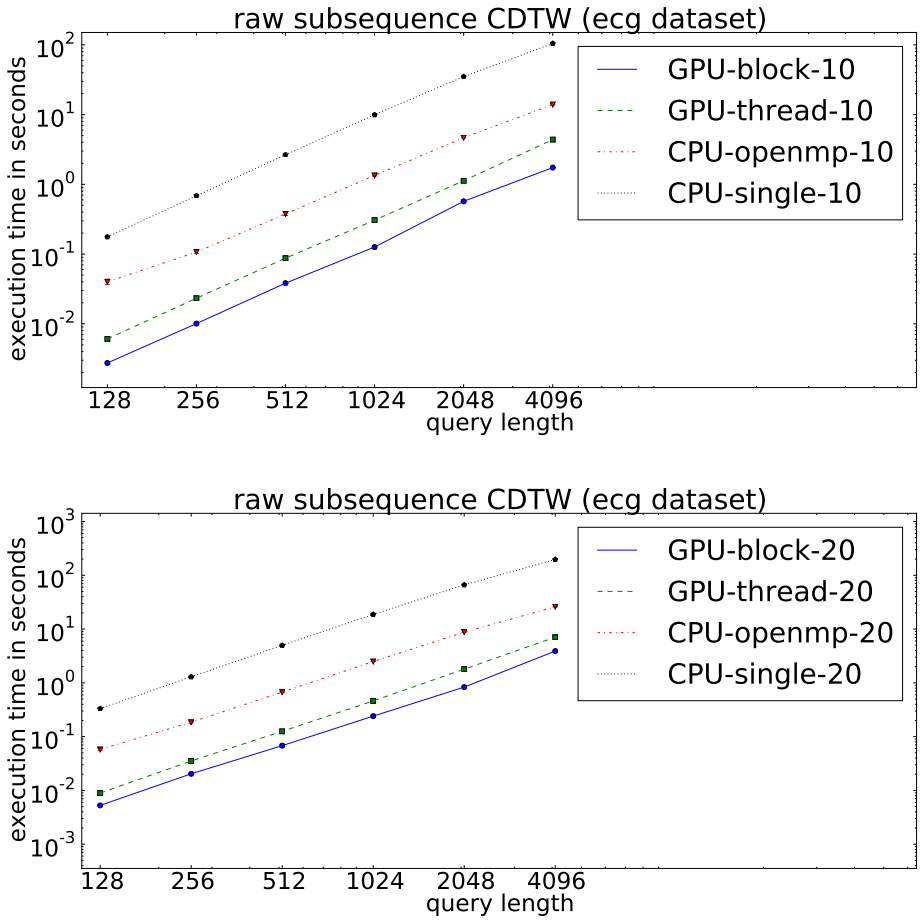


Figure 3.16: Averaged execution times for the subsequence alignment using locally z-normalized CDTW on the ECG dataset (here $|S| = 10,000$). Four different algorithms were investigated: Our combined block/batch-level-variants on the GPU (blue circles/solid line), the thread-level implementation on the GPU (green rectangles/dashed line) and the OpenMP (red triangles/dashed-dotted line) and single-core CPU versions (black pentagons/dotted line) for 10% and 20% relative band sizes of the Sakoe-Chiba band.

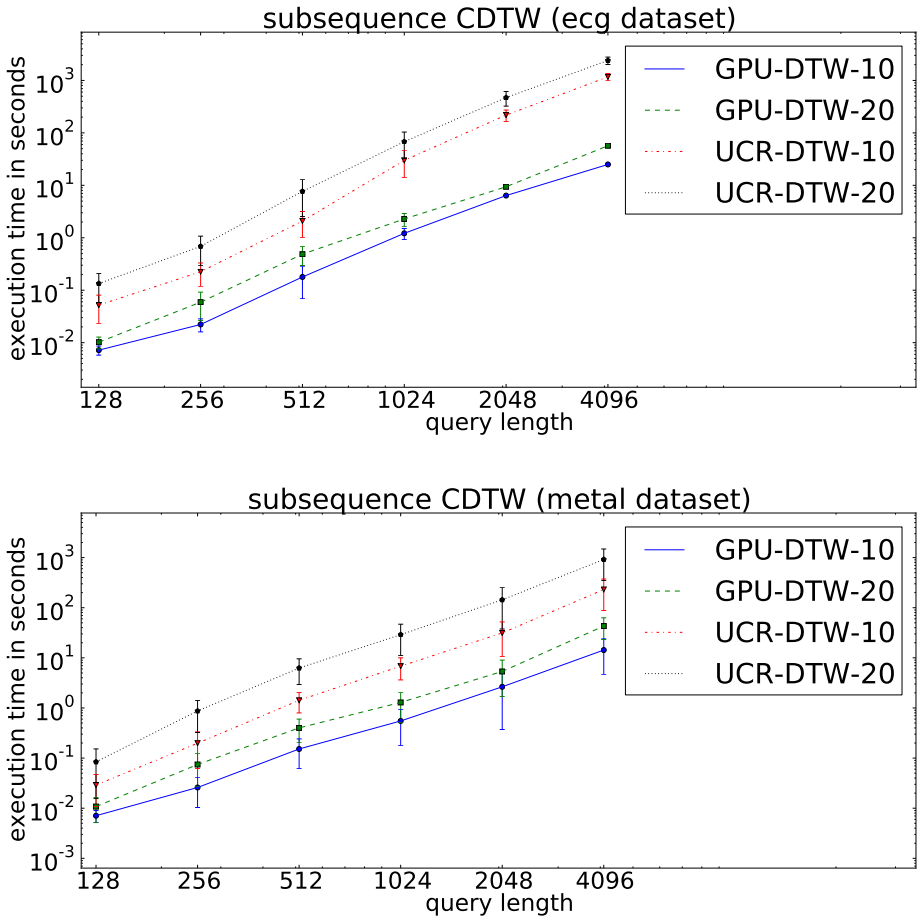


Figure 3.17: Averaged execution times for the lower-bounded subsequence alignment using locally z-normalized CDTW on the ECG and Metal dataset (here $|S| = 100{,}000$ ). Two algorithms were tested: Our lower-bounded combined block/batch-level CUDA-implementation of CDTW (blue circles/solid line for 10% and green rectangles/dashed line for 20%) and the lower-bounded CDTW implementation of the UCR-Suite (red triangles/dashed-dotted lines for 10% and black pentagons/dotted line for 20%).

Table 3.3: Raw z-Normalized Subsequence CDTW

Raw CDTW ECG 10 – averaged runtime in seconds and speedups:

| $\|Q\|$ | 128 | 256 | 512 | 1024 | 2048 | 4096 |
|---|---|---|---|---|---|---|
| Block | 0.003 | 0.010 | 0.038 | 0.126 | 0.571 | 1.738 |
| Thread | 0.006 | 0.023 | 0.087 | 0.307 | 1.126 | 4.379 |
| OpenMP | 0.040 | 0.107 | 0.376 | 1.343 | 4.662 | 13.93 |
| Single | 0.177 | 0.686 | 2.656 | 9.942 | 35.03 | 104.8 |
| Thread/Block | 2.215 | 2.312 | 2.282 | 2.428 | 1.970 | 2.519 |
| OpenMP/Block | 14.60 | 10.60 | 9.824 | 10.64 | 8.159 | 8.015 |
| Single/Block | 64.34 | 68.11 | 69.36 | 78.72 | 61.31 | 60.26 |

Raw CDTW ECG 20 – averaged runtime in seconds and speedups:

| $\|Q\|$ | 128 | 256 | 512 | 1024 | 2048 | 4096 |
|---|---|---|---|---|---|---|
| Block | 0.005 | 0.020 | 0.068 | 0.241 | 0.834 | 3.902 |
| Thread | 0.009 | 0.035 | 0.126 | 0.461 | 1.800 | 7.079 |
| OpenMP | 0.058 | 0.185 | 0.673 | 2.494 | 8.777 | 26.00 |
| Single | 0.331 | 1.294 | 4.978 | 18.73 | 66.32 | 196.9 |
| Thread/Block | 1.694 | 1.731 | 1.847 | 1.915 | 2.158 | 1.814 |
| OpenMP/Block | 10.98 | 9.077 | 9.902 | 10.37 | 10.53 | 6.662 |
| Single/Block | 62.88 | 63.53 | 73.21 | 77.82 | 79.53 | 50.46 |

Table 3.4: Lower-Bounded z-Normalized Subsequence CDTW

Lower-Bounded CDTW ECG 10 – runtime in seconds and speedups:

| $\|Q\|$ | 128 | 256 | 512 | 1024 | 2048 | 4096 |
|---|---|---|---|---|---|---|
| GPU-10 | 0.007 | 0.022 | 0.179 | 1.215 | 6.380 | 25.02 |
| UCR-10 | 0.052 | 0.225 | 2.087 | 29.93 | 218.4 | 1172 |
| UCR-10/GPU-10 | 7.226 | 10.13 | 11.68 | 24.64 | 34.23 | 46.82 |

Lower-Bounded CDTW ECG 20 – runtime in seconds and speedups:

| $\|Q\|$ | 128 | 256 | 512 | 1024 | 2048 | 4096 |
|---|---|---|---|---|---|---|
| GPU-20 | 0.010 | 0.059 | 0.486 | 2.283 | 9.386 | 56.91 |
| UCR-20 | 0.134 | 0.686 | 7.670 | 67.92 | 470.5 | 2402 |
| UCR-20/GPU-20 | 13.14 | 11.60 | 15.77 | 29.75 | 50.13 | 42.21 |

Lower-Bounded CDTW Metal 10 – runtime in seconds and speedups:

| $\|Q\|$ | 128 | 256 | 512 | 1024 | 2048 | 4096 |
|---|---|---|---|---|---|---|
| GPU-10 | 0.007 | 0.026 | 0.152 | 0.550 | 2.642 | 14.30 |
| UCR-10 | 0.029 | 0.198 | 1.420 | 6.816 | 31.38 | 230.8 |
| UCR-10/GPU-10 | 4.095 | 7.642 | 9.352 | 12.38 | 11.88 | 16.14 |

Lower-Bounded CDTW Metal 20 – runtime in seconds and speedups:

| $\|Q\|$ | 128 | 256 | 512 | 1024 | 2048 | 4096 |
|---|---|---|---|---|---|---|
| GPU-20 | 0.011 | 0.074 | 0.401 | 1.289 | 5.348 | 43.09 |
| UCR-20 | 0.084 | 0.870 | 6.251 | 28.98 | 143.9 | 917.5 |
| UCR-20/GPU-20 | 7.797 | 11.68 | 15.61 | 22.48 | 26.90 | 21.29 |

CHAPTER 4

# TOWARDS ELASTIC SUBSEQUENCE ALIGNMENT WITH LINEAR COST

*'Can we do better than UCR Suite?'* — Thanawin Rakthanmanon et al., *Searching and Mining Trillions of Time Series Subsequences under DTW*

Traditional 1NN search under elastic distance measures accounts for $\mathcal{O}\left(|S| \cdot |Q|^2\right)$ time independent of the use of lower bounds. Depending on the pruning power of the lower bound cascade, the leading constant omitted by the Landau notation can be reduced by a significant amount. Nevertheless, $\mathrm{LB}_{\mathrm{Kim}}$ as well as $\mathrm{LB}_{\mathrm{Keogh}}$ are loose in theory and practice when querying long time series.[1] Consequently, we aim for an elastic subsequence distance measure with $\mathcal{O}\left(|S| \cdot |Q|\right)$ worst time complexity that provides sufficient quality in means of 1NN classification error. During this chapter, we will discuss two traditional algorithms and introduce a novel similarity measure for the subsequence alignment of time series:

- **FastDTW (FDTW)** a hierarchical approach by Salvador and Chan [25]
- **GreedyDTW (GDTW)** an approximate DTW by MacLean and Labahn [70], later adapted as Lucky Time Warping (LDTW) by Spiegel et al. [71]
- **Gliding Elastic Match (GEM)** our fully relaxed subsequence measure [72]

[1] More specifically, $\mathrm{LB}_{\mathrm{Kim}}$ on long queries since it considers only a vanishing portion and $\mathrm{LB}_{\mathrm{Keogh}}$ on rapidly varying shapes where envelopes degenerate to global min – max enclosures.

The rest of this chapter is organized as follows: Firstly, we investigate the potential for parallelization of publicly available libraries providing classic implementations (C/C++, R, Java and Python) and linear-time approximations of DTW (FDTW and GDTW). Secondly, we introduce a fine-grained parallelization scheme for Spiegel's DTW approximation on CUDA-enabled accelerators which provides two orders-of-magnitude speedup in comparison to sequential code. Finally, our fully relaxed Itakura-flavoured subsequence DTW and its implementation on GPUs is discussed. The proposed scheme provides amortized linear cost for each alignment position while still being competitive to constrained subsequence DTW in means of 1NN classification error.

## 4.1 Performance of Publicly Available Libraries

### 4.1.1 Fast Dynamic Time Warping

A linear time algorithm for the approximate calculation of DTW scores was introduced by Chan and Salvador in 2004 [25]. The authors propose a recursive refinement strategy for the warping path starting at a coarse resolution of the penalty matrix.

Firstly, the involved time series have to be resampled at different resolutions on a dyadic grid in scale space. The implementation features a Haar wavelet decomposition as stated by the authors: '*Coarsening reduces the size (or resolution) of a time series by averaging adjacent pairs of points.*' This can be achieved in linear time using the lifting scheme for (second generation) wavelets [39] and can be proved by a simple calculation:

$$T(|C|) \propto \sum_{k=0}^{s_{\text{cutoff}}} \frac{|C|}{2^k} \leq |C| \cdot \sum_{k=0}^{\infty} \frac{1}{2^k} = \frac{|C|}{1-\frac{1}{2}} \in \mathcal{O}(|C|) \quad .$$

Secondly, the optimal warping path is processed on the coarsest resolution which accounts for constant time. Afterwards, the warping path is projected onto a grid of higher resolution (see Figure 4.1). Assuming again a dyadic refinement, the number of traversed cells will be increased by a factor of maximum four. Depending on the user's need, the resulting area may be expanded by applying a dilatation operation to ensure more variability of the warping path. The additional margin of **constant** width $r$ preserves the area's linear dependency on the length of the warping path. A detailed expression which explicitly considers the dilatation by $r$ cells is stated by the authors. The number of traversed cells $T(|C|)_0$ at the highest resolution can be upper-bounded by $|C| \cdot (4 \cdot r + 3)$ and thus the overall runtime is linear as long as the dilatation width does not depend on the length $|C|$:

$$T(|C|) \leq T(|C|)_0 \cdot \sum_{k=0}^{\infty} \frac{1}{2^k} = 2 \cdot T(|C|)_0 = |C| \cdot (8 \cdot r + 6) \in \mathcal{O}(|C|) \quad .$$

Although having a purely linear dependency in theory, the recursive refinement and dilation can be considered as heavy-weight operations acting on only small batches of consecutive memory. A a result, we expect FastDTW to outperform DTW on long time series but being slower on short ones.[2] The authors report a reasonable approximation quality of FastDTW in means of absolute distance values. Unfortunately, they do not state relative classification errors for FastDTW in comparison to fully relaxed DTW variants.

This could be of interest for future research since the refinement strategy introduces a non-trivial band constraint (the shaded area) which could possibly increase classification quality in comparison to unconstrained or Sakoe-Chiba-constrained DTW. Furthermore, $\mathrm{LB}_{\mathrm{Kim}}$ could be used to effectively lower-bound FastDTW on short time series in constant time since FastDTW is an upper bound of DTW. As a result, we could utilize a simplified lower bound cascade as used in our CUDA implementation of UCR-DTW. The FastDTW calls could be parallelized on diagonal lanes similar to the discussed wavefront scheme.

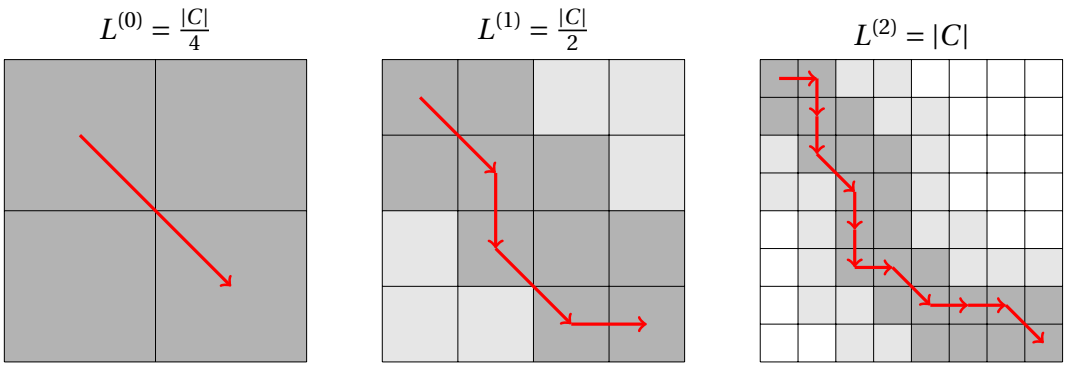


Figure 4.1: An example for the recursive refinement of the warping path (red) by repetitively projecting the traversed cells onto a grid of higher resolution (heavily shaded cells) followed by a morphologic dilatation (here $r = 1$) of the corresponding area with a single cell (grey shaded cells). Afterwards, the warping path is recalculated for each recursion level. The accumulated traversed area over all recursion levels exhibits only a linear dependency on the time series' length $|C|$.

### 4.1.2 Greedy Dynamic Time Warping

An even simpler approach for a linear cost approximation of DTW was introduced by MacLean and Labahn [70] in 2010. The authors propose a greedy approximation of the warping path $\hat{\gamma}$ by locally choosing the optimal extension for each cell of

[2] Explicit numbers are stated in Subsection 4.1.4.

the penalty matrix. The computational cost is proportional to the length of the warping path and thus upper-bounded by $\mathcal{O}(|C|)$ in contrast to the full relaxation of $\mathcal{O}(|C|^2)$ edges (see Figure 4.2). Moreover, GDTW requires only constant memory since the local extension of $\gamma$ does not depend on the preceding nodes. As a result, GDTW is a light-weight alternative to FastDTW. Unfortunately, MacLean and Labahn do not discuss constrained variants of GDTW or provide classification errors on the UCR repository [16].

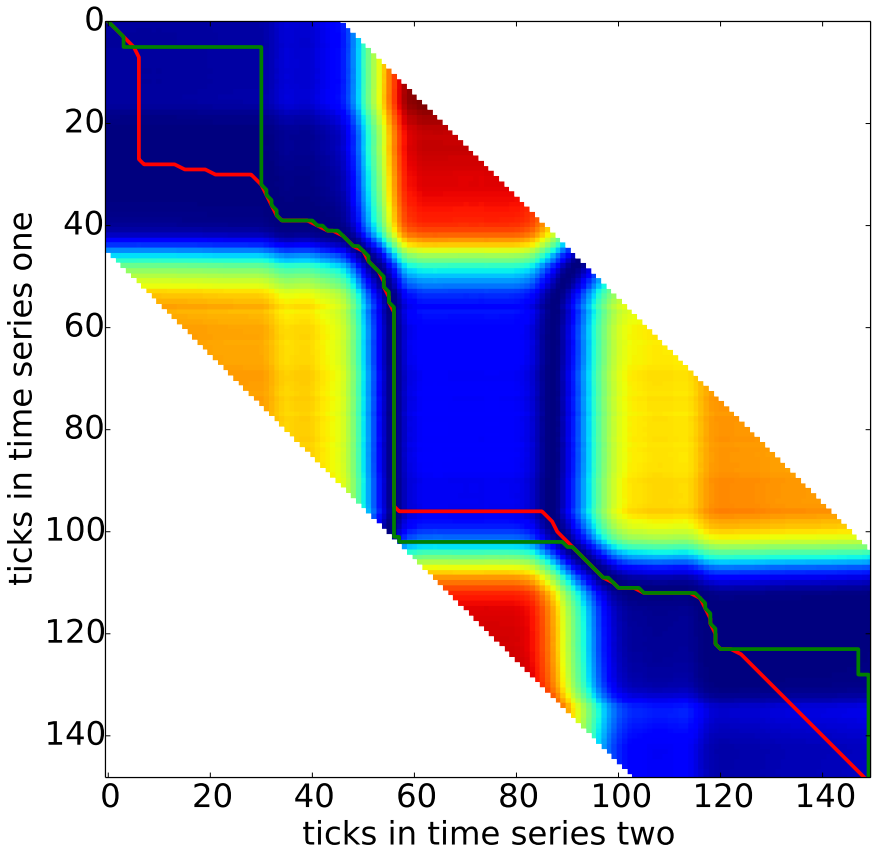


Figure 4.2: The optimal warping path (red) and the greedy warping path (green) for two time series from the Gun-Point dataset [16]. The color plot represents the entries in the Sakoe-Chiba-constrained penalty matrix ranging from small residues (blue) to higher ones (red). The majority of traversed cells are the same for the fully and greedily relaxed warping path. Minor differences can be spotted e.g. at $(60, 100)$ where the greedy path steadily follows the valley of small residues until it hits the constraint border.

In a recent paper, Spiegel et al. [71] investigated the classification quality of GDTW and its Sakoe-Chiba-constrained variants – rebranded as Lucky Time Warping similarity measure (LDTW). Omitting global constraints, we assume that local maxima on the original (fully relaxed) warping path are negligible or do not influence the approximation reasonably. In the complementary case, the approximate warping path and its optimal counterpart will not differ substantially due to the rigid restriction of the Sakoe-Chiba band. Consequently, the authors report competitive 1NN classification errors in comparison to CDTW on the whole UCR repository [16]. The corresponding pseudo-code for the Sakoe-Chiba-constrained Lucky Time Warping measure (CLDTW) is listed in Figure 4.3. Analogous to GDTW, the runtime of CLDTW is upper-bounded by the length of the warping path and therefore results in $\mathcal{O}(|C|)$ operations independent of the used window size. Unfortunately, Spiegel et al. only report accumulated costs for CLDTW in comparison to a conventional two stage $\text{LB}_{\text{Keogh}}$-CDTW lower bound cascade. Comparisons to UCR-Suite's four stage $\text{LB}_{\text{Kim}}$-$\text{LB}_{\text{Keogh}}$-$\text{LB}_{\text{reversed Keogh}}$-CDTW cascade and absolute execution times are lacking.

```
 1: function CLDTW²(Q, C, W)
 2:     (i, j) ← (0, 0)                          ▷ initial state in top left corner
 3:     distance ← (Q_0 − C_0)²
 4:
 5:     while i + 1 ≠ |Q| or j + 1 ≠ |C| do      ▷ not final corner
 6:         (diag, down, right) ← (∞, ∞, ∞)
 7:
 8:         if i + 1 < |Q| and j + 1 < |C| then
 9:             diag ← (Q_{i+1} − C_{j+1})²
10:         end if
11:         if i + 1 < |Q| and |i − j + 1| ≤ W then
12:             down ← (Q_{i+1} − C_j)²
13:         end if
14:         if j + 1 < |C| and |j − i + 1| ≤ W then
15:             right ← (Q_i − C_{j+1})²
16:         end if
17:
18:         distance += min(diag, down, right)
19:         readjust (i, j) according to the optimal direction
20:     end while
21:     return distance
22: end function
```

Figure 4.3: CLDTW pseudo-code with band constraint

### 4.1.3 Parallelization of Lucky Dynamic Time Warping

A straightforward port of the constrained Lucky Time Warping measure as stated in the pseudo-code in Figure 4.3 to the CUDA programming model is not obvious due to the `while`-loop in line five. The lengths of the greedily relaxed warping paths can differ significantly between nearby alignment positions in the stream $S$. As a result, branch divergence in a warp may drastically hurt the performance since divergent branches have to be serialized by the hardware.

Nevertheless, the maximum length of a warping path is upper-bounded by $|Q|+|C|$. Thus, we can enforce the coherent execution of instructions within a warp by steadily carrying out $|Q|+|C|-1$ relaxation steps. In cases where the relaxation should have stopped with less than the maximum amount of possible iterations, we read dummy values from the query $Q$ and alignment candidate $C$. To ensure correct results, the local weighting function $w(i,j)$ has to be multiplied with a binary auxiliary factor $\varphi(i,j)$ which is true (=1) if and only if $(i,j)$ is a valid cell in the penalty matrix and false (=0) in the opposite case. Finally, the modified weights can be written as:

$$\hat{w}(i,j) := \varphi(i,j) \cdot w\big(\varphi(i,j) \cdot i,\ \varphi(i,j) \cdot j\big) \quad .$$

The whole query and the corresponding portion of the stream can be written to shared memory since there is no need to store a penalty matrix. As a result coalesced reads are guaranteed for both the query and the stream. All threads in a block share nearby entries in the same fashion as the non-linear convolution scheme of $\mathrm{LB}_{\mathrm{Keogh}}$ in Figure 3.13. The state information $(i,j)$ and the distance variable $d$ can be kept in fast registers. Assuming 1,024 threads per block this scheme allows for the processing of queries up to the length of 5632 values:

$$\frac{48 \cdot 1024\,\mathrm{bytes}}{\mathrm{sizeof(float)}} > \underbrace{2 \cdot |Q| + \mathrm{threads} - 1}_{\text{entries in shared memory}} \quad .$$

To allow for the alignment of even longer queries, the values of $Q$ can be read from constant or texture memory. However, the performance will degrade by a factor of approximately two. Alternatively, the number of threads per block could be decreased accordingly.

Since CLDTW is an upper bound of the classic CDTW measure, the lower bounds of CDTW are still valid. However, $\mathrm{LB}_{\mathrm{Keogh}}$ has the same asymptotic time complexity as CLDTW and thus there is no benefit from using it. The less demanding $\mathrm{LB}_{\mathrm{Kim}}$ could be used but enforces a reordering of indices (see Figure 3.14) which excludes the efficient use of shared memory as described above. As a result, we limit ourselves to the brute-force computation of all alignment positions for the CLDTW measure.

### 4.1.4 Performance Evaluation

This subsection provides an overview of the performance of publicly available libraries for fully relaxed DTW and its linear cost approximations. The used protocol as well as the hardware are identical to the ones stated in Section 3.3.

#### Publicly Available Libraries

We investigate the runtimes of four publicly available libraries for the alignment under the DTW measure and compare them to our baseline implementation of DTW written in C/C++. Java-ML [73], R's DTW package [74] and mlpy [75] each provide single-threaded library calls for unconstrained DTW. Furthermore, Java-ML provides calls for FastDTW. Moreover, we reimplemented a sequential C/C++ version of Spiegel's Lucky Time Warping measure (original code written in Matlab). In the experiments, we neglect z-normalization and calculate all subsequence alignments for a given query in a ECG stream of 10,000 data points. The length of the queries ranges between 128 and 4,096. Execution times for the loading of the data are not considered. Detailed results and the code can be found on the supplementary website [69]. The runtimes are visualized in Figure 4.4 and explicit numbers are stated in Table 4.1. Our C/C++ baseline algorithm and mlpy's DTW

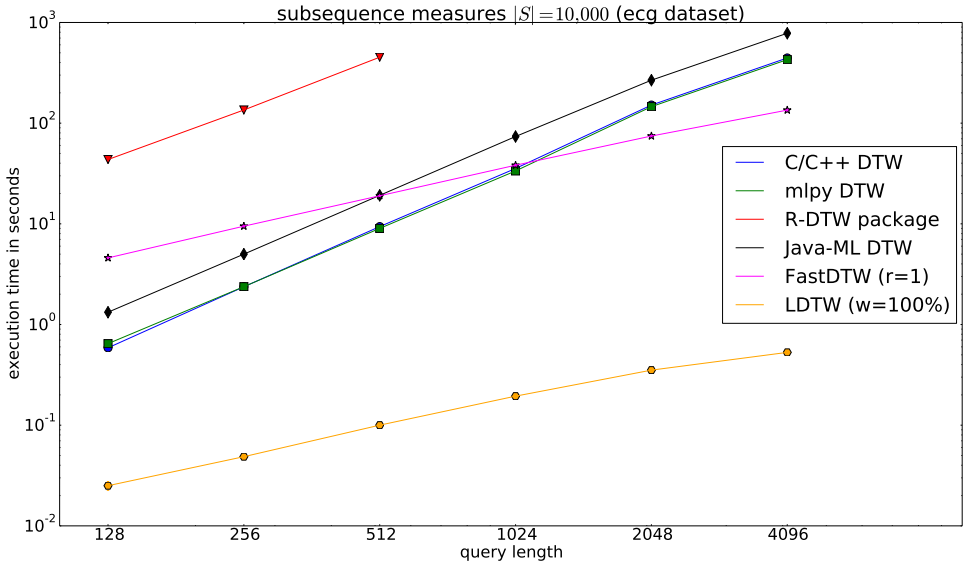


Figure 4.4: Double-logarithmic plot of the runtimes $T(|Q|)$ of the linear cost approximations FastDTW (magenta stars) and LDTW (orange hexagons) next to our baseline implementation of unconstrained DTW (blue circles), mlpy's DTW calls (green boxes), Java-ML DTW (black diamonds) and R's DTW package (red triangles).

Table 4.1: Single-threaded Subsequence Measures

averaged runtime in seconds:

| $\|Q\|$ | 128 | 256 | 512 | 1024 | 2048 | 4096 |
|---|---|---|---|---|---|---|
| native C/C++ | 0.587 | 2.371 | 9.386 | 35.47 | 151.2 | 444.1 |
| mlpy | 0.646 | 2.375 | 9.004 | 33.52 | 145.8 | 427.6 |
| Java-ML | 1.325 | 4.999 | 19.18 | 73.66 | 267.1 | 781.4 |
| R-DTW | 43.51 | 135.1 | 452.0 | intractable | intractable | intractable |
| FastDTW | 4.587 | 9.476 | 19.00 | 38.20 | 74.50 | 134.7 |
| LDTW C/C++ | 0.025 | 0.049 | 0.100 | 0.194 | 0.352 | 0.528 |

routine are the fastest among the fully relaxed DTW implementations. The Java-ML library is about two times and the DTW package of R about fifty times slower than the baseline. As expected, the graphs for all fully relaxed DTW implementations exhibit a slope $\kappa := \frac{\mathrm{d}}{\mathrm{d}k} \log_2 T(2^k)$ of approximate two and thus empirically verify DTW's quadratic dependency on $|Q|$. In contrast, the slope for the runtimes of FastDTW and unconstrained LDTW is given by $\kappa \approx 1$ which emphasizes their linear nature. The use of FastDTW is only beneficial for time series longer than 512 due to the non-negligible intercept. The sequential LDTW implementation outperforms all measures on all tested settings by at least one order-of-magnitude and thus is an ideally suited candidate for parallelization. The bend of the graphs at longer query lengths can be explained by a non-negligible distortion term of the overall runtime $T(|Q|) \propto (|S| - |Q| + 1) \cdot |Q|^\kappa = (|S| + 1) \cdot |Q|^\kappa - \underline{|Q|^{\kappa+1}}$ that can only be observed when $|S| \approx |Q|$.

### CUDA-Parallelization of the Lucky Time Warping Measure

The CUDA-parallelization of constrained LDTW is compared to a single-threaded and an OpenMP version on the CPU. The OpenMP implementation parallelizes the outer for-loop over all $|S| - |Q| + 1$ alignment positions in the stream (roughly 20 million for ECG). Since LDTW has no measurable dependency on the used dataset or the specified window size, we only report runtimes for the ECG dataset and a relative Sakoe-Chiba constraint of 10%. Execution times are illustrated in Figure 4.5 and explicitly reported in Table 4.2. The obtained speedups on the GPU increase for longer queries except for $|Q| = 4,096$. This can be explained by the utilization of the shared memory. In this case, one block uses more than 50% of the 48 KB limitation for one streaming processor. Thus, the scheduler can only assign one block for simultaneous execution. However, we constantly achieve two orders-of-magnitude faster execution times in comparison to a single CPU and speedups of more than twenty in comparison to OpenMP.

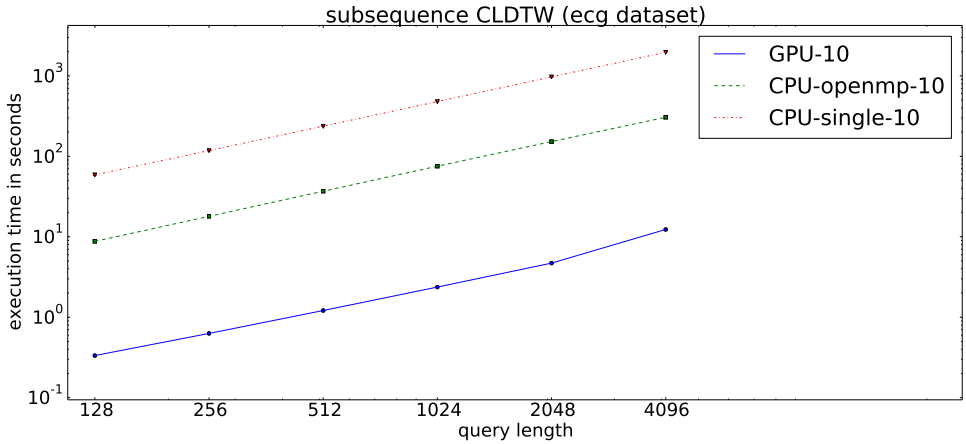


Figure 4.5: Averaged execution times for the subsequence alignment using locally z-normalized CLDTW on the ECG dataset (here $|S| = 20,040,000$ ). Three algorithms were tested: Our CUDA-implementation of CLDTW (blue circles/solid line), the OpenMP parallelization of CLDTW (green rectangles/dashed line) and a single-threaded baseline implementation of CLDTW (red triangles/dashed-dotted lines).

Table 4.2: z-Normalized Subsequence CLDTW

CLDTW ECG 10 – averaged runtime in seconds and speedups:

| $\|Q\|$ | 128 | 256 | 512 | 1024 | 2048 | 4096 |
|---|---|---|---|---|---|---|
| GPU | 0.334 | 0.629 | 1.211 | 2.371 | 4.705 | 12.32 |
| OpenMP | 8.768 | 17.95 | 36.83 | 75.35 | 152.4 | 305.3 |
| Single | 58.87 | 117.8 | 236.9 | 478.7 | 970.8 | 1964 |
| Single/GPU | 176.3 | 187.3 | 195.6 | 201.9 | 206.3 | 159.5 |
| OpenMP/GPU | 26.26 | 28.54 | 30.40 | 31.78 | 32.39 | 24.79 |
| Single/OpenMP | 6.714 | 6.561 | 6.434 | 6.354 | 6.369 | 6.434 |

## 4.2 Gliding Elastic Match

During subsequence alignment, the asymptotic cost $T_d$ for the distance computation of the query $Q$ and a single candidate $C$ influences the overall runtime linearly since $T_{\text{overall}} \in \mathcal{O}\big(|S| \cdot T_d\big)$. Thus, conventional approaches focus on the optimization of $T_d$ itself, namely FastDTW and LDTW, or alternatively try to reduce the impact of $T_d$ by lower-bounding with computational inexpensive approximations. The latter leads to heavily fluctuating runtimes depending on the characteristics of the alignment candidate, as seen in the previous chapter (see the enormous error bars in the Figures 3.15 and 3.17). Although FastDTW and LDTW guarantee linear cost, they can also be disadvantageous. FastDTW is practically intractable on short time series and LDTW may stick in local minima due to the greedy decision rule.

During the rest of this section, we pursue an alternative approach where we excessively exploit redundancy during the shifting of the alignment window to lower the overall complexity to $\mathcal{O}(|S| \cdot |Q|)$ while at the same time processing full relaxations of the penalty matrix. In 2007, Sakurai et al. [53] proposed a similar idea where the authors reused the information from nearby alignment positions to speedup the calculation of subsequence DTW. In 2010, this approach was abandoned by Sart et al. due to the missing z-normalization of each alignment candidate as stated in [76]:

> *One of the most cited methods is SPRING [27], where a query time series is searched in a larger streaming time series. The authors achieve significant speed-up by reusing computations. Unfortunately, this reuse means that the method allows false negatives [...]*

The importance of z-normalization has been stressed several times in the literature and proved empirically on different datasets [32, 77]:

> *The results are quite startling, without normalization time series similarity has essentially no meaning. More concretely, small changes in offset rapidly dwarf any information about the shape of the two time series in question.*

As a result, we can only access a candidate sequence after the z-normalization step. Thus, every globally relaxed subsequence alignment algorithm that relies on individual z-normalization of candidates will degenerate to the $\mathcal{O}(|S| \cdot |Q|^2)$ worst case. This can be avoided if we substitute z-normalization with an equivalent filter.

The rest of this section is organized as follows. Firstly, we discuss pitfalls of z-normalization and introduce a distinct windowed normalization filter, namely the exponential moving average (EMA). Secondly, the introduced EMA filter is interleaved with a translation and scale-invariant subsequence alignment algorithm similar to [31]. Thirdly, parallelization schemes for multi-core and CUDA-enabled devices are discussed followed by a performance evaluation in terms of quality and runtime. Finally, we provide an extension to multi-shape alignment tasks.

### 4.2.1 The Algorithm

The presented techniques are based on our publication at ACM SAC 2014 [72].

#### Pitfalls of z-Normalization

Removing local averages and the equalization of amplitudes is considered a crucial preprocessing step in time series data mining. During the last decade, z-

normalization has been the predominant method to achieve that:

$$z : \mathbb{R}^n \to \mathbb{R}^n \ , (C_0, \ldots, C_{n-1}) \mapsto z(C) = \frac{C - \mu}{\sigma} := \left( \frac{C_0 - \mu}{\sigma}, \ldots, \frac{C_{n-1} - \mu}{\sigma} \right) \quad ,$$

where $\mu$ is the average and $\sigma$ the standard deviation of $C$. The z-map can be decomposed into two independent normalization filters $z = A \circ O$, namely offset removal $O$ and amplitude equalization $A$. In the following, we discuss potential pitfalls of both filters:

- **(z-normalization is not elastic)** Offset removal $O(C) = C - \sum_{k=0}^{|C|-1} p_k \cdot C_k$ simply subtracts the mean from the original times series. Thus, each time tick $k$ contributes equally with $p_k = \frac{1}{|C|}$ to the estimate of the offset. However, a uniform distribution can only be assumed on the whole data set if we pair-wise compare time series under lock-step measures e.g. Euclidean or Manhattan distance. An a priori knowledge about $p_k$ is not accessible when applying elastic measures. As an example, consider two noiseless instances of the Cylinder class from the CBF data set (see Figure 1.3) $C_0 := 6 \cdot \chi_{[a,b]}$ and $C_1 := 6 \cdot \chi_{[A,B]}$. Although they should have vanishing distance under DTW, their different mean causes a non-vanishing result under DTW after z-normalization since:

  $$\mu_{C_0} = 6 \cdot \frac{b - a + 1}{128} \neq \mu_{C_1} = 6 \cdot \frac{B - A + 1}{128} \quad \text{for} \quad b - a \neq B - A \quad .$$

  We would have been better by simply omitting the offset removal. The same is true for amplitude equalization. The standard deviation and thus the final amplitude is heavily influenced by the temporal distribution of the time ticks. Thus, two time series with the same but differently warped features will have distinct amplitudes after z-normalization (see Figure 4.6).

- **(amplitude equalization removes physical units)** Many time series represent shapes of certain objects e.g. fish shapes and skull silhouettes [16]. As a result, it is desirable to remove arbitrary scaling factors caused by e.g. different camera settings or lighting conditions. However, a non-negligible amount of time series is recorded during the monitoring of physical processes e.g. motion-capturing of hand poses, voltages in an integrated circuit or pressures during the hot-rolling of metal coils. The latter intrinsically carry a physical unit which has to be taken care of during the alignment. As an example, it is meaningless to compare the velocity of a car with the speed of an aeroplane. Nevertheless, when using amplitude equalization, the two velocity profiles could match and consequently produce a false positive. In industrial settings, the absolute value of the measured amplitude is often an important indicator if the monitored system behaves as expected. In this context, amplitude equalization is prohibitive.

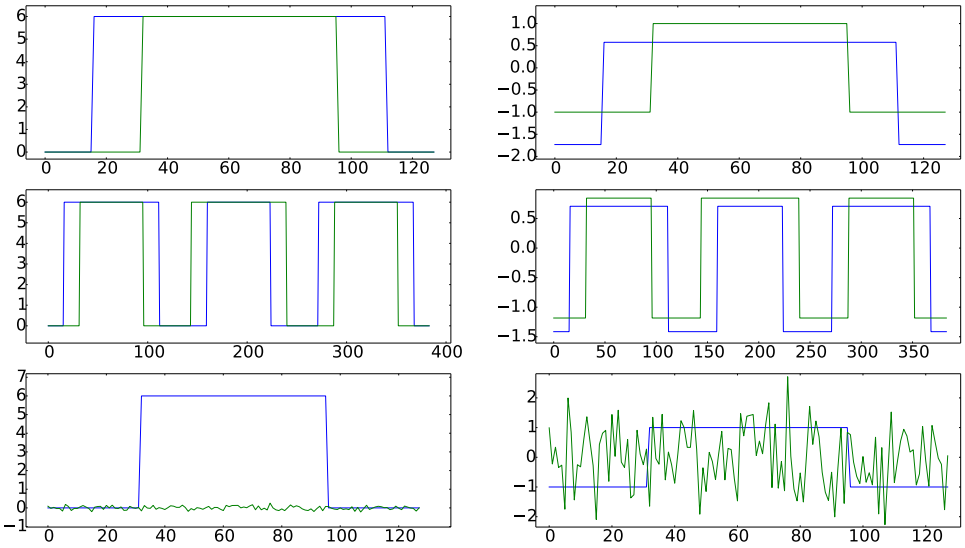

Figure 4.6: Three examples demonstrating improper use of z-normalization. The panels on the left each show two input signals (blue and green solid lines). The plots on the right depict their z-normalized variants. The two Cylinder instances $C_0 = 6 \cdot \chi_{[16,111]}$ and $C_1 = 6 \cdot \chi_{[32,95]}$ in the first row have vanishing distance under DTW. However, their z-normalized variants have non-vanishing distance. The same phenomenon can be observed when building mixed streams from the basis shapes $S_0 = (C_0, C_1, C_0)$ and $S_1 = (C_1, C_0, C_1)$ (middle row). Moreover, amplitude equalization tends to overemphasize meaningless noise on slowly-varying signals as demonstrated in the last row.

Concluding, the use of z-normalization is beneficial on many data sets but should not be imposed as an universal paradigm. Furthermore, the average $\mu$ and the standard deviation $\sigma$ obviously depend on the candidate's length. The majority of subsequence alignment algorithms including the UCR-Suite postulate this length by specifying the width of the sliding window. Another problem arises when aligning a query in a stream which has been sampled at a different rate. The optimal alignment candidate and the query will have different length due to uniform scaling. As an example, it is meaningless to align two periods onto three periods of a cosine since DTW has to match endpoints. Unfortunately, the original sample rates are often imprecise or completely lacking. In this case, z-normalization is **ill-defined** unless we brute-force all possible scaling factors between the query and the stream. Despite the unreasonable computational effort, a lower-bounded variant of this technique is considered as state-of-the-art [32]:

> *The main difficulty in creating uniform scaling invariance is that we typically do not know the scaling factor ahead of time, and are thus condemned to testing all possibilities within a given range [11].*

This problem as well as the above mentioned artefacts can be completely avoided if we abandon z-normalization as the one and only filter that has to be applied before the alignment step and substitute it by an equivalent filter which does not depend on the candidate's length.

### Restricted Affine Invariance in Time Domain

Consider again the typical use case of finding a given shape $Q$ in a huge time series $S$. As mentioned before, the global time scale of a potential candidate $C$ in $S$ may be slightly different from the one of $Q$. It is not useful to consider arbitrary small or big scaling factors of $C$ because the similarity of two time series practically implies a physical process with not necessary equal but similar scale. A domain expert can easily estimate the possible range, e.g. a drop or raise of the heart beat frequency by a factor of ten strongly indicates that it can not be caused by a living human being.

The Gliding Elastic Match (GEM) introduces a set of different monotone warping paths, denoted as $\Gamma_{\text{GEM}}$, that gives granular control over the upper and lower bounds of the occurring scale $s \geq 1$:

$$\begin{aligned}\Gamma_{\text{GEM}} := \big\{\gamma \in \Gamma \,|\, & i_0 = 0 \;\wedge\; i_{|\gamma|-1} = |Q| - 1 \\ & \wedge\; \min(i_{m+1} - i_m, j_{m+1} - j_m) = 1 \\ & \wedge\; \max(i_{m+1} - i_m, j_{m+1} - j_m) \leq s\big\}\,.\end{aligned}$$

Hence, the query $Q$ and a potential match $C$ in $S$ may only be mapped by a scale transformation in the interval $[1/s, s] \subset \mathbb{R}^+$. An exemplary set of allowed edges in the directed acyclic graph (DAG) $G$ is illustrated in Figure 4.7. Here, the query $Q = Q_{[0:3]}$ is mapped onto $M = S_{[1:6]}$ in an elastic manner. As $\frac{1}{3} < \frac{1}{s} = \frac{1}{2}$, the vertex $(3,1)$ is unreachable and cannot be matched. In addition, independent scaling factors $s_-, s_+$ for shrinking and enlarging the query $Q$ are conceivable. Furthermore, the possibility to skip values in $Q$ and $S$ provides robustness against noise from sensors and occlusion.

Every vertex $(0, j)$ in row zero is a possible starting node for the calculation of the best warping path $\hat{\gamma}$. Therefore, GEM may translate the query $Q$ to every suitable position in $S$.[3] In summary, GEM provides invariance under affine mappings of the time domain for a given restriction $[1/s_-, s_+] \subset \mathbb{R}^+$ of the scaling factor.

### Automatic Offset and Trend Adjustment

A method for automatic offset adjustment in matching algorithms was proposed by Latecki et al. [10] in connection with the MVM algorithm. The objective function

[3] Although, there are several starting nodes, it is still a SSSP problem by adding a virtual super node that is connected to every vertex $(0, j)$ with vanishing weights.

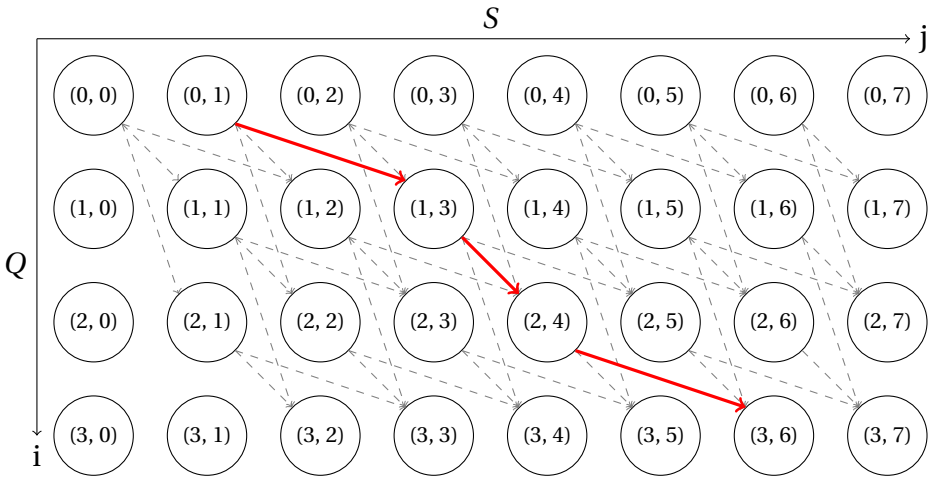


Figure 4.7: An $s = 2$ edge scheme for the GEM-algorithm and an exemplary warping path (red/bold).

$w_{\mathrm{MVM}}(i_m, j_m) = |Q_{i_m} + \Delta_m - S_{j_m}|^p$ for a given $p \in \mathbb{N}$ was introduced where the offset $\Delta_m$ is calculated by an incremental ordinary average over all residues $U_m := \left(S_{j_m} - Q_{i_m}\right)$ in the warping path $\gamma$:

$$\Delta_{m+1} = \frac{m+1}{m+2}\Delta_n + \frac{1}{m+2}U_{m+1} \quad \text{with} \quad \Delta_0 := U_0 \quad .$$

The starting nodes in row zero do not contribute to the measure since $w_{\mathrm{MVM}}(i_0, j_0) = |Q_{i_0} + (S_{j_0} - Q_{i_0}) - S_{j_0}|^p = 0$. The residue is then carried along the whole warping path by ordinary averaging.

This method is feasible to adjust a global offset. Consider the query $Q$ being a shifted variant of the subject sequence $S$ with the same length, i.e. $S = Q + N$ where $N$ is a random variable with sample mean $\mu$. The average residue will reproduce the shift exactly [10]:

$$\Delta_{|\gamma|-1} = \frac{1}{|\gamma|}\sum_{i=0}^{|\gamma|-1}\left(S_i - Q_i\right) = \frac{1}{|\gamma|}\sum_{i=0}^{|\gamma|-1} N_i = \mu \, .$$

Unfortunately, this approach is not applicable to a time-dependent trend adjustment. To illustrate this point, consider $Q$ to be a deformed variant of $S$ with the same length, i.e. $S = Q + D$ where $D$ is a deformation represented by a uniformly sampled cosine wave on the interval $[0, 2\pi)$:

$$\Delta_{|\gamma|-1} = \frac{1}{|\gamma|}\sum_{i=0}^{|\gamma|-1} D_i = \frac{1}{|\gamma|}\sum_{i=0}^{|\gamma|-1} A\cos\left(\frac{2\pi i}{|\gamma|}\right) = 0 \, .$$

The deformation $D_i$ at $i = |\gamma|$ is obviously the amplitude $A$ but would be estimated incorrectly with $0 \approx \frac{A}{|\gamma+1|} = \Delta_{|\gamma|}$. Therefore, global ordinary averaging cannot be used for a local trend removal. This problem can be approached by using a windowed averaging filter. Unfortunately, it is difficult to perform an on-the-fly calculation of an ordinary windowed average with constant memory[4] per node since all the values in the window have to be cached at the same time.

### Moving Average with Constant Memory

GEM employs a distinct low-pass filter, that can be applied for offset and time-dependent trend adjustment with constant memory utilization. In contrary to the MVM algorithm the exponential moving average (EMA) is used to propagate the residues along the warping path $\gamma$ for an $\alpha \in [0,1]$:

$$\Delta_{m+1} = (1-\alpha) \cdot \Delta_m + \alpha \cdot U_{m+1} \quad \text{with} \quad \Delta_0 := U_0 \quad .$$

Let $\left(U_m\right)_m = \left(S_{j_m} - Q_{i_m}\right)_m$ be the sequence of residues on $\gamma$, then the EMA-filter can be calculated in exactly the same manner as the ordinary average. Although, EMA is calculated on whole $\gamma$, the average $\Delta$ satisfies a local property as shown by Klinker [78].

**Proposition 6** (Klinker)**.** *Let* $\left(U_m\right)_m$ *be an admissible sequence,* $\Delta$ *the* Exponential Moving Average *and* $\Delta^{(N)}$ *its windowed variant restricted to the last N values, then*

$$\frac{|\Delta - \Delta^{(N)}|}{|\Delta|} < (1-\alpha)^N \, .$$

The influence of residues declines exponentially with the distance taken on the warping path $\gamma$. Hence, the EMA filter is essentially a windowed average that can be computed with constant memory. Furthermore, the user can control the effective window size with the averaging coefficient $\alpha$.

As seen in the previous section, GEM allows edges that skip at least one row or column for a scaling bound of $s \geq 2$. In this case fractional indices emerge, that cannot be covered by the discrete filter given above. Therefore, GEM introduces an adaptive averaging coefficient $\alpha$ depending on the step size with the help of the corresponding differential equation:

$$\frac{\mathrm{d}}{\mathrm{d}t}\Delta(t) + \frac{1}{\tau}\Delta(t) = \frac{1}{\tau}U(t) \quad \text{with} \quad \tau \in \mathbb{R}^+. \tag{4.1}$$

Without loss of generality assume $\Delta(t=0) = \Delta_m$ and further approximate locally $U(t) = U_{m+1}$ for all times $t > 0$, then the unique solution of (4.1) is given by:

$$\Delta(t) = \exp\left(-\tfrac{t}{\tau}\right) \cdot \Delta_m + \left(1 - \exp\left(-\tfrac{t}{\tau}\right)\right) \cdot U_{m+1} \, .$$

[4] Please note, we do not claim that a windowed average cannot be calculated in constant **time** which is indeed possible with the help of prefix sums.

Consequently, the adaptive averaging coefficient of the non-uniform EMA filter is defined as $\alpha_t := 1 - \exp\left(-\frac{t}{\tau}\right)$. For $\Delta_{m+1} := \Delta(t = 1)$ we assume $\alpha_1$ being the right choice for unit steps. Furthermore, $\alpha_t$ can be expressed in terms of $\alpha_1$:

$$\alpha_t = 1 - \exp\left(-t \cdot \ln\left(\frac{1}{1-\alpha_1}\right)\right) = 1 - (1-\alpha_1)^t \,.$$

The parameter $\alpha_1 \in [0,1]$ can be interpreted as flexibility of the trend adjustment. Let $\alpha_1 := 0$ and therefore $\alpha_t = 0$ for all $t \in [0,\infty)$, then the average $\Delta_m = U_0$ is a constant approximation over the whole warping path $\gamma$. Otherwise, let $\alpha_1 = 1$, then the average $\Delta_m$ reassembles exactly the sequence of residues $U_m$. The first limit can be considered as one-sample constant offset approximation and the second one as super-flexible trend adjustment, that maps any time series $Q$ onto $S$ with zero cost. In real-world application small values of $\alpha_1 \ll 1$ provide a robust and local trend adjustment.

### 4.2.2 Complexity

As discussed, two distinctive features of GEM are bounded scale invariance in the time domain and automatic local trend adjustment. Therefore, the final algorithm depends on three external parameters, the shrinking scale $s_-$, the stretching scale $s_+$ and the flexibility degree $\alpha_1$. The pseudocode of GEM is given in Figure 4.8.

The matrices $C, L$ and $\Delta$ are updated in the relaxation procedure RELAX shown in Figure 4.9. GEM minimizes the cost along the warping path $\gamma$ normalized to its length. This normalization has to be applied to prevent the algorithm favouring shorter paths. GEM therefore minimizes the mean error after trend adjustment. The relaxation of the whole DAG $G$ can be achieved in $\mathcal{O}\big(|Q| \cdot |S| \cdot (s_- + s_+ - 1)\big)$ time and $\mathcal{O}\big(|S| \cdot (s_- + 1)\big)$ space by cyclic writing to the memory. As stated above, a linear memory algorithm is possible solely because the EMA filter can be calculated with constant memory per node. Conventional windowed averaging would need space proportional to $|Q| \cdot |S|$ which would be infeasible even for moderate sized queries $Q$. Back-tracing the optimal warping path $\hat{\gamma}$ can be done on a much smaller subgraph of $G$ with a number of nodes proportional to $|Q|^2 \ll |Q| \cdot |S|$.

### 4.2.3 Parallel Relaxation

The distinct scheme for the edges in the DAG $G$ allows for an efficient fine-grained parallelization of GEM. Since there is no data dependency (horizontal edge) between individual vertices every node in a single row of $G$ can be relaxed independently. Furthermore, forward relaxation is replaced by a backward relaxation scheme to avoid any race conditions for updating matrix entries. More specifically, the GEM pseudo-code in Figure 4.8 is modified by replacing the forward calls to RELAX $\big((i,j),(i+1+k,j+1),w,\delta\big)$ and to RELAX $\big((i,j),(i+1,j+1+l),w,\delta\big)$ by

1: **function** GEM($Q$, $S$, $s_-$, $s_+$, $\alpha_1$)
2: initialize cost, length and average matrices $C, L, \Delta$
3: **for** $i = 0 \rightarrow |Q| - 2$ **do**
4: **for** $j = 0 \rightarrow |S| - 2$ **do**
5: **for** $k = 0 \rightarrow s_- - 1$ **do** ▷ jumps in rows
6: **if** $i + 1 + k < |Q|$ **then**
7: $\alpha \leftarrow 1 - (1 - \alpha_1)^{1/(k+1)}$
8: $\delta \leftarrow (1 - \alpha) \cdot \Delta_{i,j} + \alpha \cdot (S_{j+1} - Q_{i+1+k})$
9: $w \leftarrow |Q_{i+1+k} + \delta - S_{j+1}|^p$
10: RELAX $\big((i, j), (i + 1 + k, j + 1), w, \delta\big)$
11: **end if**
12: **end for**
13: **for** $l = 1 \rightarrow s_+ - 1$ **do** ▷ jumps in cols
14: **if** $j + 1 + l < |S|$ **then**
15: $\alpha \leftarrow 1 - (1 - \alpha_1)^{l+1}$
16: $\delta \leftarrow (1 - \alpha) \cdot \Delta_{i,j} + \alpha \cdot (S_{j+1+l} - Q_{i+1})$
17: $w \leftarrow |Q_{i+1} + \delta - S_{j+1+l}|^p$
18: RELAX $\big((i, j), (i + 1, j + 1 + l), w, \delta\big)$
19: **end if**
20: **end for**
21: **end for**
22: **end for**
23: find optimal length-normalized cost $\hat{d}$ in the last row of $C$
24: backtrace $\hat{\gamma}$ with memoized predecessor information
**return** optimal cost $\hat{d}$ and warping path $\hat{\gamma}$
25: **end function**

Figure 4.8: GEM pseudocode

1: **procedure** RELAX($(i_m, j_m), (i_{m+1}, j_{m+1}), w, \delta$)
2: **if** $\frac{C_{i_m,j_m} + w}{L_{i_m,j_m} + 1} < \frac{C_{i_{m+1},j_{m+1}}}{L_{i_{m+1},j_{m+1}}}$ **then** ▷ normalized to length
3: $C_{i_{m+1},j_{m+1}} \leftarrow C_{i_m,j_m} + w$ ▷ update $C$
4: $L_{i_{m+1},j_{m+1}} \leftarrow L_{i_m,j_m} + 1$ ▷ update $L$
5: $\Delta_{i_{m+1},j_{m+1}} \leftarrow \delta$ ▷ update $\Delta$
6: **end if**
7: **end procedure**

Figure 4.9: Pseudocode of the edge relaxation procedure RELAX

backward calls to RELAX $\big((i - 1 - k, j - 1), (i, j), w, \delta\big)$ and to RELAX $\big((i - 1, j - 1 - l), (i, j), w, \delta\big)$). Consequently the loop over $j$ can be parallelized independently for each value of $j$.

Several CUDA kernels, each operating on a different row of GEM matrices,

are queued to be executed in sequential order. Within a row, consecutive CUDA threads work on results of consecutive entries to ensure coalesced accesses to the global memory. Since backward relaxation is used, intermediate values can be stored in registers and only the final results are written to global memory. Memory reads to previously relaxed entries are also coalesced. For a previously relaxed entry, there are at most $\max(s_+, s_-)$ threads reading it. Thus, if the values of $s_+$ and $s_-$ are large, one can utilize CUDA shared memory to enhance speed. Figure 4.10 explains this memory access pattern.

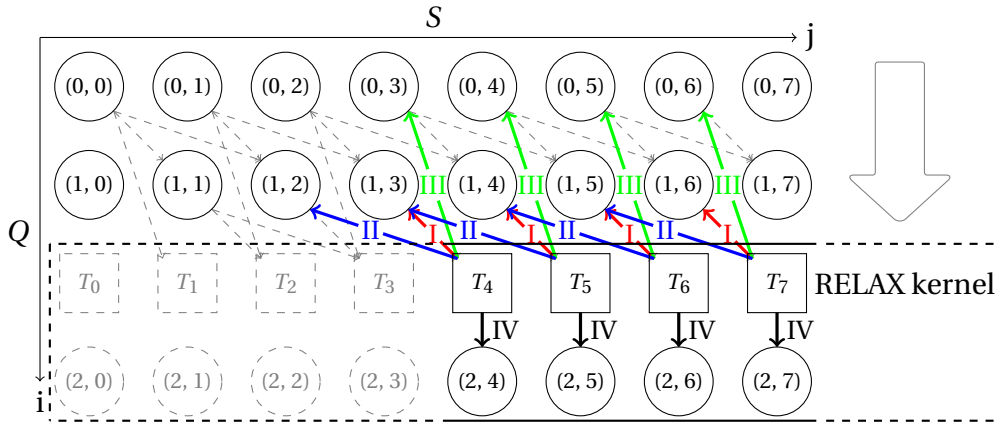


Figure 4.10: An example of coalesced memory access during the relaxation of four nodes in a single kernel invocation. Each thread relaxes the three ($s = 2$) possible backward edges consecutively (I, II, III / red, blue, green) and finally updates the score, length and averaging matrices (IV / black).

In comparison to the CUDA implementation of subsequence DTW presented in [76] (CUDA-subsequence-DTW), CUDA-GEM does not require local memory for each thread since three registers per thread are enough to store the GEM states $(C, L, \Delta)$ when relaxing an entry. Since local memory accesses are significantly slower than register accesses, our CUDA version is able to achieve a high performance.

We further optimize the code for the newest generation of CUDA-enabled devices with compute capability version 3.0 (Kepler). The new version enables launching up to $2^{31} - 1$ blocks in each kernel call. Hence, we can spawn as many threads as the size of $S$ and let the scheduler assign the thread blocks to the hardware.

### 4.2.4 Matching Quality

Although GEM features translation-invariant subsequence matching in the time domain we compare its matching quality to global matching algorithms. Given

two time series $Q$ and $S$ of the same length then translation-invariant matching degenerates to the global case since a potential match $C$ in $S$ has the approximate length of $S$ itself. The superiority of subsequence DTW over other local matching algorithms is usually justified by the matching quality of the (global) DTW similarity measure [76]. We therefore infer statements about GEM's local matching quality from the global case. In the following, matching quality will be expressed in terms of classification errors of 1NN-classifiers. Since GEM is not a symmetric distance measure[5] we always use its symmetrized variant $\text{gem}^*(Q,S) := \min\big(\text{gem}(Q,S), \text{gem}(S,Q)\big)$ during quality benchmarks to avoid ambiguity in the definition of nearest neighbours. Euclidean-flavoured versions of the distance measures are used in all experiments i.e. the parameter $p$ of the $L_p$-norm is set to 2.

### Metal Dataset

The Metal dataset consists of 1000 pairs of thickness profile measurements of metal coils. Given a thickness measurement $\text{child}^{(i)}$ we assign the corresponding time series $\text{parent}^{(i)}$ for all $i \in \{0,\ldots,999\}$. As a result, the dataset consists of 1000 classes with two members each. Since GEM and CDTW both depend on external parameters, we have to carry out parameter learning. Unfortunately, every stratified split of the dataset into a training and test set creates classes with exactly one member which renders parameter learning infeasible. Therefore, we separate a small subset for parameter learning beforehand and determine the error rates on the complementary set to avoid leakage. For our experiments we use the following protocol:

(i) Shuffle the index set $\mathscr{I} := \{0,\ldots,999\}$ with a random permutation $\pi$ to obtain $\pi(\mathscr{I}) := \{\pi(0),\ldots,\pi(999)\}$.

(ii) Split $\pi(\mathscr{I})$ into four disjoint index subsets – two pairs of parameter learning and holdout sets: $(L_{\text{left}}, H_{\text{left}})$ as well as $(L_{\text{right}}, H_{\text{right}})$ with $|L_j| = 100$ and $|H_j| = 400$ each.

(iii) For each of the two pairs $(L_j, H_j)$ do the following: Learn the parameters of the given distance measure on the indices $L_j$ by picking the parameter that yields minimum classification error of a 1NN-classifier built with the parents. After picking the most promising parameter determine the 1NN-classification error on the complementary index set $H_j$ in the same manner. Finally, one yields one error for each $H_j$. Go to (i) until finished.

[5] Exemplary, consider two time series $Q, S$ with lengths $|Q| \cdot s_+ < |S|$ for a $s_+ \geq 1$ then $\text{gem}(Q,S) \in \mathbb{R}^+$ but $\text{gem}(S,Q) = \infty$ since $S$ is too long.

The partition of the dataset is motivated by the following criteria: Parameters are learned on a small subset since this is the desired real-world setting. The parameter for CDTW is given by the relative window size of the Sakoe-Chiba Band [15] and ranges from 0% to 20%. GEM's scaling parameters $s_-, s_+ \in S$ and the averaging coefficient $\alpha_1 \in A$ are taken from the sets $S = \{1,2\}$ and $A = \left\{2^{-p} \mid p \in \{1,\ldots,8\}\right\}$. Shuffling the indices during step (i) ensures variability of parameter learning scenarios. Additionally, the dataset is split into two pairs $(L_j, H_j)$ to guarantee the same variability for the final error rate since $H_j$ is substantially altered each run. Otherwise, we would end up with nearly identical holdout sets if we provided only one pair $(L, H)$ for each iteration.

In our experiments, we investigate ED, (un)constraint DTW and symmetrized GEM during 120 iterations compliant with the given protocol. All distance measures but GEM are allowed to use z-normalization. In the following, results are only reported for CDTW and GEM since both considerably outperform unconstrained DTW and ED.[6] Hence, one obtains a sequence of 240 (relative) error rates $(E_i^d)_i$ (see Figure 4.11) for each distance measure where

$$\overline{E^{\mathrm{CDTW}}} \pm \sigma_{\mathrm{CDTW}} = 0.134 \pm 0.024 \quad \text{and} \quad \overline{E^{\mathrm{GEM}}} \pm \sigma_{\mathrm{GEM}} = 0.017 \pm 0.014 \quad .$$

The averaged error rates are determined on the same samples each iteration to guarantee fairness. Additionally, the mean $\overline{\Delta}$ and the standard deviation $\sigma_\Delta$ for the sequence of differences $(\Delta_i)_i = (E_i^{\mathrm{CDTW}} - E_i^{\mathrm{GEM}})_i$ are given by $\overline{\Delta} \pm \sigma_\Delta = 0.116 \pm 0.026$. As a result, GEM outperforms CDTW on the metal dataset with an approximate 8 times smaller error rate. This improvement is crucial for quality management in rolling mills since every alert notifying a possible mix-up enforces manual intervention of a domain expert.

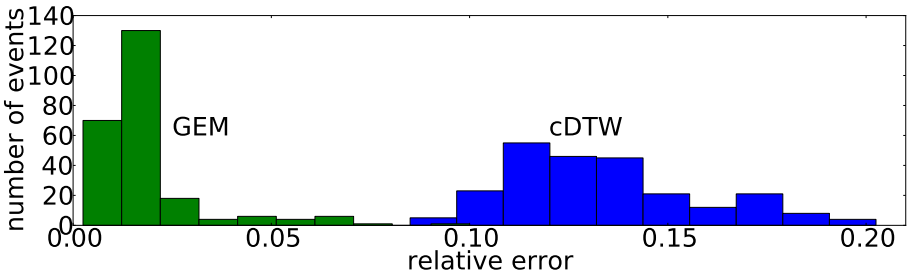


Figure 4.11: Histograms of relative errors for GEM (green/left) and CDTW (blue/right) each on the same samples for 120 shuffled splits (i.e. 240 errors).

[6] The error rates for all distance measures and detailed information about the individual iterations are provided on the supplementary website [79].

### Fish Dataset

Local deformation is not exclusively limited to time series from the metal domain. The fish dataset [3] consists of 350 time series extracted from the silhouettes of fish and exhibits seven distinguishable classes. Five members of the first category are exemplary shown in Figure 4.12. Obviously, the dataset was preprocessed

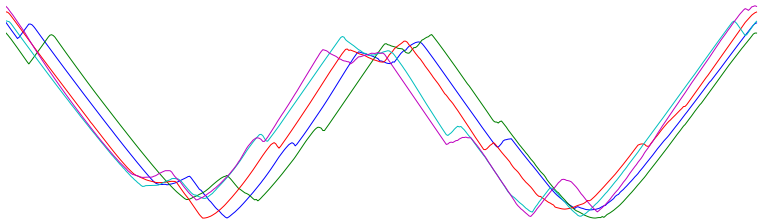

Figure 4.12: Five members of the first class of the fish dataset plotted together. The tiny local structures on top of the signals are shifted along **both** axes.

to have same length, approximate alignment and amplitudes. We therefore do not expect (C)DTW to perform considerably better than ED. However, a close look at Figure 4.12 reveals tiny structures on top of the signals that are shifted on both axes. Shifts on the time-axis may be covered by warping but not on the measurement-axis. GEM's invariance against local deformation can be exploited to improve the situation.

Again, we consider a resampling strategy for our experiments to provide means and standard deviations. The fish dataset has been included into the UCR database [16] and therefore a canonical split into a training and test set is provided. For further resampling we use the following protocol:

(i) For the first iteration take the UCR split, otherwise accomplish a stratified random split of the dataset into a training and test set conserving the split and class ratios.

(ii) Learn the parameters of the distance measures with Leave-One-Out-Cross-Validation (LOOCV) on the training set. Pick the parameter that minimizes the error rate.

(iii) Build a 1NN-classifier with the training set and the optimal parameter. Determine the final error rate on the test set with this classifier. Return to (i) until finished.

During the learning of parameters we use the same parameter sets as in the metal experiments. This protocol is executed 240 times. The results show, that

unconstrained DTW performs slightly worse than CDTW and ED.[7] In the following, we focus again on the comparison of CDTW and GEM (see Figure 4.13). The means

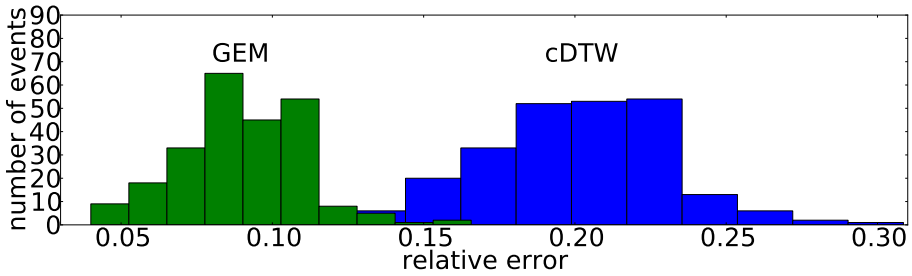


Figure 4.13: Histograms of relative errors for symmetrized GEM (green/left) and CDTW (blue/right) on the same samples for 240 stratified shuffled splits.

and standard deviations for the individual distance measures are given by

$$\overline{E^{\mathrm{CDTW}}} \pm \sigma_{\mathrm{CDTW}} = 0.200 \pm 0.029 \quad \text{and} \quad \overline{E^{\mathrm{GEM}}} \pm \sigma_{\mathrm{GEM}} = 0.090 \pm 0.020 \quad .$$

The differences of relative errors yield the following outcome $\overline{\Delta} \pm \sigma_{\Delta} = 0.111 \pm 0.034$. This improvement of the relative error rate by a factor of about 2 demonstrates that invariance against local deformation can also substantially increase accuracy offside the metal domain.

**UCR Database**

In this section, we further investigate the performance of CDTW and GEM on a range of datasets from different domains including ECG, mass spectra and motion capture data. We repeated the above mentioned protocol from the fish dataset for almost all of the 47 members of the UCR time series repository [16]. Since we want to give an impression of the general performance the number of iterations is limited to 50 instead of 240. The obtained error rates for CDTW and GEM are shown in Figure 4.14. On the first sight CDTW and GEM perform equally well but a close look reveals that GEM can considerably decrease the error rates on several datasets e.g. OSULeaf (-13%), Car (-9%) and TwoLeadECG (-8%). The use of GEM may also increase the error rate as happened in Lighting7 (+10%). Nevertheless, the benefits of GEM are predictable on small samples. Exemplary, the CBF dataset exhibits a ratio of training and test set sizes of $\frac{30}{900}$. The final performance on the usually much bigger test set can be inferred from the resubstitution error on the training set during parameter learning. Figure 4.15 shows the average difference of relative errors $\overline{\Delta}$ (CDTW-GEM) on the training and test set for each tested member of the UCR repository. The plot is divided into four disjoint regions:

[7] ED and CDTW perform identically. Please, see our website [79] for details.

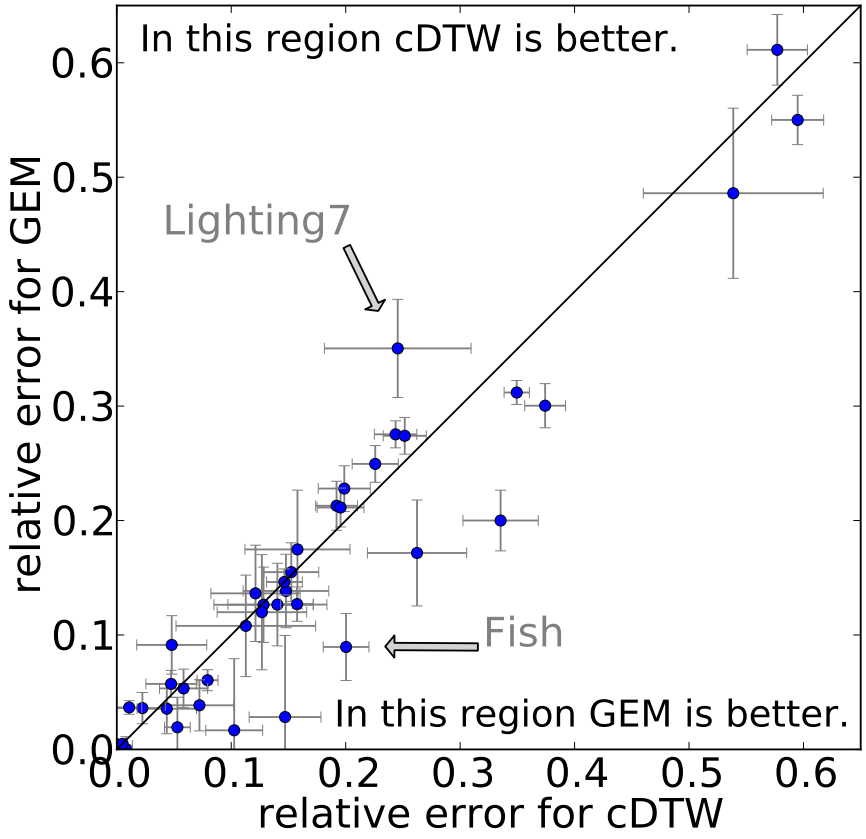


Figure 4.14: Relative errors (blue/bold dots) for CDTW and symmetrized GEM averaged over 50 runs. The error bars (grey) visualize $\pm 1\sigma$ standard deviation.

**TP** (true positive) In this regions, we predict on the training set that GEM performs better than CDTW on the test set.

**TN** (true negative) We predict correctly during training that GEM will perform worse than CDTW on the test set.

**FP** (false positive) Points in this quadrant indicate that we erroneously predict GEM to perform better than CDTW.

**FN** (false negative) Complementary to FP, the prediction that CDTW performs better than GEM is false.

The majority of datasets is located at the origin which confirms the results in Figure 4.14. However, the substantial improvements of GEM over CDTW in the upper

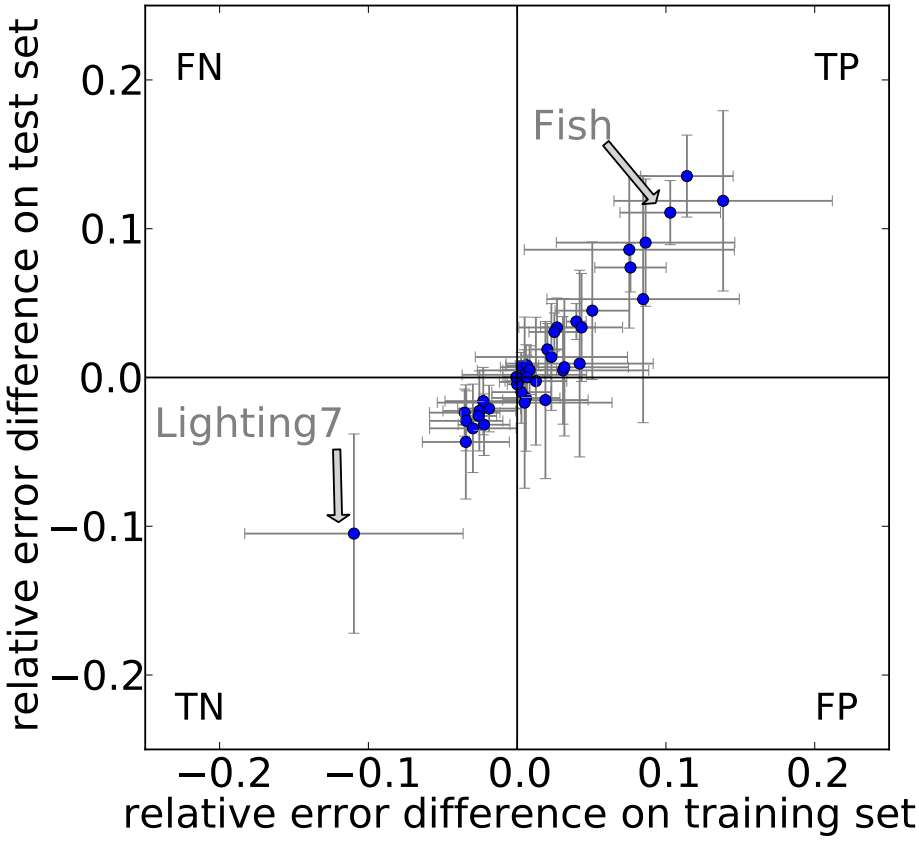


Figure 4.15: Average of the differences of relative errors (blue/bold dots) over 50 runs on the training and test set. The error bars (grey) represent $\pm 1\sigma$ standard deviation. The four regions refer to true/false positive/negative predictions.

right corner of TP are predictable. Almost all predictions are located in the regions TP and TN with a bias in favour of TP. The lower left outlier in TN represents Lighting7. Thus, a reasonable decline in classification accuracy is predictable, too. Only a few data sets lie in the region FP but not a single one with sufficient margin in relation to its standard deviation. Hence, we can confidently infer the final performance of the investigated distance measures beforehand. Finally, we can state, that GEM and CDTW can be used complementary. In addition, datasets that obey the mentioned invariances of local deformation and scaling can profit substantially during classification tasks.

### 4.2.5 Runtime and Speedups

The performance of CUDA-GEM is compared to our CPU implementation using the following platforms:

- **(CPU)** Intel Xeon X5650 Dual Hex-Core, 96 GB RAM, gcc-4.6.3 compiler with -O3 optimization, Linux Ubuntu 12.04.
- **(GPU)** NVIDIA GeForce Titan with 6 GB RAM, CUDA version 5.5, attached to an Intel Core-i7 CPU, Linux Ubuntu 12.04.

In addition to our sequential C/C++ implementation, we also have implemented a multi-core parallelization using OpenMP. The OpenMP version also parallelizes the inner loop over $j$ using multiple threads.

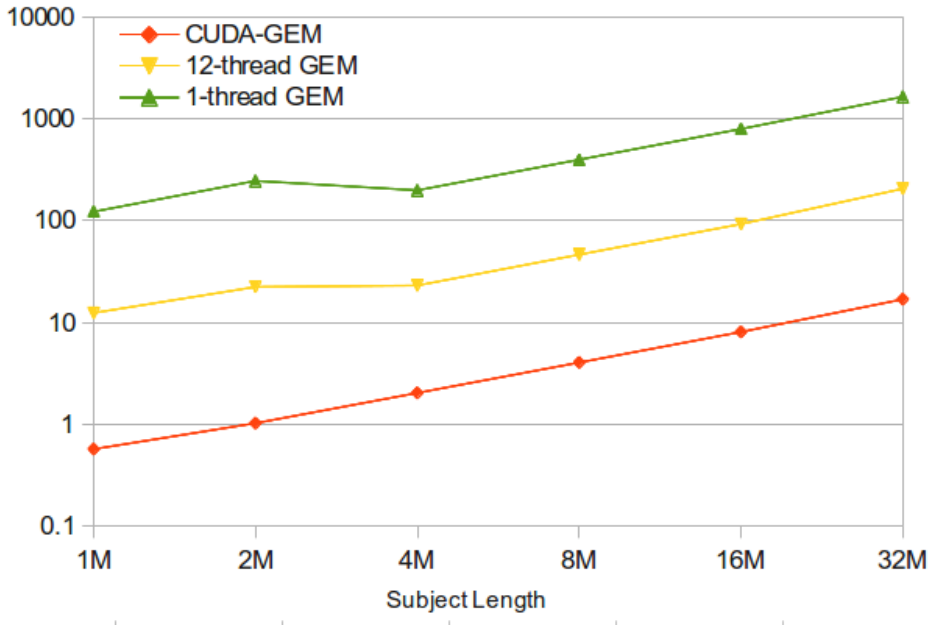


Figure 4.16: Performance in terms of execution time for the sequential (1 thread), OpenMP (12 threads) and CUDA implementation of GEM. The x-axis represents the size of $S$, the size of $Q$ is fixed at 1024. The y-axis represents the execution time (in seconds), both using logarithmic scale.

Figure 4.16 shows the performance comparison for a query $Q$ of size 1024 and a subject $S$ with size ranging from 1M to 32M (1M = $1024^2$). Our CUDA implementation achieves an average speedup of 140.3 (14.9) compared to the sequential (OpenMP) implementation. Furthermore, our OpenMP implementation achieves an average speedup of 9.4 using 12 threads compared to the sequential implementation.

We further compare the performance of GEM to the DTW implementation of the UCR Suite (UCR-DTW) using the ECG dataset (subject and queries are taken from [16]) and the metal dataset (subject and queries provided at [79]). Both the UCR and GEM CPU implementations are executed on the Xeon CPU. CUDA-GEM is exectuted on the Titan GPU. The experimental results are shown in Figure 4.17 for the ECG and in Figure 4.18 for the metal dataset.

CUDA-GEM outperforms UCR-DTW on both tested datasets for all tested settings. The execution time of GEM is linear with respect to the query length, while the execution time of UCR-DTW increases at a higher rate. This can be explained as follows: The UCR Suite uses a cascade of lower bounds. The used $LB_{Kim}$ ($\mathcal{O}(1)$ time complexity) and $LB_{Keogh}$ ($\mathcal{O}(|Q|)$ time complexity) lower bounds increasingly lose pruning power. In this case, the time complexity of UCR-DTW is considerably influenced by the $\mathcal{O}(|Q|)$ dependency of $LB_{Keogh}$ and even $\mathcal{O}(|Q|^2)$ if all lower bounds fail.

We also tested CUDA-GEM in comparison to a CUDA implementation of DTW without lower bounding in [76] for the same query and a subject of length 128K (1K= 1024) on the same GPU-based hardware. The measured runtime shows

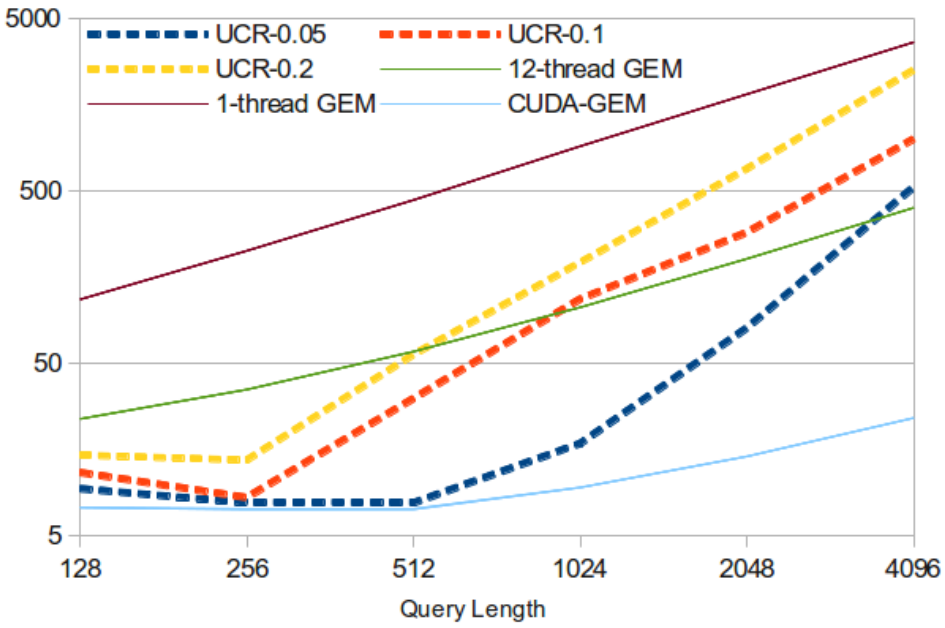


Figure 4.17: Performance in terms of execution time for the ECG dataset for the sequential (1 thread), OpenMP (12 threads), CUDA implementation of GEM in comparison to UCR-DTW with three different window settings (0.05, 0.1, 0.2). The x-axis represents the size of $Q$. $S$ is fixed at size 20.1 million. The y-axis represents the execution time (in seconds), both using logarithmic scale.

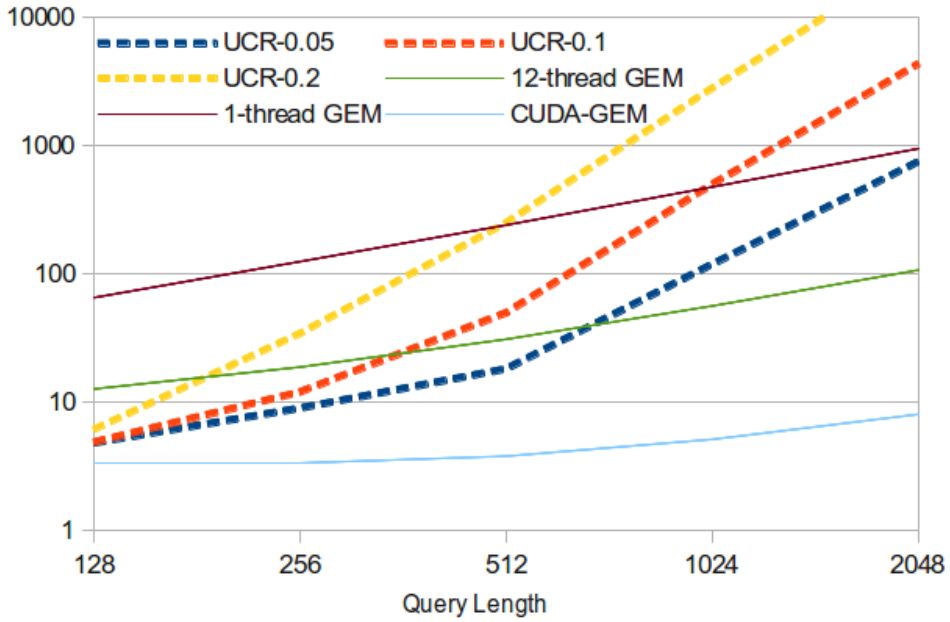


Figure 4.18: Performance in terms of execution time for the metal dataset for the sequential (1 thread), OpenMP (12 threads), CUDA implementation of GEM in comparison to UCR-DTW with three different window settings (0.05, 0.1, 0.2). The x-axis represents the size of $Q$. $S$ is fixed at size 11.34 million. The y-axis represents the execution time (in seconds), both using logarithmic scale.

that CUDA-GEM is five orders of magnitude faster. A comparison of the time complexities of both approaches ($\mathcal{O}(|S|\cdot|Q|^2)$ for subsequence-DTW and $\mathcal{O}(|S|\cdot|Q|)$ for GEM) indicates an expected speedup of about three orders of magnitude for the sequential GEM algorithm and the given problem size of $|Q|$=1K. Therefore, the achieved runtime performance of CUDA-GEM shows an additional speedup of two orders of magnitude. This result indicates that the row-independent edge scheme employed in GEM allows for a more efficient fine-grained parallelization in comparison to DTW.

### 4.2.6 Multi-Shape Alignment and Stream Segmentation

The pseudo code given in Figure 4.8 is limited to obtain only the best match of $Q$ in $S$. It can be extended to return a set of non-overlapping matches in ascending order in terms of alignments scores as follows. The starting node of a warping path has to be passed whenever an edge is relaxed. As a result one gains the first and last index of a matched subsequence $S$[start : end] together with the associated normalized cost for every entry in the last row. A standard sorting algorithm can now be applied to obtain an ascending order of the normalized costs. Finally, a balanced

binary tree can be used to determine whether two intervals are overlapping or not. In the following, our notion of non-overlapping is defined as follows:

**Definition 7** ($p$-non-overlapping intervals)**.** *Let* $[s_1, e_1]$ *and* $[s_2, e_2]$ *be two non-empty intervals and* $p \in [0,1]$*, then both intervals are called* $p$***-non-overlapping*** *if and only if*

$$\underbrace{\frac{\big((e_1 - s_1) + (e_2 - s_2)\big) - \big|(s_1 + e_1) - (s_2 + e_2)\big|}{2 \cdot \min(e_1 - s_1, e_2 - s_2)}}_{\textit{relative projection of overlap onto smaller radius}} < p$$

The choice $p = 0$ is equivalent to the usual interpretation of two non-overlapping intervals that do not intersect. For greater values the parameter $p$ refers to the maximal percentage of allowed penetration for a non-overlap. Pruning overlapping intervals is an important technique to exclude matches that differ only in minor details and gets even more crucial in anomaly detection [7]. Both, the sorting and the insertion into a tree can be accomplished in $\mathcal{O}\big(|S| \cdot \log|S|\big)$ time.

The error threshold $\varepsilon$ to decide which matches are reported strongly depends on the dataset set (even the best match is lower-bounded by the noise on top of the signals) and may be difficult to estimate. Consider the case that the query is not contained in the subject sequence then even the best match will be unsatisfactory. The interested reader may refer to [53] where additional details are given how to handle threshold-based queries in online streams.

The described procedure can be extended to the simultaneous alignment of multiple shapes. Consider a dictionary of $N$ motifs $D = \{Q^{(0)}, \ldots, Q^{(k)}, \ldots, Q^{(N-1)}\}$ and a stream $S$ that contains warped and uniform scaled variants of these basis shapes. Using GEM, we can process the score lists for each motif $Q^{(k)}$ independently. Hence, we obtain for each alignment position and each motif a distance score represented as 4-tuple $m := \big(\text{start}, \text{end}, \text{score}, \text{label}(k)\big)$ with the following ordering property

$$m < m' \quad :\Longleftrightarrow \quad \frac{m.\text{score}}{m.\text{end} - m.\text{start}} < \frac{m'.\text{score}}{m'.\text{end} - m'.\text{start}} \quad .$$

Afterwards, the sorted lists are merged and the overlaps can be removed as described above. As a result, we can provide a non-overlapping segmentation of the stream using (a subset of) the base motifs (see Figure 4.19). The corresponding pseudo-code is listed in Figure 4.20. This procedure can be extended to any subsequence alignment algorithm that returns a list of the above mentioned 4-tuples. As an example, when using Euclidean-flavoured subsequence (C)DTW, the 4-tuple for a query $Q^{(k)}$ at position $l$ is given by

$$m = \big(l,\, l + |Q^{(k)}|,\, (\text{C})\text{DTW}^2(Q, S[l : l + |Q^{(k)}|]),\, \text{label}(k)\big) \quad .$$

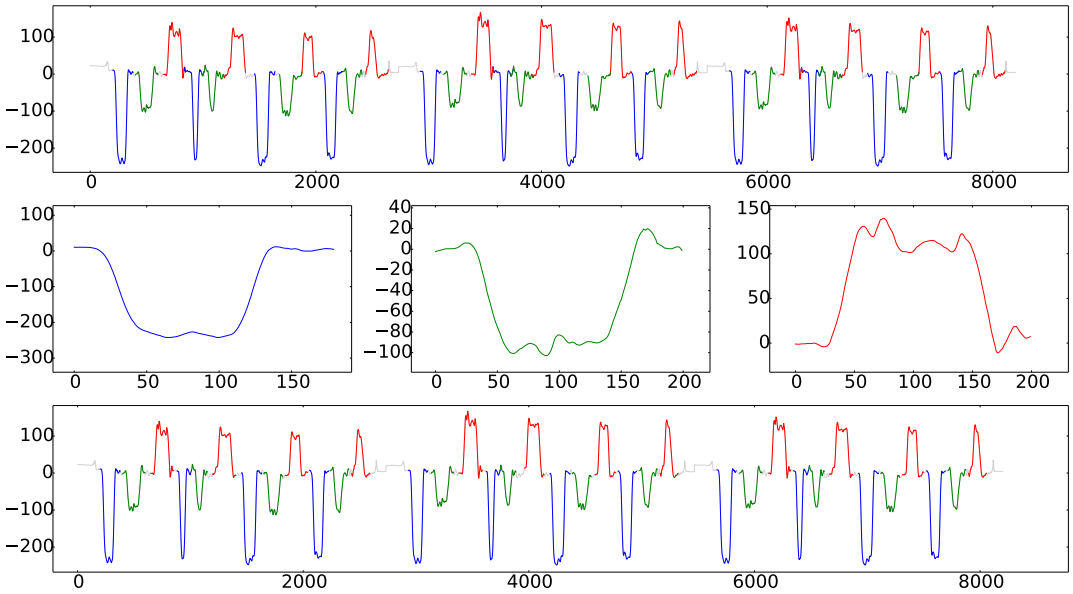


Figure 4.19: Segmentation of a stream consisting of different hand poses during an assembly task. The three basis shapes (blue, green and red time series in the second row) represent the x-coordinates of different grasping movements. The score lists are computed for both GEM (top) and unconstrained DTW (bottom) followed by a greedy removal of overlapping subsequences as described above using a relative penetration width of $p = \frac{1}{4}$. The use of z-normalization is not advisable during the computation of DTW scores due to the very similar shape of the first (blue) and second (green) shape. Both, GEM and DTW, perfectly recover the labels of the hand poses and thus are equally suited for the task. Nevertheless, GEM is capable to determine the segmentation in $\mathcal{O}\left(|S| \cdot \max(|Q^{(k)}|)\right)$ time in contrast to DTW's $\mathcal{O}\left(|S| \cdot \max(|Q^{(k)}|)^2\right)$ dependency. Note, the use of lower bounds is prohibitive since we need to access **all** alignment score at **every** position and thus have to face subsequence DTW's worst case.

The label function can be set to the identity map $\text{label}(k) = k$ for 1NN-classification with only one sample per class. In order to construct more robust classifiers, non-injective mappings from distinct queries to the same class label are conceivable. Since the dictionary $D$ has to be known beforehand, the proposed stream segmentation procedure is supervised. Usually, a domain expert provides a small set of exemplary shapes that can be used to investigate the stream. As an example, we could extract each heartbeat from a stream of heartbeats by just providing a small set of representatives. The described task gets harder if we either do not know all building blocks of the stream or have to deal with unpredictable shapes e.g. from sensor loss which do not fit in any of the classes. In this case, a binary classification task for the discrimination between outliers and shapes from $D$ has to be applied

```
1: function SUPERVISED_SEGMENTATION(M = (m_0, m_1, ...), p[= 0], ε[= ∞])
2:
3:     M ← sort(M)              ▷ sort M in ascending order using normalized scores
4:     M ← truncate(M, ε)                  ▷ remove normalized scores greater ε
5:     T ← BalancedBinaryTree(p)           ▷ tree for checking interval collisions
6:
7:     for m ∈ M do                ▷ for each match starting with the most promising
8:         if m ∉ T then                  ▷ no collision in terms of p-non-overlapping
9:             T.insert(m)                                ▷ block interval by storing in tree
10:        else
11:            M ← M \ {m}          ▷ discard match since blocked by other candidate
12:        end if
13:    end for
14:
15:    return M
16: end function
```

Figure 4.20: Pseudo-Code of a general stream segmentation algorithm.

after the segmentation. Assuming that unpredictable shapes have a reasonable inter-class distance to all entries of the dictionary $D$,[8] the $\varepsilon$-parameter can be learned on a training set to determine an appropriate threshold. If no suitable basis shapes are given at all, i.e. we cannot access a ground truth segmentation on any subset of the stream, unsupervised motif extraction algorithms [6] for the creation of the initial motif database are conceivable.

[8] In the opposite case, they could be represented by a dictionary entry.

CHAPTER 5

# LIE-GROUP-VALUED TIME SERIES

*'Ubi materia, ibi geometria.'* — Ioannes Keplerus

Up to here, the discussed algorithms for the elastic alignment of subsequences exclusively exhibit local weighting functions $w : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_0^+$ on one-dimensional and real domain. Given two values $(p, q) \in \mathbb{R} \times \mathbb{R}$ the weighting function is usually chosen to represent the length of the shortest path between $p$ and $q$ – the so-called **geodetic distance**. On flat $\mathbb{R}^m$ the shortest path between two points can be parametrized by a straight line $\gamma(\tau) = (1-\tau) \cdot p + \tau \cdot q$ for $\tau \in [0,1]$ and thus its squared arc length $w^{\mathbb{R}^m}(p,q)$ has the expected form:

$$w^{\mathbb{R}^m}(p,q) := \left( \int_0^1 \left\| \frac{\mathrm{d}}{\mathrm{d}\tau} \gamma(\tau) \right\| \mathrm{d}\tau \right)^2 = \left( \|q-p\| \cdot \int_0^1 1 \, \mathrm{d}\tau \right)^2 = \|p-q\|^2 \quad .$$

Obviously, the computation of the length involves $\dot{\gamma}(\tau)$ i.e. the velocity of the curve at each point $\gamma(\tau)$ on the geodetic parametrization between $p$ and $q$. Furthermore, we need to know how to compute the norm of a velocity vector $\|\dot{\gamma}(\tau)\|$ at each point $\gamma(\tau)$. Since $\dot{\gamma}(\tau) = q - p$, the expression can be reduced to the ordinary Euclidean distance between the points $p$ and $q$.

Consider now a curved manifold $M \subset \mathbb{R}^m$ embedded in higher dimensional flat space e.g. the unit sphere $\mathscr{S}^1 = \{p \in \mathbb{R}^2 \,|\, \|p\|^2 = 1\}$ as a subset of $\mathbb{R}^2$. A straight line between two points $p, q \in M$ does not necessarily lie in $M$. Moreover, the operations + and − have no meaning on an arbitrary manifold $M$ and thus we do not know how to evaluate the expression $\|p-q\|$ unless we report the arc length

of straight lines in the embedding space $\mathbb{R}^m$. As an example, two points $p, q \in \mathcal{S}^1$ could be written as $p = (\cos\varphi_p, \sin\varphi_p)^T$ and $q = (\cos\varphi_q, \sin\varphi_q)^T$ and thus a parametrization from $p$ to $q$ through $\mathcal{S}^1$ by an angular parameter $\tau \in [0, \varphi_q - \varphi_p]$ is given by:

$$\gamma(\tau) = \begin{pmatrix} \cos\tau & -\sin\tau \\ \sin\tau & \cos\tau \end{pmatrix} \begin{pmatrix} \cos\varphi_p \\ \sin\varphi_p \end{pmatrix} = \begin{pmatrix} \cos\tau\cos\varphi_p - \sin\tau\sin\varphi_p \\ \sin\tau\cos\varphi_p + \cos\tau\sin\varphi_p \end{pmatrix} = \begin{pmatrix} \cos(\tau+\varphi_p) \\ \sin(\tau+\varphi_p) \end{pmatrix} \quad .$$

The geodetic distance can now be computed directly using $\gamma(\tau)$ and the observation that $\langle \dot\gamma(\tau), \dot\gamma(\tau) \rangle = (-)^2 \sin^2(\tau+\varphi_p) + \cos^2(\tau+\varphi_p) = 1$ for all $\tau, \varphi_p \in \mathbb{R}$:

$$w^{\mathcal{S}^1}(p,q) := \left( \int_0^{\varphi_q - \varphi_p} \left\| \frac{\mathrm{d}}{\mathrm{d}\tau}\gamma(\tau) \right\| \mathrm{d}\tau \right)^2 = \left( \int_0^{\varphi_q - \varphi_p} 1 \, \mathrm{d}\tau \right)^2 = |\varphi_q - \varphi_p|^2 \quad .$$

Note, if $|\varphi_q - \varphi_p| > \pi$ we obviously took the wrong sign for the rotation and thus computed the longest path instead of the shortest one. This can be avoided using $w^{\mathcal{S}^1}(p,q) = \left(\arccos\langle q, p\rangle\right)^2$ the inverse cosine of the projection of $q$ on $p$ since:

$$\langle q, p \rangle = \cos\varphi_q \cos\varphi_p + \sin\varphi_q \sin\varphi_p = \cos(\varphi_q - \varphi_p) \quad .$$

Concluding, it is easy to assign meaningful distances in flat $\mathbb{R}^m$ and the unit sphere $\mathcal{S}^1$ using the length of the shortest paths between two points.

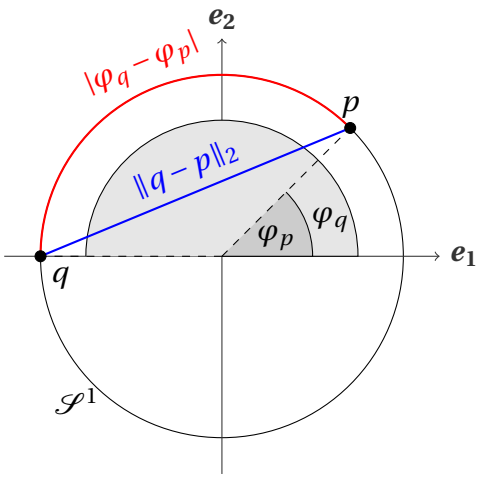


Figure 5.1: An example for the geodetic distance of two points $p, q \in \mathcal{S}^1 \subset \mathbb{R}^2$. If treated as plain vectors in $\mathbb{R}^2$ the geodetic distance is the length of the displacement vector $q - p$ (blue line). In contrast, if rigidly treated as orientations $p, q \in SO(2,\mathbb{R}) \cong \mathcal{S}^1$ with associated angles $\varphi_p$ and $\varphi_q$ the geodetic distance is given by the length of the continuous path from $p$ to $q$ through $\mathcal{S}^1$ (red arc).

However, when dealing with data from inertial measurement units (IMUs) we have to treat orientations in three-dimensional angular space and translations in three-dimensional spatial domain at the same time. Mathematically speaking, the group of rotations $SO(m,\mathbb{R})$ acting on $\mathbb{R}^m$, the so-called **special orthogonal group**,[1] is a curved and differentiable manifold of dimension $\binom{m}{2} = \frac{m\cdot(m-1)}{2}$. The computation of the squared geodetic distance $w(p,q)$ for $p,q \in SO(m,\mathbb{R})$ is not obvious without knowing how to parametrize geodesics $\gamma(\tau)$. Furthermore, we can extend the group of rotations with spatial translations i.e. the abelian group $(\mathbb{R}^m,+)$. The resulting group of all rigid (angle, length and chirality-preserving) transformations $SE(m,\mathbb{R})$ acting on $\mathbb{R}^m$ , the so-called **special Euclidean group**, is a curved and differentiable manifold of dimension $\frac{m\cdot(m-1)}{2} + m = \frac{m\cdot(m+1)}{2}$. As we will see, we can establish different notions of distance on the above mentioned manifolds. Aiming for the use in elastic measures, we have to ensure an efficient representation of the involved transformations while at the same time using as few operations as possible during the computation of the geodetic distance.

The rest of this chapter is organized as follows. Section 5.1 gives a brief introduction of the very basics of Lie groups and differential geometry. Geodetic distances on pure rotations as well as their efficient use in time series data mining applications using unit quaternions are discussed in Section 5.2. This includes a linear time algorithm for windowed frame-normalization in a stream of rotations, an analogous preprocessing step to z-normalization, the construction of lower bounds for 1NN search under DTW and multi-shape segmentation in streams of hand poses. Section 5.3 provides an extension from streams of pure rotations $SO(m,\mathbb{R})$ to streams of full rigid transformation $SE(m,\mathbb{R})$ using unit dual quaternions.

## 5.1 Mathematical Aspects of Lie Groups

This section is mainly used to fix the notation and to provide basic definitions for Lie groups and Lie algebras in order to ensure self-consistency of the thesis. As a result, it cannot cover the vast field of differential geometry and Lie groups – neither in terms of completeness nor in terms of mathematical rigidity. An interested reader may refer to Jeffrey M. Lee's excellent text book on differential geometry [80] or the introductory chapters of Florian Jung's dissertation on the construction of quantization maps using group theoretical approaches [81].

### 5.1.1 Differentiable Manifolds

Let us start with the definition of smooth manifolds.

[1] During the rest of this thesis, we neglect reflections since we have to guarantee the existence of continuous paths between all pairs of points in the manifold of transformations.

**Definition 8** (smooth manifold)**.** *An $m$-dimensional **smooth manifold** $M$ is a second countable topological Hausdorff space equipped with an $m$-dimensional $\mathscr{C}^\infty$-differential structure.*

Despite the abstract nature of the definition, smooth manifolds can be easily explained as topological spaces which locally behave like $\mathbb{R}^m$ with a notion of differentiation. Let us now decompose the definition: A topological space is a mathematical set equipped with a concept of **open sets** e.g. open balls in $m$-dimensional Cartesian space $B_\varepsilon(p) = \{q \in \mathbb{R}^m \,|\, \|p-q\| < \varepsilon\}$. Second countable means that any open subset can be written as (possibly infinite) union of a **countable** collection of open sets e.g. in $\mathbb{R}^m$ the collection of all open balls with rational radii and rational centres $\{B_\varepsilon(p) \,|\, \varepsilon \in \mathbb{Q}, p \in \mathbb{Q}^m\}$. The Hausdorff property guarantees that each pair of points $p, q \in M$ is **separable** by open sets i.e we can find two disjoint open sets $p \in U, q \in V$ with $U \cap V = \emptyset$. Up to here, the above mentioned properties ensure that a manifold can be treated locally like $\mathbb{R}^m$ when defining continuous maps (e.g. connected curves $\gamma(\tau)$ through $M$), limits and convergence. Hence, for each point $p \in M$, we can locally define a homeomorphic (invertible and continuous) map $x$ from an open subset $p \in U \subset M$ to an open subset of $x(U) \subset \mathbb{R}^m$. A tuple $(U, x)$ is called a **chart** of $M$. Figure 5.2 depicts two local homeomorphic mappings of the same point $p \in M$ to open sets of $\mathbb{R}^m$.

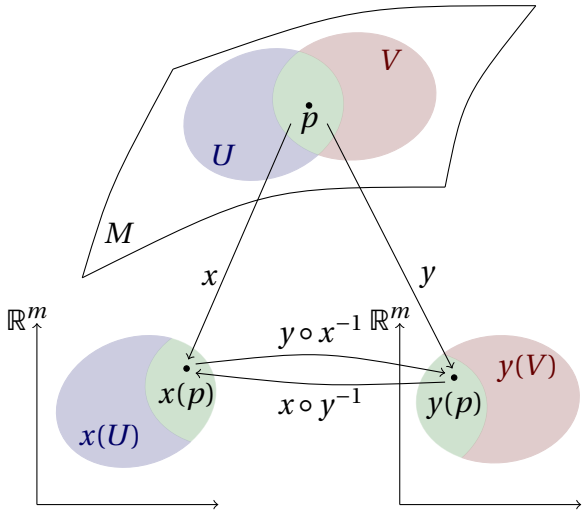


Figure 5.2: An example for homeomorphic mappings of the same point $p \in M$ with corresponding transition maps between two charts $(U, x)$ and $(V, y)$ of $M$.

The described construction ensures that every (topological) manifold is locally isomorphic to an $m$-dimensional Cartesian space in terms of open sets. However, we want to establish a notion of differentiability which can be done as follows. Assume we have two overlapping open sets $U, V \subset M$ then a point $p \in U \cap V$ can be mapped either using $x$ or $y$ to $\mathbb{R}^m$ i.e. the point $p$ can be represented in two different coordinate systems. Using $x$ and $y$ we can compute the transition maps $y \circ x^{-1}$ and $x \circ y^{-1}$ which represent the change in coordinates on the level of $\mathbb{R}^m$. Since we understand the concept of differentiability on $\mathbb{R}^m$, we define a topological manifold to be **smooth** if all transition maps between open sets are $\mathscr{C}^\infty$-functions.[2] Let us now establish the concept of tangent vectors.

**Definition 9** (smooth curve)**.** *Let $I \subseteq \mathbb{R}$ be an open subset of $\mathbb{R}$ and $M$ be an $m$-dimensional smooth manifold then $\gamma : I \to M$, $\tau \mapsto \gamma(\tau)$ is called a **smooth curve** if its coordinate representation in $\mathbb{R}^m$ is also a smooth function in each chart $(U, x)$.*

As we will see, smooth curves can be used to define the tangent plane $\mathrm{T}_p M$ for each point $p \in M$ independently. These so-called **tangent spaces** $\mathrm{T}_p M$ are isomorphic to $\mathbb{R}^m$.

**Definition 10** (tangent vector)**.** *Let $M$ be a smooth manifold and $\Gamma_p$ be the set of smooth curves $\gamma$ through $p$ with $\gamma(0) = p$. Two curves $\gamma$ and $\gamma'$ are **tangentially equivalent** $\gamma \sim \gamma'$, if they are tangent to each other in charts, i.e.*

$$\left.\frac{\mathrm{d}}{\mathrm{d}\tau}\right|_{\tau=0} x\big(\gamma(\tau)\big) = \left.\frac{\mathrm{d}}{\mathrm{d}\tau}\right|_{\tau=0} x\big(\gamma'(\tau)\big)$$

*for any chart $(U, x)$ around $p$. The equivalence classes $\dot{\gamma}(0) = [\gamma]$ are called **tangent vectors** at the point $p$ and $\mathrm{T}_p M = \{[\gamma] \mid \gamma \in \Gamma_p\}$ the **tangent space** at $p$. The disjoint union of all tangent spaces over all $p \in M$*

$$\mathrm{T}M := \coprod_{p \in M} \mathrm{T}_p M = \bigcup_{p \in M} \{p\} \times \mathrm{T}_p M$$

*is called **tangent bundle** of $M$. Without proof, $\mathrm{T}M$ itself is a smooth manifold of dimension $2m$.*

Let us illustrate the definition with a simple example. Let $M = \mathbb{R}$ the one-dimensional Cartesian space and $(U, x) = (\mathbb{R}, \mathrm{id})$ a global chart of $\mathbb{R}$ i.e. we can trivially treat $\mathbb{R}$ as smooth manifold with the identity as coordinate function. A sufficient subset of curves that translate $p$ by a constant velocity $v$ is given by $S := \{\gamma_v(\tau) = p + \tau \cdot v \mid v \in \mathbb{R}\} \subset \Gamma_p$ and obeys $\gamma_v(0) = p$ and $\dot{\gamma}_v(0) = v$ for each $\gamma_v \in S$ and thus the tangent space at $p$ is the linear hull $\mathrm{T}_p M = \mathrm{span}(v) = \mathbb{R}$.

[2] In the following, we only consider smooth manifolds and thus neglect further variants like $\mathscr{C}^k$ or analytic manifolds.

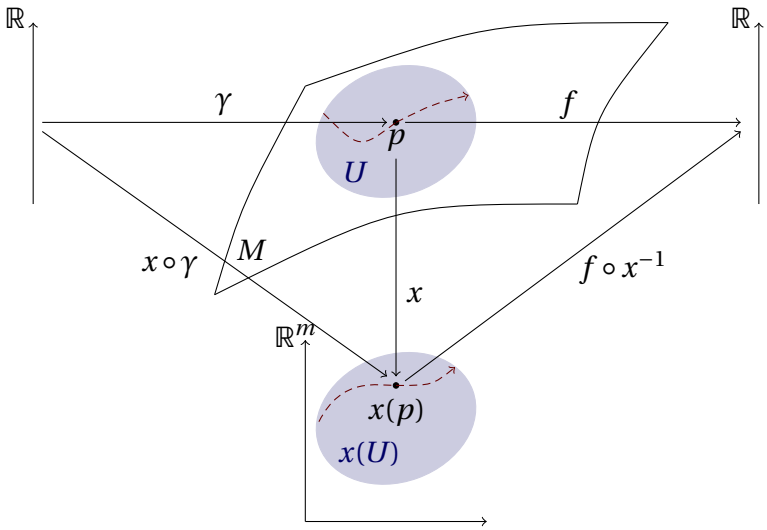


Figure 5.3: Illustration of tangent vectors in terms of smooth curves acting on smooth functions. The parametrization of a function along a curve $(f\circ\gamma)(\tau)$ can be decomposed into its coordinate representations $(f\circ\gamma)(\tau) = (f\circ x^{-1})\circ(x\circ\gamma)(\tau)$. Both $f$ and $\gamma$ are considered smooth if we can say the same about $f\circ x^{-1}$ and $x\circ\gamma$ in charts. The action of $\gamma$ on $f$ at $p$ is represented by the time derivative $[\gamma](f) := \frac{\mathrm{d}}{\mathrm{d}\tau}\big|_0 (f\circ\gamma)(\tau) = \sum_{k=0}^{m-1} \frac{\mathrm{d}}{\mathrm{d}\tau}\big|_0 (x\circ\gamma)^k \frac{\partial}{\partial x^k}\big|_p (f\circ x^{-1}) =: \big(\sum_{k=0}^{m-1} v^k \partial_k\big) f$. Hence, $\{v^k\}$ represents the set of coefficients of the tangent vector and $\{\partial_k\}$ the corresponding set of basis vectors. Since we are only interested in the time derivative in a neighbourhood of $\tau = 0$, all curves that produce the same coefficients $v^k$ lie in the same equivalence class $[\gamma]$.

As mentioned in the introduction, we are interested in arc lengths of curves through manifolds. The classic formula for the computation of arc lengths depends on the term $\langle\dot\gamma(\tau),\dot\gamma(\tau)\rangle$. As a result, an extension to smooth manifold needs an inner product on each of the tangent spaces.

**Definition 11** (Riemannian manifold)**.** *Let $M$ be a smooth manifold and $g : p \mapsto g_p$ be a smoothly varying assignment of positive definite inner products*

$$g_p : \mathrm{T}_pM \times \mathrm{T}_pM \to \mathbb{R}_0^+ , (X_p, Y_p) \mapsto \langle X_p, Y_p\rangle_p = g_p\big(X_p, Y_p\big)$$

*then the ordered tuple $(M, g)$ is called **Riemannian manifold**.*

If our manifold $M$ is Riemannian, we can define the arc length $L(\gamma)$ and the energy $E(\gamma)$ of a smooth curve $\gamma : I \to M$ by extending the classic formulas for arc length and kinetic energy (unit mass):

$$L(\gamma) = \int_I \sqrt{g_{\gamma(\tau)}\big(\dot\gamma(\tau),\dot\gamma(\tau)\big)}\,\mathrm{d}\tau \quad \text{and} \quad E(\gamma) = \frac{1}{2}\int_I g_{\gamma(\tau)}\big(\dot\gamma(\tau),\dot\gamma(\tau)\big)\,\mathrm{d}\tau \quad .$$

### 5.1.2 Fundamentals of Group Theory

Let us start with the definition of a group.

**Definition 12** (group)**.** *Let $G$ be a set and $\cdot : G \times G \to G$, $(g,g) \mapsto g \cdot g'$ a binary operation then the ordered tuple $(G,\cdot)$ is called a **group** if and only if*

1. $\forall g,g',g'' \in G : (g \cdot g') \cdot g'' = g \cdot (g' \cdot g'')$ ***(associativity)***
2. $\exists e \in G \, \forall g \in G : e \cdot g = g$ ***(neutral element)***
3. $\forall g \in G \exists g^{-1} \in G : g \cdot g^{-1} = e$ ***(inverse element)***

Obviously, a group $G$ can act on itself using left and right actions. A **left translation** $L_g : G \to G$, $g' \mapsto L_g(g') = g \circ g'$ acts on an element from the left as a group homomorphism since $(L_g \circ L_{g'})(g'') = g \cdot g' \cdot g'' = L_{g \cdot g'}(g'')$. Analogous, the **right translation** is an anti-homomorphism since $(R_g \circ R_{g'})(g'') = g'' \cdot g' \cdot g = R_{g' \cdot g}(g'')$ which inverses the order of elements when composed. Moreover, we can define an **inner action** $I_g : G \to G$, $g' \mapsto g \cdot g' \cdot g^{-1}$ which is a group homomorphism (left action) since $(I_g \circ I_{g'})(g'') = (g \cdot g') \cdot g'' \cdot (g \cdot g')^{-1} = (I_{g \cdot g'})(g'')$. The equivalent anti-homomorphism (right action) can be constructed by mapping all elements with the inverse map $\cdot^{-1} : G \to G$, $g \mapsto g^{-1}$. The above mentioned maps define structure-preserving actions of the group on itself in terms of group homomorphisms. The same can be done for actions of groups on arbitrary sets.

**Definition 13** (action of groups on sets)**.** *Let $G$ be a group and $M$ a non-empty set, then $\Phi : G \times M \to M$, $(g,p) \mapsto \Phi(g,p)$ is called a **group action of** $G$ **on** $M$ if and only if*

1. $\forall p \in M : \Phi(e,p) = p$ ***(identical action)***
2. $\forall p \in M, \forall g,g' \in G : \Phi(g,\Phi(g',p)) = \Phi(g \cdot g',p)$ ***(homomorphism)***

As an example, take the additive group of real numbers $(\mathbb{R},+)$ acting on $M = \mathbb{R}^2$ by rotating a point $p \in M$ with a rotation matrix $\Phi(\tau,p) = R(\tau) \cdot p$. Obviously, the identity acts as expected $\Phi(0,p) = p$ and moreover the homomorphism property $\Phi(\tau,\Phi(\tau',p)) = R(\tau) \cdot R(\tau') \cdot p = \Phi(\tau+\tau',p)$ holds due to the trigonometric identities of sine and cosine.

The group action does not necessarily need to be injective in the left slot e.g. the pathological assignment $\Phi(g,p) = p$ for all $p \in M$ and $g \in G$ also obeys the above stated definition. However, the partial map $\Phi(g,\cdot)$ is always invertible – simply choose $\Phi(g,\cdot)^{-1} = \Phi(g^{-1},\cdot)$ for the inverse map since $\Phi(g^{-1},\Phi(g,\cdot)) = \Phi(g^{-1} \cdot g,\cdot) = \Phi(e,\cdot) = \mathrm{id}_M$. With the help of group actions we can construct more complex groups from two basis groups.

**Definition 14** (semi-direct product of groups)**.** *Let $G$ and $H$ be two groups with associated group operations $\circ$ as well as $\bullet$. Further let $\Phi : G \times H \to H$ be a group action of $G$ on $H$ then the expression $G \ltimes_\Phi H$ is called* ***semi-direct product*** *with the corresponding group law*

$$\cdot : (G \ltimes_\Phi H) \times (G \ltimes_\Phi H) \to G \ltimes_\Phi H, \big((g,h),(g',h')\big) \mapsto \big(g \circ g', h \bullet \Phi(g,h')\big) \quad .$$

Without proof, the semi-direct product of $G$ and $H$ obeys the group laws. The identity is given by $(e_G, e_H)$, the inverse by $\big(g^{-1}, \Phi(g^{-1}, h^{-1})\big)$ and the associativity can be shown by a lengthy computation. Let us now have look at some comprehensive examples:

- **(direct product)** If the action $\Phi = \mathrm{id}_H$ of $G$ on $H$ is chosen to be trivial i.e. $\Phi(g,h) = h \ \forall g \in G$ and $h \in H$, then the group law simplifies to $(g,h) \cdot (g',h') = (g \circ g, h \bullet h')$. As a result, both groups act on themselves without interacting. Later in this thesis, we will use this approach in the **double-geodetic** representation of hand poses.

- **(one-dimensional affine group)** Let $G = (\mathbb{R}^+, \cdot)$ be the multiplicative group of scaling factors and $H = (\mathbb{R}, +)$ the additive group of translations in one-dimensional Cartesian space. Furthermore, the action of a scaling operation on a translation is simply defined by the ordinary multiplication of real numbers i.e. $\Phi : (g,h) \mapsto \Phi(g,h) = g \cdot h$. The action of two combined scaling and translation operations on a test point $p \in \mathbb{R}$ reveals the well-known group law $g \cdot (g' \cdot p + h') + h = (g \cdot g') \cdot p + (h + g \cdot h')$.

- **(rigid transformations in $\mathbb{R}^m$)** Analogous, choose $G = GL(m,\mathbb{R})$ the non-commutative matrix group of $m \times m$-matrices with non-vanishing determinant and $H = (\mathbb{R}^m, +)$ the abelian group of translations in $m$-dimensional Cartesian space. The action of $G$ on $H$ is simply defined by the ordinary matrix vector product. Important subgroups of $GL(m,\mathbb{R}) \ltimes_\Phi \mathbb{R}^m$ can be obtained by restricting $GL(m,\mathbb{R})$ to a subgroup. As an example, when choosing the rotation group $G = SO(m,\mathbb{R})$, the semi-direct product is the well known **Euclidean group** $SE(m,\mathbb{R})$ of length, angle and chirality-preserving transformations on $\mathbb{R}^m$ – often abbreviated as group of **rigid transformations**.

When dealing with hand poses, we measure for each time tick an orientation (rotation) and a position (translation) of the tracked hand. Assuming rigid bodies, a hand pose can be described by an element of the Euclidean group, namely the transformation that had to be applied to the hand in order to move it from the identity $(\mathbb{1}, 0)$ to its actual position in the angular and spatial domain. Consequently, **hand poses have to be identified with Euclidean transformations**. As a result, a notion of distance between different group elements has to be established to assign meaningful measures of similarity.

### 5.1.3 Lie Groups and Lie Algebras

In the previous sections, we discussed a notion of distance in terms of arc lengths of smooth curves on Riemannian manifolds. Fortunately, many parametrizable transformation groups can be treated as smooth manifolds, so-called Lie groups.

**Definition 15** (Lie group)**.** *A **Lie group** $G$ is simultaneously a group and a smooth manifold such that the group multiplication $\cdot : G \to G$ and the inversion operation $\cdot^{-1} : G \to G$ are smoothly varying functions in $g \in G$.*

This said, points in the manifold are identified with group elements. The differentiability can often be seen directly in charts, when utilizing standard parametrizations of the corresponding groups. As an example, an element from the matrix group $GL(m,\mathbb{R})$ of invertible matrices can be interpreted as a vector in $\mathbb{R}^{m\times m}$. Each entry in a matrix $g \in GL(m,\mathbb{R})$ represents a coordinate function of the manifold and thus the product of two matrices contains scalar products of the coordinate functions. Hence, the resulting matrix $g \cdot g'$ will be differentiable in its coordinates. A similar statement can be made about the inverse matrix whose entries are composed of polynomials in the coordinate functions (determinants of $g$ and its minors).

Remarkably, the tangent space $\mathrm{T}_g G$ at any point $g \in G$ can be computed if we know the tangent vectors at the neutral element $e$ and afterwards push them forward with the tangential map (Jacobian) of the left translation $L_g$.[3] As a result, the tangent space at the neutral element $\mathrm{T}_e G$ is a vector space whose basis vectors locally encode the structure of the group, the so-called **Lie algebra** $\mathfrak{L}G \cong \mathrm{T}_e G$. The inverse mapping from the Lie algebra to the Lie group can be computed as follows. Assume we have a curve $\gamma$ through $G$ with $\gamma(0) = e$ and $\dot\gamma(0) = v \in \mathrm{T}_e G$ as mentioned in Section 5.1.1. The integral curve of the ordinary differential equation

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\alpha(\tau) = X(\alpha(\tau)) \quad \text{where} \quad X(g) = \left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_0 L_g\big(\gamma(t)\big) = \left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_0 g\cdot\gamma(t) \quad ,$$

with the initial condition $\alpha(0) = e$ is called **exponential map**. Let us illustrate this abstract definition with some comprehensive examples.

- **(additive groups)** Let $G = (\mathbb{R},+)$ be the abelian group of translations then $L_g(g') = g + g'$ and thus $\frac{\mathrm{d}}{\mathrm{d}t}\big|_0 L_g(\gamma(t)) = \frac{\mathrm{d}}{\mathrm{d}t}\big|_0 \big(g + \gamma(t)\big) = v$. The differential equation $\frac{\mathrm{d}}{\mathrm{d}\tau}\alpha(\tau) = v$ with initial condition $\alpha(0) = 0$ can easily be solved by integration $\alpha(\tau) = \tau \cdot v$. As expected, translations are generated by steadily adding up velocity vectors. The same approach is conceivable for any additive group e.g. the group of translations in $(\mathbb{R}^m,+)$.

[3] Please note, the definition of left-invariant vector fields and their corresponding Lie bracket is left out on purpose. An interested reader may refer to [81].

- **(multiplicative groups)** Let $G = (\mathbb{R}^+, \cdot)$ be the group of scaling operations then $L_g(g') = g \cdot g'$ and thus $\frac{\mathrm{d}}{\mathrm{d}t}\big|_0 L_g(\gamma(t)) = \frac{\mathrm{d}}{\mathrm{d}t}\big|_0 \big(g \cdot \gamma(t)\big) = g \cdot v$. The differential equation $\frac{\mathrm{d}}{\mathrm{d}\tau}\alpha(\tau) = \alpha(\tau) \cdot v$ with initial condition $\alpha(0) = 1$ has the well-known solution $\alpha(\tau) = e^{\tau \cdot v}$. As expected, the exponential map of multiplicative groups is the same as the ordinary exponential function expressed as power series $\exp(\tau \cdot v) = \sum_{k=0}^{\infty} \frac{1}{k!}(\tau \cdot v)^k$. Consequently, the reverse operation of finding the tangent vector for a given group element is called the **logarithm map**.

- **(unit complex numbers)** Furthermore, choose the set of all complex numbers with unit modulus $U(1,\mathbb{C}) \cong SO(2,\mathbb{R}) \cong \mathscr{S}^1$. Its parametrization $\tau \mapsto e^{\mathrm{i}\cdot\tau}$ is obvious – one can even read off the tangential vector at the neutral element $\frac{\mathrm{d}}{\mathrm{d}\tau}\big|_0 e^{\mathrm{i}\cdot\tau} = \mathrm{i}$. The modulus of the logarithm $|\mathrm{i}\cdot\tau| = |\tau|$ is equivalent to the rotated angle. Note, since exp is not injective (choose $\tau$ and $\tau + 2\pi$), the logarithm can only be defined as right inverse of the exponential map.

At this point let us pause for a moment and reconsider the following statement: Smooth curves through Lie groups can be parametrized by simply following straight lines in the Lie algebra. This is the key property for further considerations when assigning distances between distinct group elements. But what can we state about these **one-parameter subgroups** $S_v = \{\alpha(\tau) = \exp(\tau \cdot v) \in G \,|\, \tau \in \mathbb{R}\} \subseteq G$ except that they parametrize the path between the identity $e \in G$ and an arbitrary group element $g = \exp(v) \in G$? Assume we can find a frame-independent (left and right-invariant = Ad-invariant) metric on the tangential spaces of the Lie group i.e. for all $v, w \in \mathscr{L}G$ and $g \in G$ it holds

$$\langle v, w\rangle = \langle \mathrm{Ad}_g(v), \mathrm{Ad}_g(w)\rangle = \langle (T_e I_g)(v), (\mathrm{T}_e I_g)(w)\rangle = \langle g v g^{-1}, g w g^{-1}\rangle \quad ,$$

where the adjoint map $\mathrm{Ad}_g = \mathrm{T}_e I_g$ is the tangential map (Jacobian) of the inner action $I_g : \; g' \mapsto g \cdot g' \cdot g^{-1}$ of $G$ on itself. An inner product $g_p$ on each tangent space $\mathrm{T}_g G$ can be expressed in terms of an inner product on the Lie algebra $\mathscr{L}G = \mathrm{T}_e G$ and thus $G$ exhibits the structure of a Riemannian manifold. The following proposition emphasizes the connection between geodesics and logarithms on Lie groups [82].

**Proposition 7** (norm of the logarithm)**.** *With respect to an Ad-invariant metric, geodesics of $G$ are the one-parameter subgroups, that is the curves of the form $\alpha(\tau) = \exp(\tau \cdot v)$ with $v \in \mathscr{L}G$ constant. Furthermore, the distance between the element $g$ and the identity $e$ is given by the norm of the logarithmic function*

$$\|g\|_G = \sqrt{\langle \log(g), \log(g)\rangle} \quad .$$

Let us illustrate this proposition with two comprehensive examples:

- **(translations in $\mathbb{R}^m$)** Let $(\mathbb{R}^m,+)$ be the abelian group of translations in $\mathbb{R}^m$. As we have seen, the exponential map is given by the identity map i.e. $g = \exp(v) = v$ since $\exp(\tau \cdot v) = \alpha(\tau) = \tau \cdot v$ . Consequently, $\log(v) = v$ and thus the geodetic distance on flat $\mathbb{R}^m$ is given by the ordinary Euclidean distance induced by the corresponding norm $\|g\|_{(\mathbb{R}^m,+)} = \|v\| = \sqrt{\langle v, v\rangle}$. The defined inner product is Ad-invariant due to the commutativity of the group operation i.e. $I_w(v) = w \cdot v \cdot w^{-1} = w + v - w = v$ for all $w \in G$.
- **(unit complex numbers)** Let $U(1,\mathbb{C}) = \{e^{\mathrm{i}\cdot\tau} \in \mathbb{C} \mid \tau \in \mathbb{R}\} \cong SO(2,\mathbb{R})$ the group of complex numbers with modulus one which is isomorphic to the group of rotations in the two-dimensional plane. As shown above, the exponential map for multiplicative groups $\exp(\tau \cdot v) = e^{\tau \cdot v}$ is given by the ordinary exponential function. Consequently, $\log(g) = \mathrm{i} \cdot \tau$ for $g = e^{\mathrm{i}\cdot\tau}$ and thus the geodetic distance is given by the modulus of the rotation angle since $\|g\|_{U(1,\mathbb{C})} = \sqrt{\langle \mathrm{i}\cdot\tau, \mathrm{i}\cdot\tau\rangle_{\mathbb{C}}} = |\tau|$. The used metric is Ad-invariant by the same argument as above since $U(1,\mathbb{C})$ is an abelian group.

The norm represents the distance of a group element to the identity (origin). Thus, when assigning distances between two points $g, g' \in G$, we simply report the length of the curve from the identity to the **difference transformation** $\Delta := g' \cdot g^{-1}$. This can be verified by a right translation of the original parametrization with $g$:[4]

$$\gamma : [0,1] \to G, \tau \mapsto \Delta^\tau \cdot g := \exp\big(\tau \cdot \log(\Delta)\big) \cdot g \quad \text{with} \quad \gamma(0) = g \,,\, \gamma(1) = g' \quad .$$

With the help of this formalism, we will discuss notions of distance for the Lie groups $SO(m,\mathbb{R})$ and $SE(m,\mathbb{R})$ and their efficient representation in terms of computational effort and memory consumption.

## 5.2 Time Series of Orientations in $SO(3,\mathbb{R})$

This section deals with sequences of orientations in angular space, namely time series of rotation matrices $C = (R_0, R_1, \ldots, R_{n-1}) \in SO(m,\mathbb{R})^n$. For the sake of simplicity, we choose $m = 3$, however, the construction of the corresponding Lie algebra $so(m,\mathbb{R}) := \mathscr{L}SO(m,\mathbb{R})$ as well as the geodetic distance can be extended to any dimension $m \geq 2$. Moreover, this approach is compatible with every compact Lie group as long as we can determine logarithms and find an Ad-invariant metric on them.[5] Let us start with brief discussion of the identical (usual) representation of the rotation group acting on $\mathbb{R}^3$.

[4] Please note, for the choice $G = (\mathbb{R}^m,+)$ the curve $\gamma(\tau) = \Delta^\tau \cdot g = \tau \cdot (g' - g) + g$ parametrizes a straight line between the points $g$ and $g'$ in Cartesian space (recall $\log(\Delta) = \Delta = \exp(\Delta)$ for $\mathbb{R}^m$).

[5] We can always guarantee this for **compact** Lie groups by multiplying their negative definite Killing form [82] with a negative number and thus gain a positive definite bilinear form.

### 5.2.1 The Rotation Group and its Lie Algebra

The special orthogonal group $SO(3,\mathbb{R})$ is defined as the set of automorphisms $\mathbb{R}^3 \to \mathbb{R}^3$ that preserve the scalar product and the chirality[6] of the coordinate system, the so-called **isometries** of three-dimensional Euclidean space:

$$SO(3,\mathbb{R}) := \left\{ R \in GL(3,\mathbb{R}) \,|\, \langle R\cdot p, R\cdot p\rangle = \langle p,p\rangle \text{ for all } p\in\mathbb{R}^3 \text{ and } \det(R)=1 \right\} \quad .$$

Since $\langle R\cdot p, R\cdot p\rangle = \langle p, (R^T\cdot R)\cdot p\rangle$, we can deduce $R^T\cdot R = \mathbb{1}$ and thus $R^{-1}=R^T$ for all $R\in SO(3,\mathbb{R})$. An explicit representation of the rotations around the standard axes of $\mathbb{R}^3 = \text{span}(\boldsymbol{e}_1,\boldsymbol{e}_2,\boldsymbol{e}_3)$ by the angle $\tau$ is given by:

$$R_{\boldsymbol{e}_1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\tau & -\sin\tau \\ 0 & \sin\tau & \cos\tau \end{pmatrix}, \; R_{\boldsymbol{e}_2} = \begin{pmatrix} \cos\tau & 0 & \sin\tau \\ 0 & 1 & 0 \\ -\sin\tau & 0 & \cos\tau \end{pmatrix}, \; R_{\boldsymbol{e}_3} = \begin{pmatrix} \cos\tau & -\sin\tau & 0 \\ \sin\tau & \cos\tau & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

There are various ways to assemble an arbitrary rotation in the rotation group, amongst others Euler and Tait-Bryan angles, which express an element $R\in SO(3,\mathbb{R})$ as composition of the above-mentioned rotation matrices. Nevertheless, the parametrization with Euler angles gives no direct insight in terms of geodesics and moreover suffers from singularities e.g. Gimbal locks [83]. Hence, we pursue a direct construction using the Lie algebra. Let $R^{m\times m}$ be the vector space of all $m\times m$ matrices, then the lie algebra $so(m,\mathbb{R}) = \mathscr{L}SO(m,\mathbb{R})$ is given by the set of all skew-symmetric matrices with the commutator $[\cdot,\cdot]$ as corresponding Lie bracket:

$$so(m,\mathbb{R}) = \left\{ \Omega\in\mathbb{R}^{m\times m} \,|\, \Omega^T = -\Omega \right\} \quad \text{where} \quad [\Omega,\Omega'] = \Omega\cdot\Omega' - \Omega'\cdot\Omega \quad .$$

Let $\omega = (\omega_1,\omega_2,\omega_3)\in\mathbb{R}^3$ be a normed[7] rotation axis i.e. $\sum_{k=1}^3 \omega_k^2 = 1$, then the corresponding tangent vector in the Lie algebra $so(3,\mathbb{R})$ is given by the embedding

$$\iota : \mathbb{R}^3 \to so(3,\mathbb{R}) \,,\, (\omega_1,\omega_2,\omega_3) \mapsto \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \quad \text{with} \quad \iota(\omega)\cdot p = \omega\times p \quad .$$

A standard basis of the three-dimensional tangent space can be given by $so(3,\mathbb{R}) = \text{span}\big(\iota(\boldsymbol{e}_1),\iota(\boldsymbol{e}_2),\iota(\boldsymbol{e}_3)\big)$ and thus its commutator relations determine the structure constants of the Lie algebra:

$$\big[\iota(\boldsymbol{e}_i),\iota(\boldsymbol{e}_j)\big] = \iota(\boldsymbol{e}_i)\cdot\iota(\boldsymbol{e}_j) - \iota(\boldsymbol{e}_j)\cdot\iota(\boldsymbol{e}_i) = \sum_{k=1}^3 \iota(\boldsymbol{e}_k)\cdot\varepsilon_{ijk} \quad \text{for all} \quad i,j\in\{1,2,3\} \quad ,$$

[6] Let $P = \text{diag}(-1,-1,-1)$ be a reflection at the origin with $P^T\cdot P = \mathbb{1}$ but $\det P = -1$ then the full orthogonal group can be split into two connected components $O(3,\mathbb{R}) = SO(3,\mathbb{R}) \oplus PSO(3,\mathbb{R})$.

[7] This normalization is not mandatory but offers the advantage that we can later directly read off the rotation angle $\tau$.

where $\varepsilon_{ijk}$ is the totally antisymmetric Levi-Civita symbol i.e. the sign of the permutation $\pi : (1,2,3) \mapsto (i,j,k)$. Interestingly, the vector space $\mathbb{R}^3$ with the cross product $[\cdot,\cdot] = \cdot \times \cdot$ as Lie bracket obeys the same structure i.e. $\boldsymbol{e}_1 \times \boldsymbol{e}_2 = \boldsymbol{e}_3$ (cyclic) and thus is isomorphic to $so(3,\mathbb{R})$.

Let us now compute the exponential map which is equivalent to the usual power series on square matrices. The powers of the generator $\Omega := \iota(\omega)$ can be verified by a short calculation:

$$\Omega^0 = \mathbb{1} \quad , \quad \Omega^{2\cdot k} = (-)^{k+1}\Omega^2 \quad \forall k > 0 \quad , \quad \Omega^{2\cdot k+1} = (-)^k\Omega \quad \forall k \geq 0 \quad .$$

As a result, the exponential series $\exp(\tau \cdot \Omega)$ describes a rotation around the axis $\omega$ be the angle $\tau$, the so-called **Rodrigues formula**:

$$\begin{aligned}
\exp(\tau\cdot\Omega) &= \sum_{k=0}^{\infty} \frac{(\tau\cdot\Omega)^k}{k!} = \mathbb{1} + \sum_{k=1}^{\infty} \frac{(\tau\cdot\Omega)^{2k}}{(2k)!} + \sum_{k=0}^{\infty} \frac{(\tau\cdot\Omega)^{2k+1}}{(2k+1)!} \\
&= \mathbb{1} + \Omega^2 \cdot \left(1 - \sum_{k=0}^{\infty} \frac{(-)^k \cdot \tau^{2k}}{(2k)!}\right) + \Omega \cdot \sum_{k=0}^{\infty} \frac{(-)^k \cdot \tau^{2k+1}}{(2k+1)!} \\
&= \mathbb{1} + \Omega^2 \cdot (1 - \cos\tau) + \Omega \cdot \sin\tau \quad .
\end{aligned}$$

Additionally, we can directly read off the logarithm $\tau \cdot \Omega$ for a given group element $\exp(\tau \cdot \Omega)$. The corresponding Ad-invariant Killing form [82] for $X, Y \in so(3,\mathbb{R})$ is given by $\langle X, Y\rangle_K := \mathrm{trace}(X \cdot Y)$ and thus

$$\langle \tau\cdot\Omega, \tau\cdot\Omega\rangle_K = \mathrm{trace}(\tau^2\cdot\Omega^2) = -2\cdot\tau^2\cdot(\omega_1^2+\omega_2^2+\omega_3^2) = -2\cdot\langle\tau\cdot\omega,\tau\cdot\omega\rangle_{\mathbb{R}^3} = -2\cdot\tau^2 \quad .$$

Hence, we can obtain a positive definite bilinear form $\langle\cdot,\cdot\rangle_{so(3,\mathbb{R})} := -\frac{1}{2}\langle\cdot,\cdot\rangle_K$ by simply multiplying the negative definite Killing form by $-\frac{1}{2}$. One can make another interesting observation: Let $\omega \in \mathbb{R}^3$ be an arbitrarily scaled rotation axis i.e. $\langle\omega,\omega\rangle_{\mathbb{R}^3} \in \mathbb{R}_0^+$ and $\Omega = \iota(\omega) \in so(3,\mathbb{R})$ the corresponding tangent vector then

$$\tfrac{1}{2}\|\exp(\Omega)\|^2_{SO(3,\mathbb{R})} = \frac{1}{2}\langle\Omega,\Omega\rangle_{so(3,\mathbb{R})} = \frac{1}{2}\langle\omega,\omega\rangle_{\mathbb{R}^3}$$

is exactly the kinetic energy (unit mass) that is needed to rotate a rigid body from the identity to its final position $\exp(\Omega)$ with constant angular velocity $\|\omega\|_{\mathbb{R}^3}$. Assuming a non-isotropic distribution of the mass density, an augmentation of the distance formula with a symmetric and positive definite matrix $J$, the so-called **inertia tensor**, is conceivable such that $\|\exp(\Omega)\|^2_{SO(3,\mathbb{R})} = \langle\omega, J\cdot\omega\rangle_{\mathbb{R}^3}$.

Given an arbitrary rotation $R = \mathbb{1} + \Omega^2\cdot(1-\cos\tau) + \Omega\cdot\sin\tau$, we can determine the normed tangent vector $\Omega \propto R - R^T$ since $\mathbb{1}$ and $\Omega^2$ are symmetric matrices in contrast to the skew-symmetric matrix $\Omega$. The rotation angle $\tau$ can be deduced from the computation of the trace of $R$ since $R = U^T \cdot R_{\boldsymbol{e}_3} \cdot U$ and thus

$$\mathrm{trace}(U^T\cdot R_{\boldsymbol{e}_3}\cdot U) = \mathrm{trace}(U\cdot U^T\cdot R_{\boldsymbol{e}_3}) = \mathrm{trace}(R_{\boldsymbol{e}_3}) = 2\cdot\cos\tau + 1$$

for any orthogonal matrix $U \in O(3,\mathbb{R})$. Here, we exploit the fact, that every rotation can be seen as rotation around the z-axis in a properly chosen coordinate system.

Concluding, logarithms of rotations are simply given by their rotation axis $\omega$ and their norm by the modulus of the rotation angle $|\tau|$. More explicitly, let us emphasize: **The angle-axis representation is equivalent to the logarithm** – a natural and aesthetic result.

### 5.2.2 Unit Quaternions

Up to this point, we discussed everything which is necessary to calculate the geodetic distance between two rotation matrices. Given any elastic time series alignment algorithm, we simply have to substitute the local weighting function by

$$w^{SO(3,\mathbb{R})}(R', R) = \|R' \cdot R^{-1}\|^2_{SO(3,\mathbb{R})} = \tau^2 = \arccos^2\left(\frac{\mathrm{trace}(R' \cdot R^T) - 1}{2}\right) \quad ,$$

where $R, R' \in SO(3,\mathbb{R})$ is a combination of two rotation matrices. Unfortunately, this representation of orientations needs nine real numbers to store a rotation matrix – three times the actual dimension of the Lie group $SO(3,\mathbb{R})$. Alternatively, we can use the group of **unit** Hamiltonian quaternions $\mathbb{H}_u$ to represent rotations

$$\mathbb{H}_u := \left\{(s,x) \in \mathbb{R} \times \mathbb{R}^3 \,|\, s^2 + \langle x, x\rangle_{\mathbb{R}^3} = 1\right\} = \left\{(\cos\tfrac{\tau}{2}, \tfrac{\omega}{\|\omega\|}\sin\tfrac{\tau}{2}) \in \mathbb{R} \times \mathbb{R}^3 \,|\, \tau \in \mathbb{R}\right\} \quad ,$$

with the corresponding group law and inverse element:

$$\begin{pmatrix} s \\ x \end{pmatrix} \cdot \begin{pmatrix} s' \\ x' \end{pmatrix} = \begin{pmatrix} s \cdot s' - \langle x, x'\rangle_{\mathbb{R}^3} \\ s \cdot x' + s' \cdot x + x \times x' \end{pmatrix} \quad \text{as well as} \quad \begin{pmatrix} s \\ x \end{pmatrix}^{-1} = \begin{pmatrix} s \\ -x \end{pmatrix} \quad .$$

The set of all unit quaternions is topologically isomorphic to the unit sphere $\mathscr{S}^3$ embedded in four dimensions. Hence, we only need four entries instead of nine to encode a rotation. Together with the above-stated group law $(\mathbb{H}_u, \cdot)$ is equivalent to the **double covering** of $SO(3,\mathbb{R})$, the so-called group of **special unitary transformations** $SU(2,\mathbb{C})$ i.e. there exist a two-to-one epimorphism (surjective group homomorphism) $(\mathbb{H}_u, \cdot) \cong SU(2,\mathbb{C}) \to SO(3,\mathbb{R})$ from the group of unit quaternions to the group of rotations.[8]

As an example, let us rotate the point $p \in \mathbb{R}^3$ around the normed axis $\omega \in \mathbb{R}^3$ (choose $\langle \omega, \omega\rangle_{\mathbb{R}^3} = 1$) by the angle $\tau \in \mathbb{R}$. To achieve that, $p$ has to be embedded into the space of (not necessarily unit) quaternions $p \mapsto P := (0, p)$. Afterwards, we compute the inner action $I_g(p)$ of the unit quaternion $g = (\cos\frac{\tau}{2}, \omega\sin\frac{\tau}{2})$ on the

[8] For the sake of simplicity, we omit the explicit expression and verify this claim with an example. The interested reader may refer to [84]. Furthermore, it can be shown that $SO(3,\mathbb{R}) = SU(2,\mathbb{C})/\mathbb{Z}_2$.

embedded point $P := (0, p)$. On the one hand we gain the following result:

$$\begin{aligned} I_g(p) = g \cdot P \cdot g^{-1} &= \begin{pmatrix} \cos\frac{\tau}{2} \\ \omega \cdot \sin\frac{\tau}{2} \end{pmatrix} \cdot \begin{pmatrix} 0 \\ p \end{pmatrix} \cdot \begin{pmatrix} \cos\frac{\tau}{2} \\ -\omega \cdot \sin\frac{\tau}{2} \end{pmatrix} \\ &= \begin{pmatrix} \cos\frac{\tau}{2} \\ \omega \cdot \sin\frac{\tau}{2} \end{pmatrix} \cdot \begin{pmatrix} \langle p, \omega \rangle_{\mathbb{R}^3} \cdot \sin\frac{\tau}{2} \\ p \cdot \cos\frac{\tau}{2} + (\omega \times p) \cdot \sin\frac{\tau}{2} \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ p \cdot \cos\tau + \omega \cdot \langle \omega, p \rangle_{\mathbb{R}^3} \cdot (1 - \cos\tau) + (\omega \times p) \cdot \sin\tau \end{pmatrix} \quad . \end{aligned}$$

On the other hand, the action of an arbitrary rotation $R = \mathbb{1} + \Omega^2 \cdot (1 - \cos\tau) + \Omega \cdot \sin\tau$ on a point $p \in \mathbb{R}^3$ computes to (using the above-mentioned identity $\Omega \cdot p = \omega \times p$):

$$\begin{aligned} R \cdot p &= \mathbb{1} \cdot p + \omega \times (\omega \times p) \cdot (1 - \cos\tau) + (\omega \times p) \cdot \sin\tau \\ &= p + (\omega \cdot \langle \omega, p \rangle_{\mathbb{R}^3} - p \cdot \langle \omega, \omega \rangle_{\mathbb{R}^3}) \cdot (1 - \cos\tau) + (\omega \times p) \cdot \sin\tau \\ &= p \cdot \cos\tau + \omega \cdot \langle \omega, p \rangle_{\mathbb{R}^3} \cdot (1 - \cos\tau) + (\omega \times p) \cdot \sin\tau \quad . \end{aligned}$$

As a result, the inner action of $\mathbb{H}_u$ acts like the left action of $SO(3, \mathbb{R})$ on $\mathbb{R}^3$. However, for each orientation $R \in SO(3, \mathbb{R})$ there exist two quaternions that represent the same rotation namely $g$ and $-g$ since $I_g(p) = I_{-g}(p)$ for all $g \in \mathbb{H}_u$. Although, $\mathbb{H}_u \cong SU(2, \mathbb{C})$ and $SO(3, \mathbb{R})$ are **not the same group**, they share the **same Lie algebra** i.e. $su(2, \mathbb{C}) = so(3, \mathbb{R})$. Let $\omega \in \mathbb{R}^3$ be a normed rotation axis, then the embedding[9]

$$\iota : \mathbb{R}^3 \to \mathscr{L}\mathbb{H}_u \, , \, \omega \mapsto \iota(\omega) := \begin{pmatrix} 0 \\ \frac{1}{2} \cdot \omega \end{pmatrix}$$

ensures the same commutator relations of the standard tangent vectors $\iota(\boldsymbol{e}_k)$ since

$$[\iota(\boldsymbol{e}_i), \iota(\boldsymbol{e}_j)] = \iota(\boldsymbol{e}_i) \cdot \iota(\boldsymbol{e}_j) - \iota(\boldsymbol{e}_j) \cdot \iota(\boldsymbol{e}_i) = \iota(\boldsymbol{e}_i \times \boldsymbol{e}_j) \quad ,$$

where $\cdot$ is the quaternion product. The exponential map is given by the series expansion of the exponential function. The exponential series reveals the usual parametrization of unit quaternions. Since $\iota(\omega)$ has the same algebraic properties like in the $so(3, \mathbb{R})$ case we omit an explicit computation to avoid redundancy:

$$\exp\big(\tau \cdot \iota(\omega)\big) = \sum_{k=0}^{\infty} \frac{1}{k!} \big(\tau \cdot \iota(\omega)\big)^k = \begin{pmatrix} \cos\frac{\tau}{2} \\ \omega \cdot \sin\frac{\tau}{2} \end{pmatrix} \quad .$$

Hence, the rotation angle (geodetic distance to the identity) can be extracted by $\tau = 2 \cdot \arccos(|s|)$ for any unit quaternion $(s, x)$.[10] In terms of logarithms, the same can be achieved by the computation of the Euclidean distance on $\mathscr{L}\mathbb{H}_u$:

$$\big\| \exp\big(\tau \cdot \iota(\omega)\big) \big\|^2_{\mathbb{H}_u} = 4 \cdot \big\langle (0, \tfrac{\tau}{2} \cdot \omega)^T, (0, \tfrac{\tau}{2} \cdot \omega)^T \big\rangle_{\mathscr{L}\mathbb{H}_u} = 4 \cdot \big\langle \tfrac{\tau}{2} \cdot \omega, \tfrac{\tau}{2} \cdot \omega \big\rangle_{\mathbb{R}^3} = \tau^2 \quad .$$

[9] Note, the excessive appearance of the factor $\frac{1}{2}$ is solely caused by the double covering of the angular space – a popular excuse amongst physicists to explain a missing 2 in their calculation.

[10] The modulus map $|s|$ removes the ambiguity of the double covering $SU(2, \mathbb{C})/\mathbb{Z}_2 = SO(3, \mathbb{R})$.

Concluding, unit quaternions behave similarly to rotations in $SO(3,\mathbb{R})$ but can be treated more efficiently in data mining tasks due to their moderate memory overhead and lightweight group operations. Let us now start crunching data.

### 5.2.3 Applications in Time Series Data Mining

As we have seen, the angular parameter $\tau$ is solely determined by the absolute scalar value of a unit quaternion. Thus, the weighting function between two arbitrary unit quaternions $p, q \in \mathbb{H}_u$ can be simplified to

$$w(p,q) := \|p \cdot q^{-1}\|^2_{\mathbb{H}_u} = \tau^2 = 4 \cdot \arccos^2(|\langle p,q\rangle_{\mathbb{R}^4}|) \quad .$$

Assume $p$ and $q$ lie in the same hemisphere[11] of $\mathscr{S}^3 \cong \mathbb{H}_u$ i.e. $\langle p,q\rangle_{\mathbb{R}^4} \geq 0$ then the length of straight lines $w_{\mathrm{LB}}(p,q) := \|p-q\|^2_{\mathbb{R}^4} = 2 - 2 \cdot \langle p,q\rangle_{\mathbb{R}^4}$ is a lower bound of $w(p,q)$ since $4 \cdot \arccos^2(x) - (2 - 2 \cdot x)$ is a non-negative function for $x \in [0,1]$ (see Figure 5.1 for a geometric interpretation). Thus, when executing 1NN search under (C)DTW we can utilize $\mathrm{LB}_{\mathrm{Kim}}$ using the weighting function $w_{\mathrm{LB}}(p,q)$ in order to suppress the expensive inverse cosine calls. The time series of quaternions can simply be passed to the alignment algorithm as an array of structs (e.g. `float[4]`).

Quaternions offer another advantage. Given two arbitrary rotation matrices $R, R' \in SO(3,\mathbb{R})$ then their average $\frac{R+R'}{2}$ is in general not an orthogonal matrix. As we have seen, we could use a parametrization in the Lie algebra, the so-called spherical linear interpolation

$$\mu := \mathrm{SLERP}\left(R, R', \tfrac{1}{2}\right) \quad \text{where} \quad \mathrm{SLERP}(R,R',\tau) = \exp\left(\tau \cdot \log(R' \cdot R^{-1})\right) \cdot R$$

to determine the (geodetic) average [85]. Alternatively, for every unit quaternion $p \in \mathbb{H}_u$ it holds $\langle p,p\rangle_{\mathbb{R}^4} = 1$ and thus we can define a projection from any point $x \in \mathbb{R}^4$ to the unit sphere $\mathscr{S}^3 \cong \mathbb{H}_u$ by

$$\pi : \mathbb{R}^4 \to \mathscr{S}^3 \subset \mathbb{R}^4, \, x \mapsto \pi(x) = \frac{x}{\|x\|} \quad .$$

The normalization map $\pi$ moves a point $x \in \mathbb{R}^4$ in radial direction to the surface of the unit sphere $\mathscr{S}^3$. Let $(p_0,\ldots,p_{n-1})$ be a sequence of quaternions and $(\alpha_0,\ldots,\alpha_{n-1})$ be a sequence of non-negative weights, then we can define their weighted average as $\mu := \pi(\sum_{k=0}^{n-1} \alpha_k \cdot p_k)$. Unfortunately, we have to make sure that there are no cancellations due to the double covering of $SO(3,\mathbb{R})$ such that $p \sim -p$. This could be achieved by forcing each quaternion to lie in the same hemisphere as the identity i.e. $\langle p_k, (1,0,0,0)^T\rangle_{\mathbb{R}^4} \geq 0$. However, this approach may lead to spontaneous flips in the sign of the quaternion entries if the modulus of the scalar component is near zero. Alternatively, on can exploit the continuous

[11] This can always be ensured by the mapping $p \mapsto \mathrm{sign}(\langle q,p\rangle_{\mathbb{R}^4}) \cdot p$.

nature of physical motions and define a more robust assignment of signs by the recursive equation:

$$p_{k+1} \leftarrow \operatorname{sign}\big(\langle p_k, p_{k+1}\rangle_{\mathbb{R}^4}\big) \cdot p_{k+1} \quad \text{with} \quad p_0 = p_0 \quad .$$

Averages of quaternion sequences can be used in the following situations: Assume we measure the same motion with two arbitrarily aligned gyro sensors, we obtain two time series $C = (p_0, \ldots, p_{n-1})$ and $C' = C \cdot \Delta = (p_0 \cdot \Delta, \ldots, p_{n-1} \cdot \Delta)$ where $\Delta$ is the difference transformation between the gyro sensors. Similar to the offset removal of real-valued time series, the elastic alignment $\mathrm{DTW}(C, C')$ is meaningless since both coordinate systems do not agree. We can fix that by back-rotating $C$ and $C'$ with their inverse averages. Let $\mu_C$ be the average of $C$ then $\mu_{C'} = \mu_C \cdot \Delta$ and thus

$$O(C') = C' \cdot \mu_{C'}^{-1,} = (C \cdot \Delta) \cdot (\mu_C \cdot \Delta)^{-1} = C \cdot (\Delta \cdot \Delta^{-1}) \cdot \mu_C^{-1} = C \cdot \mu_C^{-1} = O(C) \quad .$$

This preprocessing step accounts for $\mathcal{O}(|C|)$ operations and has to be applied for each time series of rotations independently.[12] However, we can compute the windowed averages $\mu_{C^{(k)}}$ for all subsequence candidates in linear time. Since $(\mathbb{R}^4, +)$ is associative in each of its components, we can mimic the streamed computation of the windowed average for real-valued time series by simply computing windowed averages for each axis independently (using prefix sums). In this case, the stream of quaternions should be stored as a struct of arrays to allow for optimal memory access patterns during the computation of the prefix sums.

In contrast to subsequence DTW, the Gliding Elastic match (GEM) uses the exponential moving average (EMA) to propagate the time-dependent drift along the warping path. GEM's local weighting function $w(p,q) = \|p + \Delta - q\|^2_{\mathbb{R}}$ is augmented with a difference transformation $\Delta$ which encodes the time-dependent offset between the two time series candidates. In terms of quaternions this expression can be rewritten as $w(p,q) = \|p \cdot \Delta \cdot q^{-1}\|^2_{\mathbb{H}_u}$ where $\Delta$ is the above-mentioned difference transformation of coordinate frames. In the real-valued case, the update step $\Delta \leftarrow (1-\alpha)\cdot\Delta + \alpha \cdot U$ of the EMA filter is obviously a weighted average of the old difference transformation and the measured residue $U = (q - p)$. In terms of unit quaternions we can simply substitute the expression by the quaternion-valued update step $\Delta \leftarrow \pi\big((1-\alpha)\cdot\Delta + \alpha \cdot (p^{-1} \cdot q)\big)$. As a result, GEM can be extended to match time series of rotations in a straight-forward manner. Interestingly, this statement is true for all compact Lie groups even if we cannot provide the projection $\pi$. The use of $\pi$ is only a shortcut for the linear interpolation of quaternions in their Lie algebra and thus we could still use the general expression $\Delta \leftarrow \big((p^{-1} \cdot q) \cdot \Delta^{-1}\big)^{\alpha} \cdot \Delta = \exp\big(\alpha \cdot \log(p^{-1} \cdot q \cdot \Delta^{-1})\big) \cdot \Delta$ for any choice of the elasticity parameter $\alpha \in [0,1]$.

[12] Note, analogous to z-normalization this can be harmful if **all** sequences are measured in the **same** reference frame and we can exclude baseline-wandering or time-dependent drift.

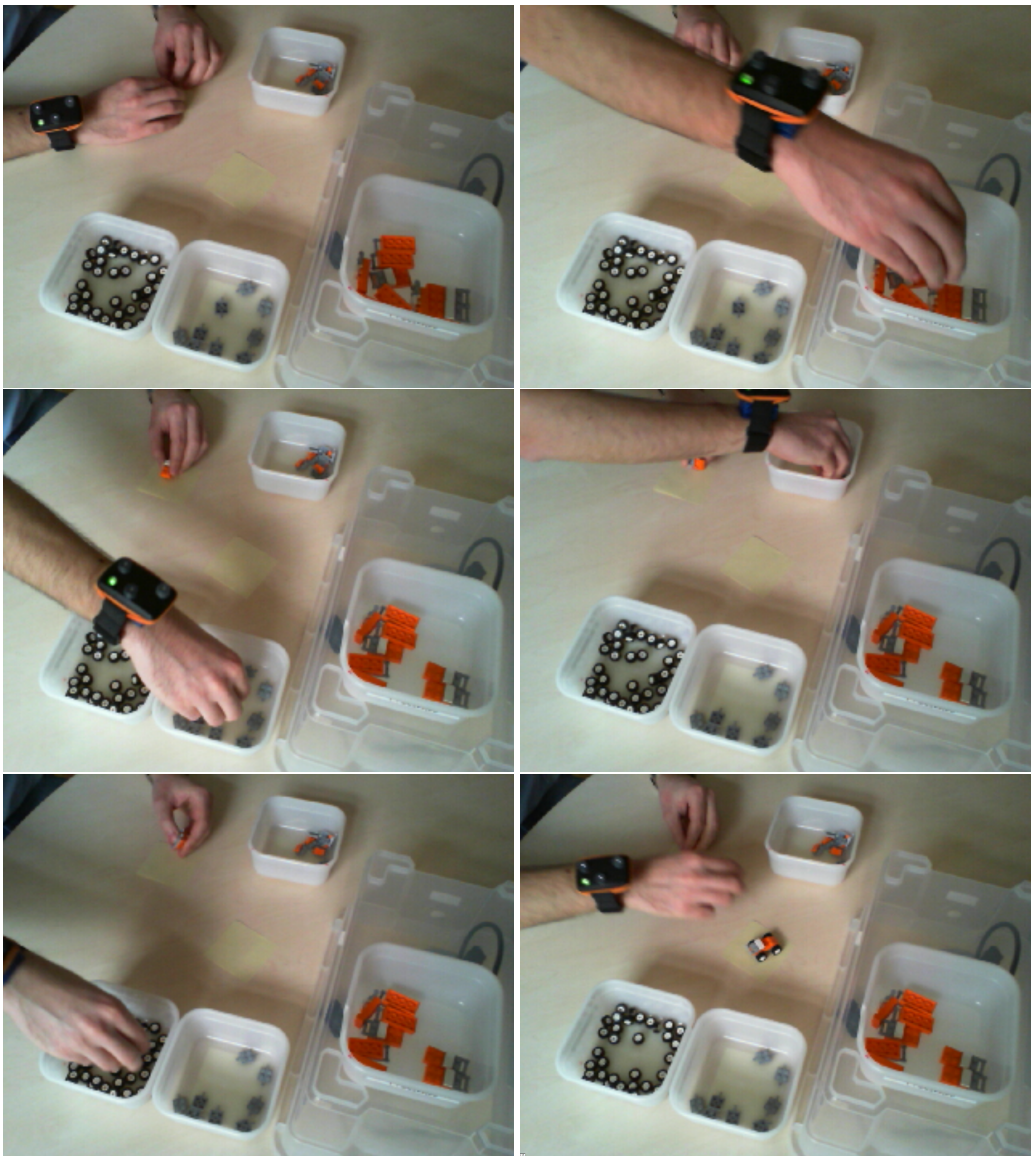

Figure 5.4: Assembly of a LEGO car recorded as a stream of quaternions with an inertial measurement unit attached to the right hand. The assembler fetches the components from different boxes and attaches them to the vehicle. Starting from the initial position (upper left panel) the chassis is fetched from the box in the centre of the table (upper right panel). The two axis for the wheels are fetched from different boxes (two panels in the middle). Afterwards, each of the four wheels is fetched independently from the corresponding box and attached to the axes (lower left panel). Finally, the complete vehicle is deposited at the target position (lower right panel). Consequently, the assembly task can be segmented into eight steps and five classes (class label in brackets): Step 1: chassis (0) → Step 2: axis one (1) → Step 3: axis two (2) → Step 4: wheel one (3) → Step 5: wheel two (3) → Step 6: wheel three (3) → Step 7: wheel four (3) → Step 8: deposit (4).

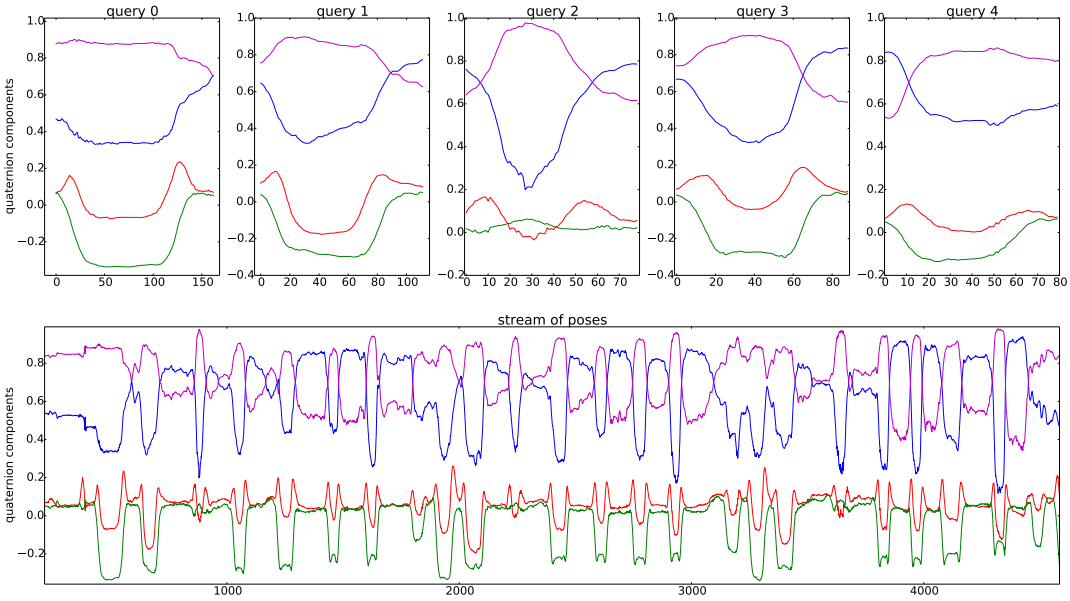


Figure 5.5: Stream of quaternions recorded during the assembly task of a LEGO car (lower panel). The four time-dependent coordinate functions of the quaternions are plotted in different color (scalar part: red, x-axis: green, y-axis: blue, z-axis: magenta). The upper panels depict the five quaternion-valued queries that correspond to the different assembly steps (same color-coding).

In order to backup our theoretical ideas with experimental data, we investigate a multi-shape segmentation task on a stream of quaternions. In our experiment, we assemble a LEGO car consisting of seven parts (one chassis, two axes and four wheels).[13] The assembly task can be split into eight steps and five classes as described in Figure 5.4. Providing one query for each class (chassis=0, axis one=1, axis two=2, wheel=3, deposit=4) we compute a segmentation of the measured stream using the algorithm listed in Figure 4.20. An excerpt of the measured stream consisting of three full periods of a construction cycle (i.e. the assembly of three individual cars) is depicted in Figure 5.5.

For the sake of simplicity we haven chosen to set the penetration parameter of the non-overlapping criterion to $p = 0$. Moreover, we have decided to omit reference frame normalization since the queries and the stream are measured in the same coordinate system. Otherwise, the segmentation quality decreases due to the strong similarity of the different assembly steps (all classes describe simple grasping movements). The segmentation algorithm uses length-normalized distance scores computed with Sakoe-Chiba-constrained DTW (relative warping window of 5%) and the Gliding Elastic Match (default scaling $s_+ = s_- = 2$) with

[13] The stream and the images were gratefully donated by soft2tec (`http://soft2tec.com/`).

disabled reference frame adjustment. Both similarity measures are equipped with the above-mentioned geodetic weighting function $w(p,q)$ for unit quaternions. Figure 5.6 depicts the obtained segmentation using alignment scores from GEM and DTW. Obviously, both similarity measures cause a reasonable segmentation of the stream. However, subsequence DTW cannot handle different query lengths without knowing the uniform scaling factor a priori. Hence, the segmentation algorithm is forced to match the similar 'axis one' class onto the last two occurrences of 'chassis' to compensate the distinct scaling of the time-axis.

Note, contrary to popular belief DTW is not invariant under uniform scaling. It can be made so, if we know the beginning and the end of the shape which is not the case in stream segmentation tasks – otherwise we would know the segmentation beforehand. To resolve this problem, we could iteratively brute-force all conceivable scaling factors. However, GEM guarantees this invariance at a computational complexity of $\mathcal{O}(|S|\cdot|Q|)$ that is even lower than subsequence DTW's $\mathcal{O}(|S|\cdot|Q|^2)$ dependency. Further evaluation steps amongst others the dependency of the segmentation quality on the size of the motif dictionary (here we used just one query per class), suitable scoring functions to assign quantitative quality measures of the segmentation and the applicability to other domains are still to be accomplished in future research. The proposed algorithm solely uses orientations in angular space. However, one can extend it to handle angular orientations and spatial positions at the same time.

## 5.3 Time Series of Rigid Transformations in $SE(3,\mathbb{R})$

This section deals with sequences of rigid transformations, namely time series of rotation matrices and displacement vectors $\big((R_0,T_0),(R_1,T_1),\dots\big)$. Analogous to the previous section, we have to find a proper data representation and a notion of distance between two rigid transformations. Unfortunately, the corresponding Lie group of **special Euclidean transformations** $SE(m,\mathbb{R}) = SO(m,\mathbb{R}) \ltimes \mathbb{R}^3$ is not compact due to the translational part. Thus, the construction of geodetic distances using parametrizations in the Lie algebra cannot be adopted directly. However, we can find similar memory-efficient representations of rigid transformations and reasonable notions of distance between them.

### 5.3.1 The Group of Rigid Transformations and its Lie Algebra

The group rigid transformations acting on $\mathbb{R}^3$ is given by the semi-direct product of the rotation group $SO(3,\mathbb{R})$ and the additive group of translations $(\mathbb{R}^3,+)$. Let $p$ be a point in $\mathbb{R}^3$ and $(R,T)$ as well as $(R',T')$ two rigid transformations then their

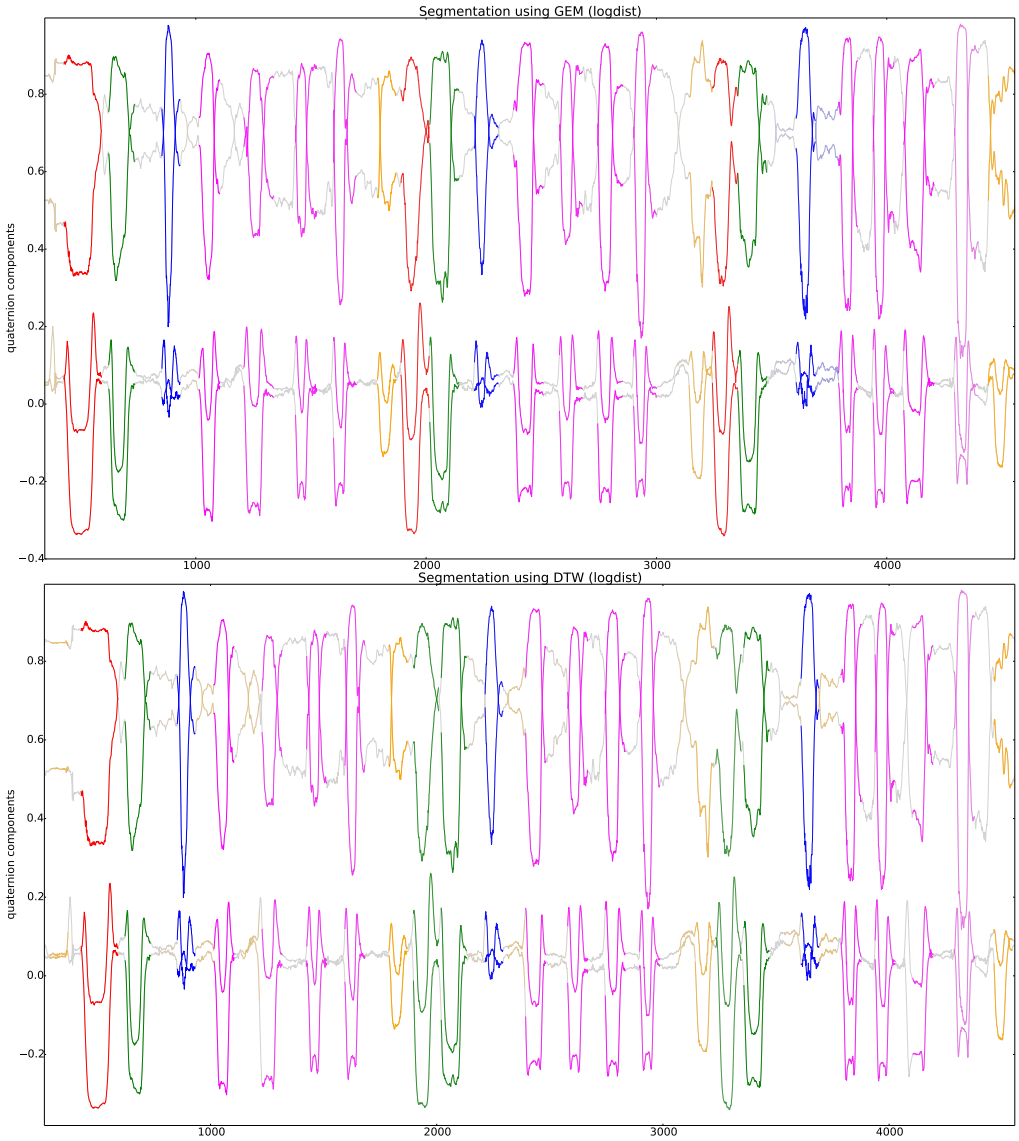


Figure 5.6: Multi-shape segmentation of a quaternion-valued stream recorded with an inertial measurement unit during the assembly of a LEGO car using GEM (upper panel) and Sakoe-Chiba-constrained DTW (lower panel). The colors encode the label of the shape (chassis=red, axis one=green, axis two=blue, wheel=magenta, deposit=orange). GEM (upper panel) recovers the correct label order – we can clearly observe three periods of 'chassis' → 'axis one' → 'axis two' → 'wheel' → 'wheel' → 'wheel' → 'wheel' → 'deposit'. In contrast, DTW (lower panel) mixes up the nearby 'chassis' and 'axis one' class. This can be explained as follows. The used query for 'chassis' is given by the red-coloured shape in the lower panel. The following 'chassis' shapes exhibit a distinct global scaling factor of the time-axis. Subsequence DTW cannot cover this invariance without knowing the scaling factor a priori (also see third wheel in the last period).

(group) action $\Phi$ on $p$ recovers the group law of the semi-direct product:

$$\Phi\big((R,T)\cdot(R',T'),p\big) = R\cdot(R'\cdot p+T')+T = \Phi\big((R\cdot R',T+R\cdot T'),p\big) \quad ,$$

where $\cdot$ is the standard matrix product and $+$ the addition of vectors. Obviously, the rigid transformation acts multiplicative and additive in this representation which makes it hard to state an exponential map. A popular solution for this problem is an embedding of $\mathbb{R}^3$ into the projective space $\mathbb{P}^3(\mathbb{R})$. The corresponding action $\Psi$ of a group element $(R,T)$ on a representative $(p,1) := (p_x,p_y,p_z,1)^T$ can than be written as an exclusively multiplicative matrix product:

$$\Psi\big((R,T),p\big) = \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix}\cdot\begin{pmatrix} p \\ 1 \end{pmatrix} = \begin{pmatrix} R\cdot p+T \\ 1 \end{pmatrix} = \begin{pmatrix} \Phi\big((R,T),p\big) \\ 1 \end{pmatrix} \quad .$$

Let $\omega$ be a rotation axis in $\mathbb{R}^3$ and $\Omega$ the associated tangent vector in $so(3,\mathbb{R}) = \mathscr{L}SO(3,\mathbb{R})$, further let $T$ be velocity vector in $\mathscr{L}\mathbb{R}^3$, then an embedding of $(\omega,T)$ into the Lie algebra $se(3,\mathbb{R}) = \mathscr{L}SE(3,\mathbb{R})$ is given as follows:

$$\iota : \mathbb{R}^3\times\mathbb{R}^3 \to se(3,\mathbb{R}) \subset \mathbb{R}^{4\times4},\ (\omega,T) \mapsto \iota(\omega,T) := \begin{pmatrix} \Omega & T \\ 0 & 0 \end{pmatrix} \quad .$$

A standard basis of the Lie algebra is given by the choice of the six basis vectors $\{\iota(\boldsymbol{e}_1,0),\iota(\boldsymbol{e}_2,0),\iota(\boldsymbol{e}_3,0),\iota(0,\boldsymbol{e}_1),\iota(0,\boldsymbol{e}_2),\iota(0,\boldsymbol{e}_3)\}$. The corresponding Lie bracket is determined by the standard commutator of square matrices. Omitting the explicit computation, one can verify that **rotations do not commute** in general

$$[\iota(\boldsymbol{e}_i,0),\iota(\boldsymbol{e}_j,0)] = \iota(\boldsymbol{e}_i\times\boldsymbol{e}_j,0) \quad \text{for all } i,j\in\{1,2,3\}$$

in contrast to pure translations which always exhibit a vanishing commutator

$$[\iota(0,\boldsymbol{e}_i),\iota(0,\boldsymbol{e}_j)] = 0 \qquad \text{for all } i,j\in\{1,2,3\} \quad .$$

The remaining relations state that **rotations and translation do not commute**

$$[\iota(\boldsymbol{e}_i,0),\iota(0,\boldsymbol{e}_j)] = \iota(0,\boldsymbol{e}_i\times\boldsymbol{e}_j) \quad \text{for all } i,j\in\{1,2,3\} \quad .$$

However, if the rotation axis $\omega$ and the displacement vector $T$ are collinear, the expression $\boldsymbol{e}_i\times\boldsymbol{e}_j$ vanishes i.e. it does not matter if the rotation or the translation is applied first. **Only in this particular case**, the exponential series of $\iota(\omega,T)$ is given by the group element $(R,T)$ since

$$\exp\big(\iota(\omega,T)\big) = \mathbb{1}_{\mathbb{R}^4} + \begin{pmatrix} \Omega & T \\ 0 & 0 \end{pmatrix} + \tfrac{1}{2!}\begin{pmatrix} \Omega^2 & 0 \\ 0 & 0 \end{pmatrix} + \tfrac{1}{3!}\begin{pmatrix} \Omega^3 & 0 \\ 0 & 0 \end{pmatrix} + \cdots = \begin{pmatrix} \exp(\Omega) & T \\ 0 & 1 \end{pmatrix} \quad .$$

As a result, the matrix $\iota(\omega,T)$ is in general **not the logarithm** of the group element $(R,T) = (\exp(\Omega),T)$. At this point, one would try to determine a positive definite

Ad-invariant inner product $\langle \cdot, \cdot \rangle_{se(3,\mathbb{R})}$ on the Lie algebra analogous to the $so(3,\mathbb{R})$ case. Unfortunately, the literature states that the only Ad-invariant choice, a linear combination of the Killing form and the Klein form [82],

$$\langle \iota(\omega, T), \iota(\omega', T') \rangle_{se(3,\mathbb{R})} = \alpha \cdot \langle \omega, \omega' \rangle_{\mathbb{R}^3} + \beta \cdot \left( \langle \omega, T' \rangle_{\mathbb{R}^3} + \langle \omega', T \rangle_{\mathbb{R}^3} \right)$$

for $\alpha, \beta \in \mathbb{R}^+$ is either indefinite or degenerate (see [82] for an exhaustive discussion). Consequently, we can find distinct group elements with vanishing or negative distance – a counter-intuitive result considering our alignment task.

Nevertheless, we can define a positive definite inner product $\langle \cdot, \cdot \rangle_{kin}$ using a weighted scalar product of the approximate representation of the logarithm

$$\langle \iota(\omega, T), \iota(\omega', T') \rangle_{kin} = \alpha \cdot \langle \omega, \omega' \rangle_{\mathbb{R}^3} + \beta \cdot \langle T, T' \rangle_{\mathbb{R}^3} \quad .$$

The corresponding norm of a group element $\|(R,T)\|^2_{kin} = \alpha \cdot \langle \omega, \omega \rangle_{\mathbb{R}^3} + \beta \cdot \langle T, T \rangle_{\mathbb{R}^3}$ can be interpreted as kinetic energy that is needed to rotate a rigid body at constant angular speed and translational velocity. Obviously, the expression is a weighted sum of squared geodetic lengths which have been calculated independently on the groups $SO(3,\mathbb{R})$ and $(\mathbb{R}^3, +)$. Hence, this assignment is often called **double-geodetic** or **kinetic** measure.

The kinetic measure has another advantage: The angular space is usually parametrized in radians in contrast to the translational component which can carry any suitable physical unit of length. Thus, the notion of similarity can differ substantially if we alter the physical units. The quotient $\frac{\alpha}{\beta}$ can be chosen adequately to relate the angular and spatial components. As an example, assume the rotational components do not carry any meaningful information – we can empirically determine the optimal choice $\frac{\alpha}{\beta} = 0$ during a cross-validation and thus completely eliminate their influence.

### 5.3.2 Unit Dual Quaternions

Similar to unit quaternions it is possible to represent rigid transformations in a memory-efficient manner. To achieve that, a rigid transformation can be written as **unit dual quaternion** $\hat{q} := q_0 + \varepsilon \cdot q_\varepsilon$ where $q_0 \in \mathbb{H}_u$ is the unit quaternion associated with a rotation and $q_\varepsilon \in \mathbb{H}$ is an arbitrarily normed quaternion carrying the translational information. The dual basis vector $\varepsilon \neq 0$ has the property $\varepsilon^2 = 0$ and commutes with all basis vectors of quaternions i.e. $\varepsilon \cdot q = q \cdot \varepsilon$ for all $q \in \mathbb{H}$. A pure rotation can be simply written as $\hat{q}_0 = q_0 + \varepsilon \cdot 0 = q_0$. Let $T = (T_x, T_y, T_z) \in \mathbb{R}^3$ be a displacement vector and $t = (0, T_x, T_y, T_z) \in \mathbb{H}$ be the associated quaternion then the dual quaternion $\hat{t} = e + \varepsilon \cdot \frac{t}{2} = (1, \frac{\varepsilon}{2} \cdot T_x, \frac{\varepsilon}{2} \cdot T_y, \frac{\varepsilon}{2} \cdot T_z)$ describes a pure translation. A quick calculation reveals the mapping of the additive group of displacements $(\mathbb{R}^3, +)$ to the multiplicative group of unit dual quaternions $\mathbb{D}_u$:

$$\hat{t} \cdot \hat{t}' = (e + \tfrac{\varepsilon}{2} \cdot t) \cdot (e + \tfrac{\varepsilon}{2} \cdot t') = e + \tfrac{\varepsilon}{2} \cdot (t + t') = \widehat{t + t'} \quad .$$

A full rigid transformation can be written as the composition of a rotation and a translation $\hat{q} = (e + \frac{\varepsilon}{2} \cdot t) \cdot q_0 = q_0 + \frac{\varepsilon}{2} \cdot t \cdot q_0$. The inverse element is given by

$$\hat{q}^{-1} = q_0^{-1} \cdot (e - \tfrac{\varepsilon}{2} \cdot t) = q_0^{-1} - \tfrac{\varepsilon}{2} \cdot q_0^{-1} \cdot t$$

and the dual conjugate of the inverse element by negating the sign of its dual part

$$\overline{\hat{q}^{-1}} = q_0^{-1} \cdot (e + \tfrac{\varepsilon}{2} \cdot t) = q_0^{-1} + \tfrac{\varepsilon}{2} \cdot q_0^{-1} \cdot t \quad .$$

A test point $p \in \mathbb{R}^3$ can be embedded into the quaternion algebra via $p \mapsto P := e + \varepsilon \cdot p = (1, \varepsilon \cdot p_x, \varepsilon \cdot p_y, \varepsilon \cdot p_z)$. Similar to unit quaternions, the action of $\mathbb{D}_u$ on the test point $P$ is given by the adjoint 'sandwich operation' [14] $\Phi(\hat{q}, P) = \hat{q} \cdot P \cdot \overline{\hat{q}^{-1}}$:

$$\begin{aligned}
\Phi(\hat{q}, P) &= (q_0 + \tfrac{\varepsilon}{2} \cdot t \cdot q_0) \cdot (e + \varepsilon \cdot p) \cdot (q_0^{-1} + \tfrac{\varepsilon}{2} \cdot q_0^{-1} \cdot t) \\
&= q_0 \cdot q_0^{-1} + \varepsilon \cdot q_0 \cdot p \cdot q_0^{-1} + \frac{\varepsilon}{2} \cdot \left( q_0 \cdot q_0^{-1} \cdot t + t \cdot q_0 \cdot q_0^{-1} \right) \\
&= e + \varepsilon \cdot \left( q_0 \cdot p \cdot q_0^{-1} + t \right) \quad .
\end{aligned}$$

As expected, the action rotates the point $p$ followed by a translation. Analogous to unit quaternions the double covering of $SE(3, \mathbb{R})$ causes an ambiguity in the sign of the transformation since $\Phi(\hat{q}, P) = \Phi(-\hat{q}, P)$. The embedding into the Lie algebra can be accomplished by $\iota(\omega, T) = (0, \frac{1}{2} \cdot \omega + \frac{\varepsilon}{2} \cdot T)$ and causes the same commutator relations as discussed in the previous section. The exponential series is computed analogously to unit quaternions using powers of dual quaternions $\hat{q}^k = \hat{q}^{k-1} \cdot \hat{q}$. Finally, we can define the **kinetic/double-geodetic** logarithm as an approximation of the analytic logarithm similar to the previous section:

$$\log_{kin}\left(q_0 + \tfrac{\varepsilon}{2} \cdot t \cdot q_0\right) := \log_{\mathbb{H}_u}(q_0) + \tfrac{\varepsilon}{2} \cdot t = \left(0, \tfrac{1}{2} \cdot \omega + \tfrac{\varepsilon}{2} \cdot T\right) = \iota(\omega, T) \quad .$$

Concluding, with the help of dual quaternions we can treat Euclidean transformations without the memory overhead of the matrix representation on the projective space $\mathbb{P}^3(\mathbb{R})$. Thus, a time series of Euclidean transformation can be passed to the alignment algorithm as an array of structs (e.g. `float[8]`).

### 5.3.3 Applications in Time Series Data Mining

With the help of the kinetic measure on $SE(3, \mathbb{R})$, we repeat the segmentation task of the car assembly experiment considering all rotational and spatial degrees of freedom. The corresponding stream of dual quaternions is depicted in Figure 5.7. For the sake of simplicity, we choose to set the weighting parameters of the kinetic measure to $\alpha = \beta = 1$. This arbitrary choice should be adjusted with a cross-validation on bigger data sets to ensure a proper mixing of angular and spatial

[14] Note, the (classic) inner action $I_{\hat{q}}(P) = \hat{q} \cdot P \cdot \hat{q}^{-1} = e + \varepsilon \cdot q_0 \cdot p \cdot q_0^{-1}$ only describes pure rotations.

coordinates. In our case, the measure is biased towards the spatial components since the dual part of the quaternions exhibit greater amplitudes than the real part. The resulting segmentation of the stream is visualized in Figure 5.8. Despite the bias towards the spatial components we can recover the correct sequence of labels. As a result, depending on the set of available sensors we can choose to exclusively use angular or spatial coordinates or a combination of both – whatever fits the problem best.

### 5.3.4 From Rigid Bodies to Kinematic Chains (Outlook)

Up to here, our treatment of orientations and displacements did not consider an underlying physical model of the tracked object. The representation in terms of points on $SE(3,\mathbb{R})$ assumes an isotropic mass distribution and thus we only know its center of mass (displacement vector) and its orientation (rotation) in relation to a reference frame. During the motion-tracking of complex systems with multiple sensors it is useful to postulate an underlying physical model to determine suitable notions of distance between individual states in the configuration space. Although we cannot cover the vast field of activity recognition in this concluding section, we aim for a brief outlook on the extension of rigid body systems to more complex manifolds of motion.

Assume a family of $N$ motion sensors attached to different positions of the monitored body. Each sensor delivers displacements vector and orientations over time. The easiest way to assign a notion of distance between two configurations $\alpha, \beta \in SE(3,\mathbb{R})^N$ of $N$ sensors $\alpha := \big((R^{(0)}, T^{(0)}), \dots, (R^{(N-1)}, T^{(N-1)})\big)$ and $\beta := \big((R^{(0)'}, T^{(0)'}), \dots, (R^{(N-1)'}, T^{(N-1)'})\big)$ is given by their $N$-geodetic distance

$$w^{SE(3,\mathbb{R})^N}(\alpha,\beta) := \sum_{k=0}^{N-1} w^{SE(3,\mathbb{R})}\big((R^{(k)}, T^{(k)}), (R^{(k)'}, T^{(k)'})\big) \quad .$$

Analogous to the double-geodetic approach on $SE(3,\mathbb{R})$, a weighting of the individual sensor contributions is conceivable. In this case, the underlying Lie group is simply the direct product of $N$ times the Euclidean group $SE(3,\mathbb{R})^N = SE(3,\mathbb{R}) \times \cdots \times SE(3,\mathbb{R})$. However, this approach ignores possible constraints between the individual sensors. To overcome this, on can model the physical system with a skeleton tree where edges represent constraints and nodes the positions of the motion sensors. In contrast to our naïve $N$-geodetic approach that models only the vertices (positions and orientations), we could extract difference transformations between the individual nodes. The sequence of parameters $(\tau_k)_k$ that is needed for a forward-parametrization of the skeleton $\gamma(\tau_k)$ can now be used to define a distance measure on the parameter space. As an example, if the sequence $(\tau_k)_k$ is given by differences of angles between the connected segments of the skeleton then the weighted sum of squared (difference) angles would be a conceivable choice for the weighting function.

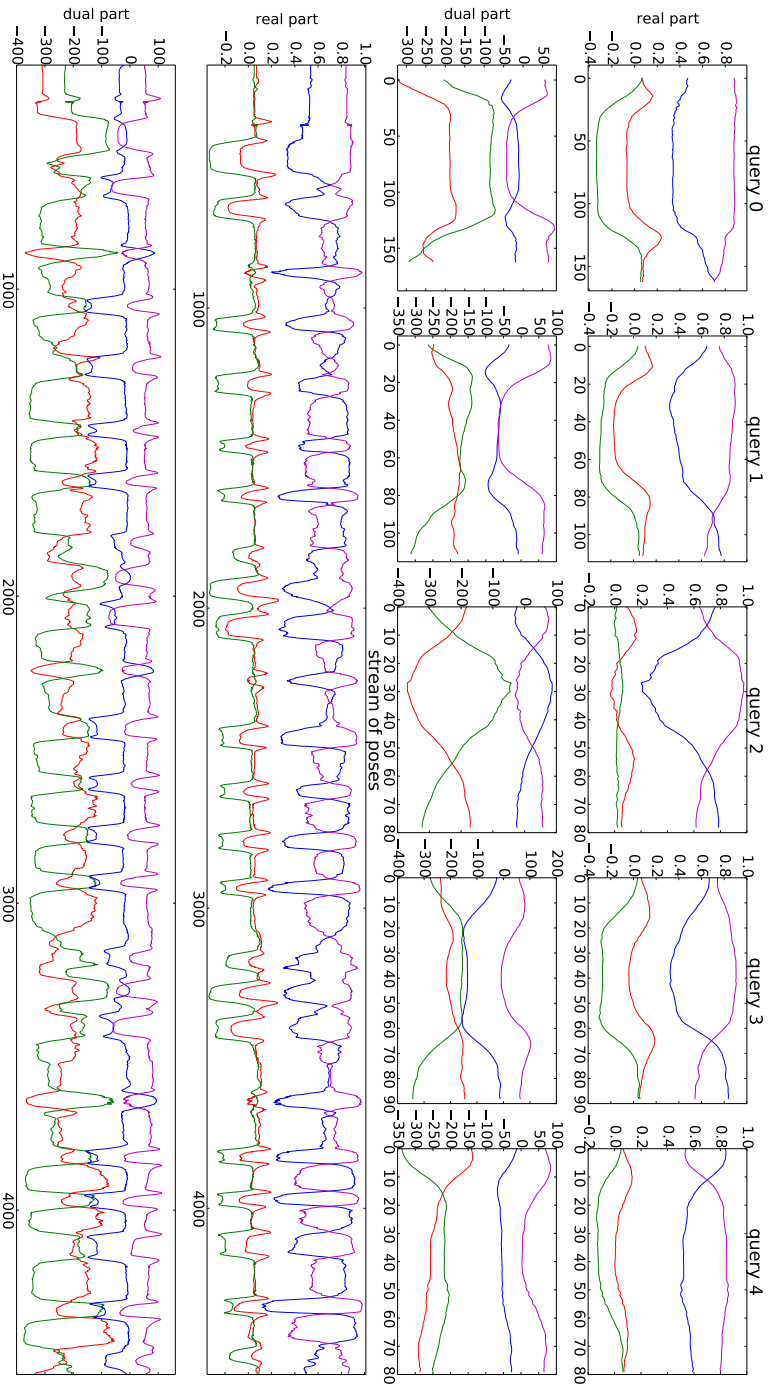


Figure 5.7: Stream of dual quaternions recorded during the assembly task of a LEGO car split into real and dual quaternion components (two panels at the bottom). The four time-dependent coordinate functions of the quaternions are plotted in different color (scalar part: red, x-axis: green, y-axis: blue, z-axis: magenta). The two panels at the top depict the five quaternion-valued queries that correspond to the different assembly steps (same color-coding).

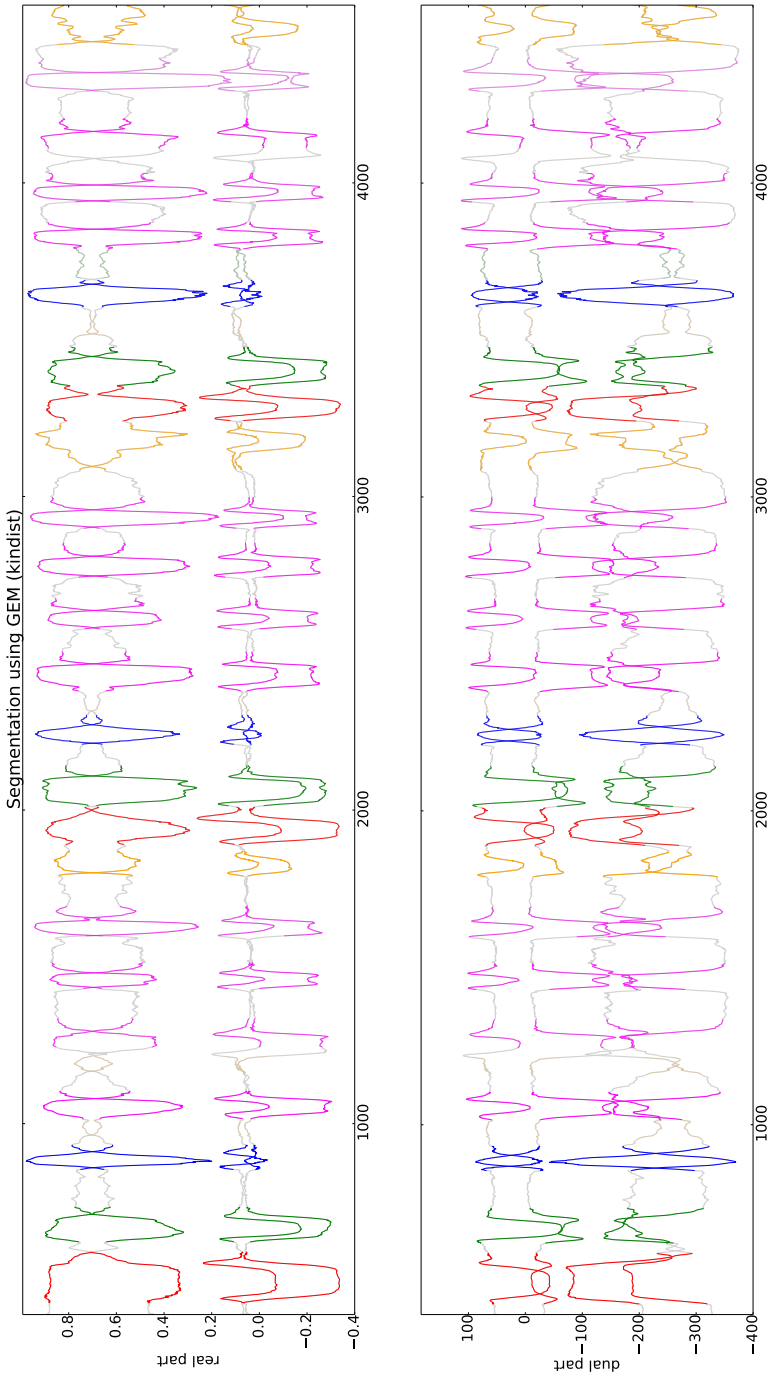


Figure 5.8: Multi-shape segmentation of a quaternion-valued stream recorded with an inertial measurement unit during the assembly of a LEGO car using GEM. The colors encode the label of the shape (chassis=red, axis one=green, axis two=blue, wheel=magenta, deposit=orange).

CHAPTER 6

# CONCLUSION AND FUTURE RESEARCH

*'If it disagrees with experiment it is wrong.'* —
Richard Phillips Feynman

The focus of this thesis is the efficient computation of subsequence alignments in streams of time series. Efficiency can be expressed in different terms. On the one hand, we aimed for high quality alignments respecting invariances of the measured data. On the other hand, the computational effort and memory consumption of the used algorithms had to be taken into account to ensure their applicability in real life. Hence, the research presented in this thesis is split evenly in the theoretical analysis of alignment algorithms **and** their efficient implementation on modern hardware accelerators.

## Chapter 2

We have seen, that Euclidean distance and its lower bounds can be rigidly treated in terms of unitary symmetries of the underlying metric space. This includes a unification of frequently used dimensional reduction techniques and the identification of z-normalized Euclidean distance with the Pearson correlation coefficient. From this point of view, traditional preprocessing steps like principal component analysis, discrete Fourier and wavelets coefficients as well as popular aggregation techniques like piecewise constant approximation and piecewise aggregate approximation of time series can be treated with a unified theoretical framework.

Further, our methodology allows for the construction of novel representations by simply choosing a suitable unitary basis of the underlying vector space.

## Chapter 3

Building upon this result, we have introduced a subsequence alignment algorithm for the computation of local alignment scores under z-normalized Euclidean distance. In contrast to the state-of-the-art implementation of **one**-nearest-neighbour search under ED, namely UCR-ED, we have proposed an analytic expression for the computation of **all** alignment scores that can be easily parallelized on any concurrent accelerator that features a Fast Fourier Transform. The discussed algorithm is superior to UCR-ED in its theoretical worst case complexity and empirically determined runtimes. Contrary to UCR-ED's linear dependency, our algorithm exhibits an immeasurable dependency on the query length and thus can be used to match queries of arbitrary size without a perceivable mitigation in performance. Our CUDA-implementation outperforms UCR-ED by up to two order-of-magnitude especially on long queries. Furthermore, we have provided an SIMT-compliant scheme for the mapping of UCR-Suite's DTW portion to CUDA-enabled devices. Again, speedups of more than one order-of-magnitude are reached in comparison to sequential code. Concluding, we provide highly efficient drop-in replacements for both UCR-ED and UCR-DTW which allows for the faster processing of even bigger streams at exactly the same accuracy.

## Chapter 4

One-nearest-neighbour search under z-normalized DTW is considered the gold standard when mining streams of time series data. Despite the possibility to lower-bound the DTW measure, its quadratic cost in the length of the candidate sequence limits its use on long queries. Firstly, we have provided theoretic and empiric evidence that the lower bound cascade of UCR-Suite degenerates to its worst case complexity for rapidly oscillating and/or long queries. Secondly, we have investigated publicly available linear time approximations of DTW amongst others Greedy DTW and FastDTW. Furthermore, we have provided a highly efficient CUDA-parallelization of Greedy DTW. The gained speedups exceed two orders-of-magnitude in comparison to optimized sequential code. Thirdly, we have introduced the novel elastic subsequence alignment algorithm GEM that interleaves subsequence candidate normalization and the full relaxation of the penalty matrix. We have documented a competitive alignment quality of GEM in comparison to subsequence DTW in terms of one-nearest neighbour classifiers on a broad variety of domains. Additionally, we could prove significant improvement in the classification quality on trended data e.g. thickness profiles of metal coils. Efficient parallelization schemes for GEM on multi-core architectures and CUDA-

enabled devices have been discussed in detail. Again, the CUDA-implementation outperforms UCR-DTW for all tested settings by up to two orders-of-magnitude. Finally, an extension of subsequence measures to supervised multi-shape segmentation of streams has been provided and evaluated on motion-tracking data.

## Chapter 5

Up to this point, all discussed algorithms have been treated to exclusively handle real-valued data. In the final chapter, we have provided a general extension of elastic similarity measures to handle measurements from any compact Lie group. Furthermore, general update rules for the removal of time-dependent trends in the reference frame have explicitly been stated. Additionally, we have discussed memory-efficient representations of orientations and rigid transformations and their notion of distance. Finally, our theoretical ideas have been evaluated on a multi-shape segmentation task using both orientations and displacements of the reference frame.

## Future Research

This thesis focuses on the fast alignment of subsequence candidates in a stream of time series data. Loosely speaking, we have answered the question to what extend a query $Q$ is contained in a stream $S$. Abusing notation, the question can be rephrased: 'Is the element $Q$ contained in the set $S$ i.e. $Q \in S$?'

From a real world perspective, the stream is always provided by the measurement but unfortunately we cannot state the same about the query $Q$. Hence, when limiting our knowledge to the stream $S$ one might ask how to choose a suitable query $Q$. Obviously, it is intractable to consider all subsequence candidates of all lengths or in case of a multi-shape segmentation task all possible combinations of them. Hence, we cannot compute any useful information unless we can provide a good guess for the building blocks of a stream. The complexity of this problem can be tracked down to the fact that time series in contrast to text do not offer a canonical segmentation. Thus, the determination of reasonable shapes for a motif library can be considered a hard task on non-tokenizable data sets. The literature states only few approaches for the unsupervised computation of motif libraries e.g. the inspiring work about unsupervised shapelets by Keogh et al. [6]. The proposed algorithm is limited to databases of time series snippets. To our best knowledge an extension to streams of time series is still lacking.

Assume, we can provide a robust procedure to extract the building blocks of a stream and thus we know its segmentation in terms of dictionary entries. Mathematically speaking, we compute the coefficient representation of the stream in terms of time-local basis vectors similar to a wavelet decomposition. This information gives no additional insight into the underlying structure unless one

can find high confidence rules that relate the building blocks to each other. As an example, we are able to partition the assembly stream of a LEGO car into its building blocks (see Figure 5.6). A future direction of research could be the automatic computation of the assembly rule 'chassis' → 'axis one' → 'axis two' → 'wheel' → 'wheel' → 'wheel' → 'wheel' → 'deposit' **without** the prior knowledge of the building blocks.

Concluding, the ubiquity of built-in sensors e.g. in mobile devices causes a vast amount of unlabelled data. Due to the lack of domain experts, a manual labelling is considered intractable. Thus, the majority of recorded data will be rendered useless unless we can provide novel unsupervised algorithms to reveal the latent structure of the recorded data.

# BIBLIOGRAPHY


[1] "Google Trends query for different scientific fields." [Online]. Available: http://www.google.com/trends

[2] "Wikimedia entry for a tachograph disk:." [Online]. Available: http://commons.wikimedia.org/wiki/File:Tachoscheibe.jpg

[3] D. Lee, J. K. Archibald, R. B. Schoenberger, A. W. Dennis, and D. K. Shiozawa, "Contour Matching for Fish Species Recognition and Migration Monitoring," in *Applications of Computational Intelligence in Biology*, ser. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2008, vol. 122, pp. 183–207.

[4] Y. Chen, B. Hu, E. Keogh, and G. E. Batista, "DTW-D: Time Series Semi-supervised Learning from a Single Example," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '13. New York, NY, USA: ACM, 2013, pp. 383–391. [Online]. Available: http://doi.acm.org/10.1145/2487575.2487633

[5] P. V. C. Hough, "Machine Analysis of Bubble Chamber Pictures," in *International Conference on High Energy Accelerators and Instrumentation*, CERN, 1959.

[6] J. Zakaria, A. Mueen, and E. Keogh, "Clustering Time Series Using Unsupervised-Shapelets," in *Proceedings of the 2012 IEEE 12th International Conference on Data Mining*, ser. ICDM '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 785–794.

[7] E. Keogh, J. Lin, and A. Fu, "HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence," in *Data Mining, Fifth IEEE International Conference on*, nov. 2005, p. 8.

[8] E. Keogh, L. Wei, X. Xi, S.-H. Lee, and M. Vlachos, "LB_Keogh Supports Exact Indexing of Shapes Under Rotation Invariance with Arbitrary Representations and Distance Measures," in *Proceedings of the 32Nd International Conference*

*on Very Large Data Bases*, ser. VLDB '06. VLDB Endowment, 2006, pp. 882–893. [Online]. Available: http://dl.acm.org/citation.cfm?id=1182635.1164203

[9] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 26, no. 1, pp. 43–49, Feb 1978.

[10] L. J. Latecki, V. Megalooikonomou, Q. Wang, R. Lakaemper, C. A. Ratanamahatana, and E. Keogh, "Partial Elastic Matching of Time Series," in *Proceedings of the Fifth IEEE International Conference on Data Mining*, ser. ICDM '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 701–704.

[11] "Big Data Meets Big Data Analytics, SAS whitepaper." [Online]. Available: http://www.sas.com/resources/whitepaper/wp_46345.pdf

[12] M. Banko and E. Brill, "Scaling to Very Very Large Corpora for Natural Language Disambiguation," in *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ser. ACL '01. Stroudsburg, PA, USA: Association for Computational Linguistics, 2001, pp. 26–33. [Online]. Available: http://dx.doi.org/10.3115/1073012.1073017

[13] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures," *Proc. VLDB Endow.*, vol. 1, no. 2, pp. 1542–1552, Aug. 2008.

[14] M. W. Kadous, "Learning Comprehensible Descriptions of Multivariate Time Series," in *In Ivan Bratko and Saso Dzeroski, editors, Proceedings of the 16 th International Conference of Machine Learning (ICML-99.* Morgan Kaufmann, 1999, pp. 454–463.

[15] C. A. Ratanamahatana and E. Keogh, "Making Time-series Classification More Accurate Using Learned Constraints," in *In proc. of SDM Int'l Conf*, 2004, pp. 11–22.

[16] E. Keogh, Q. Zhu, B. Hu, Y. Hao., X. Xi, L. Wei, and C. A. Ratanamahatana, "The UCR Repository," 2011. [Online]. Available: http://www.cs.ucr.edu/~eamonn/time_series_data

[17] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '12. New York, NY, USA: ACM, 2012, pp. 262–270.

[18] Z.-H. Zhou and Y. Yu, "Adapt bagging to nearest neighbor classifiers," *Journal of Computer Science and Technology*, vol. 20, no. 1, pp. 48–54, 2005. [Online]. Available: http://dx.doi.org/10.1007/s11390-005-0005-5

[19] C. Elkan, "Using the Triangle Inequality to Accelerate k-Means." in *ICML*, T. Fawcett and N. Mishra, Eds. AAAI Press, 2003, pp. 147–153. [Online]. Available: http://dblp.uni-trier.de/db/conf/icml/icml2003.html#Elkan03

[20] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti, "Detecting Novel Associations in Large Data Sets," *Science*, vol. 334, no. 6062, pp. 1518–1524, 2011. [Online]. Available: http://www.sciencemag.org/content/334/6062/1518.abstract

[21] "Oxford Learner's Dictionaries: entry for the word 'exact'." [Online]. Available: http://www.oxfordlearnersdictionaries.com/definition/english/exact_1

[22] D. J. Berndt and J. Clifford, "Using Dynamic Time Warping to Find Patterns in Time Series," in *KDD Workshop*, 1994, pp. 359–370.

[23] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009.

[24] E. Keogh, "Exact Indexing of Dynamic Time Warping," in *Proceedings of the 28th International Conference on Very Large Data Bases*, ser. VLDB '02. VLDB Endowment, 2002, pp. 406–417.

[25] S. Salvador and P. Chan, "Toward Accurate Dynamic Time Warping in Linear Time and Space," *Intell. Data Anal.*, vol. 11, no. 5, pp. 561–580, Oct. 2007.

[26] V. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals," *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[27] L. Chen and R. Ng, "On the Marriage of $L_p$-norms and Edit Distance," in *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, ser. VLDB '04. VLDB Endowment, 2004, pp. 792–803.

[28] L. Chen, 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*, ser. SIGMOD '05. New York, NY, USA: ACM, 2005, pp. 491–502.

[29] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh, "Indexing Multi-Dimensional Time-Series with Support for Multiple Distance Measures," in *Proceedings of the ninth ACM SIGKDD international conference on*

*Knowledge discovery and data mining*, ser. KDD '03. New York, NY, USA: ACM, 2003, pp. 216–225.

[30] S. B. Needleman and C. D. Wunsch, "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins." *Journal of molecular biology*, vol. 48, no. 3, pp. 443–453, Mar. 1970.

[31] T. F. Smith and M. S. Waterman, "Identification of Common Molecular Subsequences." *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, Mar. 1981.

[32] G. E. Batista, E. J. Keogh, O. M. Tataw, and V. M. Souza, "CID: An Efficient Complexity-invariant Distance for Time Series," *Data Min. Knowl. Discov.*, vol. 28, no. 3, pp. 634–669, May 2014.

[33] Y. Hao, M. Shokoohi-Yekta, G. Papageorgiou, and E. Keogh, "Parameter-Free Audio Motif Discovery in Large Data Archives," *2013 IEEE 13th International Conference on Data Mining*, vol. 0, pp. 261–270, 2013.

[34] E. J. Keogh and M. J. Pazzani, "Derivative Dynamic Time Warping," in *In First SIAM International Conference on Data Mining (SDM'2001*, 2001.

[35] P. Tormene, T. Giorgino, S. Quaglini, and M. Stefanelli, "Matching Incomplete Time Series with Dynamic Time Warping: An Algorithm and an Application to Post-stroke Rehabilitation," *Artif. Intell. Med.*, vol. 45, no. 1, pp. 11–34, jan 2009. [Online]. Available: http://dx.doi.org/10.1016/j.artmed.2008.11.007

[36] S. Spiegel, B.-J. Jain, and S. Albayrak, "Fast Time Series Classification Under Lucky Time Warping Distance," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, ser. SAC '14. New York, NY, USA: ACM, 2014, pp. 71–78. [Online]. Available: http://doi.acm.org/10.1145/2554850.2554885

[37] C. Ding and X. He, "K-means Clustering via Principal Component Analysis," in *Proceedings of the Twenty-first International Conference on Machine Learning*, ser. ICML '04. New York, NY, USA: ACM, 2004, pp. 29–. [Online]. Available: http://doi.acm.org/10.1145/1015330.1015408

[38] A. Haar, "Zur Theorie der orthogonalen Funktionensysteme," *Mathematische Annalen*, vol. 69, no. 3, pp. 331–371, 1910. [Online]. Available: http://dx.doi.org/10.1007/BF01456326

[39] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *Journal of Fourier Analysis and Applications*, vol. 4, no. 3, pp. 247–269, 1998. [Online]. Available: http://dx.doi.org/10.1007/BF02476026

[40] I. Daubechies, *Ten Lectures on Wavelets*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1992.

[41] J. m. Lina, "Image Processing with Complex Daubechies Wavelets," *Journal of Mathematical Imaging and Vision*, vol. 7, pp. 211–223, 1996.

[42] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases," *Knowledge and Information Systems*, vol. 3, no. 3, pp. 263–286, 2001. [Online]. Available: http://dx.doi.org/10.1007/PL00011669

[43] S. Mallat, *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*, 3rd ed. Academic Press, 2008.

[44] J. Cooley and J. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.

[45] M. Heideman, D. Johnson, and C. Burrus, "Gauss and the history of the fast Fourier transform," *Archive for History of Exact Sciences*, vol. 34, no. 3, pp. 265–277, 1985. [Online]. Available: http://dx.doi.org/10.1007/BF00348431

[46] A. Guttman, "R-trees: A Dynamic Index Structure for Spatial Searching," in *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '84. New York, NY, USA: ACM, 1984, pp. 47–57. [Online]. Available: http://doi.acm.org/10.1145/602259.602266

[47] J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975. [Online]. Available: http://doi.acm.org/10.1145/361002.361007

[48] A. Hinneburg, C. C. Aggarwal, and D. A. Keim, "What Is the Nearest Neighbor in High Dimensional Spaces?" in *Proceedings of the 26th International Conference on Very Large Data Bases*, ser. VLDB '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 506–515. [Online]. Available: http://dl.acm.org/citation.cfm?id=645926.671675

[49] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh, "Indexing Multi-dimensional Time-series with Support for Multiple Distance Measures," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '03. New York, NY, USA: ACM, 2003, pp. 216–225. [Online]. Available: http://doi.acm.org/10.1145/956750.956777

[50] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000 (June 13), circulation Electronic Pages: http://circ.ahajournals.org/cgi/content/full/101/23/e215 PMID:1085218; doi: 10.1161/01.CIR.101.23.e215.

[51] M. Cuturi, "Fast Global Alignment Kernels." in *ICML*, L. Getoor and T. Scheffer, Eds. Omnipress, 2011, pp. 929–936. [Online]. Available: http://dblp.uni-trier.de/db/conf/icml/icml2011.html#Cuturi11

[52] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel K-means: Spectral Clustering and Normalized Cuts," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '04. New York, NY, USA: ACM, 2004, pp. 551–556. [Online]. Available: http://doi.acm.org/10.1145/1014052.1014118

[53] Y. Sakurai, C. Faloutsos, and M. Yamamuro, "Stream monitoring under the time warping distance," in *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, 2007, pp. 1046–1055.

[54] T. F. Chan, G. H. Golub, and R. J. Leveque, "Algorithms for Computing the Sample Variance: Analysis and Recommendations," *The American Statistician*, vol. 37, no. 3, pp. 242–247, 1983. [Online]. Available: http://dx.doi.org/10.2307/2683386

[55] B.-K. Yi, H. Jagadish, and C. Faloutsos, "Efficient retrieval of similar time sequences under time warping," in *Data Engineering, 1998. Proceedings., 14th International Conference on*, Feb 1998, pp. 201–208.

[56] S.-W. Kim, S. Park, and W. Chu, "An index-based approach for similarity search supporting time warping in large sequence databases," in *Data Engineering, 2001. Proceedings. 17th International Conference on*, 2001, pp. 607–614.

[57] D. Lemire, "Streaming Maximum-minimum Filter Using No More Than Three Comparisons Per Element," *Nordic J. of Computing*, vol. 13, no. 4, pp. 328–339, Dec. 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1324123.1324129

[58] ——, "Faster Retrieval with a Two-pass Dynamic-time-warping Lower Bound," *Pattern Recogn.*, vol. 42, no. 9, pp. 2169–2180, Sep. 2009. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2008.11.030

[59] R. Markus and B. Hendryk, "Experiences with Intel Xeon Phi in the Max-Planck Society," in *ENES Workshop on Exascale Technologies & Innovation in HPC for Climate Models*, 2014. [Online]. Available: http://web.ipac.caltech.edu/staff/fmasci/home/miscscience/MIC_benchmarking_2013.pdf

[60] "CUDA C Programming Guide version 6.0." [Online]. Available: http://docs.nvidia.com/cuda/cuda-c-programming- guide

[61] "NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK110, NVIDIA whitepaper." [Online]. Available: http://www.nvidia.de/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf

[62] V. Volkov, "Better Performance at Lower Occupancy," in *GPU Technology Conference*, 2010. [Online]. Available: http://www.cs.berkeley.edu/~volkov/volkov10-GTC.pdf

[63] "cuBLAS." [Online]. Available: https://developer.nvidia.com/cublas

[64] "cuFFT." [Online]. Available: https://developer.nvidia.com/cuFFT

[65] C. Hundt, B. Schmidt, and E. Schömer, "CUDA-Accelerated Alignment of Subsequences in Streamed Time Series Data," in *Parallel Processing (ICPP), 2014 43rd International Conference on*, Sept 2014, pp. 10–19.

[66] "MKL." [Online]. Available: http://software.intel.com/en-us/intel-mkl

[67] Y. Liu, D. Maskell, and B. Schmidt, "CUDASW++: Optimizing Smith-Waterman Sequence Database Searches for CUDA-Enabled Graphics Processing Units," *BMC Research Notes*, vol. 2, no. 1, pp. 1–10, 2009.

[68] "supplementary webpage for the UCR-Suite." [Online]. Available: http://www.cs.ucr.edu/ eamonn/UCRsuite.html

[69] "supplementary webpage for CUDA-DTW." [Online]. Available: http://gravitino.github.io/cudadtw/

[70] S. MacLean and G. Labahn, "Elastic matching in linear time and constant space," in *International Workshop on Document Analysis Systems*, June 2010. [Online]. Available: https://cs.uwaterloo.ca/~smaclean/elastic-matching.pdf

[71] S. Spiegel, B.-J. Jain, and S. Albayrak, "Fast Time Series Classification Under Lucky Time Warping Distance," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, ser. SAC '14. New York, NY, USA: ACM, 2014, pp. 71–78.

[72] C. Hundt, B. Schmidt, E. Schömer, H. Göttler, and H.-V. Dang, "GEM: An Elastic and Translation-invariant Similarity Measure with Automatic Trend Adjustment," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, ser. SAC '14. New York, NY, USA: ACM, 2014, pp. 105–112. [Online]. Available: http://doi.acm.org/10.1145/2554850.2555041

[73] T. Abeel, Y. Van de Peer, and Y. Saeys, "Java-ML: A Machine Learning Library," *J. Mach. Learn. Res.*, vol. 10, pp. 931–934, Jun. 2009.

[74] T. Giorgino, "Computing and Visualizing Dynamic Time Warping Alignments in R: The dtw Package," *Journal of Statistical Software*, vol. 31, no. 7, pp. 1–24, 8 2009. [Online]. Available: http://www.jstatsoft.org/v31/i07

[75] D. Albanese, R. Visintainer, S. Merler, S. Riccadonna, G. Jurman, and C. Furlanello, "mlpy: Machine Learning Python," 2012.

[76] D. Sart, A. Mueen, W. Najjar, E. Keogh, and V. Niennattrakul, "Accelerating Dynamic Time Warping Subsequence Search with GPUs and FPGAs," in *Proceedings of the 2010 IEEE International Conference on Data Mining*, ser. ICDM '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 1001–1006.

[77] E. Keogh and S. Kasetty, "On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration," *Data Min. Knowl. Discov.*, vol. 7, no. 4, pp. 349–371, Oct. 2003. [Online]. Available: http://dx.doi.org/10.1023/A:1024988512476

[78] F. Klinker, "Exponential moving average versus moving exponential average," *Mathematische Semesterberichte*, vol. 58, pp. 97–107, 2011.

[79] "supplementary webpage for the Gliding Elastic Match." [Online]. Available: http://gravitino.github.io/gem/

[80] J. M. Lee, *"Introduction to Smooth Manifolds"*, ser. Graduate texts in mathematics. New York, Berlin, Heidelberg: Springer, 2003. [Online]. Available: http://opac.inria.fr/record=b1099851

[81] F. Jung, "Canonical Group Quantization and Boundary Conditions," 2012. [Online]. Available: http://ubm.opus.hbz-nrw.de/volltexte/2012/3212/index.html

[82] F. Bullo and R. M. Murray, "Proportional Derivative (PD) Control On The Euclidean Group," in *In European Control Conference*, 1995, pp. 1091–1097.

[83] B. Kenwright, "A Beginners Guide to Dual-Quaternions: What They Are, How They Work, and How to Use Them for 3D Character Hierarchies," in *The 20th International Conference on Computer Graphics, Visualization and Computer Vision*, 2012, pp. 1–13, wSCG 2012 Communication Proceedings, Conference June. 2012. [Online]. Available: http://wscg.zcu.cz/wscg2012/short/A29-full.pdf

[84] E. Zeidler, *Quantum Field Theory I: Basics in Mathematics and Physics: A Bridge between Mathematicians and Physicists*, ser. Quantum Field Theory. Springer, 2007.

[85] L. Kavan, S. Collins, C. O'Sullivan, and J. Zara, "Dual Quaternions for Rigid Transformation Blending," *Technical report, Trinity College Dublin*, 2006.

# Lebenslauf

## Zur Person

| | |
|---|---|
| Name | Christian Hundt |
| Anschrift | Kurmainzstraße 21, 55126 Mainz |
| Geburtstag | 06.07.1983 |
| Geburtsort | Blankenburg (Harz) |
| Staatsangehörigkeit | deutsch |
| Familienstand | ledig |
| Kinder | keine |

## Schulbildung

| | |
|---|---|
| 08/1990-07/1994 | Grundschule ‘auf den Höhen’ Thale |
| 08/1994-07/2003 | Europagymnasium ‘Richard von Weizsäcker’ Thale<br>Abschluss: allgemeines Abitur |

## Zivildienst

| | |
|---|---|
| 08/2003-07/2004 | Zivildienst ‘Freie Schule’ Thale |

## Hochschulstudium

| | |
|---|---|
| 10/2004-05/2010 | Johannes Gutenberg-Universität Mainz<br>Abschluss: Diplom im Fach Physik |
| Schwerpunkte | Gruppentheoretische Aspekte der Physik,<br>Quantisierung auf gekrümmten Mannigfaltigkeiten,<br>Theoretische Elementarteilchenphysik |

## Universitäre Tätigkeit

| | |
|---|---|
| Oberassistenz | Mathematik für Informatiker,<br>Datenstrukturen und effiziente Algorithmen,<br>Einführung in die Programmierung,<br>High Performance Computing,<br>Parallele Architekturen und Algorithmen |
| Seminare/Kurse | Programmierpraktikum für Lehramtskandidaten,<br>Seminar: Learn Linux the Hard Way,<br>Programmierkurs: Visualisierung mit Python |

## Sonstiges

| | |
|---|---|
| Fremdsprachen | Englisch, Französisch, Latein |