Time Series Abstraction Methods – A Survey [1]

Frank Höppner

University of Applied Sciences Wolfenbüttel
Salzdahlumer Str. 46/48
38302 Wolfenbüttel, Germany
frank.hoeppner@ieee.org

Abstract. In this paper several time series abstraction methods are reviewed with respect to their applicability in the framework of knowledge discovery (KDD) and data mining (DM). We discuss the demands of KDD/DM (noise, robustness) and show many widely used methods do not satisfy them, because they are single-scale methods. With multiscale approaches, however, the drawbacks of the single-scale methods can be addressed explicitly.

1 Introduction

In knowledge discovery from time series the goal is to detect interesting patterns in the series that may help a human to better recognize the regularities in the observed variables and thereby improve the understanding of the system. Computer programs are very good in number crunching, but *knowledge* arises only in the head of a human. Ideally, knowledge discovery algorithms therefore use time series representations that are close to those that are used by a human.

If a human is interested in gaining insight in some time series data not tabular but graphical representations are used. The impressive pattern recognition capabilities of the human brain help to establish connections between different time series or different parts of a single time series on the basis of their visual appearance. This is a reasonable approach, not only because humans are very good in visual pattern recognition, but also because (changing) trends in the output variables reflect (changes in) the internal state of system.

Many machine learning algorithms (like decision trees), however, have been designed to work with numerical feature vectors. Using subsampling such methods can be applied to time series data to induce classification trees, however, the results obtained from such an approach are usually not easily accessible by a human because the *tabular representation of a time series* rather than the *visual perception of its profile* is addressed. Besides that, frequently observed effects like translation and dilation cannot be handled by such an approach.

Thus, the raw data is converted into a more abstract representation before it is further processed (see e.g. [11, 26, 16]), and in this paper we give a survey of time series abstraction methods that have been used in the literature. Quite often the abstraction process is discussed only briefly in a short paragraph on preprocessing, but it is a very important step because even the best KDD method cannot recover relations in the data that have been falsified or biased during preprocessing. Abstraction or summarization corresponds to a segmentation of the time series and a characterization of the data within the segment. By a segment of a signal we understand a sequence of measurements being supported by the largest contiguous interval over which a certain property holds (e.g. increasing, below a certain threshold, etc.). Thus, the result of time series abstraction is a sequence of intervals where each interval is labeled with discrete and/or numeric attributes.

There is a close relationship to traditional function approximation techniques, because the elements of, let us say, a piecewise linear approximation can be used to define an abstraction. In function approximation it is quite common to compare the methods via their approximation quality. We are not interested in such a purely quantitative approach here, because the desire for high precision alone does not appropriately reflect the danger of overfitting. While a human is quite good in "guessing" what is noise and what is actually trend information, this subjective judgement is extremly difficult to simulate in a technical system. And in KDD we do not know the amount of noise nor if the noise is constant over time.

The following questions should be asked when selecting a time series abstraction method: Are the abstracted features actually supported by the data? How does the method distinguish features from noise? Are the elements of an abstraction easily interpretable by a human? Is the average length of the segments reasonably large (to get some compression effect from summarization)? If we have more than one method available providing satisfying answers, then we may choose the one with the better approximation quality.

2 Overview

We assume that we examine well-behaving signals only, we suppose that the signals are twice differentiable almost everywhere (we allow only for a small (finite) number of discontinuities in the signal or its first few derivatives). Besides that, we also assume that the signals have a finite number of extrema and inflection points and that the sampling rate has been chosen sufficiently high to detect them.

The problem of time series partitioning can be considered either as a supervised or unsupervised task. It can be supervised in the sense that the attributes of interest are defined a priori and we have to assign labels from this given set to portions of the time series. Thus, in this deductive approach, we know in advance for what kind of features we are looking for. On the other hand, we can perform unsupervised partitioning in the sense that no such set of labels is given in advance and we have to learn this set from data, too.

Inductive Approach. In the inductive approach, similar parts in the time series can be identified via clustering [14, 8, 13]. Cluster-

[1] This work has been supported by the Deutsche Forschungsgemeinschaft (DFG) under grant no. Kl 648/1.
Presented at GI Workshop on KDD, Dortmund, Germany, Sept./Oct. 2002

ing aims at partitioning data entities such that similar data objects belong to the same group and dissimilar data objects belong to different groups. If we think of small portions of time series as data objects, every cluster can be considered as an inductively derived label for a group of similar portions. The crucial point in clustering is the used notion of (dis)similarity, which is implemented by means of a distance measure. A zero distance indicates a perfect match between two subseries, a large value indicates dissimilarity. Note that in this approach noise is not handled explicitly. Using an appropriate cluster representation the clusters themselves carry some information about the variance in the associated data.

Usually, the data volume in the inductive approach is quite high, therefore the clustering algorithms have to be effective. There is a variety of clustering algorithms in the literature that have been designed explicitly to cope with larger data sets, e.g. [9, 30, 5, 12]. We will see in section 3 that the number of clusters usually has to be chosen quite large (or the average number of data per cluster quite low), which might be a reason that very simple greedy algorithms have also been reported to be successful [7].

Deductive Approach. In the deductive approach we fix the shapes of interest in advance. Among the most frequently used shape descriptions for time series in technical systems are terms like “linearly increasing”, “convexly decreasing”, or “constant” etc. There are 7 basic shapes for describing local trends in a function f, corresponding to the 3^2 possible combinations of positive/zero/negative first and second derivative f' and f'', excluding those 2 cases of zero f' and non-zero f'' which may hold only for a single point (extrema), but not for a whole segment. For time intervals of non-zero length we thus obtain the following basic shape descriptors: “constant”, “linearly increasing”, “linearly decreasing”, “convexly increasing”, “convexly decreasing”, “concavely increasing”, and “concavely decreasing”. What makes this representation attractive is that “a description that characterizes a signal by its extrema and those of the first few derivatives is a qualitative description of exactly the kind we were taught to use in elementary calculus to *sketch* a function” [29]. If we use these basic shapes to describe the time series locally, breaking down the series into subsequences is theoretically straightforward: Via differencing we obtain estimates of the first and second derivative. Whenever one of the derivatives falls into a different subset of the partition $\{(-\infty, 0), \{0\}, (0, \infty)\}$ than its predecessor, we introduce a new segment. However, in almost every application the data is noisy. Noise makes the series oscillate around the true profile, thereby introducing a large number of tiny segments and local extrema. The main problem when using this approach is therefore how to compensate the influence of noise or, putting it differently, how to distinguish noise from significant features.

There are at least two different ways how to proceed in the deductive approach: (a) We can use function approximation techniques and extract the description of the time series from the approximating function instead of the original series. In this approach the problem of handling noise is handed over to the applied regression technique. (b) We can use appropriate smoothing techniques to get more robust estimates of the first and second derivative and may use the partition $\{(-\infty, -\varepsilon), [-\varepsilon, \varepsilon], (\varepsilon, \infty)\}$. Then, handling noise correctly corresponds to selecting an appropriate smoothing filter.

3 Inductive Segmentation

In the inductive approach we have to fix (a) the representation of time series (or portions of it via a sliding window) and (b) the distance between between time series. We consider four approaches

a) *Clustering of Embedded Subsequences:* In this approach a window of constant width is slid along the series. The content of each window is transformed into a tuple of observations. If data has been measured uniformly over time, this technique embeds a constant number of n consecutive values into an n-dimensional vector of observations. This approach is very popular and used by many authors [24, 21, 15, 7].
b) *Clustering of Embedded Models:* Alternatively, a more abstract representation of the series rather than the raw data can be embedded in a vector. For instance n segments of a piecewise linear representation can be characterized by a $2n$ vector consisting of slope and length of each segment. Then, clustering is performed on these “embedded models” rather than the embedded series. We are not aware of any reference using this approach.
c) *Clustering by Warping Costs:* If other effects like vertical scaling are not significant, dynamic time-warping (DTW) can be used to locally shrink and stretch the time axis such that a point-to-point similarity of two time series is minimized [23, 4]. For any pair of series DTW yields some warping costs that can be considered as a distance between the series.
d) *Clustering using Markov Models:* Another approach is to learn a hidden Markov model (HMM) or Markov chain for each subsequence [28, 25] and to cluster via the resulting probability models.

In (a) and (b) usually distance metrics like Euclidean distance are used, which yield low distances only if one segment is almost an exact copy of the other. However, the shape of the subsequence is the main factor in perceived similarity rather than an exact match of the values. Two series may have the same shape although they have different baselines, for instance. Therefore, the attribute vector may be transformed, such as subtraction of the segment mean (to eliminate baseline effects) or normalization (zero mean and unit variance). Despite such transformation in our experiments the results were not very promising, since the distance measure is not able to assign low distance values to *dilated* window contents, resulting in very small clusters. Furthermore, only a fixed sliding window width is used to measure the similarity and quite different clusters may be found when using other window widths. When using comparatively few clusters the resulting prototypes often appear very much like dilated and translated trigonometric functions with different amplitudes. Thus, similar results can be obtained by using the first few Fourier coefficients to characterize the window content, such that we can benefit from a reduced data dimension from n to 3 or 4 as proposed in [1].

With respect to the inadequate handling of dilation, it seems more promising to embed a more abstract representation of the series as proposed in (b) rather than the raw data. The advantage of this approach is that it handles translation and dilation of the time series to some extent: A deviation of a prototype in the even components corresponds to a local shortening or lengthening of the profile, a deviation in an odd component corresponds to a scaling in the vertical axis. By using gradient information rather than absolute values we do not only compensate for different baselines in the segments, but address local differences in shape: If two segments distinguish only in a different slope at the beginning, but then behave structurally identical, their pointwise distance will be very high since the initial difference in slope separated both profiles spatially. However, the pointwise distance of their derivatives yields much smaller distance values, since the derivatives differ only at the beginning. Compared to the embedding of raw data, with embedding higher representations we have obtained much better cluster prototypes.

Translation and dilation can be handled very well by DTW techniques (c), which are especially suited to compare series of different length. However, vertical distortions in the function values tend to be compensated by time warping even if there is no temporal dilation present. Technique (c) does not transform a series into another representation, but defines an $n \times n$ symmetric dissimilarity matrix, where n is the number of series. Here, relational rather than partitional clustering algorithms have to be applied. But the space and time complexity of such algorithms is usually larger and if a window is slid along a very long time series a very large dissimilarity matrix has to be processed. Therefore it is a good idea to use subsampling to reduce run-time requirements as in [20].

HMMs assume that there is a fixed number of hidden (unobservable) states in which a system may currently be. The output of the system depends solely on the current state. Learning an HMM corresponds to estimating the state transition probabilities and state output probabilities via likelihood maximization. While this is a promising approach for long sequences, it may fail with short subsequences (windows) since they contain only little data to learn the high number of HMM parameters robustly. The state transition probabilities indirectly correspond to state durations and thus can be used to model time dilation. HMMs are used in speech recognition, replacing DTW techniques that have been used in this domain before the advent of HMMs. Instead of HMMs Markov chains can be learned from each subsequence – since Markov chains are simpler than HMMs (less parameters, no hidden states) they can also be learned from short subsequences. As with DTW it is possible to learn models from series of different lengths. By calculating the probability of being generated by the model for every series and estimated model, a dissimilarity matrix can be defined just as with the DTW approach. Difficulties arise from the fact that one has to select a priori a single model structure that is adequate for all clusters (same structure and number of hidden states).

Techniques (c) and (d) are computationally more expensive than (a) and (b). With respect to interpretability, all methods come up with a "codebook" of representative shapes, which is easily accessible only if the number of prototypes is moderately large.

4 Deductive Segmentation

In the deductive approach we seek for a segmentation at the zero-crossings of the first few derivatives. Noise introduces many such zero-crossings and therefore has to be eliminated in advance, either by function approximation or smoothing. There is a variety of different methods for function approximation or regression (see e.g. [6]). Having approximated the data by one of these methods we want to use the fitted function to extract shape information. For this approach making sense we require the following property: Any shape feature extracted from the approximating function must be supported by the original data. We will see that this property does not hold for most techniques.

The approximation can be either uniform (the approximating function f deviates from the original values by no more than a certain $\varepsilon > 0$) or least-squared fitted (sum of approximation errors is minimized but local error is not bounded in advance). In principle it is a good idea to require uniform approximation, since then the original data is approximated equally well over the complete range. Optimal uniform approximation can be quite complex, but there are also greedy, non-optimal variants. Figure 1 shows the results of an algorithm proposed in [27]. If ε has been chosen appropriately, the results are very good (example in the middle). From left to right we have increased the noise but left ε unchanged. Now, uniform approximation yields different representations for the data, although the underlying function remained the same. If ε is chosen too high (left example), the approximation gets somewhat sloppy[1], if ε is chosen too low (right example), the noise gets modelled. Ideally, we would like to get *identical* representations. In many knowledge discovery applications we cannot be sure that the amount of noise does not change over time.

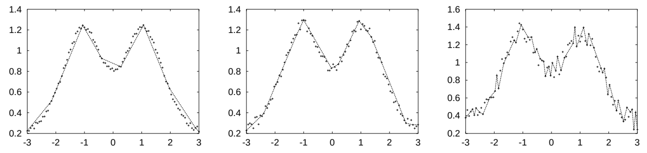


Figure 1. Uniform approximation (constant ε).

Instead of ε usually the number of segments k has to be provided with least-squares methods. If $k = 4$ is fixed then least squares methods would yield (almost) identical results in figure 1, however, in arbitrary situations determining an optimal k is not easier than fixing ε. Heuristic approaches can be used to determine whether the achieved approximation with some k is sufficiently good (again via some threshold), and some additional segments are inserted if that is not yet the case. Such an iterative refinement can also be implemented in a hierarchical (bottom-up or top-down) fashion.

Fourier and Wavelet analysis [19] can also be used to approximate the series. For noise elimination usually small coefficients of basis functions are set to zero (again via some a priori fixed threshold). Due to the locality of the wavelets, deleting coefficients does not affect the whole reconstructed time series as it is the case with the (co)sine basis functions in Fourier analysis. But since we are interested in determining the zero crossings in the time series, the wavelets should be smooth (unlike the Haar or Daubechies-4 wavelet) and should not have too many zero-crossings itself. Otherwise a 'smoothed' reconstruction of the time series exhibits a rough profile with many local extrema and inflection points that are not supported by the data itself.

Noise can also be removed by smoothing techniques. Convolving a signal with a low-pass filter [22] eliminates the high-frequent portions of the signal. However, there is an infinite number of filters from which one can choose, and each time a different 'smoothed' curve is obtained. For bad choices of filter coefficients it may even happen that the 'smoothing' process *increases* the number of local extrema, which does not only contradict our intuition, but also corresponds to an introduction of features not supported by the data. Besides that, smoothing also dislocates zero-crossings. Selecting the right filter is as difficulr as selecting the right ε in uniform approximation or the right number of segments k in least-squares approximation.

5 Multiscale Methods

Let us motivate multiscale methods by summarizing the drawbacks of the previously mentioned approaches. The methods in section 3 may become pretty expensive in terms of computational costs. Usually, the number of clusters will be quite large and thus prototypes

[1] Using this approximation we have a very bad estimate for the position of the local minimum, the extracted location is not supported by the raw data.

may become quite similar. The discrimination between shapes requires to compare *all* elements in the whole set of shapes, therefore the approach may become hard to understand for a human as the set is large.

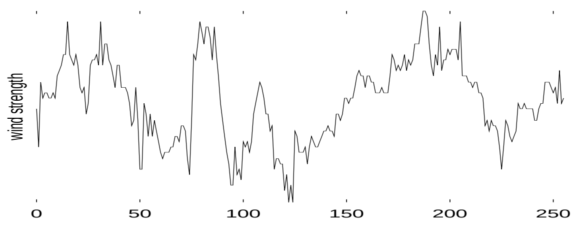
All methods in section 4 implicitly assume that the noise level does not change over time – only under this assumption one can select a representative data set and tailor the parameters (k, ε, filter coefficients) of the respective method. However, this is not a realistic assumption in data mining applications, where devices may record data over long periods of time and failures and equipment degradation may affect the amount of noise. Badly chosen parameters lead to abstractions that are not supported by the raw data (and thus hinder the further KDD process).

If the clustering algorithms are crisp, that is, every data object is assigned unambiguously to a cluster, then all methods from section 3 and 4 provide a single abstraction of the time series – although the correctness of the parameters and the applied heuristics cannot be guaranteed. A human is usually much more imprecise (or flexible) in the classification of shapes.

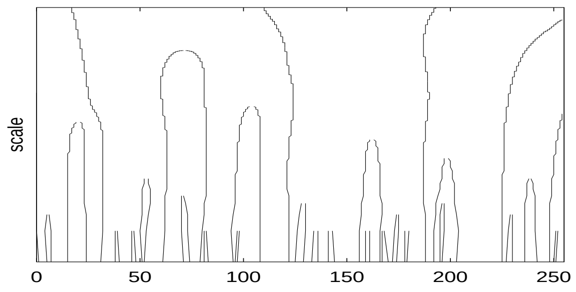
Recently, multiscale methods became increasingly popular, since they address the problems mentioned above. Instead of relying on a correct parameter setting and abstracting the time series at a single scale[2] multiple abstractions at multiple scales are derived. But different descriptions of the same time series cause ambiguity. Either the knowledge discovery method can handle ambiguity or the descriptions are compared to derive a most stable single description.

Although in principle one could use parameters like k or ε together with the respective methods, traditionally smoothing and thus the scale s of the filter is varied to obtain the different representations. A prerequisite of this approach is that the smoothing operation corresponds to our intuition, that is, peaks are only removed but not introduced (which may happen if the filter is not appropriate, as already mentioned in the previous section). It has been shown in [2] that for the continuous case the Gaussian kernel is the only kernel that guarantees that no zero crossings will be introduced in the smoothed signal. For the discrete case, Lindeberg [18] characterizes the *scale-space kernels* that guarantee this property.

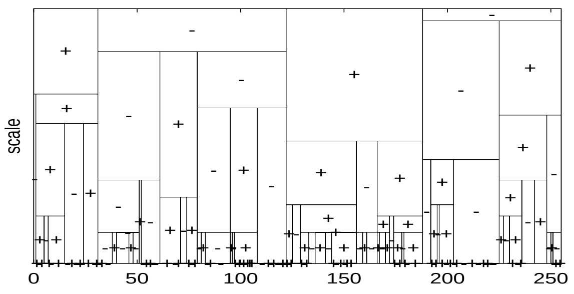
The advantage of a multiscale approach as proposed by Witkin [29] is that we can influence which features we want to hand over to the KDD process on the basis of some robustness criterion. To get reliable results, we have to make sure that the abstraction is not obtained "by chance" due to a good or bad parameter setting, but is robust against some variation. This is of special interest for large data volumes where we cannot fix the parameters in advance since the noise level may vary over time.



(a) A noisy signal.



(b) Location of zero-crossings versus scale (size of filter).



(c) Interval Tree of Scales.

Figure 2. Multiresolution analysis of a signal.

Compensating Dislocation We have already seen in figure 1 that badly chosen parameters result in misleading abstractions (the position of the local minimum in the left figure does not correspond to the position in the curve). This effect of *dislocation* also occurs with smoothing techniques.

By considering a curve (e.g. figure 2(a)) at multiple levels of smoothing and locating the positions of extrema for each of the resulting representations, it is possible to track the zero-crossing locations versus scale (the higher the scale the more smoothing has been applied). Rather than considering the zero-crossings in the different representations as a set of unrelated events, they can be treated as manifestations of the same physical events, which is why corresponding zero-crossings are connected with lines (fig. 2(b)). If no dislocation would take place, all these lines were perfectly straight.

Since minima and maxima can only vanish pairwise (it is not possible to have two local maxima without a minimum in between), any time interval between two lines either remains a single interval or is subdivided into three successor intervals as the scale decreases. We thus may construct a tree describing the successive partitioning of the signal into finer subintervals as new zero-crossings appear at finer scales. This allows us to compensate the spatial distortion of the zero-crossings at coarser scales: we use a coarser scale to identify important features, and use coarse-to-fine tracking to localize the features exactly in the raw data. Having identified the true locations, we can propagate them from the zero scale up to higher scales, thereby turning all contours into straight lines, as shown in figure 2(c). The resulting (ternary) tree is called *interval tree of scales* [29] and is shown in figure 2(c). The "+" and "−" signs in the rectangles indi-

[2] Here, scale refers to the width of a Gaussian filter in kernel smoothing, however, one could also say "at a single parameter setting".

cate whether the segment between the zero-crossings represents an increasing or decreasing segment. The rectangles tessellate the time-scale plane completely.

Stable Features Since we are interested in the robustness of the extracted features, we want to use the lifetime over which a feature persists versus scale as a measure of robustness or significance. Tall rectangles in the interval tree of scale persist against smoothing and thus can be considered as being robust, while small vertical expansion indicates that the interval will disappear soon as we start to smooth the data.

Once we have a numerical measure for segment stability [17], we can seek for a single scale which maximizes the mean stability. However, once we have done a multiscale analysis we do not have to restrict ourselves to features that appear in a single scale. Witkin proposes to descend the tree from the top to the bottom, as long as the mean lifetime of the offsprings is larger than the lifetime of any of the parents. The latter criterion gives signal descriptions that correspond very well to the human perception of the time series.

Thus, we use the interval tree of scales to convert a numerical time series into a stable symbolic description consisting of labeled intervals. The labels address increasing/decreasing behaviour if the first derivative is used, or concave/convex behaviour in case of the second derivative, or both. The description is *stable* in the sense that small changes in the scale parameter do not change the symbolic description. No thresholds were necessary. The time consuming kernel smoothing may be replaced by efficient wavelet analysis [3], allowing also for compensating the usual flattening effects of smoothing.

6 Conclusions

In this paper we have summarized a number of time series abstraction methods. We believe that (a) the use of higher-representations is very important if we want a human to *understand* the discovered relationships and think that this is a prerequisite for *knowledge* discovery. Since humans are used to work with graphical representations of time series, abstracted representations are necessary to account for the human way of perceiving time series. And while there are many different abstraction methods, we believe that (b) most of these methods are not very well suited for KDD and data mining, where assumptions about constant noise etc. are not valid: If data is measured over many years, the equipment may degrade and/or replaced, which influences the amount of noise. It is important to have a method that provides a robust and valid abstraction. Using multiscale methods in the deductive approach, robustness can be obtained by considering only features that persist over a broad range of scales, and validity of the extracted features is achieved by tracking the intervals from coarse to fine scales to compensate dislocation effects.

REFERENCES

[1] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In *Proc. of the 4th Int. Conf. on Foundations of Data Organizations and Algorithms*, pages 69–84, Chicago, Oct. 1993.

[2] J. Babaud, A. P. Witkin, M. Baudin, and R. O. Duda. Uniqueness of the Gaussian kernel for scale-space filtering. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(1):26–33, Jan. 1986.

[3] B. R. Bakshi and G. Stephanopoulos. Reasoning in time: Modelling, analysis, and pattern recognition of temporal process trends. In *Advances in Chemical Engineering*, volume 22, pages 485–548. Academic Press, Inc., 1995.

[4] D. J. Berndt and J. Clifford. Finding patterns in time series: A dynamic programming approach. In [10], chapter 9, pages 229–248. MIT Press, 1996.

[5] P. S. Bradley, U. M. Fayyad, and C. A. Reina. Scaling EM (expectation-maximization) clustering to large databases. Technical Report 15, Microsoft, Feb. 1999.

[6] V. S. Cherkassky and F. Mulier. *Learning from Data*. Wiley, 1998.

[7] G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth. Rule discovery from time series. In *Proc. of the 4th Int. Conf. on Knowl. Discovery and Data Mining*, pages 16–22. AAAI Press, 1998.

[8] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[9] M. Ester, H.-P. Kriegel, J. Sander, and X. Xiaowei. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery in Databases*, pages 226–331, Portland, Oregon, 1996.

[10] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. MIT Press, 1996.

[11] F. Höppner. Discovery of temporal patterns – learning rules about the qualitative behaviour of time series. In *Proc. of the 5th Europ. Conf. on Principles of Data Mining and Knowl. Discovery*, number 2168 in LNAI, pages 192–203, Freiburg, Germany, Sept. 2001. Springer.

[12] F. Höppner. Speeding up fuzzy c-means: Using a hierarchical data organisation to control the precision of membership calculation. *Fuzzy Sets and Systems*, 128(3):365–378, 2002.

[13] F. Höppner, F. Klawonn, R. Kruse, and T. A. Runkler. *Fuzzy Cluster Analysis*. John Wiley & Sons, Chichester, England, 1999.

[14] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[15] M. W. Kadous. Learning comprehensible descriptions of multivariate time series. In *Proc. of the 16th Int. Conf. on Machine Learning*, pages 454–463, 1999.

[16] P.-S. Kam and A. W.-C. Fu. Discovering temporal patterns for interval-based events. In *Proc. of the 2nd Int. Conf. on Data Warehousing and Knowl. Discovery*, volume 1874 of *LNCS*, pages 317–326. Springer, 2000.

[17] T. Lindeberg. Effective scale: A natural unit for measuring scale-space lifetime. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(10):1068–1074, Oct. 1993.

[18] T. Lindeberg. *Scale-Space Theory in Computer Vision*. Int. Series in Engineering and Computer Science, Robotics: Vision, Manipulation and Sensors. Kluwer Academic Publishers, Dordrecht, 1994.

[19] S. G. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, Inc., 2nd edition, 2001.

[20] T. Oates. Identifying distinctive subsequences in multivariate time series by clustering. In *Proc. of the 5th Int. Conf. on Knowl. Discovery and Data Mining*, pages 322–326, 1999.

[21] M. Ortolani, H. Hofer, D. Patterson, F. Höppner, and M. Berthold. Fuzzy information granules in time series data. In *Proc. of the World Congress on Computational Intelligence*, Honolulu, Hawai, May 2002.

[22] J. O. Ramsay and B. W. Silverman. *Functional Data Analysis*. Springer Series in Statistics. Springer, 1997.

[23] D. Sankoff and J. B. Kruskal. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison Wesley, 1983.

[24] I. Savnik, G. Lausen, H.-P. Kahle, H. Spiecker, and S. Hein. Algorithm for matching sets of time series. In *Int. Conf. on Principles of Data Mining and Knowledge Discovery*, pages 277–288, 2000.

[25] P. Sebastiani, M. Ramoni, P. R. Cohen, J. Warwick, and J. Davis. Discovering dynamics using bayesian clustering. In *Proc. of the 3rd Int. Symp. on Intelligent Data Analysis*, pages 199–209, Amsterdam, The Netherlands, 1999. Springer, Berlin.

[26] Y. Shahar and M. A. Musen. Knowledge-based temporal abstraction in clinical domains. *Artificial Intelligence in Medicine*, 8:267–298, 1996.

[27] J. Sklansky and V. Gonzalez. Fast polygonal approximation of digitized curves. *Pattern Recognition*, 12:327–331, 1980.

[28] P. Smyth. Clustering sequences with hidden markov models. In *Advances in Neural Information Processing 9*. MIT Press, 1997.

[29] A. P. Witkin. Scale space filtering. In *Proc. of the 8th Int. Joint Conf. on Artifial Intelligence*, pages 1019–1022, Karlsruhe, Germany, 1983.

[30] X. Xiaowei, M. Ester, H.-P. Kriegel, and J. Sander. A distribution-based clustering algorithm for mining in large spatial databases. In *Proceedings of the IEEE International Conference on Data Engineering*, pages 324–331, Orlando, Florida, 1998.