Data Mining a Trillion Time Series Subsequences Under Dynamic Time Warping*

Thanawin Rakthanmanon[1], Bilson Campana, Abdullah Mueen, Gustavo Batista[2], Brandon Westover[3], Qiang Zhu, Jesin Zakaria, Eamonn Keogh
UC Riverside [1]Kasetsart University [2]University of São Paulo [3]Brigham and Women's Hospital
{eamonn, bcampana, mueen, qzhu, jzaka}@cs.ucr.edu, thanawin.r@ku.ac.th, gbatista@icmc.usp.br

Abstract

Most time series data mining algorithms use similarity search as a core subroutine, and thus the time taken for similarity search is the bottleneck for virtually all time series data mining algorithms. The difficulty of scaling search to large datasets largely explains why most academic work on time series data mining has plateaued at considering a few millions of time series objects, while much of industry and science sits on billions of time series objects waiting to be explored. In this work we show that by using a combination of four novel ideas we can search and mine truly massive time series for the first time. We demonstrate the following extremely unintuitive fact; in large datasets we can exactly search under DTW much more quickly than the current state-of-the-art Euclidean distance search algorithms. We demonstrate our work on the largest set of time series experiments ever attempted. We show that our ideas allow us to solve higher-level time series data mining problems at scales that would otherwise be untenable.

1 Introduction

Most time series data mining algorithms require similarity comparisons as a subroutine, and in spite of the consideration of dozens of alternatives, there is increasing evidence that the classic Dynamic Time Warping (DTW) measure is the best measure in most domains [Ding *et al.*, 2008]. It is difficult to overstate the ubiquity of DTW. It has been used in robotics, medicine, biometrics, music/speech processing, climatology, aviation, gesture recognition, user interfaces, industrial processing, geology, astronomy, space exploration, wildlife monitoring, etc.

As ubiquitous as DTW is, we believe that there are thousands of research efforts that would like to use DTW, but find it too computationally expensive. For example, consider the following: "*Ideally, dynamic time warping would be used to achieve this, but due to time constraints…*" [Chadwick *et al.*, 2011] and [Adams *et al.*, 2005] notes, even "*a 30 fold speed increase may not be sufficient for scaling DTW methods to truly massive databases.*" As we shall show, our subsequence search (called the *UCR suite*) removes all of these objections. We can reproduce all of the experiments in all of these papers in well under a second.

We make an additional claim for our UCR suite which is almost certainly true, but very hard to prove, given the variability in how search results are presented in the literature. We believe our exact DTW sequential search is much faster than any current *approximate* search or exact *indexed* search. In a handful of papers the authors *are* explicit enough with their experiments to see this is true. Consider [Papapetrou *et al.*, 2011], in which the authors introduce a technique that can answer queries of length 1,000 under DTW with 95% accuracy, in a random walk dataset of one million objects in 5.65 seconds. We can *exactly* search this dataset in 3.8 seconds (on a very similar machine). An influential paper on gesture recognition on multi-touch screens laments that "*DTW took 128.26 minutes to run the 14,400 tests for a given subject's 160 gestures*" [Wobbrock *et al.*, 2007]. However, we can reproduce these results in less than three seconds.

1.1 A Brief Discussion of a Trillion

In this work, we search a trillion (one million million, or 10^{12}, or 1,000,000,000,000) objects and, to our knowledge, such a large dataset has never been considered in a data mining/database paper before.

As large as a trillion is, there are thousands of research labs and commercial enterprises that have this much data. For example, many research hospitals have trillions of datapoints of EEG data, NASA Ames has tens of trillions of datapoints of telemetry of domestic flights, etc.

1.2 Explicit Statement of Our Assumptions

Our work is predicated on several assumptions that we will now enumerate and justify.

Time Series Subsequences Must be Normalized

In order to make meaningful comparisons between two time series, both must be normalized. This was demonstrated a

* The paper on which this extended abstract is based was the recipient of the best paper award of the SIGKDD 2012 [Rakthanmanon *et al.*, 2012].

decade ago in a widely cited paper [Keogh and Kasetty, 2003]. This is critical because some speedup techniques *only* work on the un-normalized data; thus, the contributions of these research efforts may be largely nullified.

Dynamic Time Warping is the Best Measure
Recent empirical evidence strongly suggests that none of alternative measures routinely beats DTW. When put to the test on a collection of forty datasets, the very *best* of these measures are sometimes a little better than DTW and sometimes a little worse [Ding *et al.*, 2008]. After an exhaustive literature search of more than 800 papers, we are not aware of any distance measure that has been shown to outperform DTW by a statistically significant amount on reproducible experiments [Ding *et al.*, 2008; Keogh and Kasetty, 2003]. Thus, DTW is *the* measure to optimize.

Arbitrary Query Lengths cannot be Indexed
If we know the length of queries ahead of time we can mitigate at least some of the intractability of search by indexing the data [Fu *et al.*, 2008;]. Although to our knowledge no one has built an index for a trillion real-valued objects (Google only indexed a trillion webpages as recently as 2008), perhaps this could be done. However, there are no known techniques to support similarity search of arbitrary lengths once we have datasets in the billions.

There Exists Data Mining Problems That We Are Willing to Wait Some Hours to Answer
This point is almost self-evident. If a team of entomologists has spent three years gathering 0.2 trillion datapoints, or astronomers have spent billions of dollars to launch a satellite to collect one trillion datapoints of star-light curve data per day [Keogh *et al.*, 2009], or a hospital charges $34,000 for a daylong EEG session to collect 0.3 trillion datapoints. Then, it is not unreasonable to expect that these groups would be willing to spend hours of CPU time to glean knowledge from their data.

2 Related work

Our review of related work on time series indexing is necessarily superficial, given the vast amount of work on the topic and page limits. Instead, we refer the interested reader to two recent papers [Ding *et al.*, 2008; Papapetrou *et al.*, 2011], which have comprehensive reviews of existing work. We are interested in datasets that are five to six orders of magnitude larger than anything else considered in the literature [Ding *et al.*, 2008].

3 Background and Notations

Definition: The Euclidean distance (ED) between subsequences Q and C, where $|Q| = |C|$, is defined as $ED(Q,C) = \sqrt{\sum_{i=1}^{n}(q_i - c_i)^2}$

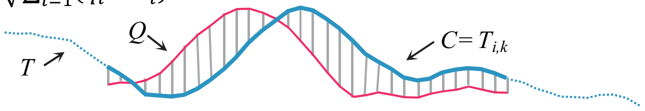
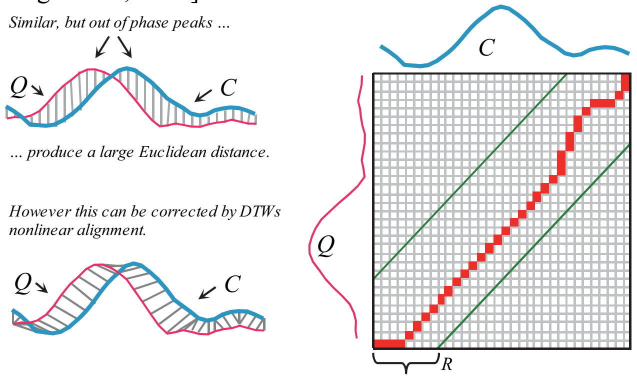


Figure 1. A long time series T can have a subsequence C extracted and compared to a query Q under the Euclidean distance.

The Euclidean distance, as shown in Figure 1, which is a one-to-one mapping of the two sequences, can be seen as a special case of DTW, which allows a one-to-many alignment, as illustrated in Figure 2. For brevity we ask unfamiliar readers to refer to [Fu *et al.*, 2008; Ding *et al.*, 2008; Keogh *et al.*, 2009] for more detail of DTW.



Similar, but out of phase peaks …
Q
C
… produce a large Euclidean distance.
However this can be corrected by DTWs nonlinear alignment.
Q
C
C
Q
R

Figure 2. *left*) Two time series which are similar but out of phase. *right*) To align the sequences we construct a warping matrix, and search for the optimal warping path (red/solid squares). Sakoe-Chiba Band with width R is used to constrain the warping path.

4 Algorithms

4.1 Known Optimizations

We begin by discussing previously known optimizations of sequential search under ED and/or DTW.

Using the Squared Distance
Both DTW and ED have a square root calculation. However, if we omit this step, it does not change the relative rankings of nearest neighbors, since both functions are monotonic and concave.

Lower Bounding
A classic trick to speed up sequential search with an expensive distance measure such as DTW is to use a cheap-to-compute lower bound to prune off unpromising candidates. The LB_{Keogh} bound is well-documented elsewhere, for brevity we ask the unfamiliar reader to refer to [Fu *et al.*, 2008; Keogh *et al.*, 2009] for a review.

Early Abandoning of ED and LB_Keogh
During the computation of the Euclidean distance or the LB_{Keogh} lower bound, if we note that the current sum of the squared differences between each pair of corresponding datapoints exceeds the *best-so-far*, then we can stop the calculation, secure in the knowledge that the distance had we calculated it, would have exceeded the *best-so-far*.

Early Abandoning of DTW
We can incrementally compute the DTW and admissibly stop if the minimum distance at the any cut exceeds the *best-so-far* distance.

Exploiting Multicores
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.

4.2 Novel Optimizations: The UCR Suite

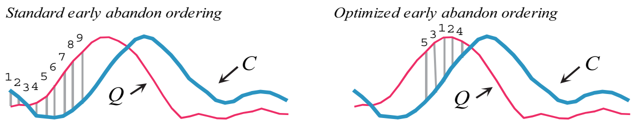
We are finally in a position to introduce our four original optimizations of search under ED and/or DTW.

Early Abandoning Z-Normalization

To the best of our knowledge, no one has ever considered optimizing the *normalization* step. Our insight here is that we can interleave the early abandoning calculations of Euclidean distance (or *LB_Keogh*) with the online Z-normalization. The mean and standard deviation of a stream of numbers can be incrementally calculated and maintained. Thus, we could be pruning not just distance calculation steps, but also unnecessary *normalization* steps.

Reordering Early Abandoning

Consider Figure 3.*left*, which shows the normal left-to-right ordering in which the early abandoning calculation proceeds. In this case *nine* of the thirty-two calculations were performed before the accumulated distance exceeded *b* and we could abandon. In contrast, Figure 3.*right* uses a different ordering and was able to abandon earlier, with just *five* of the thirty-two calculations.



Standard early abandon ordering
Optimized early abandon ordering
1 2 3 4 5 6 7 8 9
5 3 1 2 4
Q
C

Figure 3. *left*) ED early abandoning. We have a best-so-far value of b. After incrementally summing the first nine individual contributions to the ED, we have exceeded b; thus, we abandon the calculation. *right*) A different ordering allows us to abandon after just five calculations.

The proof to show that the *universal* optimal ordering is to sort the indices based on the absolute values of the Z-normalized *Q* is available at [UCRsuite, 2012].

Reversing the Query/Data Role in LB_Keogh

Normally the *LB_Keogh* lower bound builds the envelope around the *query*, a situation we denote *LB_KeoghEQ* for concreteness, and illustrate in Figure 4.*left*. If we built the envelope around each *candidate* too, a situation we denote *LB_KeoghEC*. This only needs to be done once, and thus saves the time but *triples* space overhead.

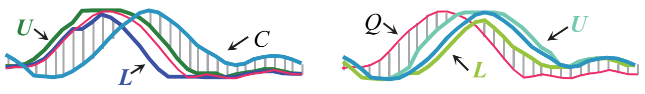


Figure 4. *left*) Normally the *LB_Keogh* envelope is built around the query *Q*, and the distance between *C* and the closer of {*U*,*L*} acts as a lower bound. *right*) However, we can reverse the roles such that the envelope is built around *C* and the distance between *Q* and the closer of {*U*,*L*} is the lower bound.

However, we can selectively calculate *LB_KeoghEC* in a "just-in-time" fashion, *only* if all other lower bounds fail to prune. This removes *space* overhead, and as we will see, the *time* overhead pays for itself by pruning more full DTW calculations. Note that in general, *LB_KeoghEQ* ≠ *LB_KeoghEC* and that on average each one is larger about half the time.

Cascading Lower Bounds

One of the most useful ways to speed up time series similarity search is the use of lower bounds to admissibly prune off unpromising. This has led to a flurry of research on lower bounds, with at least eighteen proposed for DTW [Ding *et al.*, 2008; Keogh *et al.*, 2009; Kim *et al.*, 2001; Yi *et al.* 1998]. In general, it is difficult to state definitively which is the best bound to use, since there is a tradeoff between the tightness of the lower bound and how fast it is to compute. Moreover, different datasets and even different queries can produce slightly different results.

However, as a starting point, we implemented all published lower bounds and tested them on fifty different datasets from the UCR archive, plotting the (slightly idealized for visual clarity) results in Figure 5. Following the literature, we measured the *tightness* of each lower bound as *LB*(A,B)/*DTW*(A,B) over 100,000 randomly sampled subsequences A and B of length 256.

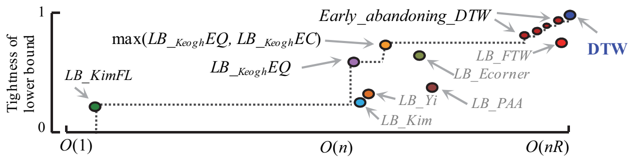


Figure 5. The mean tightness of selected lower bounds from the literature plotted against the time taken to compute them.

The reader will appreciate that a *necessary* condition for a lower bound to be useful is for it to appear on the "skyline" shown with a dashed line; otherwise there exists a faster-to-compute bound that is at least as tight, and we should use that instead. Using this technique we can prune more than 99.9999% of DTW calculations for a large-scale search.

5 Experimental Results

To ensure our experiments are reproducible, all data and code will be available at [UCRsuite, 2012]. We consider the following methods:

- **Naive**: Each subsequence is Z-normalized from scratch. The full DTW (or ED) is used at each step.
- **State-of-the-art (SOTA)**: Each sequence is Z-normalized from scratch, early abandoning is used, and the *LB_Keogh* lower bound is used for DTW.
- **UCR Suite**: We use all of our speedup techniques.

DTW uses *R* = 5% unless otherwise noted. For experiments where Naive or SOTA takes more than 24 hours to finish, we terminate the experiments and present the interpolated values, shown in gray. Where appropriate we also compare to an oracle algorithm:

- **GOd's ALgorithm** (GOAL) is an algorithm that only *maintains* the mean and standard deviation using the online *O*(1) incremental calculations.

It is critical to note that our implementations of Naive, SOTA and GOAL are incredibly efficient and tightly optimized. In particular, the code for Naive, SOTA and GOAL is exactly the same code as the UCR suite, except the relevant speedup techniques have been commented out.

While very detailed spreadsheets of all of our results are archived in perpetuity in the supporting webpage. We present subsets of some results below by considering wall clock time on a 2 Intel Xeon Quad-Core E5620 2.40GHz with 12GB DDR3 RAM (using just one core unless otherwise explicitly stated).

5.1 Baseline Tests on Random Walk

We begin with experiments on random walk data. In Table 1 we show the length of time it takes to search large datasets with queries of length 128. The numbers are averaged over 1000, 100 and 10 queries, respectively.

Table 1. Time taken to search a random walk dataset with |Q| =128.

	Million (*Seconds*)	Billion (*Minutes*)	Trillion (*Hours*)
UCR-ED	0.034	0.22	3.16
SOTA-ED	0.243	2.40	39.80
UCR-DTW	0.159	1.83	34.09
SOTA-DTW	2.447	38.14	472.80

These results show a significant difference between SOTA and UCR suite. However, this is for a very short query; what happens if we consider longer queries? As we show in Figure 6, the ratio of SOTA-DTW over UCR-DTW *improves* for longer queries. Remarkably, UCR-DTW is even faster than SOTA *Euclidean* distance. Even though 4,096 is longer than any published query lengths in the literature, there is a need for even *longer* queries.

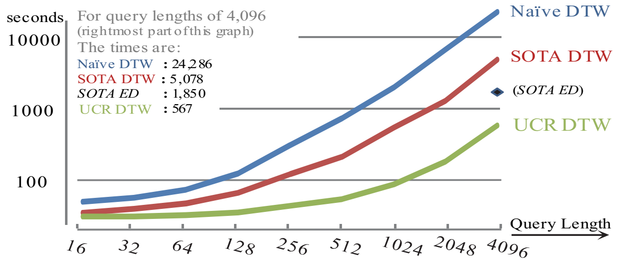


Figure 6. The time taken to search random walks of length 20 million with increasingly long queries, for three variants of DTW. In addition, we include just length 4,096 with SOTA-ED for reference.

It is also interesting to consider the results of the 128-length DTW queries as a ratio over GOAL. Recall that the cost for GOAL is independent of query length, and this experiment is just 23.57 seconds. The ratios for Naive, SOTA and UCR suite are 5.27, 2.74 and 1.41, respectively. This suggests that we are asymptomatically closing in on the fastest possible subsequence search algorithm for DTW. Another interesting ratio to consider is the time for UCR-DTW over UCR-ED, which is just 1.18. Thus, the time for DTW is not significantly different than that for ED, an idea which contradicts an assumption made by almost all papers on time series in the last decade.

Because the space limitation, we encourage the readers to see more interesting results at [UCRsuite, 2012] or in the original version of the paper [Rakthanmanon *et al.*, 2012].

5.2 Speeding up Existing Mining Algorithms

In this section, we demonstrate that we can speed up much of the code in the time series data mining literature with minimal effort, simply by replacing their distance calculation subroutines with the UCR suite. In many cases, the difference is small, because the algorithms in question already typically try to prune as many distance calculations as possible. Nevertheless, even though the speedups are relatively small (1.5X to 16X), they are "free", requiring just minutes of cut-and-paste code editing.

Time Series Shapelets have garnered significant interest since their introduction in 2009 [Ye and Keogh, 2009]. We obtained the original code and tested it on the Face (four) dataset, finding it took 18.9 minutes to finish. After replacing the similarity search routine with the UCR suite, it took 12.5 minutes to finish.

Online Time Series Motifs generalize the idea of mining repeated patterns in a batch time series to the streaming case [Mueen and Keogh, 2010]. We obtained the original code and tested it on the EEG dataset used in the original paper. The fastest running time for the code assuming linear space is 436 seconds. After replacing the distance function with the UCR suite, it took just 156 seconds.

Classification of Ancient Coins [Huber-Mörk *et al.*, 2011]. 2,400 irregularly shaped coins are converted to time series of length 256, and rotation-invariant DTW is used to search the database, taking 12.8 seconds per query. Using the UCR suite, this takes 0.8 seconds per query.

Clustering of Star Light Curves [Keogh *et al.*, 2009] is an important problem in astronomy, as it can be a preprocessing step in outlier detection. We consider a dataset with 1,000 (purportedly) phase-aligned light curves of length 1,024, whose class has been determined by an expert [Rebbapragada *et al.*, 2009]. Doing spectral clustering with DTW (R=5%) takes about 23 minutes for all algorithms, and averaged over 100 runs we find the Rand-Index is 0.62. As we do not trust the original claim of phase alignment, we further do *rotation-invariant* DTW that dramatically increases the Rand-Index to 0.76. Using SOTA, this takes 16.57 days, but if we use the UCR suite, this time falls by an order of magnitude, to just 1.47 days on a single core.

6 Discussion and Conclusions

While our work has focused on fast *sequential search*, we believe that for DTW, our work is faster than all known *indexing* efforts. We also have made a strong and unintuitive claim in the abstract. We said that our UCR-*DTW* is faster than all current *Euclidean* distance searches.

Thus, the contributions of this paper are twofold. First, we have shown that much of the recent pessimism about using DTW for real-time problems was simply unwarranted. If carefully implemented, *existing* techniques, especially lower bounding, can make DTW tractable for many problems. Our second contribution is the introduction of the UCR suite of techniques that make DTW and Euclidean distance subsequence search significantly faster than current state-of-the-art techniques. We have avoided presenting full pseudo-code to enhance the readability of the text; however, full pseudo-code (and highly useable source-code) *is* readily available at [UCRsuite, 2012].

Acknowledgements

We would like to acknowledge the financial support for our research provided by NSF grants 0803410 and 0808770, FAPESP 2009/06349-0, and Royal Thai Gov. Scholarship.

References

[Adams *et at.*, 2005] Adams, N., Marquez, D., and Wakefield, G. 2005. Iterative deepening for melody alignment and retrieval. *ISMIR*, 199-206.

[Alon *et al.*, 2009] Alon, J., Athitsos, V., Yuan, Q., and Sclaroff, S. 2009. A unified framework for gesture recognition and spatiotemporal gesture segmentation. *IEEE PAMI* 31, 9, 1685-1699.

[Chadwick *et al.*, 2011] Chadwick, N. A., McMeekin, D. A., and Tan, T. 2011. Classifying eye and head movement atifacts in EEG Signals. *IEEE DEST,* 285-291.

[Ding *et al.*, 2008] Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., and Keogh, E. J. 2008. Querying and mining of time series data: experimental comparison of representations and distance measures. *PVLDB* 1, 2.

[Chen *et al.*, 2009] Chen, Y., Chen, G., Chen, K., and Ooi, B. C. 2009. Efficient processing of warping time series join of motion capture data. *ICDE*, 1048-1059.

[Fornés *et al.*, 2007] Fornés, A., Lladós, J., and Sanchez, G. 2007. Old handwritten musical symbol classification by a dynamic time warping based method. *Graphics Recognition* 5046, 51-60.

[Fu *et al.*, 2008] Fu, A., Keogh, E. J., Lau, L., Ratanamahatana, C., and Wong., R. 2008. Scaling and time warping in time series querying. *VLDB J.* 17, 4.

[Gillian *et al.*, 2011] Gillian, N., Knapp, R., and O'Modhrain, S. 2011. Recognition of multivariate temporal musical gestures using n-dimensional dynamic time warping. *Proc of the 11th Int'l conference on New Interfaces for Musical Expression*.

[Goldberg, 1991] Goldberg, D. 1991. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys* 23, 1.

[Hsiao *et al.*, 2005] Hsiao, M., West, K., and Vedatesh, G. 2005. Online context recognition in multisensor system using dynamic time warping. *ISSNIP*, 283-288.

[Kahveci and Singh, 2004] Kahveci, T., and Singh, A. K. 2004. Optimizing similarity search for arbitrary length time series queries. *IEEE Trans. Knowl. Data Eng.* 16, 4.

[Keogh and Kasetty, 2003] Keogh, E. J., and Kasetty, S. 2003. On the need for time series data mining benchmarks: a survey and empirical demonstration. *Data Mining and Knowledge. Discovery* 7, 4, 349-371.

[Keogh *et al.*, 2009] Keogh, E. J., Wei, L., Xi, X., Vlachos, M., Lee, S. H., and Protopapas, P. 2009. Supporting exact indexing of arbitrarily rotated shapes and periodic time series under Euclidean and warping distance measures. *VLDB J.* 18, 3, 611-630.

[Kim *et al.*, 2001] Kim, S., Park, S., and Chu, W. 2001. An index-based approach for similarity search supporting time warping in large sequence databases. *ICDE*, 607-61.

[Mueen and Keogh, 2010] Mueen, A., and Keogh, E. J. 2010. Online discovery and maintenance of time series motifs. *KDD*, 1089-1098.

[Mueen et al., 2011] Mueen, A., Keogh, E. J., Zhu, Q., Cash, S., Westover, M. B., and Shamlo, N. 2011. A disk-aware algorithm for time series motif discovery. *Data Min. Knowl. Discov.* 22, 1-2, 73-105.

[Papapetrou et al., 2011] Papapetrou, P., Athitsos, V., Potamias, M., Kollios, G., and Gunopulos, D. 2011. Embedding-based subsequence matching in time-series databases. *ACM TODS* 36, 3, 17.

[Raghavendra et al., 2011] Raghavendra, B., Bera, D., Bopardikar, A., and Narayanan, R. 2011. Cardiac arrhythmia detection using dynamic time warping of ECG beats in e-healthcare systems. *WOWMOM*, 1-6.

[Rakthanmanon *et al.*, 2012] Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., and Keogh E., 2012. Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping, *SIGKDD*, 262-270.

[Rebbapragada et al., 2009] Rebbapragada, U., Protopapas, P., Brodley, C., and Alcock, C. 2009. Finding anomalous periodic time series. *Machine Learning* 74, 3, 281-313.

[Sakurai et al., 2007] Sakurai, Y., Faloutsos, C., and Yamamuro, M. 2007. Stream monitoring under the time warping distance. *ICDE*, 1046-55.

[Sakurai et al., 2005] Sakurai, Y., Yoshikawa, M., and Faloutsos, C. 2005. FTW: fast similarity search under the time warping distance. *PODS*, 326-337.

[Srikanthan et al., 2011] Srikanthan, S., Kumar, A., and Gupta, R. 2011. Implementing the dynamic time warping algorithm in multithreaded environments for real time and unsupervised pattern discovery. *IEEE ICCCT.*

[Stiefmeier et al., 2007] Stiefmeier, T., Roggen, D., and Tröster, G. 2007. Gestures are strings: efficient online gesture spotting and classification using string matching. *Proceedings of the ICST 2nd international conference on Body area networks.*

[Vlachos et al., 2003] Vlachos, M., Hadjieleftheriou, M., Gunopulos, D., and Keogh, E. J. 2003. Indexing multi-dimensional time-series with support for multiple distance measures. *KDD*, 216-225.

[Wobbrock et al., 2007] Wobbrock, J. O., Wilson, A. D., and Li, Y. 2007. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. *ACM UIST*, 159-168.

[Ye and Keogh, 2009] Ye, L., and Keogh, E. J. 2009. Time series shapelets: a new primitive for data mining. *KDD*.

[Yi et al., 1998] Yi, B., Jagadish, H., and Faloutsos, C. 1998. Efficient retrieval of similar time sequences under time warping. *ICDE*, 201-208.

[Zinke and Mayer, 2006] Zinke, A., and Mayer, D. 2006. *Iterative Multi Scale Dynamic Time Warping*. Universität Bonn, Tech Report # CG-2006-1.

[UCRsuite, 2012] UCR Suite Supporting Website. 2012. *www.cs.ucr.edu/~eamonn/UCRsuite.html*