Shape Grammar Extraction for Efficient Query-by-Sketch Pattern Matching in Long Time Series

Prithiviraj K. Muthumanickam Katerina Vrotsou Matthew Cooper Jimmy Johansson*

Linköping University, Sweden

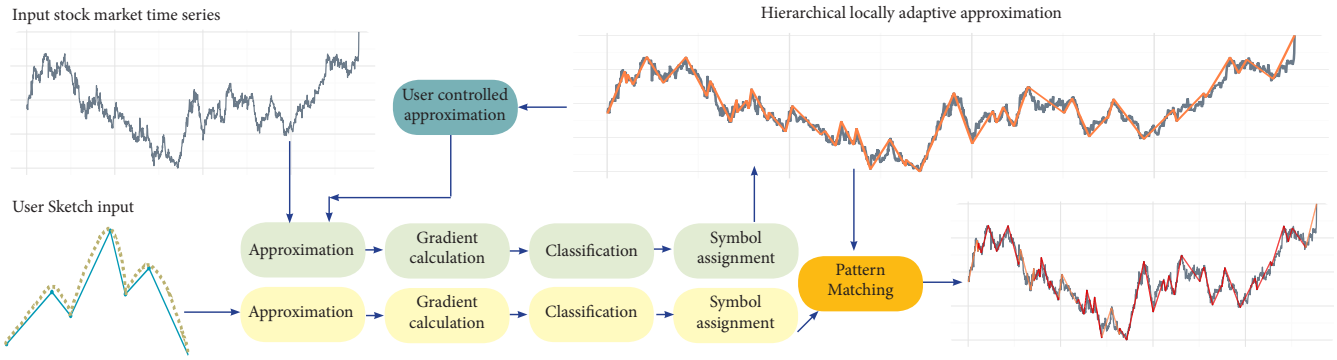


Figure 1: A schematic representation of the work flow. A sketched pattern of interest is matched to the time-series data. Efficient approximation, classification and symbol assignment, based on gradient ratios, enables real-time pattern searching within very large time-series. The steps depicted with green boxes are executed when a new input time series is loaded or when it undergoes hierarchical approximation, yellow boxes only when a new sketch is entered.

Abstract

Long time-series, involving thousands or even millions of time steps, are common in many application domains but remain very difficult to explore interactively. Often the analytical task in such data is to identify specific patterns, but this is a very complex and computationally difficult problem and so focusing the search in order to only identify interesting patterns is a common solution. We propose an efficient method for exploring user-sketched patterns, incorporating the domain expert's knowledge, in time series data through a shape grammar based approach. The shape grammar is extracted from the time series by considering the data as a combination of basic elementary shapes positioned across different amplitudes. We represent these basic shapes using a ratio value, perform binning on ratio values and apply a symbolic approximation. Our proposed method for pattern matching is amplitude-, scale- and translation-invariant and, since the pattern search and pattern constraint relaxation happen at the symbolic level, is very efficient permitting its use in a real-time/online system. We demonstrate the effectiveness of our method in a case study on stock market data although it is applicable to any numeric time series data.

Keywords: User-queries, Sketching, Time Series, Symbolic approximation, Regular Expression, Shape Grammar.

Index Terms: Information Systems [Information retrieval] ; Human-centered computing [Visualization] : Interaction Techniques—Visual analytics;

1 Introduction

Modern experiments, simulations and sensor networks in many application domains often produce very long time series data which can be very difficult to explore through visual representations, particularly when the data is non-periodic. Analysts are often seeking to identify short sequences of interest within these long time series, such as repeating 'motifs' or infrequent anomalies within the data. Time-series graphs, however, become cluttered when long time series are being explored and so the visual identification of interesting short sequences of samples can be almost impossible. For this reason researchers have developed a variety of algorithmic approaches which attempt to identify such interesting short sequences in time series using local characteristics of the samples. Such searches are often conducted using user-sketched sequences or sequence primitives but searching using these local characteristics can be computationally heavy, making each search a time-consuming problem and reducing the interactive exploration of the data to a rather tedious task. In this work we present an approach based on the extraction of a representative but simplified grammar, involving only a few tens of symbols, from the time series and search sequence. The search problem then reduces to a search within the grammar data, a process which can be completed in milliseconds even for time series involving millions of symbols. The interactive exploration process can then involve many searches all being carried out apparently instantaneously. This symbolic search also permits the analyst to interactively carry out a stepwise relaxation of the constraints placed upon the search to find more sequences that match the general shape of the search sequence but less precisely. This relaxation requires only an editing of the grammar and no recomputation from the time series and so can also be carried out in milliseconds, even for millions of time samples. Employing a freehand or stepwise sketch-based approach in the definition of the search sequence enables us to keep the domain user in the visual analytics loop by allowing them to define sequences of interest within their domain and exploration task. With simple controls to enable the creation of a sketched sequence, users can perform searches and carry out the search constraint relaxation processes to enable a full exploration of the data series. The main contributions are the following:

*e-mail:prithiviraj.muthumanickam, katerina.vrotsou, matthew.cooper, jimmy.johansson@liu.se

- **Ratio approximation and symbolic representation.** While searching for similar patterns, users often ignore differences across amplitude, scale and translation [7, 25]. Any pattern search algorithm should be able to handle these three conditions in an efficient manner. In our method, translation invariance can be handled using string matching algorithms, and scale and amplitude variance are handled through our shape grammar induced ratios. The ratios are used to transform the data into a symbolic representation. The relevance of amplitude, scale and translation invariance in pattern matching is described in [47] using the "GoalPost Fever" pattern in patient body temperature datasets.

- **User-sketched pattern matching.** Our method allows the user to sketch approximate patterns in a separate sketching space, either through line strips defined through a sequence of mouse clicks or free form sketching. Ratio approximation and symbolic representation are applied to the user input and a string matching, based on regular expressions (RE), is then performed in real-time to identify similar patterns.

- **Hierarchical locally adaptive approximation.** Users are given the flexibility of performing local approximation of raw time series data in a hierarchical manner. No extra arithmetic calculation is necessary since the entire approximation process is performed on the symbolic data using RE.

The remainder of the paper is structured as follows. In Section 2, we will discuss previous work related to dimensionality reduction, shape grammar based pattern matching, user sketch based search and regular expression based queries. In Section 3 we present an overview of our new approach. In Section 4 we describe our shape grammar algorithm for symbolic approximation. Section 5 presents our intuitive approach for user input based, locally adaptive hierarchical approximation and the flexibility of our shape grammar in handling noisy input data. In Section 6 we describe the user-sketch based pattern input and the RE based string matching for finding patterns similar to the input sketch. Section 7 describes a case study on a stock market data set to demonstrate the effectiveness of our approach. Section 8 discusses the performance and scalability of our algorithm and is followed by a final concluding section.

2 Related Work

Time series data mining is a vast field with numerous papers targeted at query-by-content, clustering, classification, segmentation, prediction, anomaly detection and motif discovery. A thorough review of time series data mining can be found in [10, 34, 12, 14]. In our work we are interested in user interaction based pattern matching and hence our method falls under Query-By-Content (QBC), specifically where the input user queries are entered in the form of sketches. This section presents the most closely related work.

2.1 Dimensionality Reduction

For high-dimensional time series working directly on the raw data is computationally expensive. Hence it is common to reduce its dimensionality by converting it to a lower-dimensional representation [10]. One of the most popular methods for dimensionality reduction is symbolic approximation of time series (SAX) [32]. SAX achieves dimensionality reduction by segmenting the data based on a user specified segment length, assigning a symbol to each segment, and uses a sliding window of user specified length to generate a symbolic approximation. Different methods have tried to visualize the symbolically approximated time series data using bitmaps [26], VizTree based representations [33] and coloured rectangles [17] that allow users to visually explore the data. The SAX algorithm uses piecewise aggregate approximation (PAA) [23] for dimensionality reduction, but this method cannot capture local trends, and in turn the overall general shape of the time series, due to smoothing of perceptually important points (PIP) [13]. In order to overcome this limitation recent methods suggest storing additional data with each symbol, such as slope information, maximum and minimum amplitude points, regression and PIPs [3, 42, 29, 53, 48, 35, 37]. In [30] Li et al. stored additional information along with SAX symbols and visually represent them using Sector Visualization and VizTree. Identification of repeating patterns and anomalies [45] can be performed through grammar based approaches as in Sequitur [41] but, again, they are greedy grammar induction algorithms and so they are not always complete [46]. Searching for similar motifs using partial match criterion by modifying the Sequitur algorithm were implemented in [2] without the loss of original performance. This approach helps us to find sufficiently similar motifs with a tolerance margin for distortion in the presence of noise. Finding all possible patterns using a smallest complete grammar is an NP hard problem and [31] improves upon the Sequitur algorithm by considering trigrams (sets of three symbols that appear consecutively in a sequence), but the method still does not generate a smallest complete grammar. Methods such as [54] operate on large time series to find clusters of similar patterns. The dimensionality of the data, along with inherent noise, is reduced using the Douglas-Pecker algorithm which also retains the PIPs. Data segments in multiscale data from the systems biology domain were described symbolically by the sign of its slope in [36]. Even though they use a simple symbolic approximation based on trend similarity for piecewise linearly approximated segments [28], they lack query-by-sketch support and a shape grammar to search for user-defined patterns at different scales.

2.2 User-Sketched Pattern Matching in Time Series

Semi-automatic approaches, such as user-sketched pattern matching in time series data, play an important role in specifying the input pattern and searching for patterns of interest. Different user input mechanisms have been used in sketching over a particular part of the raw data. Rectangular selection techniques called Timeboxes were used to specify the extent of time points on the horizontal axis and the range of values on the vertical axis [18, 19]. The rigidity of the query specification of this technique was improved upon using Variable timeboxes in [24], angular queries and slopes in [20], options for query adjustment in [3], timeboxes augmented by predefined shape templates in [4], while Querylines [43] accept soft-constraints along with user preferences while, at the same time, ranking the resulting matches based on their importance. In [21] Holz et al. provided a relaxed selection technique that allows the user to pan through the data to find patterns of interest, sketch an approximate pattern over them, along with circles placed along the sketched line that support user relaxation. But this approach provides limited scale invariance due to the use of circles for query relaxation and to adapt a query on the screen pixel-precise input was necessary. When users know the pattern of interest it is common to manually sketch an approximate pattern and search for similar patterns in the time series data [51]. Some methods identify the most common shapes such as spikes, sinks, rise, drop, plateau, valley and gaps, and display them as pattern templates [16]. Users can then search for matches in the data. In [40] a tool was described that includes the facility for pattern templates that can be created and stored by domain users. User-sketched queries were used to perform comparison of rankings of pattern matches produced by algorithms against human-annotated rankings. It was concluded that human annotated rankings can differ drastically from algorithmically generated rankings [9].

2.3 Shape Grammar Based Pattern Matching

The shape definition language of Agrawal et al. [1] provides a language to describe an alphabet that can be constructed using a set

of symbols and operators. Shape grammar based approaches have been used in song retrieval by rhythm in music databases [6] where basic units of rhythm strings called *mubols* were used as building blocks. They are constructed by ignoring the pitch value of the notes and by considering only the duration of each note. They then propose methods for exact matching or k-similar matching of rhythm strings.

2.4 Regular Expression Based Query Matching

The work on approximate queries [47] finds a possible set of shapes that are represented using curves or functions and uses regular expression queries for pattern matching but does not provide a user interface for a visual query language in which the user can draw the shape of the sequence they are looking for. Such a language can allow the user to point out the important dimensions for comparison and the error tolerance in each dimension. Methods such as [27] search for temporal trends in multivariate time-varying data and trend specification is either through domain users who identify certain common trends in the data or using a clustering algorithm. But they highlight the need to use regular expressions for manual specification of any kind of input user queries. The advantages, such as intuitiveness and flexibility (fuzzy queries), of using regular expressions for specification of user queries are discussed in [15].

3 Our Approach

We extend our idea presented previously as a poster abstract [22] with additional details and significant improvement. User queries using sketches is an interactive method for creating search patterns in time series data. While regular expression based user queries can be intuitive and flexible, as discussed in [15], the method described in our paper allows a user to search for any pattern of interest by sketching an approximation of it in a graphical user interface. Previous user query methods used the extensive process of browsing through the entire data to find patterns of interest, specify methods for selecting such patterns and search for them in the rest of the data. As this method is tiresome, there is a need for creating an intuitive visual interface to sketch any patterns of interest by the domain specialist. So it becomes imperative to create a shape grammar based approach for user-sketched pattern matching and data smoothing. Since we are dealing with user input based queries, preserving the shape characteristics on a local scale is important. While the discussed dimenionality reduction methods were effective in reducing the size of raw data by transforming them to a different representation, the need for preserving the shape characteristics on a local scale is important for a user query based system. As we are interested in a real-time system involving domain user interaction, the computational cost should also be minimal. Our method approaches this problem using a shape grammar based method that is optimized for user-sketched pattern matching and we take inspiration from the symbolic approximation of time series, shape grammar based pattern matching, regular expression based queries and semi-automatic user sketched pattern matching algorithms. Our method has been implemented within a web-based interface and involves four basic steps. **(1)** Initially the time series data to be explored is simplified into a symbolic representation by computing ratios of the gradients between adjacent time series points. **(2)** Classification of the gradient ratios into a set of basic cases making up the building blocks of any time series. **(3)** As we are interested in searching for local trends in the input data, we perform a progressive, hierarchical, locally adaptive, user controlled smoothing operation on the raw data. The computation is performed in the symbol space using regular expressions instead of performing arithmetic computations on raw data. **(4)** Following this, a user can search for patterns of interest in the time series by sketching an approximate pattern. Translation, amplitude and scale-invariant pattern matching is achieved using regular expressions at the symbolic level. The previous step can be repeated to iteratively smooth the data to search for long term trends. We demonstrate the effectiveness of our approach in a case study on stock market data although it is applicable to any numeric time series data. While many previous visual analytical methods [44, 54, 55] on financial data analysis proposed many visualization models for finding patterns in time series data, our method is based on simple user queries for finding patterns in the data.

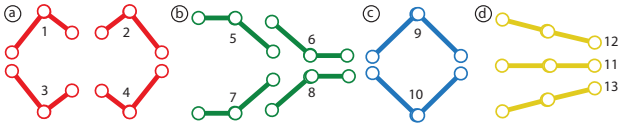


Figure 2: The set of unique building block cases numbered from 1-13 are identified as elementary shapes within a time series. Each case containing two gradient slopes is the smallest unit that can be represented as a ratio value. The range of values from 0.0-1.0 can be divided and approximated into 'n' symbols. **(a)**Ratio value computed by dividing absolute values of lowest gradient by highest gradient. **(b)**Ratio value is the absolute value of non-zero gradient. **(c)**Ratio value of absolute gradients is 1.0 and represented using a single symbol. **(d)**Linearly increasing, decreasing and zero gradients, each approximated to a single symbol. These are the simple building blocks of an intuitive user sketch based input. s_1 *to* s_{13} are the set of symbols used for approximating each building block case.

4 Symbolic Approximation using Shape Grammar

The idea behind a visual query language is to provide a simple sketching interface allowing domain-experts to roughly draw a pattern of interest. This pattern can then be matched against the raw input data and similar patterns that are invariant across scale, amplitude and translation are identified. As the method is based on user input and interaction, the above process should be intuitive and perform in real time. Input from user-sketching will be in the form of three simple shapes: linearly increasing, decreasing, constant and their mutual combinations that form a visual grammar which conforms to an intuitive user-input model. Also the visual representation of time series is a combination of the above three simple shapes. In the process of building this visual grammar, we identify both the user sketch and time series data as a combination of basic elementary shapes that are positioned across different amplitudes (see Figure 2). The elementary shapes are constructed in a simple manner so that each of them can be represented as a ratio value that aids in scale invariant pattern matching and also can be reduced to symbols (see Figure 5). The basic building block contains only 13 patterns, rather than other possible combinations. We identify only the minimal number of building block patterns that can be used to build any time series representation. Such a minimal definition is needed because it helps us to build simple regular expressions to perform data smoothing at later stages in the algorithm to handle the inherent noise in the data. We now describe the process of our approach in detail as illustrated in Figure 1.

4.1 Normalization and Approximation

The input data and user sketched pattern are first normalized to the range 0.0 to 1.0. The nature of the input can either be standalone or streaming data and our algorithm is well suited to handle both cases. We perform an initial approximation on both the input data and the user sketch, where consecutive positive/negative gradients of different lengths across n time steps are replaced with n linear gradients of equal length as shown in Figure 4. The dissimilarities in pattern matches that arise from amplitude and translation are handled using the ratio of adjacent gradients, while the scaling issue is addressed by our approximation step.

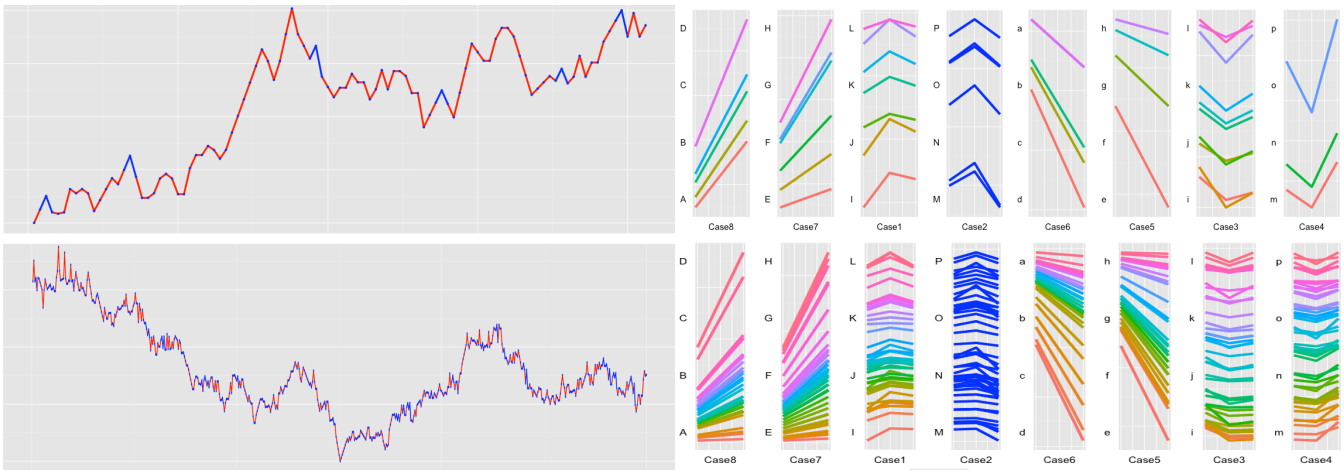


Figure 3: **Top:** Small random input time series of length *103*. Adjacent gradients that fall under any of the multi-symbol cases of Figure 2 are mapped to their corresponding columns. For example, 6 instances of Case 2 are highlighted in blue on the time series graph towards the left of the Figure and these 6 instances are binned accordingly based on their ratio values and highlighted in blue towards the right. **Bottom:** Random input time series of length *600*. The bins on the bottom right contain ratio values that are almost uniformly distributed across the entire value range of ratio values from 0.0 to 1.0. We observe that when the size of the input data becomes large there is a basic building block shape for every possible ratio value.

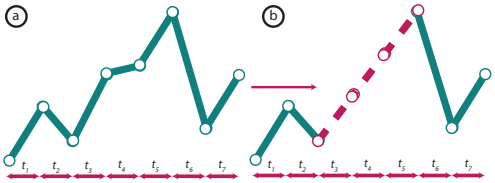


Figure 4: Initial approximation of consecutive positive or negative gradients. (a) Input data, (b) Approximated data.

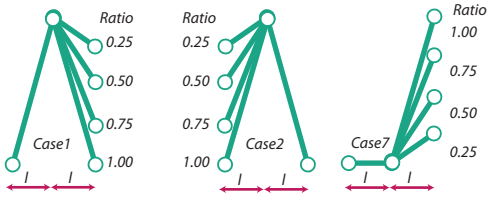


Figure 5: Example cases for ratio calculation: Each building block case is represented by a ratio value from 0.0 to 1.0. A smaller gradient followed by larger gradient and vice-versa are differentiated into separate building block cases meaning that the ratio value is always between 0.0 and 1.0. Ratio values falling under a certain range, for example 0.0 to 0.25, can be approximated to a single symbol.

4.2 Basic Building Blocks of a Time Series

After the normalization and initial approximation step, we calculate the gradients g_j and g_{j+1} from time series points t_j, t_{j+1}, t_{j+2} by sliding a window with a length of three time points across the time series. Since we consider equal time steps, the gradient between two time series point is $g_j = \frac{amplitude(t_{j+1}) - amplitude(t_j)}{d}$, where d is the distance between two time steps. g_j and g_{j+1} can be approximated to a single ratio value based on the classification shown in Figure 2. With our sliding window step, we compute the ratios of all gradients, g_j and g_{j+1}, falling under each particular case. We find the smallest among g_j and g_{j+1} and then compute the ratio value. For example, consider the building blocks in Figure 2(a), case 1: $g_j > g_{j+1}$, the ratio value is g_{j+1}/g_j, case 2: $g_j < g_{j+1}$, the ratio value is g_j/g_{j+1}. In case 1, the entire group of building blocks having $g_j > g_{j+1}$ can be represented using ratio values in the range 0.0 to 1.0. In Figure 2(b), where any one of the gradients g_j or g_{j+1} is zero, we consider the non-zero gradient to be our current ratio value. Since d is uniform we consider d=*1.0* and hence the range of ratio values for this case will also be between 0.0 and 1.0.

4.3 Classification and Symbol Assignment

Each set of three time series points t_j, t_{j+1}, t_{j+2} provided by the sliding window step is categorized into its respective building block case as described in the previous section and the ratio is computed. Building blocks of a particular case that are similar to each other will have similar ratio values and hence can be approximated using a symbol. Also, when the size of the input time series increases, the ratio values are distributed across the entire range from 0.0-1.0 (see Figure 3). This removes the need for explicit binning algorithms and hence a simple uniform binning can be applied. For example, if the size of the alphabet for a particular building block bin (case 1) is *4*, the uniform binning algorithm will approximate the ratio values falling under *0.0-0.25* to *a*, *0.25-0.5* to *b*, *0.5-0.75* to *c* and *0.75-1.0* to *d*. Uniform binning is applied to all the building block cases except the cases 9-13 in Figure 2 since they can be represented with a symbol. Cases 9 and 10 have equal absolute value of the gradients, g_j and g_{j+1}, and hence the same amplitude value. Since we are interested in scale invariant matches, all these cases containing different amplitude values will be approximated to a symbol. Cases 11, 12 and 13 form the simple building blocks of linearly increasing or decreasing, and zero gradients and each one of them can also be approximated to a symbol. The size of distinct alphabets required for symbolic approximation of a particular building block can be kept to a minimum due to the high resemblance in user specification of a building block during sketching. At this stage, the input time series data and user-sketched pattern have been converted to an approximate symbolic representation.

4.4 Building Blocks in the form of Regular Expressions

One interesting feature of our visual query language is that, due to the approximation step and shape definition based classification, the time series data can be expressed as a regular expression (RE) of the form below,

$$(z*|p*|n*)?(s(z*|p*|n*))* \quad (1)$$

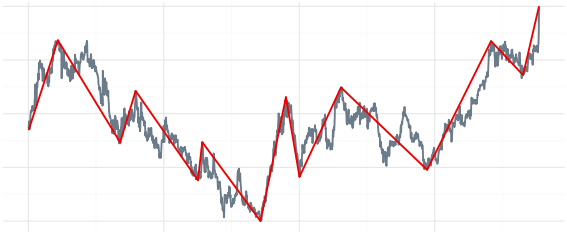
where the symbols z, p, n represent the zero, positive and negative gradient shapes *(cases 11, 12 and 13)*, respectively, and s represents the *cases 1 to 10* in Figure 2. ? represents one or zero occurrences while $*$ represents zero or more occurrence. This common representation is exploited for RE based string matching and hierarchical approximation of time series in the following sections. When we construct a finite automaton from our RE the output is a deterministic finite automaton (DFA).

5 Smoothing the Data Using Shape Grammar

Two primary concerns need to be addressed while pattern matching the user sketch with input time series data, **(1)** As we are interested in searching for local trends in the time series data, insignificant spikes and valleys that would not alter the local trend can be approximated. Since they are approximated to symbols, as explained earlier, they are a hindrance to our regular expression (RE) based shape matching. **(2)** A domain user may be interested in finding short term trends across a few time points or long term trends across a larger time interval and hence a user input based hierarchical smoothing will help them to search for trends across short or long time intervals. By taking these two factors into consideration we perform a hierarchical locally adaptive approximation, i.e. the input data can undergo smoothing in a hierarchical fashion based on user interaction. The advantage of performing a locally adaptive approximation rather than a piecewise adaptive approximation (PAA) is that the latter can potentially smooth out important points in the data set due to the constant sized bins used to compute the average of all the time points within that bin. Smoothing of time series data by dimension reduction techniques such as the Douglas-Peucker (DP) algorithm [8] has been used in time series simplification methods [49, 54]. While this is an effective method for time series smoothing, our algorithm operates on a symbolic approximation of the raw data and so avoids arithmetic calculation at each approximation step. The advantage of our approach lies in the central theme of using REs for user-query specification, smoothing and pattern search in time series data. RE based user-query specification is intuitive, flexible (fuzzy queries) [15] and allows pattern searching based on RE based string matching which is efficient [50]. The distance epsilon parameter in DP is similar to that of our ratio approximation threshold used for controlling the error induced in each hierarchical smoothing step. Figure 6 portrays the hierarchical smoothing applied to stock market data where *(a)* depicts the minimum level of approximation and *(c)* the near maximum level of approximation. The stock market data covers a time period of roughly 7 years and hence the maximum level of smoothing can be used to find a pattern that extends over a long period of time. Figures 7(a), 7(b), 7(c) show the smoothing algorithm applied to other kinds of input data (Datasets courtesy of [52, 46]). The algorithm is explained below.

(a) Least level of approximation

(b) Medium level of approximation



(c) Near Maximum level of approximation

Figure 6: Multiple levels of smoothing on Stock market data. The chart at the top depicts the minimum level of approximation and after every user input based smoothing iteration, the approximation increases. The red lines indicate the approximated input data.

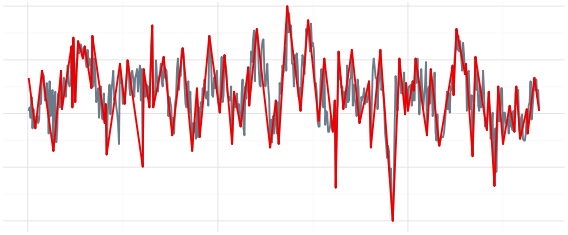
5.1 Approximation algorithm

Smoothing the input data can be seen as a transformation of *cases 1 to 8* in Figure 2 represented using multiple alphabets/ratios into *cases 9 to 13* that can be represented using a single alphabet. Hence the smoothing process is a further simplification of the symbolically approximated data. Our approximation algorithm operates in the symbol space of the raw data. While performing the smoothing operation of symbolic data, sufficient care should be taken to ensure that the minimum amount of smoothing error is introduced at each approximation step. We always consider a combination of two building block cases from Figure 2 for approximation as shown in Figure 9. We started our symbolic approximation algorithm in the previous section with two consecutive gradients of raw data for symbolization and approximation. Now, in our next step of smoothing of symbolic data, we consider two consecutive symbols corresponding to building blocks for smoothing. In Figure 9(a) the pattern is a combination of building block cases 1 and 4. While carrying out the approximation, we follow a general rule that minimizes the smoothing error at each step as portrayed in Figure 9. **(1)** In *Figure 9(a)* the ratio of gradients g_2/g_1 and g_2/g_3 is less than zero and the pattern is approximated to a positive or negative gradient slope accordingly. **(2)** In *Figures 9(b) and (c)* the ratio of gradients g_2/g_1 and g_2/g_3 is greater than zero and so the pattern is approximated to a positive or negative gradient slope based on a user defined ratio approximation value (see Section 5.4). **(3)** In *Figures 9(d) and (e)* one of the ratios of gradients g_2/g_1 and g_2/g_3 is greater than zero and the other less than zero. In such scenarios, there is a high smoothing error and they are skipped for the next iteration of the approximation step. In addition, two building block combinations are the basic unit of symbolically approximated data and by considering only two of the fundamental building blocks at a time, we minimize the local approximation error. Combinations of building block shapes can extend across two adjacent symbols (four time series points) or can be scaled and extend across multiple symbols (many time series points). In the former case, a sliding window and alphabet comparison can be performed to smooth the data, while in the latter case regular expressions (RE) ought to be used. Even though the two adjacent symbols can also be approximated using REs in a single step, we prefer to separate them into two steps. The reason for separating the adjacent symbol and RE based approximation is due to the fact that a smoothing step to approximate the two adjacent symbols can consume other nearby trends over long intervals as well. So we first remove the scenarios pertaining to two symbol cases and then continue with RE based

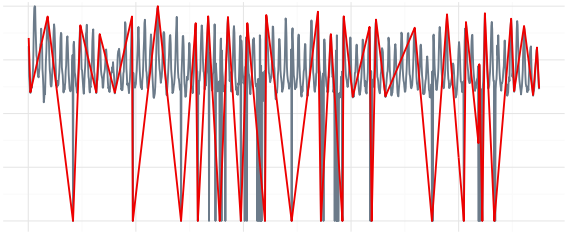
approximation. The two step process is explained below.

5.2 Approximation of consecutive symbols

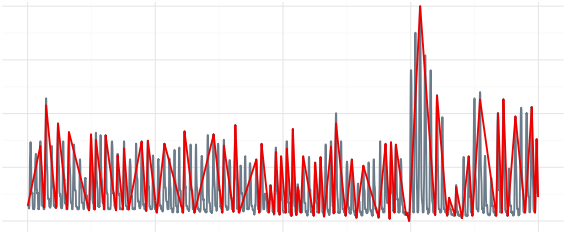
We start by sliding a window of length two across the symbolic approximation. All the patterns in column 1 of Figure 8 are approximated to their counterparts in column 3. While performing the approximation, the original time series data conforming to the particular alphabet set is updated with positive, negative or constant slope. The index of a symbol in approximated space corresponds to the starting time series point of the shape in the raw data.



(a) Data smoothing of monthly values of the The Southern Oscillation Index.



(b) Data smoothing of 10-day mean light intensity recordings of Carinae variable star.



(c) nprs43 data smoothing. Similar trends in the neighboring regions are approximated, while the disparities in trends in the neighboring regions are retained for the next smoothing iteration.

Figure 7: Locally adaptive data smoothing across different datasets. The red lines indicate the approximated input data. Significant anomalies are still retained in the approximation step.

5.3 Approximation using regular expressions

When a combination of two building block cases extend across multiple symbols, regular expression (RE) based string matching can be performed. We apply a sliding window based approach by employing RE based prefix match function to check for matches. All the REs mentioned in column 2 of Figure 8 are applied to find a match. The last two entries of the column contain only two combinations of positive, negative and zero gradient slopes while the other combinations are omitted due to space constraints. If there is a match, the set of symbols corresponding to the match are replaced with symbols for positive, negative or constant slope. The next RE prefix match then starts from the end position of the previous match. Based on the user input for approximation error (see below section), if we cannot approximate a RE, then the index is moved to the immediate symbol that does not correspond to positive, negative or constant slope of RE. Algorithm 1 (next page) explains the above process step by step.

5.4 User controlled approximation

As explained in Section 5.1, in order to minimize the local smoothing error we omitted the cases *(d) and (e)* of Figure 9 from further approximation. We do, however, give the user the ability to control the approximation in cases *(b) and (c)* by setting a ratio approximation threshold below which the patterns will not be approximated further. In Figures 9(a) and (b) the ratio between gradient ratios $rg_1/rg_2$where $rg_1 = g_2/g_1$ and $rg_2 = g_2/g_3$ is close to 1.0 while in Figure 9(c) it is approximatedly *0.5*. If we set the ratio threshold to be *0.25*, then the above cases will also be approximated. After the approximation the raw time series points are updated accordingly and the symbolic approximation step is re-run to generate the new symbolic approximation.

Pattern for approximation	*Regular expression for pattern search*	*Pattern after approximation*
1	$((s_{11})^*(s_7)(s_{13})^*(s_8)(s_{11})^*)$	
2	$((s_{13})^*(s_8)(s_{11})^*(s_7)(s_{13})^*)$	
3	$((s_{12})^*(s_6)(s_{11})^*(s_5)(s_{12})^*)$	
4	$((s_{11})^*(s_5)(s_{12})^*(s_6)(s_{11})^*)$	
5	$((s_{13})^*(s_1\|s_2\|s_9)(s_{12})^*$ $(s_3\|s_4\|s_{10})(s_{13})^*)$	
6	$((s_{12})^*(s_3\|s_4\|s_{10})(s_{13})^*$ $(s_1\|s_2\|s_9)(s_{12})^*)$	
7	$((s_{12})^*(s_6)(s_{11})^*)$	
8	$((s_{13})^*(s_8)(s_{11})^*)$	

Figure 8: First column: Combination of two building block shapes that can be approximated to their counterparts in the third column. Second column: Corresponding regular expressions used to search for such patterns in the symbolically approximated data using a sliding window based approach.

5.5 Final symbolic approximation for pattern matching

We perform the smoothing operation on the symbolic approximation of the raw data and all the building blocks from case 1 to case 10, each represented by multiple symbols, approximated to a single symbol building block case of 11, 12 and 13. At this stage, the symbolic approximation will contain chunks of symbols s_{11}, s_{12} and s_{13} and there is always a symbol associated with every time interval of the original data as shown in Figure 10(b). At a higher level of approximation, this increases further and hence it is redundant to use this symbolic approximation for pattern matching using regular expressions (RE). We can shorten this even fur-

ther by running our symbolic approximation of Section 4, except in this case we use regular expressions for searching for each individual building block case of Figure 2. We can do this by using one RE for cases 1 and 2, one RE for cases 3 and 4, and one RE for each case from 5 to 8. For example, cases 1 and 2 have the RE $s^*_{11}(s_1|s_2|s_9)s^*_{12}$. Based on the gradients of matching patterns, we can classify the building blocks as shown in Figure 10(c). This extra step also avoids false matches during user sketch based pattern matching. Since the smoothing operation is carried out in the symbol space, in order to find the ratios of building blocks we index the original timeseries data point. We can also use the length of the RE matches to find approximate ratios in case we wish to discard the raw data after the initial approximation.

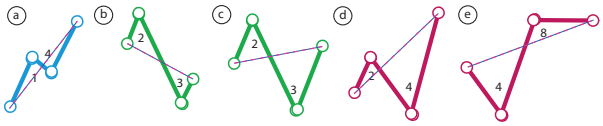


Figure 9: Controlled approximation: At each stage of approximation we make sure that minimum error is introduced, thereby preventing the shape of local trends from distortion. *(a), (b) and (c)* have the minimum level of approximation error while *(d) and (e)* have higher levels of smoothing error.

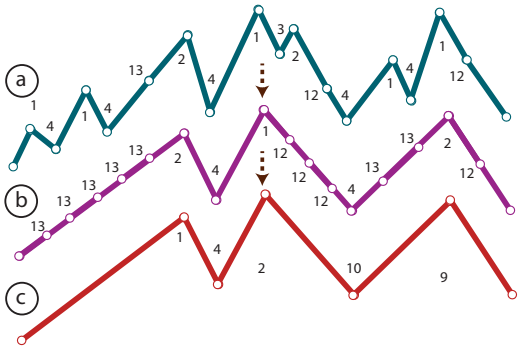


Figure 10: Final approximation step: (a) Raw time series data. (b) After smoothing operation on symbolic data (c) Final approximation step where the symbolic data in itself is approximated further thereby removing redundant data for string search and avoiding false matches. Numbers indicate the building block cases corresponding to the adjacent gradient pairs.

6 User Sketched Pattern Matching

After setting the input time series to the desired level of approximation, users can search for specific trends in the data by sketching a rough pattern. Our application interface, shown in Figure 12(a), consists of an input data display, a user sketching space and interaction controls. A user can draw a rough sketch of the sought pattern in the sketching space and it is processed in the same way as the input time series data. Our algorithm does not impose hard constraints on the patterns and the users are free to draw only a rough sketch of the pattern. Such soft-constraints reduce the need to perform tiresome user interaction during user sketching. Our interface allows the user to sketch input data using line strips constructed with mouse clicks or perform a free form sketching by holding the mouse button and drawing a rough sketch of interest. The later case is handled using finite difference methods (FDM) [38] to compute the skeleton points representing the input. Since we deal with data generated at uniform time intervals, we convert the input data to the same format by adjusting the horizontal coordinates of the input points into equal spacing. The sketch is converted to a symbolic approximation in the same way as the input time series data. Since we use regular expression (RE) based pattern matching, the symbolic representation of the user sketch is converted to a RE format. As our method does not impose hard constraints during the pattern search, sufficient care should be taken while building the REs to find similar matches. Let us take the so-called 'Head and Shoulders' pattern from the financial domain as an example. When constructing the RE with soft constraints, sufficient care should be taken to avoid false matches. Every odd number alphabet of the symbolic data is critical in maintaining the overall shape and trend of the pattern. In Figure 11(a) contains a sample input user sketch. The building block cases of the same are numbered accordingly. In (b) we find that the 1st, 3rd and 5th building blocks are updated to their counterparts and still the overall trend of the pattern remains the same. Cases 1 and 2 are upward patterns, while cases 3 and 4 are downward patterns and they are treated as similar patterns in odd number building blocks. Figure 11(c) clearly portrays the change in the overall trends of the pattern when the even number building blocks are replaced with their equivalent counterparts. Such an approach helps to avoid false positives being matched through REs. For this example, the RE for user sketched pattern will be,

$$(s_1|s_2)^+(s_4)^+(s_1|s_2)^+(s_3)^+(s_1|s_2)^+ \quad (2)$$

Using this pattern as input, we perform a RE based string matching with the user controlled symbolically approximated input time series data. The resulting matches are then highlighted accordingly in the input data as shown in Figure 12(a). Our approach provides

Algorithm 1 User input based data smoothing

```
1: function SMOOTHING(sym, data)
2:     i ← 0
3:     while i ≠ symbol.length do
4:         if ((m = rE1.matchAsPrefix(sym,i)) ≠ null)∨
5:         ((m = rE2.matchAsPrefix(sym,i)) ≠ null)∨
6:         ((m = rE3.matchAsPrefix(sym,i)) ≠ null)∨
7:         ((m = rE4.matchAsPrefix(sym,i)) ≠ null) then
8:             i ← m.end
9:             updateRawData(data, m.start, m.end)
10:        else if ((m = rE5.matchAsPrefix(sym,i))) ≠ null∨
11:        ((m = rE6.matchAsPrefix(sym,i))) ≠ null then
12:            Calculate length of gradients
13:            Calculate ratio of length of gradients
14:            if (ratio1 ≤ 1.0) ∧ (ratio2 ≤ 1.0) then
15:                i ← m.end
16:                updateRawData(data, m.start, m.end)
17:            else if (ratio1 > 1.0) ∧ (ratio2 > 1.0) then
18:                if (ratio1/ratio2) ≥ ratioThreshold then
19:                    i ← m.end
20:                    updateRawData(data, m.start, m.end)
21:                else
22:                    i ← m.nextAlphabet
23:                end if
24:            end if
25:            i ← m.nextAlphabet
26:            updateRawData(data, m.nextalphabet, m.end)
27:        else if ((m = rE7.matchAsPrefix(sym,i)) ≠ null)∨
28:        ((m = rE8.matchAsPrefix(sym,i)) ≠ null) then
29:            i ← m.nextAlphabet
30:            updateRawData(data, m.nextalphabet, m.end)
31:        else
32:            i ← m.nextAlphabet
33:        end if
34:    end while
35: end function
```

the user with flexibility for relaxation by allowing them to search for similar matches with soft constraints. Users can perform a hierarchical approximation of the input time series data to find patterns covering a long time interval. In contrast to the previous approaches, our user relaxation approach does not involve arithmetic re-computation. One disadvantage with our method, however, is that the string matching algorithm skips overlapping matches but this can be overcome with a small modification whereby we always check neighboring set of symbols for a match.

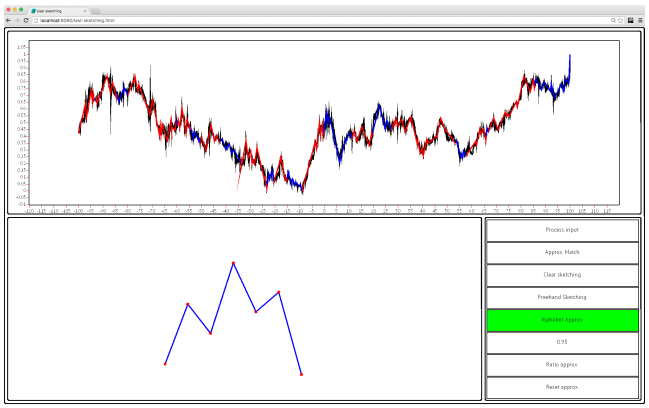
Figure 11: Construction of REs for pattern matching. Alphabets indicate building block cases. (a) Input sketch. (b) The overall trend of the pattern remains unchanged even when odd number building blocks are updated. (c) When even number building blocks are changed, the overall trend of the pattern can become distorted.

7 Case Study

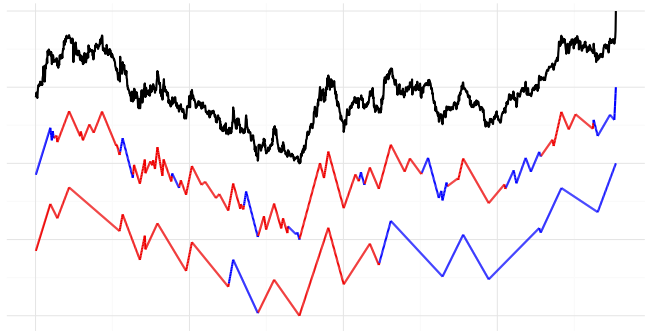
We will study some test cases that demonstrate our method with an input data from the stock market as shown in Figure 12(a). The data was downloaded from the NYSE database for the ticker symbol ADM from 1st May 1997 to 30th October 2004. The input data consists of 1886 time points. One of the prime features of the algorithm is the ability to identify both short and long term patterns with soft constraints that are invariant across scale, amplitude and translation. The input will be in the form of rough user sketches and a RE based string matching is performed in real-time to search for matches. The original raw time series data will be displayed in black, the hierarchically approximated time series in blue and the pattern matches in red. **Case 1: 'Head and shoulders' pattern.** This pattern in the financial domain is made up of a peak followed by a higher peak and then a lower peak. Domain users can draw a rough pattern of it in the sketching space followed by setting a desired level of approximation and smoothing on the raw time series data as shown in the Figure 12(a). Also, we can notice the neighbouring overlapped matches are not highlighted, which can be verified visually, but with small changes to the algorithm we can highlight overlapped matches with sufficient visual identification to differentiate from non-overlapped matches. **Case 2: 'Inverted head and shoulders'.** The inverted head and shoulders pattern in the financial domain is made up of a trough followed by a deeper trough and then a shallower trough as shown in the Figure 12(c). **Case 3: 'Double top'.** Double tops or double bottoms in the financial domain consist of two peaks or troughs of similar magnitude as shown in the Figure 13(a). Due to the flexibility of our algorithm in providing for soft-constraints, two similar peaks are highlighted. At the same time, the algorithm does not allow for false matches that completely distort the trend in the pattern. **Case 4: 'Triple top'.** Triple tops or bottoms are identified by three peaks or troughs of similar height as shown in the Figure 13(b).

8 Results and Discussion

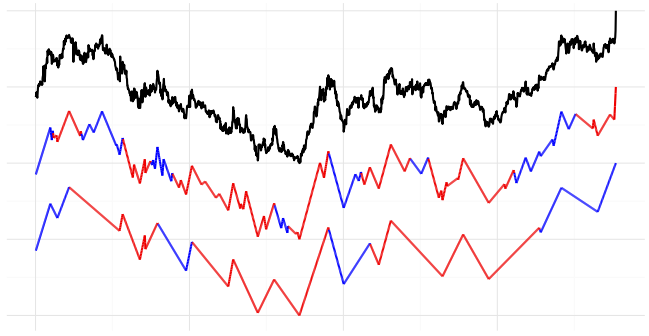
Our method has been implemented on a Dart and WegGL based framework and tested on an iMac running OS X El Capitan with 3.5GHz Intel Core i7 processor, 16GB RAM and an NVIDIA GeForce GTX 775M graphics card with 2GB memory. The processing time for the algorithm to perform the symbolic approximation and regular expression based string search are the main components in the data exploration and so the prime candidates for performance analysis. We tested our algorithm with financial data sets and other real world data sets containing thousands of time points



(a) Interactive web-based application. Case 1: Searching for the head and shoulders pattern with lower level of smoothing on the raw data.



(b) Case 1: Incremental smoothing of the raw data and searching for long term head and shoulders pattern matches.

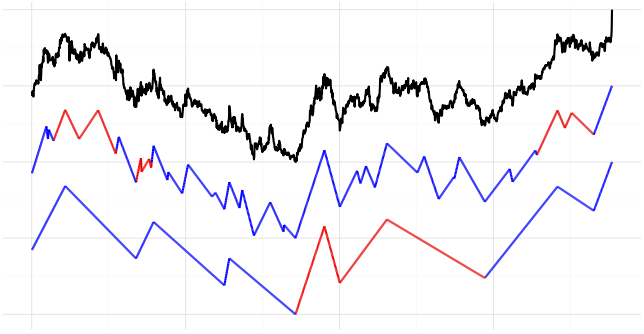


(c) Case 2: Incremental smoothing, searching of inverted head and shoulders pattern.

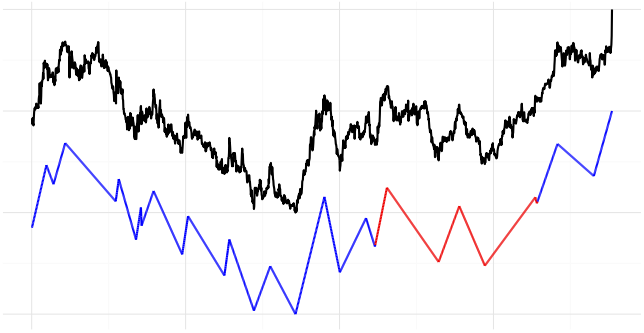
Figure 12: Hierarchical smoothing and pattern matching. The resulting matches are shown in red.

and synthetic data sets with one hundred thousand time points. As our algorithm has linear time complexity, it is evident that the processing time is linear and of the order of milliseconds. Regular expression based string matching is necessary to search for matches at different scales. In the shape grammar, *cases 11,12 and 13* are responsible for the presence of multiscale patterns. Our method utilizes the regular expression string search of the Dart framework and we tested it against user sketches having *5,10 and 15* time points in a large input time series data and the time taken is in the order of few milliseconds. Advanced regular expression string search algorithms as presented by Thompson [50] can increase the performance for millions of time series points. In the current implementation, the Dart framework supports only non-overlapping regular expression matching but, with a small modification, it is simple to check for all

possible overlapping matches by starting the search process from the index of the previously matched pattern and looping it until there are no more matches.The method is not suited for low resolution periodic data as the regular expression based smoothing operation can distort the periodicity of the data based on local trends contributing to noise. When we decrease the ratio approximation threshold to a very low value of 0.05, as explained in Section 5.4 the data can be over-approximated and hence it may lose essential local trends as shown in Figure 14. Keeping the threshold close to 1.0 during the initial steps of smoothing will make the approximation step less error prone.

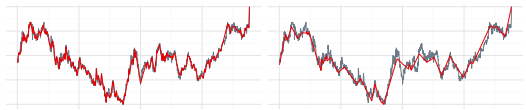


(a) Case 3: Incremental smoothing and searching of Double top pattern.



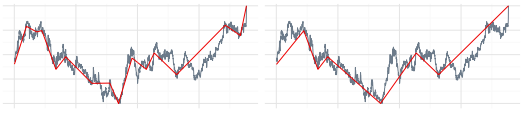
(b) Case 4: Incremental smoothing and searching of Triple top pattern.

Figure 13: Hierarchical smoothing and pattern matching. The resulting matches are shown in red.



(a) Smoothing step1

(b) Smoothing step2



(c) Smoothing step3

(d) Smoothing step4

Figure 14: Ratio approximation threshold of 0.05. Top left chart depicts the minimum level of approximation and in four smoothing iteration, we get the maximum approximation in bottom right chart.

The method described in this paper is a first step towards creating a visual query based shape grammar that is suitable for user-sketched pattern matching in real-time and performing all the operations in the symbolic space rather than on raw data. The shape grammar can approximately model any time series graph. The algorithm is also suited to work with streaming time series data where we always wait for a time point *t* and using the previous two time points *t-1, t-2* we constitute a basic shape and symbolize them accordingly. While the method does not involve absolute value matching, an absolute value range can be provided by the user as input and the search results restricted accordingly by comparing with the raw time series data. In future work we will compare the efficiency of regular expression based user query-specification and pattern matching against popular existing pattern matching approaches across different domains such as [39, 5, 11].

9 Conclusion

We have introduced our initial work on an interactive sketch-based pattern search algorithm that works in real-time with a significant level of accuracy and without complex user interaction. For a time series of length 100k, the time taken for symbolic approximation, relaxation and pattern matching is of the order of milliseconds. The pattern matching is also amplitude, scale and translation-invariant. This is achieved through careful approximation by breaking the raw data into a set of basic shapes and computing their ratios, whereby removing the amplitude information associated with them. One important contribution of our work is that we perform constraint relaxation, data smoothing and all operations in the symbol space rather than on the raw data. Such a method removes the need for arithmetic recomputation for pattern relaxation and matching. In the future we plan to explore equiprobable symbols that can be used for indexing and storage, along with prefix tree based approaches to search for similar patterns of different lengths.

Acknowledgements

We thank all reviewers for their constructive feedback. This work is funded by Swedish Research Council grant 2013-4939.

References

[1] R. Agrawal, G. Psaila, E. L. Wimmers, and M. Zaït. Querying shapes of histories. In *VLDB'95, Proceedings of 21th Int'l Conf. on Very Large Data Bases, Zurich, Switzerland.*, pages 502–514, 1995.

[2] A. Balasubramanian and B. Prabhakaran. Flexible exploration and visualization of motifs in biomedical sensor data. In *Proc. of Workshop on Data Mining for Healthcare, in conjunction with ACM KDD*, 2013.

[3] P. Buono, A. Aris, C. Plaisant, A. Khella, and B. Shneiderman. Interactive pattern search in time series. In *Electronic Imaging 2005*, pages 175–186. International Society for Optics and Photonics, 2005.

[4] P. Buono and A. L. Simeone. Interactive shape specification for pattern search in time series. In *Proceedings of the working conference on Advanced visual interfaces*, pages 480–481. ACM, 2008.

[5] K.-P. Chan and A. W.-C. Fu. Efficient time series matching by wavelets. In *Data Engineering, 1999. Proc of 15th Int'l Conf on*, pages 126–133. IEEE, 1999.

[6] J. C. Chen and A. L. Chen. Query by rhythm: An approach for song retrieval in music databases. In *Research Issues In Data Engineering, 1998.'Continuous-Media Databases and Applications'. Proceedings., Eighth International Workshop on*, pages 139–146. IEEE, 1998.

[7] A. Chortaras and A. Schopenhauer. Efficient storage, retrieval and indexing of time series data. Master's thesis, Imperial College of Science, Technology and Medicine, University of London, UK, 2002.

[8] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[9] P. Eichmann and E. Zgraggen. Evaluating subjective accuracy in time series pattern-matching using human-annotated rankings. In *Proc of 20th Int'l Conf on Intelligent User Interfaces*, pages 28–37. ACM, 2015.

[10] P. Esling and C. Agon. Time-series data mining. *ACM Computing Surveys (CSUR)*, 45(1):12, 2012.

[11] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. *Fast subsequence matching in time-series databases*, volume 23. ACM, 1994.

[12] T. Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

[13] T.-c. Fu, F.-l. Chung, K.-y. Kwok, and C.-m. Ng. Stock time series visualization based on data point importance. *Engineering Applications of Artificial Intelligence*, 21(8):1217–1232, 2008.

[14] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining data streams: a review. *ACM Sigmod Record*, 34(2):18–26, 2005.

[15] M. Glatter, J. Huang, S. Ahern, J. Daniel, and A. Lu. Visualizing temporal patterns in large multivariate data using modified globbing. *IEEE TVCG*, 14(6):1467–1474, 2008.

[16] M. Gregory and B. Shneiderman. Shape identification in temporal data sets. In *Expanding the Frontiers of Visual Analytics and Visualization*, pages 305–321. Springer, 2012.

[17] M. C. Hao, M. Marwah, H. Janetzko, U. Dayal, D. A. Keim, D. Patnaik, N. Ramakrishnan, and R. K. Sharma. Visual exploration of frequent patterns in multivariate time series. *Information Visualization*, 11(1):71–83, 2012.

[18] H. Hochheiser and B. Shneiderman. Interactive exploration of time series data. In *Discovery Science*, pages 441–446. Springer, 2001.

[19] H. Hochheiser and B. Shneiderman. Visual queries for finding patterns in time series data. *University of Maryland, Computer Science Dept. Tech Report, CS-TR-4365*, 2002.

[20] H. Hochheiser and B. Shneiderman. Dynamic query tools for time series data sets: timebox widgets for interactive exploration. *Information Visualization*, 3(1):1–18, 2004.

[21] C. Holz and S. Feiner. Relaxed selection techniques for querying time-series graphs. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 213–222. ACM, 2009.

[22] P. K. Muthumanickam, K. Vrotsou, M. Cooper, and J. Johansson. Smart series: Sketch-based matching through approximated ratios in time series. In *Poster Abstracts of IEEE VIS 2015*. IEEE.

[23] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3(3):263–286, 2001.

[24] E. Keogh, H. Hochheiser, and B. Shneiderman. An augmented visual query mechanism for finding patterns in time series data. In *Flexible Query Answering Systems*, pages 240–250. Springer, 2002.

[25] E. J. Keogh and M. J. Pazzani. Relevance feedback retrieval of time series data. In *Proc of the 22nd annual Int'l ACM SIGIR Conf on Research and development in information retrieval*, pages 183–190. ACM, 1999.

[26] N. Kumar, V. N. Lolla, E. J. Keogh, S. Lonardi, and C. A. Ratanamahatana. Time-series bitmaps: a practical visualization tool for working with large time series databases. In *Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005, Newport Beach, CA, USA, April 21-23, 2005*, pages 531–535, 2005.

[27] T.-Y. Lee and H.-W. Shen. Visualization and exploration of temporal trend relationships in multivariate time-varying data. *IEEE TVCG*, 15(6):1359–1366, 2009.

[28] G. Li, Y. Wang, L. Zhang, and X. Zhu. Similarity measure for time series based on piecewise linear approximation. In *Wireless Communications & Signal Processing, 2009. WCSP 2009. International Conference on*, pages 1–4. IEEE, 2009.

[29] G. Li, L. Zhang, and L. Yang. Tsx: A novel symbolic representation for financial time series. In *PRICAI 2012: Trends in Artificial Intelligence*, pages 262–273. Springer, 2012.

[30] H. Li and L. Yang. Time series visualization based on shape features. *Knowledge-Based Systems*, 41:43–53, 2013.

[31] Y. Li and J. X. Chen. A nondeterministic approach to infer context free grammar from sequence. In *Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 2014 11th International Computer Conference on*, pages 1–9. IEEE, 2014.

[32] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11. ACM, 2003.

[33] J. Lin, E. Keogh, S. Lonardi, J. P. Lankford, and D. M. Nystrom. Viztree: a tool for visually mining and monitoring massive time series databases. In *Proc. of the 30th Int'l conference on Very large data bases-Volume 30*, pages 1269–1272. VLDB Endowment, 2004.

[34] J. Lin, S. Williamson, K. Borne, and D. DeBarr. Pattern recognition in time series. *Advances in Machine Learning and Data Mining for Astronomy*, 1:617–645, 2012.

[35] B. Lkhagva, Y. Suzuki, and K. Kawagoe. Extended sax: Extension of symbolic aggregate approximation for financial time series data representation. *DEWS2006 4A-i8*, 7, 2006.

[36] M. Luboschik, C. Maus, H.-J. Schulz, H. Schumann, and A. Uhrmacher. Heterogeneity-based guidance for exploring multiscale data in systems biology. In *Biological Data Visualization (BioVis), 2012 IEEE Symposium on*, pages 33–40. IEEE, 2012.

[37] S. Malinowski, T. Guyet, R. Quiniou, and R. Tavenard. 1d-sax: A novel symbolic representation for time series. In *Advances in Intelligent Data Analysis XII*, pages 273–284. Springer, 2013.

[38] J. H. Mathews and K. D. Fink. *Numerical methods using MATLAB*, volume 31. Prentice hall Upper Saddle River, NJ, 1999.

[39] M. Meyer, T. Munzner, A. DePace, and H. Pfister. Multeesum:a tool for comparative spatial and temporal gene expression data. *IEEE TVCG*, 16(6):908–917, 2010.

[40] J. P. Morrill. Distributed recognition of patterns in time series data. *Communications of the ACM*, 41(5):45–51, 1998.

[41] C. G. Nevill-Manning and I. H. Witten. Identifying hierarchical strcture in sequences: A linear-time algorithm. *J. Artif. Intell. Res.(JAIR)*, 7:67–82, 1997.

[42] N. D. Pham, Q. L. Le, and T. K. Dang. Two novel adaptive symbolic representations for similarity search in time series databases. In *Web Conf (APWEB), 12th Int'l Asia-Pacific*, pages 181–187. IEEE, 2010.

[43] K. Ryall, N. Lesh, T. Lanning, D. Leigh, H. Miyashita, and S. Makino. Querylines: Approximate query for visual browsing. In *CHI'05 Extended Abstracts on Human Factors in Computing Systems*, pages 1765–1768. ACM, 2005.

[44] M. Schäfer, F. Wanner, R. Kahl, L. Zhang, T. Schreck, and D. Keim. A novel explorative visualization tool for financial time series data analysis. In *VAW2 011: The Third Int'l UKVAC Workshop on Visual Analytics*, 2011.

[45] P. Senin, J. Lin, X. Wang, T. Oates, S. Gandhi, A. P. Boedihardjo, C. Chen, and S. Frankenstein. Time series anomaly discovery with grammar-based compression. In *Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015, Brussels, Belgium, March 23-27, 2015.*, pages 481–492, 2015.

[46] P. Senin, J. Lin, X. Wang, T. Oates, S. Gandhi, A. P. Boedihardjo, C. Chen, S. Frankenstein, and M. Lerner. Grammarviz 2.0: a tool for grammar-based pattern discovery in time series. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 468–472. Springer, 2014.

[47] H. Shatkay and S. B. Zdonik. Approximate queries and representations for large data sequences. In *Proceedings of the 12th International Conference on Data Engineering.*, pages 536–545. IEEE, 1996.

[48] Y. Sun, J. Li, J. Liu, B. Sun, and C. Chow. An improvement of symbolic aggregate approximation distance measure for time series. *Neurocomputing*, 138:189–198, 2014.

[49] G. K. Tam, Q. Zheng, M. Corbyn, and R. W. Lau. Motion retrieval based on energy morphing. In *Multimedia, 2007. ISM 2007. Ninth IEEE International Symposium on*, pages 210–220. IEEE, 2007.

[50] K. Thompson. Programming techniques: Regular expression search algorithm. *Communications of the ACM*, 11(6):419–422, 1968.

[51] M. Wattenberg. Sketching a graph to query a time-series database. In *CHI'01 Extended Abstracts on Human factors in Computing Systems*, pages 381–382. ACM, 2001.

[52] M. West. Some time series data sets. `http://www2.stat.duke.edu/~mw/ts_data_sets.html`.

[53] W. Zalewski, F. Silva, H. D. Lee, A. G. Maletzke, and F. C. Wu. Time series discretization based on the approximation of the local slope information. In *Advances in Artificial Intelligence–IBERAMIA 2012*, pages 91–100. Springer, 2012.

[54] H. Ziegler, M. Jenny, T. Gruse, and D. A. Keim. Visual market sector analysis for financial time series data. In *Visual Analytics Science and Technology (VAST)*, pages 83–90. IEEE, 2010.

[55] H. Ziegler, T. Nietzschmann, and D. A. Keim. Visual analytics on the financial market: Pixel-based analysis and comparison of long-term investments. In *Information Visualisation*, pages 287–295, 2008.