

Special Section on EuroVA 2022

SAXRegEx: Multivariate time series pattern search with symbolic representation, regular expression, and query expansion

Yuncong Yu^{a,b,*}, Tim Becker^a, Le Minh Trinh^a, Michael Behrisch^b^a IAV GmbH, Rockwellstr. 16, 38518 Gifhorn, Germany^b Visualization and Graphics Group, Utrecht University, Princetonplein 5, 3584 CC Utrecht, Netherlands

ARTICLE INFO

Article history:

Received 14 October 2022

Received in revised form 10 January 2023

Accepted 11 March 2023

Available online 15 March 2023

Keywords:

Time series

Visual pattern search

Query representation

ABSTRACT

We present SAXRegEx, a method for pattern search in multivariate time series in the presence of various distortions, such as duration variation, warping, and time delay between signals. For example, in the automotive industry, calibration engineers spontaneously search for event-induced patterns in new measurements under time pressure. Current methods do not sufficiently address duration (horizontal along the time axis) scaling and inter-track time delay. One reason is that it can be overwhelmingly complex to consider scaling and warping jointly and analyze temporal dynamics and attribute interrelation simultaneously. SAXRegEx meets this challenge with a novel symbolic representation adapted to handle time series with multiple tracks. We employ methods from text retrieval, i.e., regular expression matching, to perform a pattern retrieval and develop a novel query expansion algorithm to deal flexibly with pattern distortions. Experiments show the effectiveness of our approach, especially in the presence of such distortions, and its efficiency surpassing the benchmarking methods. We have developed a user interface with the emphasis on multivariate query definition with inter-track time shifts and algorithm parameter setting. While we design the method primarily for automotive data, it is well transferable to other domains.

© 2023 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Searching for patterns in time series is a ubiquitous task, like identifying astronomical activities in light brightness observation records [1], querying segments in genetic sequences [1–3], tracking anomalies of equipment in sensor measurements [4], or discovering similar market behavior in stock price development [5–7]. Our use cases are mainly in the automotive domain. Engine engineers await a signal to pinpoint the best entry point of their novel algorithm for a smoother air–fuel ratio control; transmission engineers trace gear change intervals to analyze the moments on different shafts for less traction loss; Noise Vibration Harshness (NVH) engineers hunt for periods when some cylinders are deactivated or reactivated for cars with variable displacement to study the abrupt vibration during these periods. These use cases are all backed up by techniques for pattern search in time series.

Locating sub-sequences in time series similar to the given query is a frequent prerequisite for further data processing. Numerous time series indexing techniques [5,8,9] can already find

univariate time series patterns with minor distortions efficiently. Univariate retrieval with certain distortion types, especially time shifts and sometimes horizontal scaling along the time axis, is also addressed [10–12]. Recently, multivariate cases have received increasing attention [4,13], because an event can reveal itself in multiple signals. Multivariate distortion-invariant time series retrieval, however, remains untouched. In practice, this problem occurs in our datasets. Our target signals constantly vary in duration and are often distributed in several tracks with significant inter-track time delay. Horizontal length scaling and time shifts within a track can be regarded as heterogeneous horizontal resampling. The dummy data in Fig. 1 illustrate the distortions. These distortions are also likely to exist in patterns in data from other domains. Heterogeneous horizontal scaling is hard to capture, due to the de facto preprocessing step with a sliding window, usually assuming a fixed pattern length silently. Inter-track time shifts are even more challenging due to the complexity when considering temporal dynamics and track interrelation simultaneously.

We propose an easy-to-implement method, SAXRegEx, based on symbolic representation to enable text retrieval methods and regular expressions as our search engine. Thereupon, we design query expansion to deal with the mentioned distortions. Our method shows comparable accuracy, sometimes minor accuracy loss, to state-of-the-art techniques on datasets without these

* Corresponding author at: IAV GmbH, Rockwellstr. 16, 38518 Gifhorn, Germany.

E-mail addresses: yuncong.yu@iav.de (Y. Yu), tim.becker@iav.de (T. Becker), le.minh.trinh@iav.de (L.M. Trinh), m.behrisch@uu.nl (M. Behrisch).

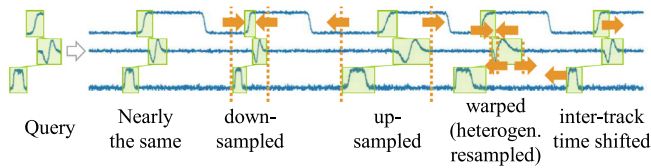


Fig. 1. Distorted patterns: the query on the left is searched for in the multivariate time series on the right. Target patterns in the time series exhibit distortions in various forms.

distortions and better performance in multivariate datasets with mentioned distortions. On the other hand, even for data without such distortions, SAXRegEx outperforms the benchmarking methods in terms of speed.¹ We have designed a user interface to facilitate the method and demonstrate it with a case study.

2. Related work

Research on time series retrieval mainly focus on two directions: first, novel similarity measures between two time series that better describe the notion of similarity (accuracy) [11,12]; second, indexing techniques focusing on efficiency [6,9,15]. Usually, sliding windows are assumed for preprocessing. Our work breaks the convention by employing neither a sliding window approach nor an explicit measure of similarity.

Besides random noise, patterns in the time series can have various distortion forms. Horizontal translation along the time axis is trivial; horizontal scaling is only addressed in a few works, such as [10,16]; vertical translation and scaling on the value/amplitude axis is often handled by normalizing time series windows [2,9]. Warping is captured by a variety of “elastic” distances [17,18], especially Dynamic Time Warping (DTW) [11,13,19]. In multivariate cases, most methods handle the temporal dynamics and interrelation between tracks independently and assume that the tracks are synchronized. For instance, DTW_l calculates distance profiles along the time axis for each track individually and merges them subsequently [13]; Locality-Sensitive Hashing (LSH) merges tracks first, followed by univariate processing along the time axis [20]. All these approaches assume that the pattern is synchronized in all tracks. Driven by our domain problems, we target this problem without this assumption and scan the temporal dynamics with inter-track relation jointly.

Recently, machine learning penetrates time series retrieval [2, 21,22]. They do not suit, however, highly dynamic or user-driven retrieval cases, where the queries and the datasets change rapidly. In our case, engineers start the search spontaneously and often wish an immediate answer.

Regarding accuracy, it is accepted that no similarity measure accounts for the human notion of similarity [23]; therefore, different datasets favor different similarity measures. Some works propose an active learning strategy to adapt the similarity measure [2,24–26]. The extensive benchmark in [27] shows that DTW enjoys top ranking accuracy among 20 methods for time series classification. As for speed, MASS is so far the fastest similarity search algorithm [28,29]. We, consequently, include both for benchmarking SAXRegEx in Section 5.

SAXRegEx uses SAX [30,31] as the data representation. It converts a univariate time series into a string to reveal certain patterns in the data and enables text processing methods. Specifically, it applies Piecewise Aggregate Approximation (PAA) to calculate piecewise means to reduce data points. Next, it

discretizes the means by partitioning the value range with breakpoints arranged to equally distribute the data points. Finally, it assigns each value range partition a symbol. While SAX proposes a distance metric between symbols, we run the search directly in the symbolic space without direct distance calculation, yielding speed boost. We exploit SAX’s less utilized “numerosity reduction” feature (combining repetitive symbols) with regex quantifier to add elasticity to the query to deal with distortions.

SAXRegEx’s regex-based search in symbolically encoded time series, resembles SSTs [32]. In comparison, SAXRegEx’s contributions lie in the ability to handle multivariate time series and various distortions.

According to [1], the term Visual Query System (VQS) was first introduced in [33] to refer to systems that allow users to “specify and search for desired line chart patterns via visual interfaces”.² According to this definition, SAXRegEx including the user interface, which is the major extension of our previous work [14], fall into the category of VQS. VQSSs are often classified based on the query definition approach. There are two prevailing query definition approaches, namely query-by-example and query-by-sketch. The former defines a query by providing an example, like marking a box in the line chart [2,26,36]. The latter specifies the query by drawing it on a canvas [1,16,23]. We compare recent VQSSs and methods in Appendix F. SAXRegEx adopts the query-by-example approach, because the patterns in our datasets can be too complex to draw, let alone considering multiple tracks and the time shifts between them. On the other hand, the engineers usually know and can provide at least one example. Our user interface focuses on the definition of multivariate time shifted query, an untouched area in VQS design.

3. Method

SAXRegEx consists of five steps illustrated in Fig. 2. First, it encodes with SAX each track in the query and in the time series. It reduces data volume, smooths curves, and enables methods for text retrieval. Next, SAXRegEx intertwines tracks in the query and in the time series. This step enables simultaneous processing of multiple tracks through a single regex. The word “intertwine” refers to the step-wise merging of sequences. In the third step, repetitive symbol groups are merged through regex quantifiers, which exploits SAX’s “numerosity reduction” property [31]. The penultimate step conducts query expansion to cope with distortions. As shown in Fig. 2, it adds a tolerance band to the query by allowing character classes rather than individual characters; thereupon, it makes the query elastic along the time axis by substituting the fixed quantifier with ranges. Finally, SAXRegEx searches for the query regex in the time series string and reconstructs the pattern in each track by a fine regex matching in the predicted intervals. In the following sections, we motivate the method choices, explain our modifications to the established method in Step 1, and describe Step 3 and Step 4 in detail with examples.

3.1. Motivations of method choices

The concrete choices for the time series representation and the search technique are based on the following motivations.

We opt for SAX, not only because it is one of the state-of-the-art symbolic representations for time series, but also considering its two properties. Firstly, the symbols in SAX’s alphabet have an order, allowing a tolerance band for the subsequent

¹ The MATLAB version of MASS is much faster than SAXRegEx. We were not aware of the issue in the previous work [14], which used a slower Python version in the experiments. Please refer to Section B.2 and Appendix E for details.

² The term Visual Query System (VQS) has been used since even earlier, like in [34,35], but in a broader sense.

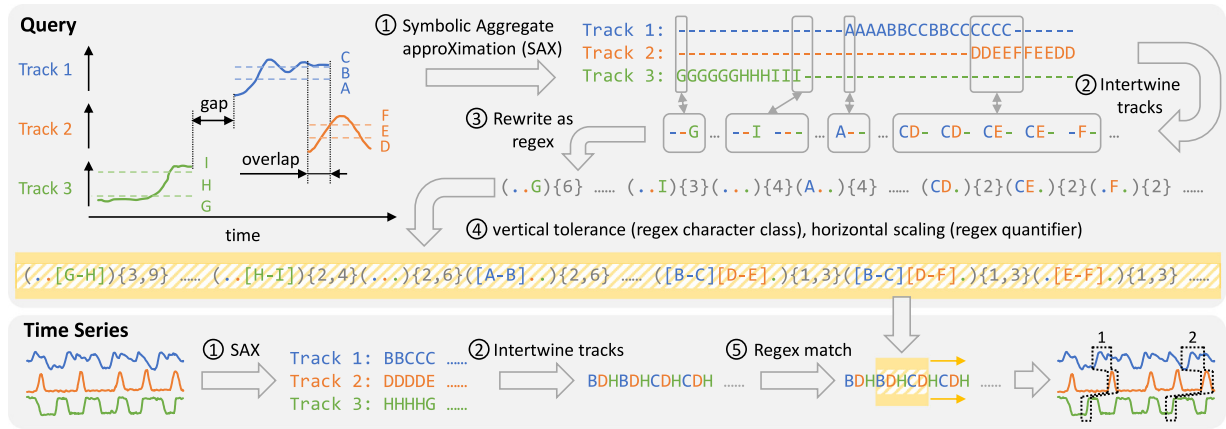


Fig. 2. Pipeline: (1) convert with SAX each track with a different alphabet to enable text retrieval methods; (2) intertwine tracks in the query and in the time series to process multiple tracks with a single regex, leave wildcards “.” for “gaps” to capture inter-track time shifts; (3) extract regex from the query string by combining repetitive symbol groups; (4) manipulate the regex with character classes and quantifiers to deal with distortions; (5) search for the query regex in the time series string and reconstruct the shape in each track.

search. Secondly, SAX’s numerosity reduction property (consecutively repeated symbols can be merged) inspired us to the query expansion for duration-scaling-invariant search.

We decide on regex, because it naturally handles duration-scaling-invariant search, together with other possible tricks, which distortion-invariant pattern search in time series may benefit from. Furthermore, it is potentially very fast, because it stops similarity matching immediately when the pattern partially mismatches the query. On the other hand, regex is well established and simple to use. Lastly, SAXRegEx may also benefit from future development of regex.

3.2. Symbolic encoding

We use SAX to convert the numerical time series into strings. Introduced in [30] and extensively explained in [31], it is an established method. SAX contains two major steps. Firstly, it performs PAA, merging temporal consecutive time steps into one by calculating their average. Secondly, it quantifies the values with quantiles as breakpoints and map the values, that are within a range bounded by the breakpoints, to a symbol.

While it is common practice to conduct indexing (SAX) on the whole time series dataset, we fit SAX (parameters: bin size horizontally along the time axis and breakpoints along the value axis) with the query, and then conduct SAX on the time series dataset with the fitted parameters. We make such an alternation, because there is no clue about the appropriate bin size and breakpoints. Especially when the patterns in the time series are quite small, either in terms of length or value range. This is exactly the case in our APST dataset. In fact, conceptually similar practice is proposed for LSH [20,37], where the hash tables are based on the query, not the dataset. They call the new LSH query-aware and the previous query-oblivious. Though hashing is needed every time the query changes, it significantly improves accuracy. On the other hand, the elapsed time required by SAX is negligible compared to that by the search process.

The original SAX assumes normal distribution of the values within each track. Based on this assumption, it calculates quantiles to quantify the values and categorize them into symbols, so that each symbol has approximately the same number of values/time steps/data points. This helps zooming in on the value ranges where the values concentrate. The values in the tracks in many of our datasets are clearly not normally distributed. Therefore, we calculate the quantiles based on the true value distribution in each track. Again, we do not calculate the value

distribution based on the whole time series dataset but on the query. Because we want to focus on the value range of interest, which is given by the query.

3.3. Regex extraction

The first two steps of our processing pipeline typically leave repetitive symbols in the text sequences. They can be bundled with quantifiers. For instance, the string ACCCCGGGBAAA can be rewritten as `AC{4}G{3}BA{3}`. As we will see later, these quantifiers also enable horizontal-scaling-invariant search.

Generally, regex is meant for exact search instead of “fuzzy” search like pattern search in time series. It can only match patterns strictly satisfying the restrictions imposed by the regex. To allow fuzzy search, we add a tolerance band to the query with character classes analogous to the L^∞ norm. For instance, the previously encoded query `AC{4}G{3}BA{3}` can be further augmented as `[A-B][B-D]{4}[F-H]{3}[A-C][A-B]{3}`. In this example, we allow a tolerance band width of two symbols.

3.4. Query expansion

We conduct query expansion to address the necessary retrieval invariances required in our use cases, particularly heterogeneous horizontal scaling and inter-track time shifts.

For heterogeneous horizontal scaling, we use value ranges rather than fixed regex quantifiers. Pertaining to our running example, we can further augment the query to `[A-B]{0,2}[B-D]{2,8}[F-H]{1,6}[A-C]{0,2}[A-B]{1,6}`. This modification captures similar patterns with a half to double duration. Moreover, it allows different fragments in the pattern to have different scaling factors, thus capturing heterogeneous horizontal scaling or time warping.

For inter-track time shifts, we use a different alphabet per track and intertwine the tracks to a single string. As shown in Fig. 2 Step 2, we intertwine the query while taking time shifts between the shapes in different tracks into consideration. We can add this flexibility by introducing the wildcard character “.” for “gaps”. This way, the query ignores the potentially interfering context in each track and focuses only on the given shape in the track. As a result, the shape in a track within the query does not have to span the whole query length. Note the term `(...){2,6}` in the yellow box in Fig. 2, which captures the elastic gap between the shapes in Track 3 and in Track 1.

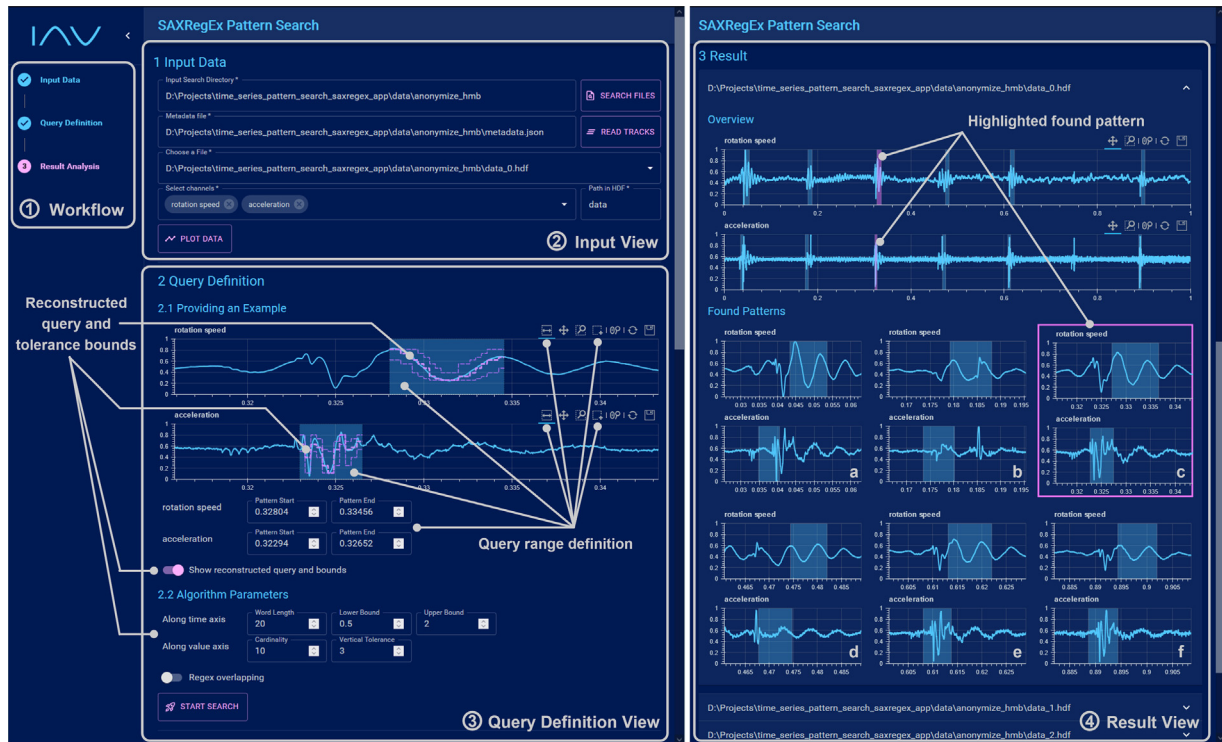


Fig. 3. User interface: assists the user with a streamlined workflow. It highlights the specification of track-wise time shifted query and the on-the-fly visualization of the expanded pattern perceived by the algorithm with the current parameter setting.

4. User interface

We have designed a user interface to assist domain users, especially during query definition and parameter setting. We derive our design from the requirements of our domain users, who request a tool with a streamlined workflow, little configuration overhead, and quick response. In line with the findings in [38], our domain users disapprove of visualizations for understanding the algorithm's internal mechanism. They are reluctant to learn the algorithm itself, but appreciate intuitive visual assistance, especially during query definition and parameter setting.

We use React as the frontend library, BokehJS to render charts, and Django to access the algorithm presented in Section 3 and previously published in [14]. In the following, we will describe individual features of the user interface and present a coherent workflow in a case study in Section 5.3.

4.1. Sidebar and input view

To satisfy our workflow requirements, we choose a layout allowing the user to proceed sequentially through the workflow steps “Input Data”, “Query Definition”, and “Result Analysis” shown in the sidebar. The accomplished steps are in blue, in-processing steps in the highlighting color pink, as well as coming steps in gray, as shown in Fig. 3.1. In our proprietary system, more workflow steps are listed in the sidebar and SAXRegEx acts as the first quick search. We have removed preprocessing steps like dataset description, resampling, and steps after SAXRegEx like the accuracy-centered feedback-driven search based on the user's relevance feedback, because they are outside the scope of this work. In the depicted user interface, the user needs to provide the dataset and relevant tracks in the Input View in Fig. 3.2.

4.2. Query definition view

As the primary visual component of the user interface, the Query Definition View in Fig. 3.3 assists the user in specifying the desired query and setting the algorithm parameters.

4.2.1. Providing an example

As explained in Section 2, we opt for the query-by-example approach. Based on this premise, we find the most efficient way to define an inter-track time-shifted query is to approximately mark one interval for all tracks simultaneously first, then fine-tune the interval in each track individually. The process is depicted in Fig. 4. To define an approximate query without time shifts, the user activates the box select tool . It is configured to allow marking a box along the time axis and the selection will be synchronized across all tracks. A box is technically a Bokeh range tool , which is automatically activated in all tracks after marking the first query without time shift. This allows the user to move the box and fine-tune its left and right time stamps in each track, i.e., to modify the spread of a pattern. This design emphasizes the relationship between the patterns in different tracks. Compared with drawing boxes in each track individually, the user gets a sense of the positions of the patterns in different tracks relative to each other, as the user moves the box, that was marked for the pattern in another track, to align its left edge with the starting time of the pattern in the current track. Likewise, the user feels the difference between pattern durations when adjusting the box size from the length of the first pattern to that of the next. If there are no inter-track time shifts between the patterns in several tracks, the user does not need to mark the same time interval multiple times. The edge positions or the time intervals are listed additionally as numbers in the input fields below.

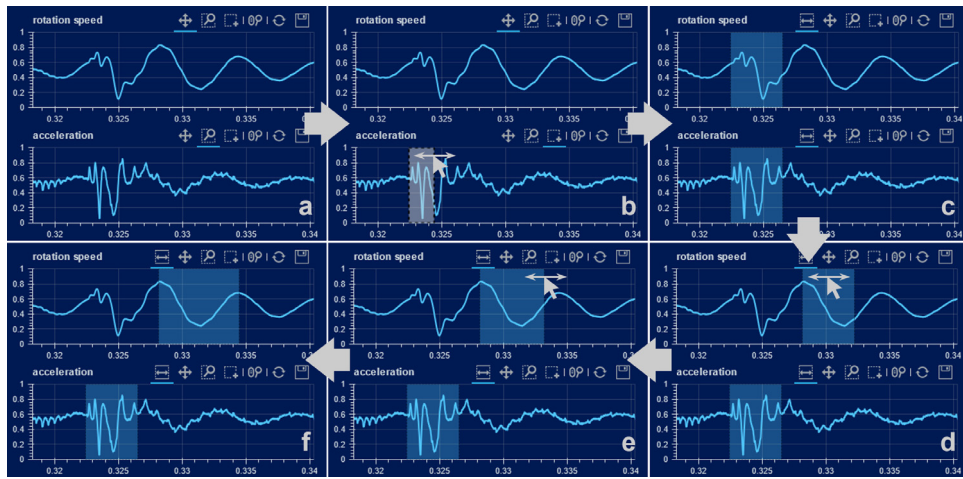


Fig. 4. Labeling an example as the query: the user draws a box in any track to define a query without inter-track time shifts, then fine-tune the boxes in different tracks individually by moving it or adjusting its left and right edges.

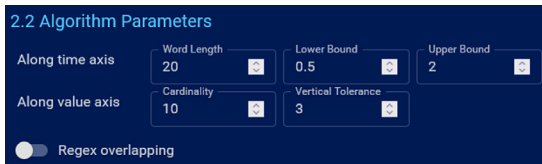


Fig. 5. Algorithm parameters: they influence the strictness of the search by controlling the resolutions and query expansion.

4.2.2. Algorithm parameters

The user can tune six parameters shown in Fig. 5. They control the resolutions, the query expansion, and thus the fuzziness or strictness of the search. They have empirically reasonable values or can be adjusted individually and intuitively.

The parameters introduced by SAX, namely word length and cardinality, regulate the horizontal (along the time axis) and vertical (along the value axis) resolutions, respectively. For instance, word length 20 encodes the pattern in the first track to a string of length 20. The patterns in other tracks are converted to strings with proportional lengths. Cardinality 10 means discretizing the values in each track with 10 symbols. We find word length 20 usually sufficient. According to the experiments in [31], cardinality 10 is enough.

The lower and upper bounds along the time axis determine the horizontal elasticity of the query. Lower bound 0.5 means that the lengths of the target patterns are not allowed to be shorter than half of the query length, while upper bound 2 means that the lengths of the target patterns cannot exceed the doubled length of the query. These two parameters are task-dependent, but users can often set them before starting search. Note that the scaling can be heterogeneous, i.e., some segments in the query are down-scaled while others can be up-scaled.

Vertical tolerance refers to the allowed deviation of symbols. For example, the deviation between A and D is 3 in the symbolic space. It resembles the Chebyshev distance, or L^∞ norm, and plays a filtering role analogous to a similarity threshold or the desired number of found patterns. It is difficult to estimate this parameter a priori. The best setting is affected by how similar the target patterns are compared to the query (homogeneity within the positive class) and how dissimilar the rest of the data are with the query (difference between positive and negative classes). It is likely that users need to adjust this parameter to find less or more patterns.

The last parameter regex overlapping is boolean and controlled by a switch. It handles cases when the patterns can overlap, like Pattern 3 and Pattern 4 in Fig. 9 (in the last row for SAXRegex). It is likely that the same pattern will be found multiple times if this feature is activated. This feature is less used and deactivated by default.

To assist users with parameter tuning, we overlay the query with its reconstruction from the symbolic space and a pair of bounds visualizing the most dynamic parameter, namely vertical tolerance, if the user activates the switch “Show reconstructed query and bounds”, as shown in Fig. 6. The reconstruction and the bounds will update automatically when the user adjusts the query boxes, the word length, the cardinality, or the vertical tolerance. Fig. 7 shows the resulting reconstruction and tolerance bounds with different parameter settings.

The reconstruction and tolerance bounds are generated by converting a symbol with the mean of its lower and upper breakpoints. A breakpoint is the borderline between two symbols. If a symbol does not have a lower or upper breakpoint, we use the minimum or maximal value in the track of the query instead. For instance, a track in the query has a minimum value of 0 and a maximal value of 10. Supposing that we use A, B, and C to discretize the tracks, the breakpoint between A and B is 4, and between B and C 6. Then, the reconstructed value for A is $(0 + 4) \div 2 = 2$, for B is $(4 + 6) \div 2 = 5$, and for C is $(6 + 10) \div 2 = 8$. Because we use the mean of the breakpoints rather than the breakpoints themselves, the tolerance bounds appear visually tighter than the actual bounds. Nonetheless, the imprecise tolerance bounds help the user obtain an intuition of the vertical tolerance before starting the search.

4.3. Result view

The result view in Fig. 3.4 lists all data files. For each input data file, we show an overview plot and plots of the individual found patterns. The overview plots show the entire data and found patterns. Clicking a plot for an individual found pattern will highlight the plot with a pink frame. The corresponding pattern in the overview plot will also be highlighted in pink.

5. Evaluation

We evaluated the accuracy and speed of SAXRegex with eight labeled datasets, among which two contain patterns with heterogeneous horizontal scaling or inter-track time shifts. They

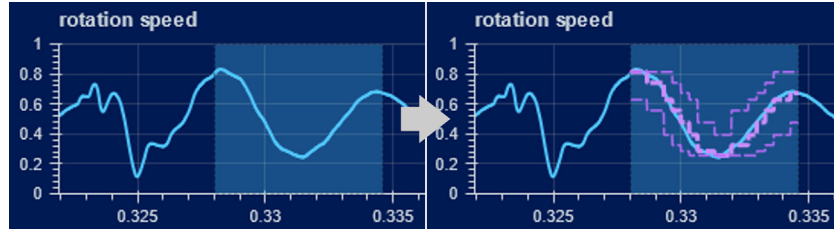


Fig. 6. Query reconstruction and tolerance bounds: the original query is overlaid with its reconstruction from the symbolic space and the tolerance bounds, if the user activates the switch “Show reconstructed query and bounds”. They assist the user with parameter setting.

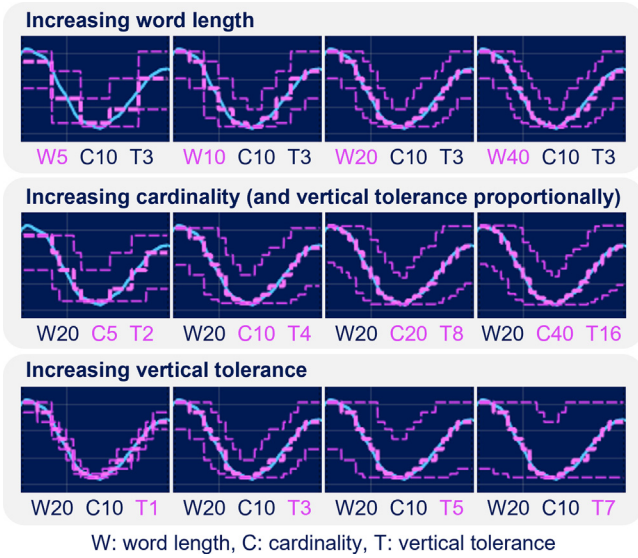


Fig. 7. Query reconstruction and tolerance bounds with different parameter settings: The query reconstruction and the tolerance bounds are updated whenever the original query or the parameters change. They help the user gain an intuition of some parameters before starting the search.

are mainly automotive measurement data provided by different engineers in our industrial collaborator IAV. While the datasets show different characteristics like the number of tracks, the pattern shape, and the extent of duration variation, by no means do they cover all possible cases, like high-frequency audio patterns. Appendix A describes the datasets in details. We chose four state-of-the-art methods for our comparative evaluation of SAXRegEx: (1) correlation: one of the standard similarity measures; (2) DTW: the most popular elastic distance measure for time series with “hard-to-beat” accuracy for time series classification [27]; (3) Euclidean Distance (ED): specifically MASS, the so-far fastest similarity search tool [28,29]; (4) SAX: our baseline time series representation, together with its similarity measure [31], can be used for similarity search. Details of the benchmark methods are in Appendix B. Finally, we conducted a case study to demonstrate, how SAXRegEx’s user interface assists users in applying the algorithm.

We conducted all experiments on a standard laptop running on 64-bit Windows 10 Enterprise with Intel i7-8650U CPU, 16 GB RAM, and 1 TB HDD. Details of the hardware and software setup are in Appendix B.

		Prec.	Recall	F1	mAP30	mAP50
Cable Cutter	Corr.	0.79	0.86	0.82	0.89	0.56
	DTW	0.48	0.98	0.64	0.84	0.19
	ED	0.36	0.52	0.43	0.27	0.18
	SAX	0.74	0.71	0.72	0.69	0.31
	SRe	0.55	0.87	0.67	0.89	0.55
Deep Valve	Corr.	0.88	0.47	0.61	0.82	0.29
	DTW	0.46	0.94	0.62	0.46	0.19
	ED	0.88	0.53	0.67	0.85	0.28
	SAX	0.87	0.93	0.90	0.73	0.66
	SRe	0.62	0.84	0.71	0.83	0.50
EEG Eye State	Corr.	0.76	0.77	0.76	0.57	0.40
	DTW	0.73	0.87	0.79	0.57	0.43
	ED	0.63	0.77	0.69	0.58	0.36
	SAX	0.58	0.80	0.67	0.40	0.37
	SRe	0.59	0.92	0.72	0.62	0.62
CAN 1 (with heterogeneous horizontal scaling)	Corr.	0.68	0.93	0.79	0.79	0.75
	DTW	0.93	1.00	0.96	1.00	1.00
	ED	0.68	0.93	0.79	0.90	0.78
	SAX	0.74	0.86	0.80	0.90	0.81
	SRe	0.99	0.99	0.99	1.00	1.00
CAN 2 (with inter-track time shifts)	Corr.	0.82	0.99	0.90	0.97	0.97
	DTW	0.80	0.75	0.77	0.89	0.70
	ED	1.00	0.78	0.87	0.94	0.94
	SAX	0.76	0.93	0.84	0.84	0.84
	SRe	1.00	1.00	1.00	1.00	1.00

Fig. 8. Accuracy benchmark: best performance of all methods. Complete version in Figure D.1. Best F1-score, mAP30 and mAP50 among five methods are highlighted bold and red. Each method suits different datasets. The proposed method (denoted as SRe) outperforms the other, when there are heterogeneous horizontal scaling or inter-track time shifts, as in the last two cases.

5.1. Accuracy benchmark

We compared the performance of all methods quantitatively with standard metrics and qualitatively through a visual inspection. Besides the standard metrics accuracy, balanced accuracy, precision, recall, and F1-score, which regard the problem as the binary classification for each time step (inside/outside a pattern), we want to introduce the metric mean Average Precision (mAP) from object detection in computer vision from a segment-wise perspective. We have chosen 30% and 50% as the Intersection over Union (IoU) threshold required by this metric. They are denoted mAP30 and mAP50, respectively. For details, please refer to Section B.3. The table in Fig. 8 (complete version in Appendix

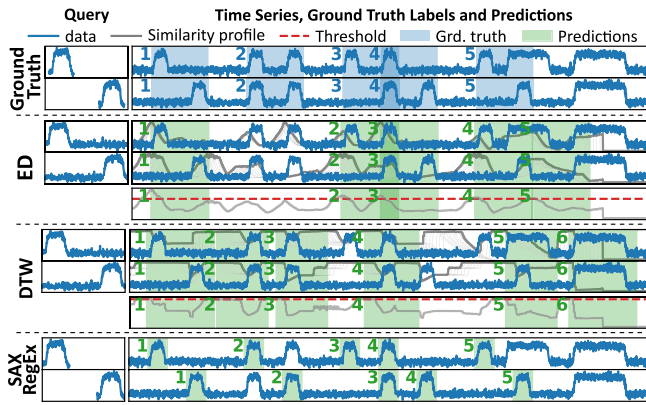


Fig. 9. Visual inspection: result on the dataset CAN 2. SAXRegEx perfectly finds inter-track time-shifted patterns, while the benchmark methods suffer from false negatives and false positives. Description of the elements in the figure and a complete version for all methods and datasets are in Appendix C.

D) depicts our five evaluation metrics for all methods on various datasets. The result suggests that different data favor different methods and no method constantly outperforms the other, confirming the finding in [23]. However, SAXRegEx can better capture heterogeneous horizontal scaling and inter-track time shifts. We can observe this specifically on the last two datasets in Fig. 8 and Figure D.1.

We have plotted the search results and ground truth labels with all methods on all publishable datasets in Appendix C, together with the description of the plots. Fig. 9 shows an example. From the visual inspection, we can infer pitfalls of the methods. Firstly, the benchmark methods spread focus evenly over all time steps rather than critical transitions (usually large ramps). This leads to the false positive Pattern 5 with ED. Pattern 5 does resemble the query in terms of the general shape, but misses an upward ramp at the beginning in the first track and a downward ramp at the end in the second track. Next, the benchmark methods take the context of the pattern shape in each track into account (please note the difference between the query for SAXRegEx and the benchmark methods). Consequently, the search can be misled by a changing context, as implied by the missing ground truth label 2 for ED and DTW. In this case, the target pattern contains a distracting plateau in each track. Finally, while DTW well addresses the time shifts within a track, as proven by the performance on the CAN 1 dataset in Fig. 8 and Figure C.5, it has a hard time when it comes to time shifts between tracks, as indicated by Prediction 2, 3 and 4 for DTW. In the presence of such distortions, SAXRegEx outperforms the benchmarks and fulfills our needs. On the flip side, SAXRegEx may find excessively long patterns when they do not have unclear boundaries, as in Figure C.4. Because the quantifiers in regex are greedy by default. Thus, the regex tries to match the longest possible patterns.

5.2. Speed benchmark

We calculated the elapsed time of all methods on all datasets. Every experiment is repeated five times. The complete result can be found in Fig. 10. In Figure E.2, we report on the relative performance gain compared to DTW.

In short, SAXRegEx is nearly x50 faster than DTW and x4.6 faster than MASS, the so-far fastest similarity search tool for time series retrieval. Please note the caveat regarding MASS in Appendix E. We attribute this speed boost to two reasons. First,



SAXRegEx naturally captures horizontally scaled patterns. In contrast, the benchmark methods use eight sliding windows with varying window lengths, costing roughly x8 the time. However, even reducing the sliding window numbers to four, which is already too coarse, SAXRegEx still outperforms all benchmark methods. Second, regex terminates a matching as soon as the search engine notices a partial mismatch.

5.3. Case study

In this section, we demonstrate a use case with a real-world dataset. Please also refer to Appendix G for full screenshots and the video https://www.youtube.com/watch?v=pb8ZUCjI_8Q for details of interactions.

We use the variable displacement dataset described in details in Appendix A for our demonstration. It contains measurements of a car switching between the four-cylinder and two-cylinder drive-mode. The used tracks are domain-specific extracted features, but roughly correspond to the engine rotational speed and acceleration. All tracks including time are normalized to the range [0, 1] for anonymization, which is not a required preprocessing step by SAXRegEx.

When the user starts the user interface, only the Input View is visible. After providing input data and clicking the button “Plot data”, the Query Definition View appears. It has a line chart plotting the chosen tracks in the given data file and input fields populated with default algorithm parameters.

The user interacts with the line chart by panning and zooming to have a quick look at the data and spots a pattern that is characteristic of the transition event. As shown in Fig. 4.a, there is a period with abrupt increase of amplitude in the second track (acceleration). This turbulence can be the cause of the uncomfortable jerks that the driver senses and the user as domain expert investigates. After the acceleration turbulence, the engine rotational speed track shows a damped oscillation. The user drags a span with the box select tool  to label it (Fig. 4.b). There appear boxes with the same labeled range in both tracks (Fig. 4.c). In principle, the user can use this label as a query. Let us assume that the user is interested in the pattern in the first track shown in Fig. 4.f. In this case, the user may move the box in the first track, so that the left edge of the box matches the left side of the desired pattern (Fig. 4.d). Next, the user adjusts the right edge of the box (Fig. 4.e). The corresponding range tool  is activated automatically after the first marking to allow these interactions. When moving the box in the first track, the user senses that the pattern in the first track lags behind for a certain time; when adjusting the right edge of the box, the user notices that the pattern in the first track lasts longer to some extent.

Then, the user moves on to the algorithm parameters (Fig. 5). The user turns on the switch “Show reconstructed query and bounds” and noticed the reconstructed query together with the tolerance bounds, as in Fig. 6. Without a deep understanding of the algorithm, the user can interpret the thicker pink line in the middle, right on the blue query curve, as a reflection of how the algorithm perceives the pattern and the thinner pink curve pairs as a tolerance band. The user finds the resolution sufficient and decides not to change the word length or the cardinality. The user also keeps the lower bound 0.5 and upper bound 2 for horizontal scaling. So the query or any part of it can be compressed or stretched with the maximal factor 2. The user is unsure about the vertical tolerance and tries multiple values (last row in Fig. 7). With the help of the visualized tolerance bound, the user finds the preset reasonable. Finally, the user clicks the “Start search” button. The Result View appears and eight panes corresponding to eight data files appear in a second. The user opens one pane, scans the six found patterns, clicks interesting ones, and inspects

	APST	Cable Cutter	Deep Valve	EEG Eye State	Filling Prediction	Variable Displ.	CAN 1	CAN 2
DTW	24.54±2.29	43.56±3.39	9.98±0.37	2.00±0.57	1.36±0.07	94.66±5.25	0.320 ± 0.031	0.410 ± 0.028
Correlation	27.90±3.25	55.09±4.11	12.35±1.24	3.61±0.38	1.41±0.03	107.44±7.57	0.644 ± 0.636	0.542 ± 0.288
SAX	2.59±0.21	7.89±1.83	1.26±0.04	0.20±0.01	0.23±0.01	8.28±0.83	0.222 ± 0.016	0.282 ± 0.006
ED (MASS)	15.05±0.87	27.18±1.48	6.19±0.29	1.23±0.30	0.83±0.06	48.36±2.36	0.062 ± 0.003	0.074 ± 0.005
SAXRegex	0.65±0.03	4.39±0.55	0.22±0.01	0.05±0.00	0.04±0.01	1.36±0.16	0.022 ± 0.003	0.022 ± 0.007

Fig. 10. Speed benchmark: unit sec, methods in ascending order of elapsed time. We repeat the same measurement five times and recorded their mean and standard deviation. SAXRegex outperforms the other methods for all datasets. Please note the caveat regarding MASS in Appendix E.

the clicked ones highlighted in the overview chart. We show a mediocre result in Fig. 3.4 as a common first outcome, and keep a better result after some tuning in Figure G.15.

In Fig. 3.4, the user notices that some patterns indeed resemble the query, like the Pattern c, e, and f. The Pattern b and d pass for the desired pattern to some extent, but may not indicate the same event, because the acceleration shows different dynamics. Pattern a indicates a potentially desired pattern, but the found interval in the acceleration track does not align well with the pattern. If only in search of some samples of a target event, the user may check several files and collect the desired patterns. If aiming at all similar patterns in all data files with high recall and precision, the user may take next steps. Specifically, the user may add more relevant tracks (but possibly without visually similar patterns), provide feedback on the found patterns, run a supervised-learning-based search with the labeled findings, and finally conduct domain-specific analysis with the refined result, which go beyond the scope of this work.

6. Discussion and limitations

Our industrial collaborator applies SAXRegex to data from engine control units, transmission control units, and CAN bus, where heterogeneous horizontal scaling of the pattern and inter-track time shifts occur regularly. It satisfactorily meets automotive engineers' flexible needs for prompt search results.

One shortcoming of SAXRegex is the neglect of vertical translation (bias) and vertical scaling (amplitude scaling) of the pattern. The user should ensure that they do not affect the patterns in the database. Ideally, future query expansion should be able to covered them.

Notwithstanding the above limitation regarding query expansion, there is still potential in modifying the regex. For instance, groups in regex can be used to denote a pattern with multiple phases; positive/negative lookarounds allow retrieving patterns with/without certain preceding or following patterns; conditionals in regex allow a part of the pattern to vary according to another part. However, their usefulness and effectiveness remain to be examined.

SAXRegex does not calculate similarity profiles. On the one hand, it accelerates the algorithm because the regex search stops immediately when a part of the pattern does not match the query. On the other hand, it is not possible to tune a similarity threshold to improve the found patterns. Therefore, it functions better as the first quick-and-dirty search. The user can improve the search results with fine-grained methods.

We plan to investigate the time complexity of SAXRegex and carry out experiments to study its scalability with increasing query sizes (regex length) and the increasing number of tracks in a contour plot. We would also like to examine the degree of distortions that SAXRegex can bear.

We use one vertical tolerance for all tracks, providing that the characteristics of the tracks do not differ significantly. However, this assumption does not always hold. We could allow an individual vertical tolerance for each track, but still need a way to avoid overwhelming the user.

Overall, our evaluation of the user interface is incomplete.

7. Conclusion

With SAXRegex, we propose an algorithm for multivariate time series retrieval based on SAX, regex, and query expansion. It excels in capturing distorted patterns of various types. In particular, it is ideal to search for patterns scaled horizontally along the time axis heterogeneously or patterns showing inter-track time shifts, while remaining efficient. We also present a user interface featuring multivariate query definition with inter-track time shifts and parameter setting, assisting users in applying the algorithm. SAXRegex helps automotive engineers quickly find patterns related to various events in the measurements. Nonetheless, the method itself is not limited to any domain-specific prerequisites and can be used in other domains as well. In the future, we plan to extend the evaluation with a scalability test for the algorithm and expert studies for the user interface.

CRedit authorship contribution statement

Yuncong Yu: Methodology, Software, Validation, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Tim Becker:** Conceptualization, Resources, Data curation, Writing – review & editing, Supervision. **Le Minh Trinh:** Software, Writing – review & editing. **Michael Behrisch:** Conceptualization, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Authors Yuncong Yu and Tim Becker are employed by IAV GmbH. IAV GmbH has funded the research work and uses the result commercially.

Data availability

The data that has been used is confidential.

Acknowledgments

We would like to express our sincere gratitude to Dr. Anna Bertram and Dr. Thomas Müller for their in-depth review and support during software development.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at <https://doi.org/10.1016/j.cag.2023.03.002>.

References

- [1] Lee DJ, Lee J, Siddiqui T, Kim J, Karahalios K, Parameswaran A. You can't always sketch what you want: Understanding sensemaking in visual query systems. *IEEE Trans Vis Comput Graphics* 2020;26(1):1267–77. <http://dx.doi.org/10.1109/TVCG.2019.2934666>.
- [2] Lekschas F, Peterson B, Haehn D, Ma E, Gehlenborg N, Pfister H. PEAX: Interactive visual pattern search in sequential data using unsupervised deep representation learning. *Comput Graph Forum* 2020;39(3):167–79. <http://dx.doi.org/10.1111/cgf.13971>.
- [3] Siddiqui T, Luh P, Wang Z, Karahalios K, Parameswaran A. ShapeSearch: A flexible and efficient system for shape-based exploration of trendlines. In: *Proceedings of the 2020 ACM SIGMOD international conference on management of data*. New York; 2020, p. 51–65. <http://dx.doi.org/10.1145/3318464.3389722>.
- [4] Laftchiev E, Liu Y. Finding multidimensional patterns in multidimensional time series. In: *KDD workshop on MiLeTS*. London; 2018.
- [5] Faloutsos C, Ranganathan M, Manolopoulos Y. Fast subsequence matching in time-series databases. In: *Proceedings of the 1994 ACM SIGMOD international conference on management of data*. New York; 1994, p. 419–29. <http://dx.doi.org/10.1145/191839.191925>.
- [6] Agrawal R, Lin K, Sawhney HS, Shim K. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In: *Proceeding of the 21th international conference on very large data bases*. Zurich; 1995, p. 490–501.
- [7] Negi T, Bansal V. Time series: Similarity search and its applications. In: *Proceedings of the international conference on systemics, cybernetics and informatics: ICSCI-04*. Hyderabad. 2005, p. 528–33.
- [8] Gunopulos D, Das G. Time series similarity measures and time series indexing. In: *Proceedings of the 2001 ACM SIGMOD international conference on management of data*. New York; 2001, p. 624. <http://dx.doi.org/10.1145/375663.375808>.
- [9] Chatzigeorgakidis G, Skoutas D, Patrourmpas K, Palpanas T, S. A, Skiadopoulou S. Efficient range and kNN twin subsequence search in time series. *IEEE Trans Knowl Data Eng* 2022. <http://dx.doi.org/10.1109/TKDE.2022.3167257>.
- [10] Keogh E. A fast and robust method for pattern matching in time series databases. In: *Proceedings of western users of SAS software*. Universal City; 1997.
- [11] Zhao J, Itti L. Shapedtw: Shape dynamic time warping. *Pattern Recognit* 2018;74:171–84. <http://dx.doi.org/10.1016/j.patcog.2017.09.020>.
- [12] Gharghabi S, Imani S, Bagnall A, Darvishzadeh A, Keogh E. Mpdist: A novel time series distance measure to allow data mining in more challenging scenarios. In: *IEEE international conference on data mining*. Singapore; 2018, p. 965–70. <http://dx.doi.org/10.1109/ICDM.2018.00119>.
- [13] Shokoohi-Yekta M, Wang J, Keogh E. On the non-trivial generalization of dynamic time warping to the multi-dimensional case. In: *Proceedings of the SIAM international conference on data mining*. Vancouver; 2015, p. 289–97. <http://dx.doi.org/10.1137/1.9781611974010.33>.
- [14] Yu Y, Becker T, Behrisch M. Multivariate time series retrieval with symbolic aggregate approximation, regular expression, and query expansion. In: *EuroVis workshop on visual analytics*. Rome; 2022. <http://dx.doi.org/10.2312/eurova.2022.1081>.
- [15] Peng B, Fatourou P, Palpanas T. SING: Sequence indexing using GPUs. In: *IEEE 37th international conference on data engineering*. Chania; 2021, p. 1883–8. <http://dx.doi.org/10.1109/ICDE51399.2021.00171>.
- [16] Mannino M, Abouzied A. Qetch: Time series querying with expressive sketches. In: *Proceedings of the 44th international conference on management of data*. New York; 2018, p. 1741–4. <http://dx.doi.org/10.1145/3183713.3193547>.
- [17] Chen L, Ng R. On the marriage of Lp-norms and edit distance. In: *Proceedings of the 30th international conference on very large data bases*. Toronto; 2004, p. 792–803.
- [18] Stefan A, Athitsos V, Das G. The move-split-merge metric for time series. *IEEE Trans Knowl Data Eng* 2013;25(6):1425–38.
- [19] Berndt DJ, Clifford J. Using dynamic time warping to find patterns in time series. In: *Proceedings of the 3rd international conference on knowledge discovery and data mining*. Seattle; 1994, p. 359–70.
- [20] Yu C, Luo L, Chan LL, Rakthanmanon T, Nutanong S. A fast LSH-based similarity search method for multivariate time series. *Inform Sci* 2019;476:337–56. <http://dx.doi.org/10.1016/j.ins.2018.10.026>.
- [21] Grabocka J, Schmidt-Thieme L. NeuralWarp: Time-series similarity with warping networks. *Comput Res Repos* 2018. [abs/1812.08306](https://arxiv.org/abs/1812.08306).
- [22] Hou L, Jin X, Zhao Z. Time series similarity measure via siamese convolutional neural network. In: *Proceeding of the 12th international congress on image and signal processing, biomedical engineering and informatics*. Suzhou; 2019, p. 1–6. <http://dx.doi.org/10.1109/CISP-BMEI48845.2019.8966048>.
- [23] Correll M, Gleicher M. The semantics of sketch: Flexibility in visual query systems for time series data. In: *Proceedings of the IEEE conference on visual analytics science and technology*. Baltimore; 2016, p. 131–40. <http://dx.doi.org/10.1109/VAST.2016.7883519>.
- [24] Keogh E, Pazzani MJ. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In: *Proceedings of the 4th international conference on knowledge discovery and data mining*. New York; 1998, p. 239–43.
- [25] Schlegl T, Schlegl S, Tomaselli D, West N, Deuse J. Adaptive similarity search for the retrieval of rare events from large time series databases. *Adv Eng Inform* 2022;52:101629. <http://dx.doi.org/10.1016/j.aei.2022.101629>.
- [26] Yu Y, Kruffy D, Jiao J, Becker T, Behrisch M. PSEUDO: Interactive pattern search in multivariate time series with locality-sensitive hashing and relevance feedback. *IEEE Trans Vis Comput Graphics* 2022;29(1):33–42. <http://dx.doi.org/10.1109/TVCG.2022.3209431>.
- [27] Bagnall A, Lines J, Bostrom A, Large J, Keogh E. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Min Knowl Discov* 2017;31(3):606–60. <http://dx.doi.org/10.1007/s10618-016-0483-9>.
- [28] Yeh CM, Zhu Y, Ulanova L, Begum N, Ding Y, Dau HA, Silva DF, Mueen A, Keogh E. Matrix profile I: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In: *Proceedings of the IEEE 16th international conference on data mining*. Barcelona; 2016, p. 1317–22. <http://dx.doi.org/10.1109/ICDM.2016.0179>.
- [29] Mercer R, Alaei S, Abdoli A, Singh S, Murillo A, Keogh E. Matrix profile XXIII: Contrast profile: A novel time series primitive that allows real world classification. In: *Proceedings of the 21st IEEE international conference on data mining*. Auckland; 2021, p. 1240–5. <http://dx.doi.org/10.1109/ICDM51629.2021.00151>.
- [30] Lin J, Keogh E, Lonardi S, Chiu B. 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*. New York; 2003, p. 2–11. <http://dx.doi.org/10.1145/882082.882086>.
- [31] Lin J, Keogh E, Wei L, Lonardi S. Experiencing SAX: a novel symbolic representation of time series. *Data Min Knowl Discov* 2007;15(2):107–44. <http://dx.doi.org/10.1007/s10618-007-0064-z>.
- [32] Rodrigues J, Folgado D, Belo D, Gamboa H. SSTS: A syntactic tool for pattern search on time series. *Inf Process Manag* 2019;56(1):61–76. <http://dx.doi.org/10.1016/j.ipm.2018.09.001>.
- [33] Ryall K, Lesh N, Lanning T, Leigh D, Miyashita H, Makino S. QueryLines: Approximate query for visual browsing. In: *CHI '05 extended abstracts on human factors in computing systems*. New York; 2005, p. 1765–8. <http://dx.doi.org/10.1145/1056808.1057017>.
- [34] Catarci T, Costabile MF, Levialdi S, Batini C. Visual query systems for databases: A survey. *J Vis Lang Comput* 1997;8(2):215–60. <http://dx.doi.org/10.1006/jvlc.1997.0037>.
- [35] Angelaccio M, Catarci T, Santucci G. Query by diagram: A fully visual query system. *J Vis Lang Comput* 1990;1(3):255–73. [http://dx.doi.org/10.1016/S1045-926X\(05\)80009-6](http://dx.doi.org/10.1016/S1045-926X(05)80009-6).
- [36] Hochheiser H, Shneiderman B. Dynamic query tools for time series data sets: Timebox widgets for interactive exploration. *Inf Vis* 2004;3(1):1–18. <http://dx.doi.org/10.1057/palgrave.ivs.9500061>.
- [37] Huang Q, Feng J, Zhang Y, Fang Q, Ng W. Query-aware locality-sensitive hashing for approximate nearest neighbor search. *Proc VLDB Endow* 2015;9(1):1–12. <http://dx.doi.org/10.14778/2850469.2850470>.
- [38] Simon S, Mittelstädt S, Keim DA, Sedlmair M. Bridging the gap of domain and visualization experts with a liaison. In: *Proceeding of eurographics conference on visualization*. Cagliari; 2015, p. 127–31. <http://dx.doi.org/10.2312/eurovisshort.2015.1137>.