Eindhoven University of Technology
Department of Mathematics and Computer Science

Master Thesis

Clustering and Alignment of Physiological Time Series

Author:
Sander den Heijer

Supervisors:
dr. Natalia Sidorova
dr. Joyce Westerink

A thesis submitted in fulfilment of the requirements for the degree of Master of Science

January 2015

Abstract

Microprocessors which can be used to collect data are embedded in more and more everyday objects. This phenomenon is known as pervasive computing and results in large amounts of data. These data can contain valuable information, but extracting this information is not a trivial task. For example, a data set containing the values of a person's heart rate can be collected using a smart watch; however, the size of such a data set will rapidly grow beyond the point where manual inspection is possible.

The aim of this master project is to determine a description of a characteristic pattern which reoccurs in a given data set. This description can be either interpersonal or intrapersonal; one may think of the development of the heart rate of athletes during a race, the recognition of words based on speech patterns, or the development of a person's skin temperature during sleep. A major problem when trying to achieve this goal is that it is not necessarily the case that all data sets contain similar patterns. Therefore, clusters containing data sets with similar patterns must be determined, followed by the construction of a data set which describes all the data sets in the cluster; such a data set is called an alignment.

Several alignment algorithms are evaluated in this master project, after which the one most suitable is extended in such a way that it generates an alignment based on multiple data sets. The algorithm also visualizes the reliability of the alignment; the more data sets containing a similar part in the alignment, that more reliable is that part.

Data sets often contain outliers, which have a negative effect on the quality of the determination of clusters and alignments. Therefore, an outlier removal algorithm is developed based on an existing one; this algorithm is compared with existing outlier removal algorithms and performs better on data sets with varying characteristics. The main advantage of this algorithm is that it is fully automated, which means that no knowledge of the frequency of outliers occurring in the data set is required.

To validate these developed algorithms on a real world data set, a case study is conducted on a data set consisting of the skin temperature, skin conductance, and heart rate data of one person. The aim of the case study is to determine the characteristic patterns of these physiological features during sleep. The nights are clustered on the basis of similarity of the data sets of the feature under inspection and an alignment for each cluster is generated. Both the clustering algorithm and the alignment algorithm provide desirable results.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Pervasive computing is a growing trend where computing is made available in everyday objects such as a mobile phone or a smart watch. Pervasive computing is applied in numerous areas of human activities and allows one to deduce valuable information. For example, pervasive sensor technologies can be used for unobtrusive monitoring of a person's level of stress. However, if a device stores data at a frequency of 1 Hz, this will already lead to a data set consisting of $604,800$ measurements in one week. It is not possible for one person to manually analyze this amount of data. Since these so-called time series consist of a sequence of values over time, visualizing these data is easy and can give rise to insights. However, if a data set contains time series of several instances of the same event, simply depicting all data often yields indistinct results as can be seen in Figure 1.1; this figure contains the time series of a person's skin temperature during 19 nights. In order to be able to understand these data, it is of interest to construct one time series which contains characteristic patterns of all the others; such a time series is called an alignment. A method commonly used in fields of study outside computer science is to take the average value of all other time series for each moment in time. The result of this approach on the time series illustrated in Figure 1.1 is shown in Figure 1.2.

One of the disadvantages of the averaging method is that it does not align the same pattern in different time series if it is shifted in time. The main disadvantage of alignment algorithms in general is that the alignment is based on all the time series; however, not all time series possess common patterns most of the time. Hence, the aim of this project is stated as follows:

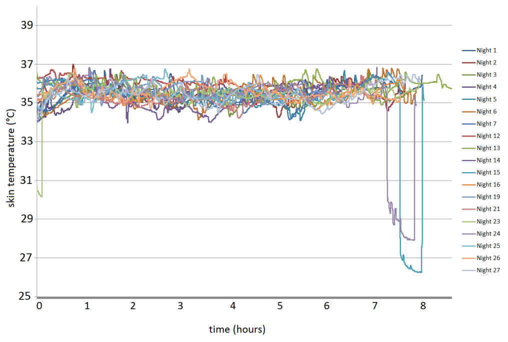

FIGURE 1.1: A person's skin temperature during 19 nights

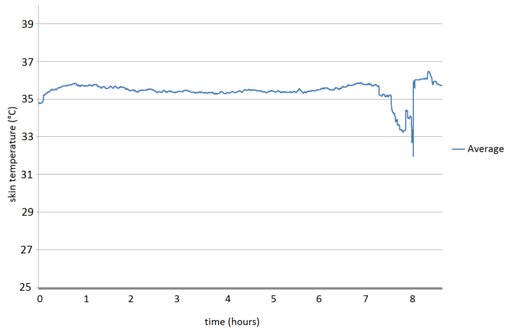

FIGURE 1.2: Alignment using average values

The goal of this project is, given a set of time series of instances of the same event:

- *to determine clusters of time series with similar patterns;*
- *to determine an alignment for the time series in each cluster.*

However, real-world time series are prone to noise and may contain outliers which disrupt the patterns to be distinguished. To increase the quality of both clustering and alignment, noise and outliers need to be removed first. To accomplish this goal, a number of outlier detection algorithms are evaluated after which the one most suitable for application on large data sets containing physiological data is used as a basis for the algorithm developed in this master project. The algorithm is parameter-free as it uses samples of the existing data set to determine the amount of noise and outliers and uses this information to set its parameter. The algorithm is compared to existing algorithms by means of a number of time series with different characteristics of which the ground truth is known.

After the removal of outliers from the time series, the next step is the clustering of the time series. A case study on a real-world data set is conducted in order to validate the clustering algorithm, which is used to obtain clusters of nights for skin temperature, skin conductance, and heart rate data during sleep. Since the clustering algorithm makes use of the alignment algorithm to obtain distances between time series and the case study also shows the functionality of the outlier detection algorithm and the alignment algorithm on the data set of the physiological features mentioned above, the case study is presented after the alignment algorithm.

The last step to the attainment of the aim of this project is the alignment of the time series. The same approach as used for the outlier detection algorithm is applied to the alignment algorithm: several algorithms are evaluated after which the fastest and most accurate one is extended to obtain desirable results on large data sets consisting of physiological time series. The algorithm generates a mapping between points of two time series, which is used to determine the distance between the time series. This mapping is also used to generate an alignment if the time series are in the same cluster. The resulting algorithm is validated by means of a time series of which the ground truth is known.

This thesis has the following structure: Chapter 2 presents the developed outlier detection algorithm and gives a comparison of this algorithm with existing outlier detection algorithms; the developed alignment algorithm is discussed in Chapter 3; Chapter 4 discusses the data set examined in the case study and contains the results of the case study conducted; in Chapter 5, the implementation of the developed application is elaborated; finally, in Chapter 6, conclusions are drawn with respect to this master project and improvements are suggested for future work.

Chapter 2

Outlier detection

Outliers have a negative effect on the quality of the clustering and alignment of the time series as they disturb the general, characteristic patterns occurring in a time series. The goal of this chapter is to develop an algorithm which removes outliers, given a time series.

An outlying observation, or "outlier", is one that appears to deviate markedly from other members of the sample in which it occurs. In this connection, the following two alternatives are of interest: (i) an outlying observation may be merely an extreme manifestation of the random variability inherent in the data or (ii) an outlying observation may be the result of gross deviation from prescribed experimental procedure or an error in calculating or recording the numerical value [1]. The latter case is also referred to as "noise".

This chapter starts with a discussion of a number of existing algorithms and their limitations, followed by a description of the algorithm developed for this master project; finally, this algorithm is compared to a number of previously discussed alternatives.

2.1 Related work

The Local Outlier Factor [2] (LOF) is based on the concept of local density. The locality is given by the k nearest neighbors in the data set, the distance of which is used to estimate the density. By comparing the local density of an item with the local densities of its neighbors, regions of similar density can be identified. Points that have a substantially lower density than their neighbors are considered to be outliers.

Hyndman and Khandakar developed the Forecast algorithm [3] which is based on a multiplicative error model; it calculates a prediction interval for the next n values based on the previous m values. This algorithm can be used to predict an interval for each value of the time series ($n = 1$), if the actual value is outside this interval, it will be marked as an outlier.

Hill and Minsker [4] developed an anomaly detection method based on univariate autoregressive data-driven models of the sensor data stream and a prediction interval. It relies on a threshold deviation from a model prediction to delineate the boundary between anomalous and non-anomalous data. An advantage of this approach is that it does not require the user to set the threshold, as that is derived from the data. How to determine the optimal window size for this algorithm, however, is not elaborated.

The HOT SAX algorithm [5] finds time series discords, which are subsequences of a longer time series that are maximally different to all the rest of the time series subsequences. It requires the time series to have a recurring pattern at a certain frequency, which is not necessarily the case in the time series used for this master project.

Another common approach to the removal of outliers is median filtering. This is done by replacing the value of each element by the median found in a window around the element. If the window size is s, the median of an element e in a time series is found by using its own value and (i) using $(s-1)/2$ (it is assumed s is odd) *elements* in front of and after e or (ii) using $(s-1)/2$ *seconds* in front of and after e. If the number of elements in front of or after e is smaller than $(s-1)/2$, the median is not defined. Since the frequency in each time series is constant in this master project, the former approach is applicable.

For example, given the following observations:

time	0	1	2	3	4	5	6	7
value	0.2	0.2	0.2	0.3	0.3	0.3	300.0	0.3

Applying the median filter with $s = 3$ to the series yields:

time	1	2	3	4	5	6
value	0.2	0.2	0.3	0.3	0.3	0.3

The outlier at time 6 is eliminated [6].

This method gives a robust estimation, having a breakdown value of 50%, higher than that of many other robust procedures [7]. The breakdown value is a popular measure of the robustness of an estimator against outlying observations. Roughly speaking, it indicates the smallest fraction of contaminants in a sample that causes the estimator to break down, that is, to take on values that are arbitrarily bad or meaningless [8].

A disadvantage of this method is that actual values of peaks and troughs get lost as they get flattened. In the most extreme case a peak or trough as a whole will be flattened, but this will only happen if the window size is too large.

Basu and Meckesheimer [9] propose an algorithm based on median filtering, which predicts the value at time t as a median of the values in a window size of 2κ from $t - \kappa$ to $t + \kappa$. This predicted value is compared to the actual value; if it deviates more than a certain threshold τ, then it is marked as an outlier and replaced with the value of the median. Unfortunately, the authors do not elaborate on how to find correct values for κ and τ. It seems that the values for these variables are chosen manually in the case studies they present.

2.2 Evaluation of outlier detection algorithms

This section contains an evaluation of algorithms discussed in the previous section.

2.2.1 Local Outlier Factor

The main problem with the LOF algorithm is that the density values cannot be interpreted the in the same way for each data set, or even within a data set itself, due to the locality of the algorithm. This means that both the threshold for density values and the value of k (the number of nearest neighbors to be taken into account) have to be set by the user or determined automatically. Especially the former determination is far from trivial.

However, R contains the DMwR package [10], which implements the LOF algorithm in R. R[1] is a free software environment for statistical computing and graphics. Figure 2.1

[1]http://www.r-project.org/

shows part of a skin conductance time series with outliers at 14.5 seconds, 18.0 seconds, and 18.5 seconds.

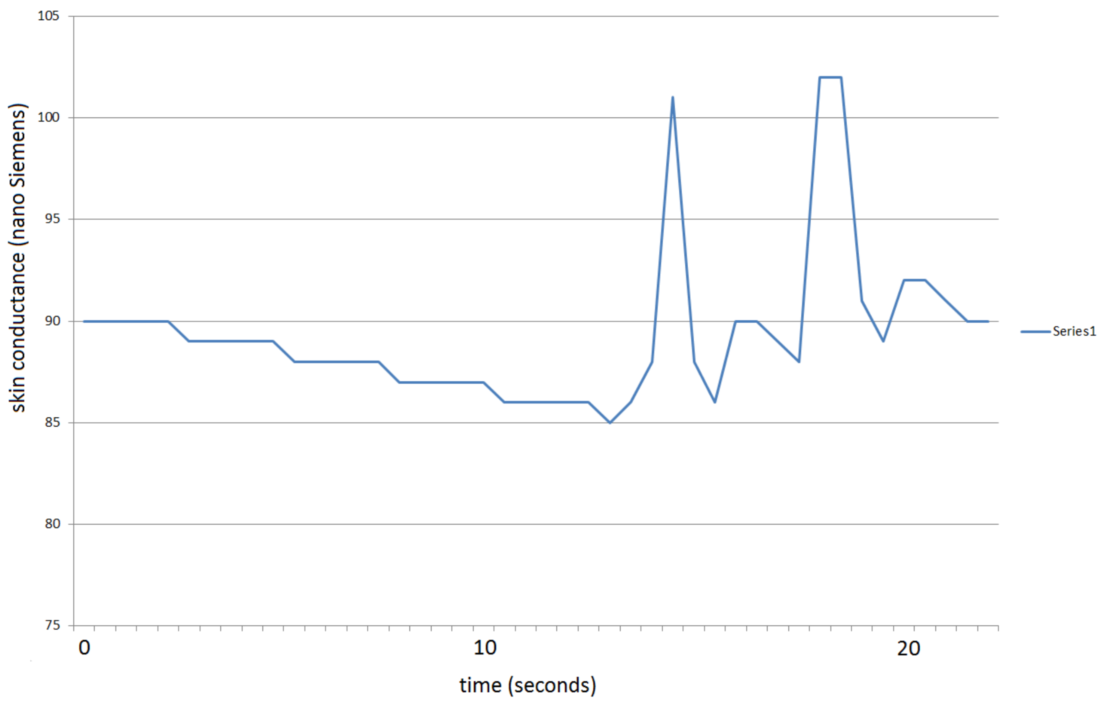

Figure 2.1: Time series with outliers at 14.5s, 18.0s, and 18.5s

Table 2.1 shows the top five of points with the lowest density, determined by the algorithm for different values of k. It can be concluded that the value of the parameter k is important, however, the outcome needs to be checked in order to verify if k is large enough. Another disadvantage of the algorithm is that it generates a list of points, starting with the point with lowest density. The length of this list has to be set by the user, therefore, it is not known at which moment in the list the outliers stop and the list continues with inliers; it is even possible that the list is too short and not all outliers are contained in the list. For these reasons, the approach is not of interest for this master project.

Table 2.1: Times of the points with lowest density for several values of k

k	Lowest density				
5	8.0s	8.5s	9.0s	9.5s	10.0s
7	3.0s	3.5s	4.0s	4.5s	5.0s
9	18.0s	18.5s	14.5s	20.0s	20.5s
11	18.0s	18.5s	14.5s	20.0s	20.5s

Several extensions of the LOF algorithm exist. One of them is Local Outlier Probability [12], which uses inexpensive local statistics to make it become less sensitive to the choice of the parameter k. In addition, the resulting values are unity-based normalized, i.e., scaled to a range of $[0, 1]$, in order to become independent of the specific data distribution in a given data set. This is an interesting improvement but it only solves one of the two problems the original algorithm faced.

2.2.2 Forecast algorithm

The Forecast algorithm is implemented in R as well. When using this approach on a test sample of 7200 values, i.e., one hour with a frequency of 2 Hz, it took 14 minutes and 38 seconds to compute all the prediction intervals using ten values to determine the prediction interval of the next value. This running time is not acceptable and this algorithm will not be used in this master project.

2.2.3 Median filtering

The median filtering algorithm is an effective alternative if it does not matter that the original values get changed, which is the case in this master project, since its goal is to find general patterns over a longer period of time. Its running time depends on the chosen window size, but a data set containing $172,800$ values (24 hours with a frequency of 2 Hz) is processed within 2 seconds with a large window size of 501. The only disadvantage of this algorithm is that the window size needs to be set. If a data set contains long sequences of noise, the window size needs to be larger in order to be able to filter out this noise. The next section presents an approach to automatically determine the window size given a specific data set.

2.3 Expanding the median filter algorithm

This section is confidential.

2.4 Linear regression

This section is confidential.

2.5 Verification and comparison of algorithms

The following algorithms are compared in this section:

- The basic median filtering algorithm described in Section 2.1, with varying window sizes;
- Basu and Meckesheimer's algorithm, also described in Section 2.1;
- The automated median filtering algorithm proposed in Section 2.3.

The performance of the algorithms will be evaluated on the basis of running time, true positives and false positives found, deviation from the original time series values excluding outliers, and the number of parameters that have to be set. Three test sets will be used, depicted in Appendix C, each one with specific characteristics:

- Set 1: a skin conductance time series which contains a high level of variance (Figure **??**);
- Set 2: a skin conductance time series which contains a considerable amount of outliers (Figure **??**);
- Set 3: an ambient light time series containing several outliers. The changes are more sudden in this time series in comparison with the skin conductance time series (Figure **??**);

All test sets consist of a time series of one hour with a frequency of 2 Hz, i.e., 7200 values.

The following algorithms are compared in this section:

Basic median filtering algorithm

A simple approach is to use the median filter as described in Section 2.1 with fixed window size on the time series. Since the automated median filtering algorithm checks window sizes of 61 and higher, smaller window sizes will be used when testing with a constant window size, namely: 5, 11, 21, and 31. A positive side-effect of these small window sizes is that the computation will take little time.

Basu and Meckesheimer's algorithm

This algorithm is described in Section 2.1 as well. After comparing the window sizes for the basic median filtering algorithm, the window size of 31 performed best. Therefore, the window size for this algorithm is set to this size. Several values for the threshold will be used, namely: 20, 40, and 100.

Automated median filtering algorithm

The automated median filtering algorithm proposed in Section 2.3, which determines its window size automatically.

2.5.1 Running time

The running time is an important aspect of an algorithm; earlier in this chapter, the Forecast algorithm got rejected as it took a significant amount of time to generate the results. The algorithms are all based on median filtering, therefore, an analysis of this algorithm is given.

When finding the median of k values ($k \in \mathbb{Z}^+$), the values first have to be ordered. The fastest sorting algorithms, e.g., Mergesort or Heapsort, take $O(k \log(k))$ time. After the ordering step, the median is the element at position $(k+1)/2$ (assuming k is odd); getting the value at that position takes constant time ($O(1)$). It is not clear if the application used to perform this analysis uses an optimization to calculate consecutive medians in the same data set, therefore, it is assumed that the application computes the median as described above every time when using a moving window on a time series.

This means that the total running time is $O(nk \log(k)))$, with n the size of the input and k the window size. In practice, for the window sizes used for the basic median filtering algorithm and Basu and Meckesheimer's algorithm, a time series of 24 hours with a frequency of 2 Hz is processed in between 1 and 2 seconds. It takes longer for the automated median filtering algorithm as it uses a larger window size (e.g., 10 seconds for a window size of 361). This is, however, still only a fraction if compared to the size of the data set and an upper bound as the window size usually is variable.

2.5.2 True and false positives

The quality of the detection of outliers is measured by determining the number of true and false positives. This shows how many outliers are detected and how many inliers are incorrectly marked as an outlier. Outliers were manually marked in a copy of the test sets and replaced by values based on the neighbors that are inliers. True and false positives are distinguished as follows for the basic and automated median filtering algorithms. If the difference between the value computed by the algorithm and the corrected value is smaller than a certain threshold, it is a true positive. The threshold is set to 10 for the skin conductance data sets and 100 for the ambient light data set. If the value of an inlier deviates from the original value more than the threshold, it is a false positive. Basu and Meckesheimer's algorithm marks values larger than the threshold τ as outliers, other values are inliers. This makes it trivial to determine the true and false positives.

If the percentage of detected true positives is below 50, it is marked red in the table; the same holds for false positive if the percentage is above 10.

Basic median filtering algorithm

The number of true and false positives found for each test set while using different window sizes can be seen in Table 2.2.

When using a window size of five, it takes three consecutive outliers to miss a true positive. Since this is not uncommon in most time series, this window size rectifies the least outliers in most cases. The fact that the window size of 21 outperforms the window size of 31 on Set 2 is an interesting outcome. This is caused by sequences of the following form: a high number of outliers with some inliers in between, where there are more than

TABLE 2.2: True positives and false positives found for the median filtering algorithm using different window sizes

	True positives			False positives		
Window size	Set 1	Set 2	Set 3	Set 1	Set 2	Set 3
5	15/15	50/159	36/61	0/7185	18/7041	462/7139
11	15/15	65/159	41/61	0/7185	32/7041	613/7139
21	15/15	74/159	48/61	0/7185	55/7041	779/7139
31	15/15	73/159	52/61	0/7185	78/7041	1,010/7139

11 inliers in a window of 21 but less than 16 in a window of 31. However, Figure 2.2 shows that an increasing window size does seem to have a positive effect.

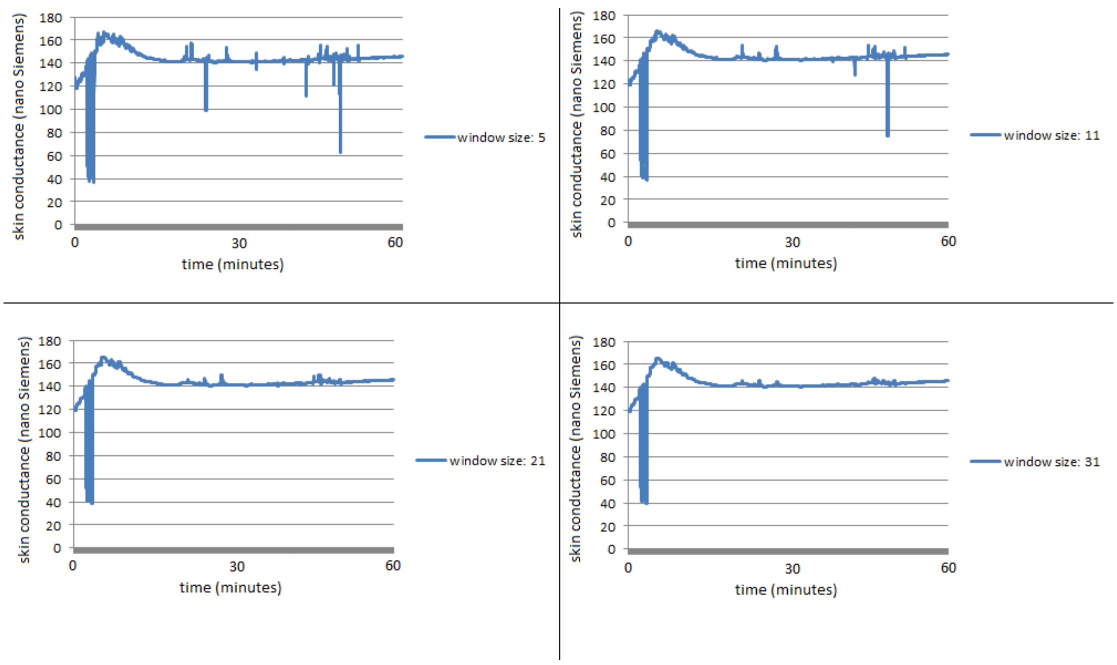

FIGURE 2.2: Results for median filtering when applying different window sizes on Set 2

In order to detect a false positive, the number of outliers in the window needs to be larger than the number of inliers. Since the number of inliers always form a sequence longer than half the window size in Set 1, no false positives occur.

Basu and Meckesheimer's algorithm

The number of true and false positives found for each test set can be seen in Table 2.3.

TABLE 2.3: True positives and false positives found for Basu and Meckesheimer's algorithm

	True positives			False positives		
Threshold	Set 1	Set 2	Set 3	Set 1	Set 2	Set 3
20	15/15	76/159	61/61	8/7185	27/7041	2,386/7139
40	12/15	45/159	61/61	1/7185	19/7041	1,758/7139
100	2/15	0/159	61/61	0/7185	0/7041	986/7139

Automated median filtering

The number of true and false positives found for each test set can be seen in Table 2.4.

TABLE 2.4: True positives and false positives found for the automated median filtering algorithm

True positives			False positives		
Set 1	Set 2	Set 3	Set 1	Set 2	Set 3
15/15	159/159	61/61	0/7185	101/7041	1,869/7139

Conclusion

In general, increasing the window size increases the number of outliers found and the number of inliers incorrectly marked as outliers.

The quality of the detection of outliers decreases rapidly when the chosen threshold for Basu and Meckesheimer's algorithm is not optimal. The most accurate threshold can only be chosen when the user has knowledge of the tendencies of the outliers in the data set.

Overall, the results of the basic and automated median filtering algorithms are better. All outliers have been removed in all sets when the automated median filtering algorithm is applied, which is an excellent result. After inspection, most false positives in Set 2 appeared to be changed just over the threshold. This is the consequence of a larger window size and is not as harmful as it appears to be on first sight. All three algorithms perform less well on the ambient light time series than the skin conductance time series. Apparently, the large and sudden deviations in this time series make it hard to distinguish outliers from inliers.

2.5.3 Average deviation

For each point that is not an outlier, the difference between the original value and the value determined by the algorithm is calculated; the sum of these values divided by the number of points in the data set is used to determine to what extent the time series is changed. The inliers in Set 1 range from 76 to 314, in Set 2 from 117 to 168, and in Set 3 from 0 to 6,048.

Basic median filtering algorithm

The average deviation found for each test set while applying different window sizes can be seen in Table 2.5. Please note that the values in Set 3 are higher and deviate more from each other than in the other two test sets, this results in a higher sum of deviations.

Table 2.5: Average deviation found for the median filtering algorithm using different window sizes

	Average deviation		
Window size	Set 1	Set 2	Set 3
5	0.60	0.23	20.69
11	1.03	0.49	56.72
21	1.13	0.61	69.01
31	1.57	0.71	81.85

Basu and Meckesheimer's algorithm

The average deviation for each test set can be found in Table 2.6.

Table 2.6: Average deviation found for Basu and Meckesheimer's algorithm

	Average deviation		
Threshold	Set 1	Set 2	Set 3
20	0.09	0.24	79.37
40	0.01	0.21	76.84
100	0	0	70.09

Automated median filtering

The average deviation for each test set can be found in Table 2.7.

Table 2.7: Average deviation found for the automated median filtering algorithm

	Set 1	Set 2	Set 3
Average deviation	3.69	2.12	135.23

Conclusion

With respect to this metric, the Basu and Meckesheimer algorithm performs best. This can be explained by the fact that a value is only changed if the threshold is exceeded.

In each column the values of the basic median filtering algorithm are increasing, this is not surprising since a larger window size results in a smoother graph, which means the original values are changed to a larger extent.

The automated median filtering algorithm changes the time series significantly more than the other two algorithms. This is a result of the window size being at least 61, while the other algorithms have a maximum window size of 31. The fact that the time series is changed more is not necessarily a disadvantage; it can even be an advantage if deviations that distract from the general pattern are removed. However, it is good to keep in mind that the values of the peaks and troughs can be more extreme in the original data set.

2.5.4 Parameters

The median filtering algorithm requires the user to set the window size. In most cases, after trying a number of different values for this parameter, an approximate lower bound and upper bound can be found which eliminate most outliers while maintaining the main characteristics of the data set.

Basu and Meckesheimer defined one additional parameter, when compared to the median filtering algorithm: the maximum acceptable deviation from the median before a value is considered to be an outlier. In order to set this parameter correctly, the user needs to know what a realistic change between two consecutive values is. Otherwise it is hard to get satisfying results using this algorithm.

The automated median filtering algorithm does not require any parameters to be set by the user.

2.6 Conclusion

The Forecast algorithm and the Local Outlier Factor are not suitable for analyzing this specific type of time series; the former needs a considerable amount of time, which is not desirable since the data sets are large, the latter is faster but produces invalid results. This is why these two algorithms are not included in the comparison of algorithms. The remaining algorithms are the basic median filtering algorithm, Basu and Meckesheimer's algorithm, and the automated median filtering algorithm developed in this master project. These three algorithms all are variants of the median filtering algorithm and require, therefore, approximately the same running time, which is acceptable.

If the user wants to change the time series as little as possible and possesses knowledge about specific characteristics of the data set, i.e., maximum deviation per time unit and quality of the data, Basu and Meckesheimer's algorithm is preferable since it only changes the original data if a certain threshold is exceeded. A disadvantage of this algorithm is that a wrong choice of threshold can have important consequences with respect to the detected number of outliers.

If the user does not know what the maximum deviation per time unit is, reasonable results can still be acquired when using the median filtering algorithm. However, it cannot be guaranteed that all outliers are eliminated from the time series.

If the characteristics are unknown, the algorithm developed in this chapter can be used. It is aimed at eliminating as many outliers as possible and can, as consequence, change the original values more than the other algorithms. However, the goal of this project is to find general patterns, therefore the exact values are less important and this is in favour of this algorithm. The fact that the resulting time series contain less deviation from the general pattern even improves the quality of the alignment, which is discussed in the next chapter.

Chapter 3

Alignment

An alignment is a time series which contains general, characteristic patterns occurring in a set of time series. The goal of this chapter is to generate an algorithm which determines an alignment and its reliability for a given set of time series. The reliability reflects how often each part of the alignment occurs in the given set of time series.

This chapter starts with an example which shows the flaws in the averaging method, a simple algorithm often applied in practice. Next, a discussion and evaluation of existing alignment algorithms is presented, after which the alignment algorithm developed for this master project is elaborated and validated. This algorithm is based on an algorithm which takes two time series as input; therefore, it is extended in such a way that it can be applied to a set of time series of arbitrary size.

In order to determine the alignment of two time series, a mapping from points of one time series to points of the other time series is needed. The average values of each two points mapped to each other form the alignment. If this mapping is done correctly, the alignment contains patterns occurring in both time series.

A rather simple way, often applied in practice, to align two time series is to make a mapping based on the timestamps. Figure 3.1 shows time series S and T, which are intuitively similar to each other. Figure 3.2 shows the resulting alignment A, which has the average values of S and T at each timestamp. Time series A clearly does not capture the patterns of S and T, since it has two peaks in the middle part with a small trough in between.

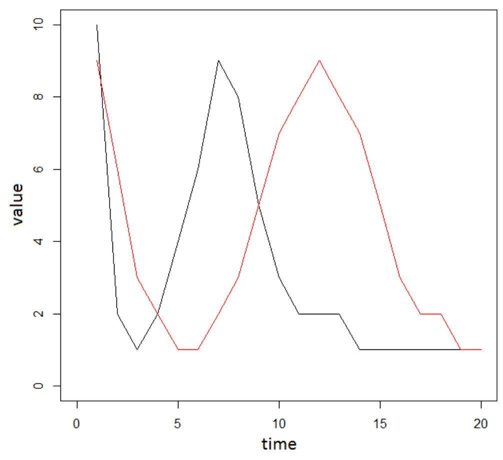

FIGURE 3.1: Two time series S (black) and T (red)

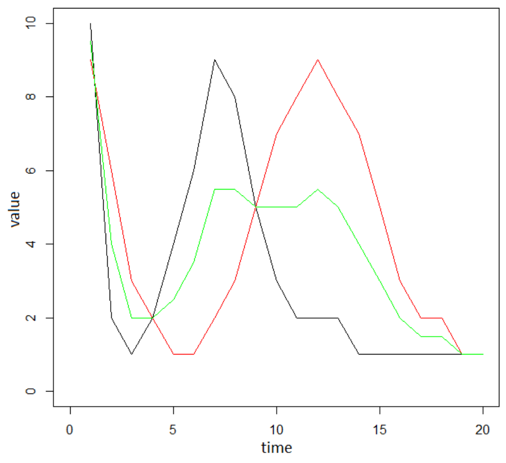

FIGURE 3.2: Alignment A (green) of S and T, using average values

In the next section, related work is presented and an algorithm that is used as the basis for many other algorithms is discussed in more detail.

3.1 Related work

A common problem which has to be dealt with, is that the time series can be shifted in height and can differ from each other in amplitude and frequency. Speech recognition is a field of study in which this is inescapable. A new signal is compared to known patterns in order to recognize a word; since even a single person varies the way in which he speaks, different periods and amplitudes of the signal occur. Non-stationary Hidden Markov models have been used to recognize patterns [13]. In this case, a mapping is computed between the new signal and a known pattern to determine to what extent they match. Maintaining a library of known patterns makes the determination of which word was said possible.

In order to compute a mapping that is not influenced by amplitude or frequency, Dynamic Time Warping was introduced by Kruskal and Liberman [14]. DTW generates a mapping with minimal distance between two time series. The distance measurement applied by this algorithm is called the warping distance and is based on the sum of distances of each pair of points in the mapping. DTW uses dynamic programming as follows, to find the minimal warping distance and the corresponding mapping between two time series:

For time series $S = s_1, s_2, \ldots, s_n$ and $T = t_1, t_2, \ldots, t_m$, where s_i and t_i are the values at timestamp i for S and T respectively, the minimal warping distance wd_{min} is defined as follows:

$$\begin{aligned}
wd_{min} &= w(n, m) \\
w(0, 0) &= 0 \\
w(i, 0) &= \infty \quad (1 \leq i \leq n) \\
w(0, j) &= \infty \quad (1 \leq j \leq m) \\
w(i, j) &= d(s_i, t_j) + min(w(i-1, j), w(i, j-1), w(i-1, j-1))
\end{aligned}$$

Where, for two values x and y, $d(x, y)$ is the chosen distance between the two values, typically the Euclidean distance. The n-by-m matrix w is called the warping matrix. Both axes of the matrix represent time, one axis represents the time of S, the other the time of T. Each cell in the warping matrix uses a value from any of the three previously computed neighbors, which are the optimal solutions for smaller parts of the time series. This means it is possible to trace back the values used to compute the minimal warping distance, i.e., $w(n, m)$, thus getting the mapping with minimum distance between S and T. If the value of $min(w(i-1, j), w(i, j-1), w(i-1, j-1))$ is either $w(i-1, j)$ or $w(i, j-1)$, two points in one time series are mapped to one point in the other time series. Since multiple points of one time series can be mapped to one point of the other time series, time series do not have to be of equal length. The mapping must start at the beginning of both time series ($w(1, 1)$) and finish at the end of both time series ($w(n, m)$), ensuring that every index of both time series is used in the mapping.

Petitjean [15] introduces a variant of DTW, based on Barycentric Averaging. This algorithm produces a time series after each step of its iterative process; this time series converges to the alignment of the time series used as input.

Salvador and Chan [16] present the FastDTW algorithm; it uses an approximation of DTW, which leads to linear time and space complexity instead of quadratic. It uses a multilevel approach inspired by the multilevel approach used for graph bisection [17]. Their presented approach consists of three key operations:

1. Coarsening - shrink a time series into a smaller time series that represents the same curve as accurately as possible with fewer data points;

2. Projection - find a mapping with minimum distance at a lower resolution, and use that mapping as an initial guess for a higher resolution's mapping;

3. Refinement - refine the mapping projected from a lower resolution through local adjustments of the mapping.

Several of the algorithms discussed in this section are evaluated in the next section.

3.2 Evaluation of existing alignment algorithms

This section contains an evaluation of algorithms discussed in the previous section, explaining the choice made for the algorithm used as the basis of the algorithm developed in this study.

3.2.1 Dynamic Time Warping

The practical evaluation of DTW was performed using R. R contains the dtw package [18], which implements DTW in R. Figure 3.3 shows the mapping generated by R, Figure 3.4 shows the time series (green line) based on the average values of each two points mapped together. Intuitively, the mapping seems correct.

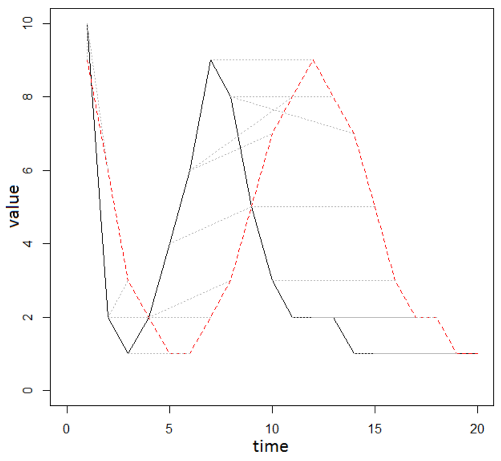

Figure 3.3: Alignment of time series S (black) and T (red) in R

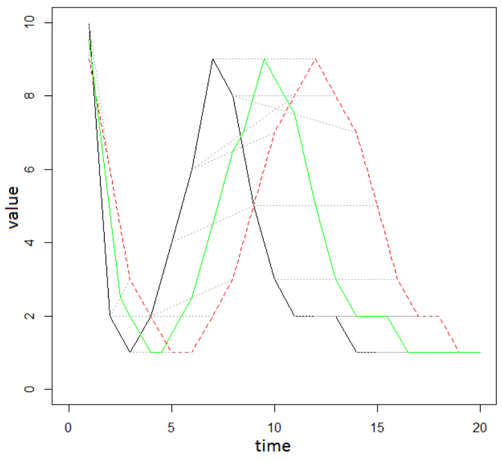

Figure 3.4: Average values of the alignment in Figure 3.3

However, when a time series is shifted upwards and mapped to its original values, a number of points with high density occur. All points mapped to the same point indicate a pattern, according to the algorithm, which is not present in the other time series. This defect can be seen in Figure 3.5 and Figure 3.6. In this trivial case, it can be concluded that the mapping is not the desired one. A bigger problem arose when the algorithm was applied on two time series of 15,000 values: R ran out of memory. This is most probably a consequence of quadratic space complexity.

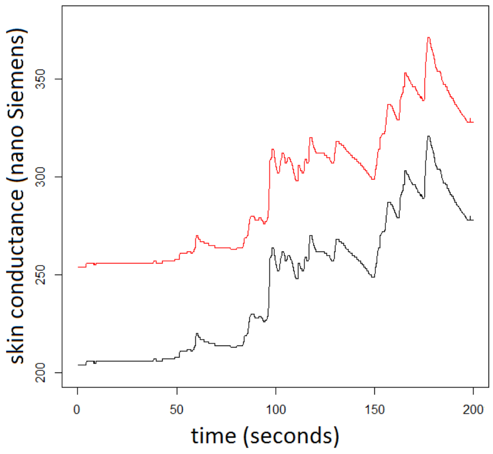

FIGURE 3.5: Time series S (black) and S' (red) (S shifted upwards by 50)

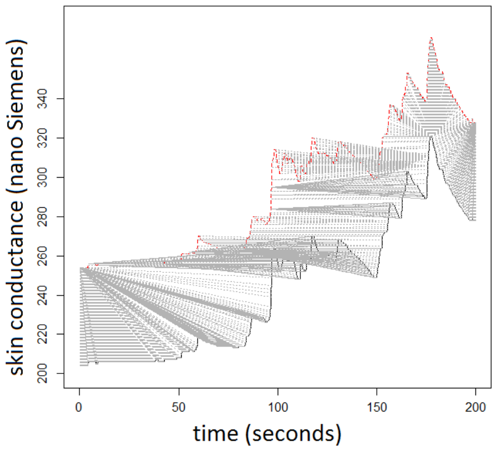

FIGURE 3.6: Alignment of S and S' generated by R's dtw package

3.2.2 DTW Barrycenter Averaging

A simplified version of the DBA algorithm[1] is available as open source; Figure 3.7 to Figure 3.10 show the result after the first four iterations when applied on the same time series as depicted in Figure 3.5. The first one contains a significant number of outliers, while the subsequent ones seem to converge to a straight line; this line is obviously not the desired result.

3.2.3 FastDTW

Salvador and Chan implemented the FastDTW algorithm[2]. The application takes two time series as input and outputs the mapping and the warping distance between the two time series. After implementing a tool that visualizes the mapping, a similar problem emerged as with the R package, i.e., a small number of points with high density. According to Chen et al. [19], DTW is inadequate in handling shifting and scaling in amplitude dimension. This observation gave rise to the idea of normalizing the time series before

[1]https://zenodo.org/record/10432
[2]https://code.google.com/p/fastdtw/downloads/list

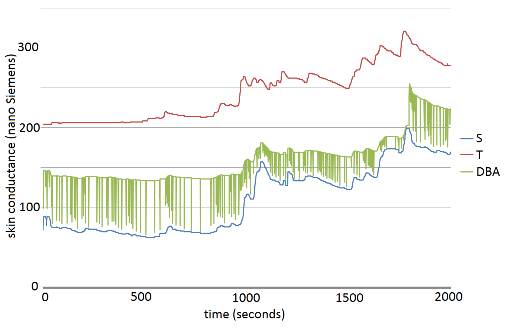

FIGURE 3.7: Alignment by DBA after one iteration

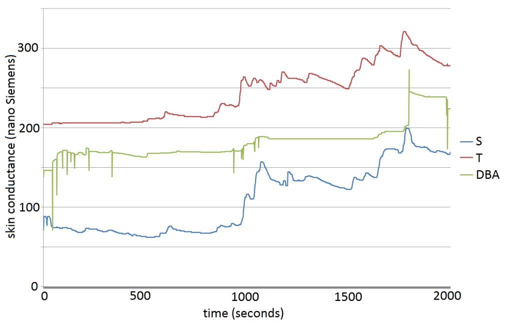

FIGURE 3.8: Alignment by DBA after two iterations

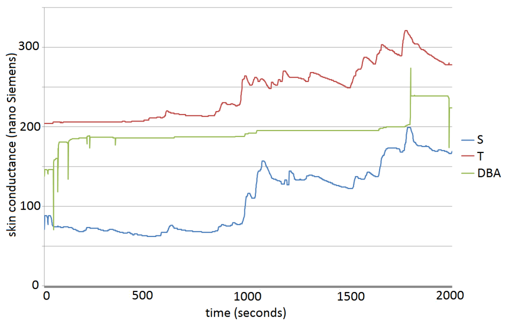

FIGURE 3.9: Alignment by DBA after three iterations

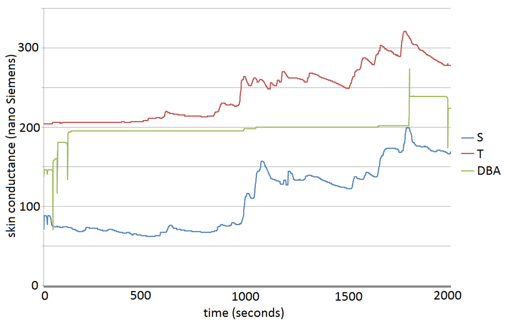

FIGURE 3.10: Alignment by DBA after four iterations

applying DTW. The results significantly improved when the values of the time series were normalized first. A more elaborate evaluation and expansion of this algorithm is discussed in the next section.

3.3 Expanding FastDTW

This section is confidential.

3.4 Aligning multiple time series

This section is confidential.

3.5 Visualizing the reliability of a mapping

This section is confidential.

3.6 Clustering

This section is confidential.

3.7 Conclusion

This section is confidential.

Chapter 4

Case study

This chapter is confidential.

Chapter 5

Application

This chapter is confidential.

Chapter 6

Conclusion

This chapter is confidential.

Appendix A

-

This appendix is confidential.

Appendix B

-

This appendix is confidential.

Appendix C

-

This appendix is confidential.

Appendix D

-

This appendix is confidential.

Appendix E

-

This appendix is confidential.

Appendix F

-

This appendix is confidential.

Bibliography

[1] Frank E Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, 1969.

[2] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *ACM Sigmod Record*, volume 29, pages 93–104. ACM, 2000.

[3] Rob J Hyndman and Yeasmin Khandakar. Automatic time series for forecasting: the forecast package for R. Monash University, Department of Econometrics and Business Statistics, 2007.

[4] David J Hill and Barbara S Minsker. Anomaly detection in streaming environmental sensor data: A data-driven modeling approach. *Environmental Modelling & Software*, 25(9):1014–1022, 2010.

[5] Eamonn Keogh, Jessica Lin, and Ada Fu. Hot sax: Efficiently finding the most unusual time series subsequence. In *Data mining, fifth IEEE international conference on*, pages 8–pp. IEEE, 2005.

[6] Ziv Yaniv. Median filtering. *School of Engineering and Computer Science The Hebrew University, Jerusalem, Israel*, 2009.

[7] Andrew F Siegel. Robust regression using repeated medians. *Biometrika*, 69(1): 242–244, 1982.

[8] Mia Hubert and Michiel Debruyne. Breakdown value. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(3):296–302, 2009. ISSN 1939-0068.

[9] Sabyasachi Basu and Martin Meckesheimer. Automatic outlier detection for time series: an application to sensor data. *Knowledge and Information Systems*, 11(2): 137–154, 2007.

[10] L. Torgo. *Data Mining with R, learning with case studies.* Chapman and Hall/CRC, 2010.

[12] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Loop: local outlier probabilities. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1649–1652. ACM, 2009.

[13] Li Deng, Michael Aksmanovic, Xiaodong Sun, and CF Jeff Wu. Speech recognition using hidden markov models with polynomial regression functions as nonstationary states. *Speech and Audio Processing, IEEE Transactions on*, 2(4):507–520, 1994.

[14] Joseph B Kruskal and Mark Liberman. The symmetric time-warping problem: from continuous to discrete. *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*, pages 125–161, 1983.

[15] François Petitjean, Alain Ketterlin, and Pierre Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44(3):678–693, 2011.

[16] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[17] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. Multilevel hypergraph partitioning: applications in vlsi domain. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 7(1):69–79, 1999.

[18] Toni Giorgino. Computing and visualizing dynamic time warping alignments in r: the dtw package. *Journal of statistical Software*, 31(7):1–24, 2009.

[19] Yueguo Chen, Mario A Nascimento, Beng Chin Ooi, and A Tung. Spade: On shape-based pattern detection in streaming time series. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 786–795. IEEE, 2007.