Relaxed Selection Techniques for Querying Time-Series Graphs

Christian Holz
Hasso Plattner Institute[1]
Potsdam, Germany
christian.holz@hpi.uni-potsdam.de

Steven Feiner
Department of Computer Science
Columbia University, New York, NY
feiner@cs.columbia.edu

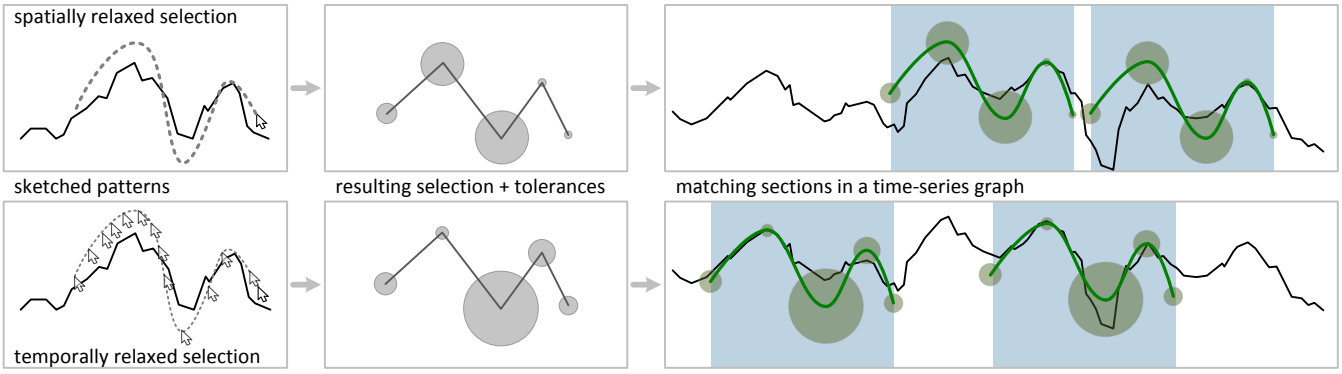

Figure 1: Relaxed selection techniques. Spatially relaxed selection (top): Tolerances are derived from spatial deviations between displayed graph and user sketch on a point-by-point basis. Temporally relaxed selection (bottom): Tolerances are derived from input speed. (Center) In both cases, tolerances can be visualized with circles around points in the selection. (Right) Tolerant selections allow similarity matches for patterns that recur in a time series.

ABSTRACT

Time-series graphs are often used to visualize phenomena that change over time. Common tasks include comparing values at different points in time and searching for specified patterns, either exact or approximate. However, tools that support time-series graphs typically separate query specification from the actual search process, allowing users to adapt the level of similarity only after specifying the pattern. We introduce relaxed selection techniques, in which users implicitly define a level of similarity that can vary across the search pattern, while creating a search query with a single-gesture interaction. Users sketch over part of the graph, establishing the level of similarity through either spatial deviations from the graph, or the speed at which they sketch (temporal deviations). In a user study, participants were significantly faster when using our temporally relaxed selection technique than when using traditional techniques. In addition, they achieved significantly higher precision and recall with our spatially relaxed selection technique compared to traditional techniques.

ACM Classification: H3.3 [Information Search and Retrieval]: Query formulation; 5.2 [Information Interfaces and Presentation]: User Interfaces—*Graphical user interfaces*.

General terms: Design, Human Factors.

Keywords: Time-series data, similarity queries.

UIST'09, October 4–7, 2009, Victoria, British Columbia, Canada.

INTRODUCTION

Time-series graphs are one of the most frequently used types of diagram for visualizing datasets [16]. Mapping time to a single spatial axis can make it relatively easy for a naïve viewer to comprehend the depicted behavior over time and visually search for large-scale patterns in visible portions of the graph. When datasets are large, however, graphs cannot be displayed at once, and manual search for potentially recurring patterns can become tedious. Therefore, tools for visualizing time-series graphs often allow users to specify patterns for exact or approximate automated search (e.g., [1,8,18,19]). However, these systems typically separate the process of specifying a pattern from the search process. Users can often control the level of similarity only after the pattern is specified, and often solely on a global level.

To address these problems, we propose *relaxed selection techniques* for querying time-series graphs. As shown in Figure 1, relaxed selection techniques allow *relative search* (to match the pattern wherever it occurs) [20] with a single input gesture. The user not only selects the relevant part of a displayed time-series graph, but simultaneously defines similarity constraints with the same gesture. We have developed two kinds of relaxed selection techniques: spatially relaxed selection and temporally relaxed selection. *Spatially relaxed selection* determines tolerances for the input query by examining spatial deviations from the graph: The user traces part of the displayed graph more or less precisely and the amount the sketch deviates from the graph, from

[1] The work reported in this paper was carried out during the first author's visit to Columbia University.

one part of the pattern to another, determines the level of similarity required to match that query pattern (Figure 1 top). *Temporally relaxed selection* uses input speed to derive tolerances: Temporal deviations in gesture speed are evaluated such that slower input speed indicates the need for a closer match, while faster input speed allows for bigger variations (Figure 1 bottom).

If a user ignores features of the underlying time-series graph while sketching, those points are filtered from the selection. The extent to which the filtered points stand out is used to determine a smoothing threshold for matching. The user can optionally refine a query after specifying it: The pattern can be altered in shape and tolerances can be adapted point-by-point. Thus, only a visual understanding of the displayed pattern is required to specify the search.

In the remainder of this paper, we first discuss related work on searching time-series graphs. Then, we describe our techniques and their implementation. Next, we present a user study comparing relaxed selection with two baseline approaches commonly used for searching time-series data. Finally, we conclude with a discussion of future work.

RELATED WORK

Interactive Query Facilities

Many data visualization tools provide extensive filtering capabilities that operate directly on the visualized data. The user must fine-tune these filters to obtain only the intended parts of the dataset. For example, Spotfire 2.2[2] and Tableau 4.1[3] concentrate on exploring and filtering entire datasets that are visualized based on spreadsheet-like tables. Users can select portions of a graph by "rubber-band rectangle" selection, much like selecting cells in a spreadsheet. Spotfire supports automatic clustering of records of equal length that exhibit similar behavior. It takes smoothing, normalization, and transformation into account when comparing two line graphs and does not compute similarity based on a rigid distance metric. Tableau offers automatic trend discovery and allows specification of user-defined functions to compare datasets. However, neither supports relative searching for individual patterns in graphs.

In TimeSearcher [5], users can place "timeboxes" (rectangular regions) over a set of graphs to select just those that pass horizontally through the timeboxes. Variable timeboxes add an adjustable degree of uncertainty in the time axis and provide somewhat tolerant queries [8]. Angular queries restrict pattern slopes to a specified range on the time axis [6]. Users can also create "searchboxes" around patterns to issue relative queries [1]. Queries can be adapted by the user after initial selection [2]. Similarity is judged using a Euclidean distance metric and the similarity threshold can be modified with a slider. However, using rectangular regions for selections can be problematic: graphs will be dismissed if they pass through a timebox vertically [4] or if just one data point falls outside a timebox [6]. Users cannot assign different similarity thresholds to different parts of a relative query and there is no visual aid to explain why a particular part of a time series is a match.

Line Graph Explorer [10] visualizes large collections of time-series data such that single lines of pixels encode line graphs and colors encode values. Users can sort or cluster the list of line graphs by similarity in user-specified time ranges to visually compare absolute distances between line graphs. Line Graph Explorer uses either Euclidean distance or Pearson correlation to judge similarity across graphs, but does not support relative pattern searching in a graph.

In contrast to directly selecting a portion of a graph, QuerySketch [21] supports query-by-example: The user draws a sketch and QuerySketch performs an *absolute search* (to match the pattern at the position in the graph at which it is specified), ranking the closest matches based on Euclidean distance. Users must sketch in an empty area, which can be difficult if the desired pattern is hard to sketch or the user is unsure of what they are looking for.

Keogh and Pazzani [9] found that human understanding of similarity does not necessarily coincide with strict Euclidean distance, because a user might ignore transformations, such as amplitude scaling and translation, when judging similarity. They employ subjective "relevance feedback" from users to iteratively determine the similarity metric used for matching. The user specifies a query by drawing it or selecting a subsequence of a time-series graph. Patterns are reduced to sequences of lines and associated weight factors express relative importance within the pattern. The best matches can be rated by the user, resulting in a modified query. However, the initial query does not specify similarity criteria, which only result from iterative, potentially time-consuming, refinement through rating.

QueryLines [18] supports approximate queries by drawing lines on the graphs. The simplest lines indicate minimum and maximum restrictions. Trend lines are similar to angular queries in TimeSearcher, and goal lines allow for three-point pattern specification (e.g., peaks and valleys). Relative and absolute queries can be created, and queries can be refined by adjusting the lines, but the user cannot adjust the level of similarity desired. As query lines are distinct objects and cannot be connected or grouped, absolute search for complex patterns requires many goal lines.

Symbolic Search Facilities

Many tools are based on symbolic representations of time-series graphs. Patterns [15] introduced the idea of fuzzy queries. Queries can be roughly defined by symbolized slopes of subsequences, and resulting matches are represented as symbols. Domain experts can arrange these symbols to define shapes of interest that will be found automatically in a time-series graph.

VizTree [13] automatically analyzes time-series data to detect anomalies. Recurring subsequences are visualized as branches in a tree, with frequent patterns drawn more prominently than rare ones. Thus, users can spot previously unknown patterns as non-prominent branches [12]. VizTree discretizes a time series into a string of symbols [11, 14] and employs a sliding window approach to construct the tree. When a branch is selected, VizTree visualizes the

[2] http://spotfire.tibco.com, retrieved November 20, 2008.
[3] http://www.tableausoftware.com, retrieved November 5, 2008.

distribution of that pattern throughout the graph. However, a user must specify the length and sample rate for the sliding window, which differ across datasets. To find a particular pattern, a user must translate it into its symbolic representation and select the corresponding branch in the tree [13], but cannot specify the level of similarity for a match.

Recognizing that users are frequently more interested in the overall shape of a subsequence than in its exact values, work on "approximate queries" [19] abstracts from raw data using an alphabet of symbols. Graph regions are described by simple functions or linear approximations. Users can then roughly specify a relative query symbolically.

RELAXED SELECTION TECHNIQUES

Unlike previous approaches for querying time-series graphs, relaxed selection techniques allow users to simultaneously select a portion of the graph as the search pattern and specify levels of similarity required when matching that can vary from one part of the pattern to another. Relaxed selection techniques rely on sketching a selection directly on top of the graph. By sketching, the user specifies not only the portion of the underlying dataset to select, but adapts it to the shape they have in mind. Therefore, they implicitly point out which parts of the selected data subsequence are meaningful for the search and how accurately they should be matched, and which parts can be ignored.

By either following or passing over peaks and valleys in the data while sketching, the user indicates the level of detail in which they are interested. They can ignore variations in the displayed graph that they consider unimportant or regard as noise by drawing a line that does not track these features. This behavior determines the noise threshold for later smoothing of the searched graph. Features that are ignored are dropped from the selection and do not participate in the matching process. Thus, as described below, a local maximum or minimum in the displayed graph is ignored during selection if the sketch does not have a corresponding local maximum or minimum. In the remainder of this paper, we will refer to the user's drawing, which is essentially just another time series, as the *(user) sketch*.

Filtering

Since processing queries on raw datasets is time-consuming and computationally intensive, systems often search on complexity-reduced representations of datasets [3]. Discussions with five financial analysts in an informal pilot study revealed that they perceive the local extrema (minima or maxima) of a time series to be indicative of its shape. Related work [7,17] supports this approach, which also leads to significant complexity reduction. In the remainder of this paper, we will use the term *filtering* to refer to this process of reducing a time series to its start point, end point, and all local extrema in between. As a result of filtering, a time series can lose detailed shape and consist of only straight lines thereafter. Internally, our implementations of relaxed selection techniques operate only on filtered sets of the displayed time series and user sketches.

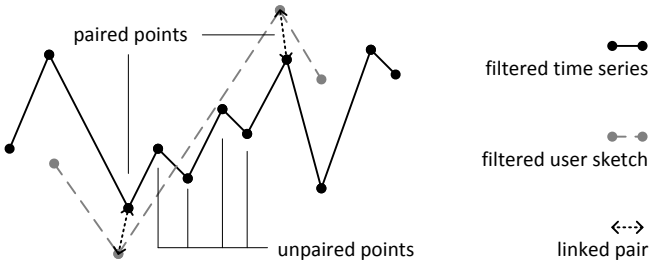

Figure 2: Pairing. Corresponding points (i.e., corresponding local extrema of the same type) from either series are linked together.

However, the user always sees the time-series graphs in their original form, comprising all measured values.

Pairing

Filtering a time series, as described above, facilitates a solution to tolerance specification. Relaxed selection techniques are based on the assumption that the user sketch resembles the displayed subsequence to a certain extent. Corresponding local extrema of the same type (i.e., both local minima and local maxima) can be found in each series. Therefore, relaxed selection relies on comparing the filtered user sketch with the filtered subsequence. We call corresponding points *pairs*. Every point in the filtered subsequence of the displayed data graph that does not correspond to a point in the filtered user sketch will be considered an outlier or irrelevant for the specified query. If there are more local extrema in the filtered user sketch than in the underlying filtered subsequence, the points in the filtered user sketch that have no corresponding points in the filtered subsequence will be discarded. Figure 2 exemplifies the process of pairing the filtered user sketch to the filtered version of the underlying time series. The maximum distance between two subsequent points of the filtered subsequence that have no corresponding points in the filtered user sketch determines the *noise threshold* for a relaxed selection. Therefore, the user implicitly communicates the amount of variation that they consider unimportant for the matching process. We will call the filtered subsequence of the underlying graph without outliers the *(user) selection* in the remainder of this paper.

Deriving Tolerances

Tolerances are assigned to all points in the filtered user selection: the start and end points and the set of local extrema in between. Tolerances express how far away corresponding points of a potential match can lie and still be part of a matching subsequence. Tolerances can be visualized by circles around local extrema—corresponding points in potential matches must lie in their respective circles; thus, the larger the circle, the greater the tolerance.

Spatially Relaxed Selection

The distances between paired data points determine the tolerances on a point-by-point basis. Figure 3 illustrates this process. Thus, users can exaggerate features that occur in a time-series graph and thereby define tolerances for each individual feature. They can draw peaks and valleys larger or smaller, depending on what they have in mind.

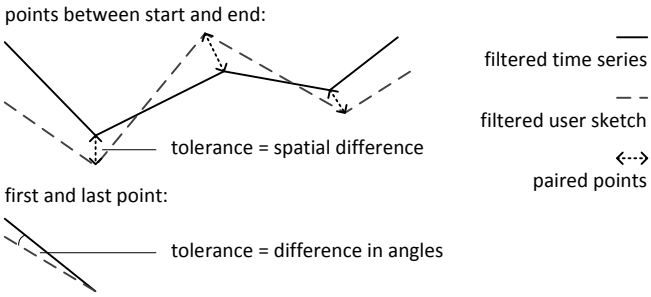

Figure 3: Spatially relaxed selection. The tolerance for each point in the selection is determined by the distance to its corresponding point in the displayed pattern. Tolerances for the first and last point of the user selection encode the angles of the rays that start at their immediate neighbors.

In our pilot study, we found that users begin and end sketching their selection rather sloppily, but tend to sketch intervening points in the time-series accurately. Therefore, relaxed selections consider spatial tolerances only for intervening points of a selection. Tolerances for end points are derived from differences in angles (Figure 3 bottom).

Temporally Relaxed Selection

This approach assesses the user sketch over time, so that portions that were drawn slowly specify the need for a high level of similarity (low tolerance), while quickly drawn portions indicate a low level of similarity (high tolerance). The underlying concept is that a user specifies a query carefully, and thus slowly, if they intend to define an accurate sketch, and sloppily, and thus quickly, if they intend to define a vague sketch. Figure 4 shows how tolerances are derived from input speed. A *section* contains all points of the sketch that are closer to its base than to another base, where a *base* is a local extremum of the sketch or a start or end point. The time spent drawing a section defines the tolerance of the point to which the section belongs. We use an inversely proportional function to convert duration for each segment to the tolerance value for the corresponding point of each base. We thereby employ lower and upper boundaries for tolerances. Similar to a spatially relaxed selection, tolerances for end points encode angles.

IMPLEMENTATION

We implemented the concepts described here in our application *SoftSelect*. SoftSelect was written in C++ and uses the Standard Template Library for data processing and the Qt Toolkit[4] for visualization. Our application was designed to be platform independent and compiles under Windows, Linux, and Mac OS X.

SoftSelect can be operated with a mouse cursor, but was designed to work with touch or stylus input. (We rely on touch-sensitive display manufacturers that provide mouse drivers that translate touch input into mouse operations.) We tested SoftSelect with a touch-sensitive 17" Planar PT170M monitor and a touch-sensitive 4.5" Sony Vaio VGN-UX280P. For data exploration purposes, we provide a circling gesture to zoom into the displayed graphs.

[4] http://trolltech.com/products/, retrieved November 20, 2008.

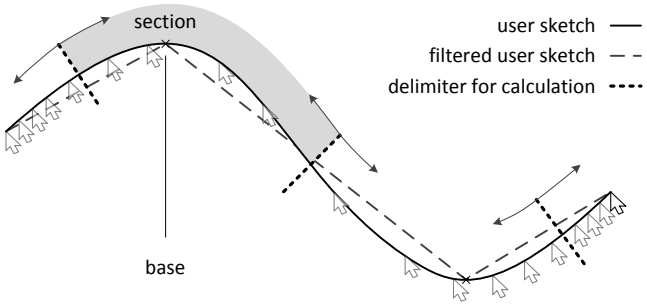

Figure 4: Temporally relaxed selection. Tolerances of local extrema and start and end points are determined by the time the user spends sketching the sections around them. A section contains all points that are closer to its base than to another base.

Matching

Since our focus is on user interface design, not search algorithm efficiency, we based the matching process in our implementation on a naïve sliding window approach [3]. (While a commercial product would use a faster algorithm to handle large datasets, we note that all examples in this paper and the study run in under one second.) To avoid the limitations of a fixed-size window, which we discuss below, we use a pointer that traverses the time series in which the query pattern should be found. As mentioned before, we focus on finding matches similar to the filtered user selection in a filtered time series. Figure 5 exemplifies the process. The user selection is successively translated on top of this series such that the second point of the filtered user selection coincides with the point in the filtered time series indexed by the pointer (Figure 5a). (We will take the selection start point into account later.) For the query pattern to match at the current position, the following steps have to be checked.

1. For each intervening point in the user selection, the closest (in Euclidean distance) corresponding point of the same type—either local maximum or minimum—must be found in the time series (Figure 5b). Whenever no corresponding point can be found (e.g., if a peak of the selection corresponded to a valley in the time series), the matching process can be stopped for the current position and the matching pointer can advance.
2. The user selection must be moved on top of the time series, so that every point of the subsequence has a distance to its corresponding point in the user sketch no greater than the tolerance of that corresponding point. Recall that since the tolerance represents the maximum distance allowed between a point in the user selection and its correspondent for a match, the tolerance can be visually interpreted as the radius of a circle centered about that point. Graphically speaking, for a match at the current position, all corresponding points in the subsequence need to lie within the circles around their respective points in the user selection (Figure 5c).
3. The maximum distance between two points of the filtered subsequence that have no corresponding point in the user selection has to be smaller than the noise threshold. That means that an outlier in the time series can be

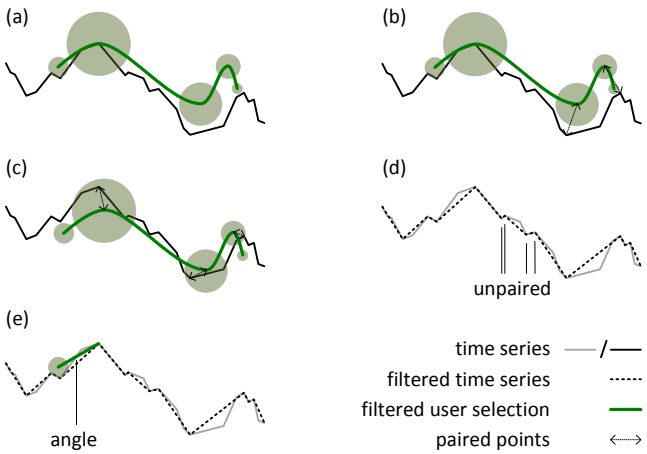

Figure 5: Matching process. (a) The user selection is overlaid on the filtered time series. (b) Corresponding points are determined. (c) The selection is repositioned so that pairs lie within the same circle. (d) Unpaired points are compared with the noise threshold. (e) For start and end points, angles are compared.

considered noise and, thus, insignificant for the matching process (Figure 5d).

4. In the displayed time series, there is not necessarily a point in the vicinity of the first or last point in the user selection. Therefore, differences in angles of the lines connecting the first two points and the last two points in either time series are compared against the tolerance value for the respective end point in the user selection (Figure 5e).

If all requirements are met, then there is a match at the pointer's current position.

Visualization of Matches

Matches are highlighted where they occur, as shown in Figure 6. The user selection is placed on top of a matching subsequence. Tolerances for each point are visualized as circles. All points in the time series that correspond to points in the user selection when this particular position was tested for a match will lie inside the circles around their corresponding point. This is intended to help the user understand why this position is a match and see how this section of the displayed graph resembles the query.

Query Adjustment

Related work has often used sliders to adjust a query. Buono et al. [1] describe how users of TimeSearcher 2 stated that tolerance sliders had no meaning to them. They proposed dedicated plus and minus controls for raising and lowering the tolerance instead. In SoftSelect, queries can be adjusted in two ways. First, the user can alter their selection by moving a point of the pattern. Second, the tolerance of an individual point can be adapted by dragging the edge of a circle around a data point. In each case, the revised search is reexecuted. The user can also replace the selection by sketching on top of the displayed graph again.

Two additional controls are provided to fine-tune all tolerances and modify the noise threshold. The first control adjusts all tolerances proportionally. Circles representing the

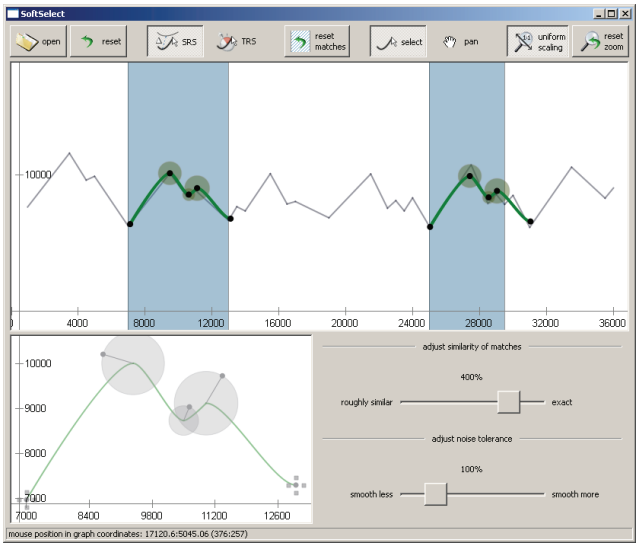

Figure 6: SoftSelect screen. The top part is the graph-display and main-interaction area. The bottom part allows query refinement (dragging query points) and tolerance adjustment (resizing circles).

tolerance of each point are updated as the user interacts with the system and the query is reexecuted. The second control scales the noise threshold. Smoothing and complexity-reduction is only performed internally, so changing the noise threshold has no effect on the displayed graphs.

DESIGN DECISIONS

Similarity and Transformation Resistance

Perng et al. [17] and Chortaras [3] added normalization as an obligatory step during matching, to account for transformations of the query pattern. Financial analysts in our pilot study expressed interest in restricting similarity thresholds in order not to leave too much room for interpretation: Deviations should be taken into account, but the selection should not be distorted, stretched, or adapted automatically. Therefore, we decided to use Euclidean distance at each inflection point, based on the point's tolerance. The user controls transformation invariance to some degree, such as amplitude stretching or offset translation. Possible query distortions for time warping [22] can be defined and matches become apparent through visualization. Perng et al. also noted the importance of smoothing raw data and proposed a smoothing function with rigid parameters. We defer smoothing until actual matching and smooth according to the user's sketch and refinements. Consequently, smoothing is highly user-dependent in relaxed selections and no step is executed before user interaction.

Sequential Matching Process

Chortaras [3] has outlined the disadvantages of a rigid sliding window: a constant window size is too static to allow for similarity matching and varying the size is computationally expensive. While it is possible to incorporate window size into the similarity metric [3], we believe that window size should have no influence at all on the matching process. Consequently, we work with only a pointer to one position in the searched time-series.

Absolute Queries

Relaxed selection, as implemented for this paper, is not suited to search for absolute matches. Techniques, such as QueryLines [18] can process queries that look for relative matches in the horizontal direction, but absolute matches in the vertical direction. We regard this as relative matching in both directions, preceded by eliminating all data points outside the selected vertical range. Since mechanisms to do this are widespread in most applications, we did not integrate them into our technique. Instead, we assume that this step can be done beforehand and our concepts work with the data that remains. Ryall et al. [18] also suggest displaying near matches. We offer this feature with the overall tolerance control. If the user adjusts it a little, SoftSelect will show formerly near matches amongst the matches.

EXPERIMENTAL EVALUATION

We conducted a within-subject, single-session, 60-minute user study in a controlled laboratory setting to examine the effectiveness of our proposed techniques, compare them to traditional approaches, and collect feedback. We compared our spatially relaxed (*SRS*) and temporally relaxed (*TRS*) selection techniques with two baselines: the common rubber-band rectangle selection technique (*RB*), as employed in commercial tools, and a query-by-example technique (*QE*), based on that used in QuerySketch [21], which we implemented to do relative search. We created the study application by extending SoftSelect with the two baseline techniques and their matching algorithms as employed in TimeSearcher and QuerySketch (A. Simeone, personal communication, 12/15/08; M. Wattenberg, personal communication, 12/12/08). We recruited 18 participants (13 male/5 female, 18–38 years old) through email and flyers. All participants were students, stated that they had no experience interacting with time-series data, and used a computer on a daily basis.

We selected nine real-world data sets from Hyndman's data library[5] and the time-series database of the Department of Mathematics at University of York[6], relating to traffic accidents, passenger rates on public transport, and temperature and precipitation. (We chose these because we believe that finding similarities in the resulting graphs could identify seasonal patterns.)

Experimental Setup

All participants used an AMD Athlon MP 2800+ 2.13 GHz computer with 2GB of RAM, running Microsoft Windows XP Professional SP3. An NVIDIA GeForce4 Ti 4600 provided video output to two monitors: a touch-sensitive 17" Planar PT170M (1280×1024 resolution), with which users interacted exclusively using a stylus, and a 20.1" Dell 2005FPW (1680×1050 resolution), which served as a secondary display for instructions, feedback, and status messages. Audio feedback was provided through an Aureal Vortex 8830 and desktop speakers.

[5] http://robjhyndman.com/TSDL/, retrieved December 5, 2008.

[6] http://www.york.ac.uk/depts/maths/data/ts/, retrieved December 5, 2008.

Task

To establish a standard for correct results, the participant was shown the pattern that they had to find and select in a time-series graph, along with a set of similar matches and their locations. The participant's task was to perform the selection with a specified technique such that they found all matches that were shown on the screen.

After loading a dataset, the application highlighted in red the area that we wanted the participant to select. Desired subsequences to match in the same time-series graph were highlighted in green. The task was to formulate a query with the given technique based on the subsequence highlighted in red. The application would search for matches in response to the query and highlight them in gray. Feedback was provided on the second screen in a separate window. This message window showed the technique to use to specify the query and an area colored either green or red, which provided visual feedback about success or failure. For invalid input, the application colored the area red, displayed a message on the screen, and gave audio feedback.

After formulating their initial query, the participant was asked to adjust or replace it, if necessary, to come as close to the presented set of matches as possible. When satisfied with the achieved matches, they could finish the task by pushing a button labeled "done" and proceed to the next task (*manual transition*). We additionally introduced a threshold for the maximum number of interactions, so that participants did not get frustrated during the study. If a user made 15 repeated adjustments to their query, the application would move on to the next dataset (*automatic transition*). Transitions to the next dataset or method were accompanied by a sound in both cases, whether manual or automatic.

Procedure

The experimenter gave each participant a brief introduction to the domain of time-series graphs, the need for query facilities, and the meaning of similar search results. He explained all four techniques in detail to the participant and then presented the study application. He demonstrated the process of specifying a selection or query, automatic and manual transitions to the next task, facilities for exploring the dataset, and the options for adjusting a query in each technique. In a subsequent practice phase, the participant could try out the system and learn how to use it with all selection techniques and two datasets. During this phase, the experimenter commented on interaction and gave feedback to improve performance. The final phase was the actual study session, in which each participant was observed using all four techniques with all nine datasets. After the user study, each participant filled out a post hoc questionnaire regarding personal preference and self-reflection after using the techniques.

Timing for all interactions was measured and stored automatically, along with the participant's sketches and selections. Actions such as query refinement with sliders or by changing the query pattern were also saved. This allowed us to reproduce the application's state before and after every interaction. We are therefore able to not only visualize matches to a user query after the study, but also recreate

the corresponding state of the application during a trial. In total, each participant used 4 selection techniques on 9 different datasets with up to 15 interactions per technique-dataset pair. The order in which the search patterns, selection techniques, and datasets were presented was counterbalanced across participants.

Hypotheses

We formulated five hypotheses:

H1. Specifying a query using *TRS* will be faster than when using any other technique. Participants will purposely sketch faster to achieve high tolerances for certain parts of a query, and therefore will take less time than they normally would.

H2. *QE* will be faster than *RB* and *SRS* for query specification. Participants will roughly and quickly reproduce the pattern of interest that is presented to them on the screen. With the other techniques, they will be more careful, because they are operating directly on the graph.

H3. *RB* will take the longest to make an accurate query. Since inclusion and exclusion of points is crucial with *RB,* we assume that participants will pay more attention to their selection, trying to include only those points they need.

H4. Query refinement for *SRS* and *TRS* will take longer on average than for *RB* or *QE*. We assume that given the ability to do point-by-point refinement, participants will take more time to adjust a query. We further believe that with *RB* and *QE*, participants will use the slider for quick and frequent adjustments.

H5. *SRS* will have the highest rate of relevant matches retrieved, because it will allow participants to express tolerances more accurately on a point-by-point basis.

Results

Our analysis includes only data collected during each participant's study session, not their earlier practice session.

Time Analysis

We applied two one-way repeated measure ANOVAs on mean selection time and mean adjustment time, respectively, in the completed trials with our participants as the random variable. We found a significant main effect on selection time ($F_{3,15}$ = 13.013, p < .001) and adjustment time ($F_{3,15}$ = 9.778, p < .002) for α = .05. With post-hoc t-tests using Bonferroni correction, we found three significant differences: On average, *TRS* was 0.6 sec faster than *RB* (p < .001) and 0.5 sec faster than *QE* (p < .02); in addition, *SRS* was 0.4 sec faster than *RB* (p < .03).

With regard to query adjustment, pairwise comparisons with Bonferroni correction showed three statistically significant differences: On average, refinement of *QE* queries was 0.15 sec faster than when refining *SRS* queries (p < .001), and 0.1 sec faster than when adjusting *TRS* queries (p < .05); furthermore, refining *RB* queries was 0.1 sec faster than refining *SRS* queries (p < .03). Selection and adjustment times are displayed in Figure 7.

Success Analysis

We compared the matches found by a participant to the matches that our user-study application highlighted in terms of precision and recall. First, we counted only exact matches and dismissed all matches that intersected, were enclosed in, enclosed, or lay outside a highlighted match. Second, we also took matches into account that intersected, were enclosed in, or enclosed a highlighted match. We considered those kinds of matches valid if their width was within 75–150% of the true match that they intersected. From a user's point of view, finding only the approximate position and extents of a similar match can still prove helpful. However, matches that are too big or too small do not contribute to the desired result. Results are shown in the left part of Figure 8.

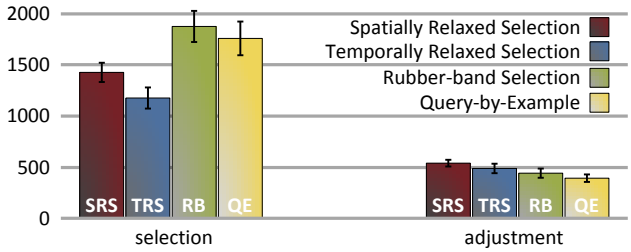

Figure 7: Average selection and adjustment times in milliseconds. Error bars indicate standard errors.

Exact matches

We applied a one-way ANOVA on precision of exact matches with our participants as the random variable. We found a significant main effect ($F_{3,15}$ = 39.421, p < .001) for α = .05. Pairwise comparisons with Bonferroni correction showed that *SRS* had significantly higher precision than all other techniques for exact matches. On average, the precision of *SRS* was 20% higher than *TRS*, 26% higher than *RB*, and 39% higher than *QE* (all p < .001). We found two additional significant differences. On average, *TRS* had 19% higher precision than *QE* (p < .001) and RB was 12% more precise than *QE* (p < .002).

Regarding recall rates, we applied a one-way ANOVA with our participants as the random variable and found a significant main effect ($F_{3,15}$ = 61.450, p < .001) for α = .05. In pairwise comparisons using Bonferroni correction, we found that our relaxed selection techniques had significantly higher recall rates than traditional techniques, but could not find a significant difference between *SRS* and *TRS* or *RB* and *QE*. On average, *SRS* had 31.5% higher recall than *RB* and 33.5% higher than *QE*; *TRS* had 20% higher recall than *RB* and 22% higher than *QE* (all p < .001).

Intersecting matches

We applied a one-way ANOVA on precision of intersecting matches with participants as the random variable. We found a significant main effect of the query technique on precision ($F_{3,15}$ = 41.545, p < .001) for α = .05. Using Bonferroni correction in pairwise comparisons, we observed that *SRS* had significantly higher precision than all other techniques: 21.5% more precise than *TRS* (p < .002), 16.5% more than *RB* (p < .03), and 35% more than *QE* (p < .001). In addition, we found that *QE* had significantly less precision than all other techniques. *TRS* was 13.5% more precise than *QE* (p < .004) and *RB* was 18.5% more precise (p < .002).

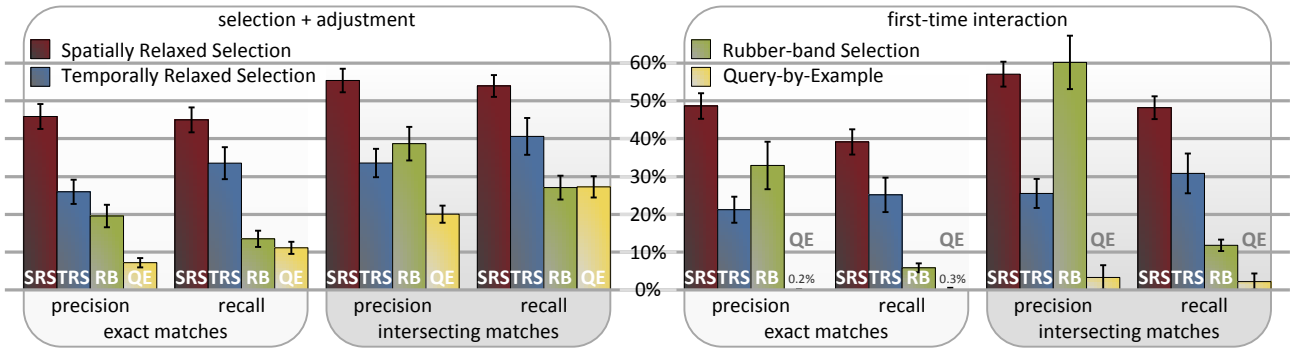

Figure 8: Precision and recall for exact and intersecting matches (more tolerant counting). The first group represents selection and query adjustments; the second shows only initial interaction. Error bars indicate standard errors.

In terms of recall, we applied a one-way ANOVA for $\alpha = .05$ with our participants as the random variable. We found a significant main effect of the technique on recall rates ($F_{3,15} = 55.245$, $p < .001$). Pairwise comparisons using Bonferroni correction showed that *SRS* had a significantly higher recall rate than all other techniques. On average, SRS was 13.5% more precise than *TRS* ($p < .05$), 27% more than *RB*, and 26.5% more than *QE* (both $p < .001$). We also observed a significantly higher recall rate for *TRS* compared to *RB* (13.5% on average, $p < .03$).

Success Analysis of Initial Interactions

We argued earlier that in contrast to traditional techniques, relaxed selection techniques let the user specify a degree of similarity while creating a selection. Unlike other techniques, regions of tolerances are part of relaxed selections from the outset. For that reason, we investigated techniques in terms of precision and recall again, considering only participants' initial interactions for each combination of data set and technique, not counting adjustments or redone selections. Results are shown in the right part of Figure 8.

Exact matches

We ran a one-way ANOVA on precision of exact matches with participants as the random variable. We found a significant main effect of the technique on the precision of the first interactions for $\alpha = .05$ ($F_{3,15} = 83.606$, $p < .001$). We found four significant differences in pairwise comparisons using Bonferroni correction and observed that *QE* had significantly lower precision than all other techniques. *SRS* had 27.5% higher precision than *TRS* and 48.5% higher than *QE*; *TRS* was 21% more precise than *QE* and *RB* was 32.5% more precise than *QE* (all $p < .001$).

Regarding recall, we ran a one-way ANOVA for $\alpha = .05$ and found a significant main effect of technique ($F_{3,15} = 57.869$, $p < .001$). Bonferroni–corrected pairwise comparisons showed a significantly higher recall of relaxed selection techniques compared to traditional techniques. On average, *SRS* was 33.5% more precise than *RB* and 39% more precise than *QE* (both $p < .001$); *TRS* had 19% higher precision than *RB* ($p < .003$) and 25% higher precision than *QE* ($p < .001$). In addition, we found a significant difference of recall between *RB* and *QE* (5.5% higher, $p < .004$).

Intersecting matches

In order to investigate precision of intersecting matches on first interactions, we applied a one-way ANOVA for $\alpha = .05$ with our participants as the random variable. We observed a statistically significant main effect ($F_{3,15} = 67.686$, $p < .001$). Post-hoc t-tests using Bonferroni correction showed five significant differences. On average, *SRS* was 31.5% more precise than *TRS* and 53.5% more precise than *QE* (both $p < .001$). TRS had 22% higher precision than *QE* ($p < .002$). Furthermore, *RB* had 34.5% higher precision than *TRS* and 57% higher precision than *QE* (both $p < .001$).

With regard to recall, we ran a one-way ANOVA for $\alpha = .05$ and found a statistically significant main effect ($F_{3,15} = 92.395$) of technique. Pairwise comparisons using Bonferroni correction showed that our relaxed selection techniques had significantly higher recall rates than traditional techniques. On average, *SRS* had 36.5% higher recall than *RB* and 46% higher than *QE* (both $p < .001$); *TRS* had 19% higher recall than *RB* ($p < .007$) and 28.5% higher than *QE* ($p < .002$). In addition, *RB* had significantly higher recall than *QE* (9.5%, $p < .007$).

Analysis of Qualitative Feedback

The post-hoc questionnaire featured room for comments and five-point Likert scale questions (1 = most negative, 5 = most positive) about ease of use, intuitiveness, and satisfaction. Figure 9 shows the results for rating the selection and adjustment processes. Non-parametric analysis of the results using 2-related samples tests (Wilcoxon) and k-related samples tests (Friedman) did not show significant differences in participants' responses. However, we could identify certain trends. For query specification, participants evaluated *QE* and *RB* higher than *SRS* and *TRS* for ease of use and intuitiveness. Participants mentioned problems with drawing rectangular selections around points in a time series. *SRS* and *QE* were rated highest for satisfaction. Figure 9 shows a bimodal response toward *TRS*: most participants either liked or disliked it.

We also asked participants to rank the four techniques based on intuitiveness, perception of speed, and personal preference for using the technique (1 = highest, 4 = lowest). Figure 10 shows the results of the ranking and confirms that participants considered *QE* and *RB* more intuitive than *SRS* and *TRS*. Surprisingly, participants ranked *RB* as a fast technique, whereas statistics show that they were, on average, significantly faster using *TRS*. Figure 10 also shows a bimodal response in the preference ranking of

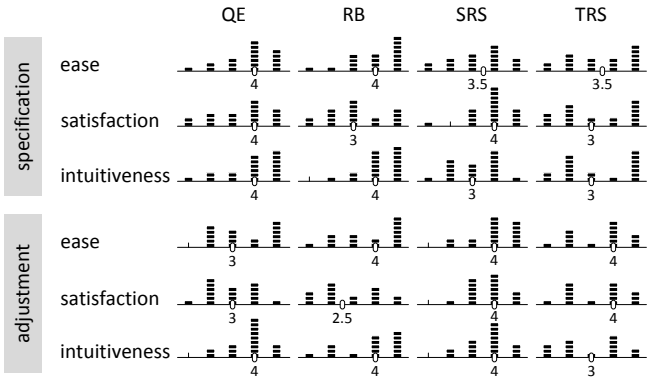

Figure 9: Qualitative feedback. Histograms of participant ratings of the four techniques on a five-point (low to high) Likert scale. Each item in a stack represents a participant who rated the technique with that value. Small numbers are median ratings.

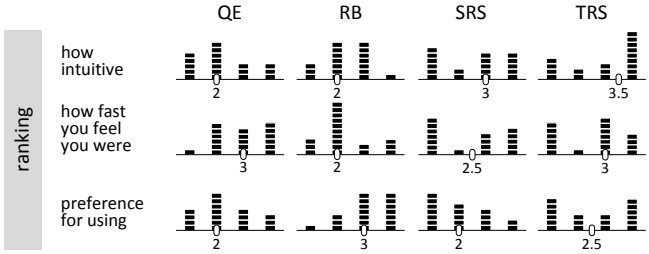

Figure 10: Qualitative ranking. Histograms of participant rankings of techniques from 1 to 4 (high to low). Each item in a stack represents a participant who gave the technique that rank. Small numbers are median ranks.

TRS. Despite some of the ease and intuitiveness ratings displayed in Figures 9 and 10, participants liked using *SRS* and *TRS*, and wrote in their comments that they favor sketching a query over other selection approaches.

Discussion

Our user study partially supports H1. *TRS* was significantly faster than the traditional techniques for specifying a query, although not significantly faster than *SRS*. We believe that users deliberately interacted faster using *TRS* because they knew that interaction speed influenced the results. We observed that participants were more careful when drawing a sketch using *SRS*, but were still faster than with *RB*.

From observations and feedback, we can understand the slow average values for *RB* and *QE*. Our study disproves H2, because *TRS* was faster than *QE*. We anticipated that participants would draw a sloppy sketch with *QE*. Instead, they carefully tried to reproduce the highlighted part of the time-series graph.

RB was slower than *SRS* and *TRS*, which partially supports H3. Participants tried to make a highly accurate selection so that all highlighted points fell inside their selection. We observed that sometimes it took them a long time to make sure that they had included all relevant points while excluding irrelevant points—especially when starting a selection. With relaxed selection techniques, they simply started and ended sketching without being too careful, which explains the difference in speed. However, this finding contradicts participants' feeling of speed; they ranked *RB* higher than both *SRS* and *TRS*.

Adjustment time for *QE* was lower than for relaxed selections. We believe this is due to the refinement options; as mentioned before, our application offered just a slider for adjusting *RB* or *QE*. As expected, participants moved the slider to where they felt they would obtain the best results rather quickly and frequently. Although we also provided sliders for *SRS* and *TRS*, participants often adjusted tolerances on a point-by-point basis and altered the query shape, which took them longer to accomplish. This partially supports H4. However, the data that we collected suggests that query refinement on the actual user selection is not much slower than with a simple slider.

The user study supported H5, because *SRS* had higher precision than all other techniques for both exact and intersecting matches. In addition, *SRS* had higher recall than all other techniques for intersecting matches. When comparing only exact matches, we understand the low precision of traditional techniques; the matching algorithm works on a point-by-point basis and compares a constant number of points—the number of points in the user selection. If a match in the time series had more or less points, *RB* was not able to match this result exactly. From observations and feedback, we believe that *QE* achieved only low precision, because participants were not able to reproduce the pattern of interest precisely with the same dimensions. When comparing intersecting results, *RB* and *QE* achieved much better results and *RB* notably reduced its difference to *SRS*.

We noted that one drawback of traditional techniques is that they separate the process of query specification from the process of finding matches. Only in the latter process can users adapt options for similarity matching. In contrast, relaxed selection techniques allow users to communicate levels of tolerance implicitly while specifying a query. Results achieved by participants' first interactions with the datasets disprove H5 in this case. For intersecting matches on first-time interactions, *SRS* and *RB* both had higher precision than *TRS* and *QE*. Regarding recall, however, our study showed that the two relaxed selection techniques both have higher possibilities of retrieving a relevant match than the traditional techniques. This held for both exact and intersecting matches on first-time interactions.

Overall, our user study showed that relaxed selection techniques benefit from enabling the user to specify tolerances at query time, and to adapt both shape and tolerance of the query on a point-by-point basis. Of the 18 participants, 13 chose *SRS* or *TRS* as their most preferred technique. We received various comments from participants that *SRS* and *TRS* are not intuitive, but—once one is used to them—are easy to use and achieve satisfactory results. While many pilot participants expressed concerns about the importance of input speed, formal study participants often verbally mentioned *TRS* as their favorite technique. Preference ranking for *TRS* was bimodal: out of all 18 participants, a third ranked it first and a third ranked it last. The differences in evaluations of query adjustment for *RB* and *QE*, and

SRS and *TRS* were surprising, since *RB* and *QE* shared the same slider for query refinement, and *SRS* and *TRS* shared the same refinement concepts.

CONCLUSIONS AND FUTURE WORK

We presented relaxed selection techniques, which evaluate a sketching gesture with which the user describes a search pattern within a time-series graph by drawing over part of the graph. The selected portion of the graph is augmented with tolerance constraints that are derived from the gesture on a point-by-point basis. These constraints are used during the search process to find matches that are sufficiently close to the user query. Our user study, which compared relaxed selection techniques with two baseline techniques, showed that participants achieved high precision and recall using relaxed selection techniques, including higher precision and recall for the spatially relaxed selection technique than for the baselines. Participants were also faster using the temporally relaxed selection technique when creating a selection than using the baselines. A majority chose either the spatially or temporally relaxed selection technique as their most preferred technique. Overall, we believe that relaxed selection techniques can allow users to achieve better results than traditional techniques after a short training period.

There are many possibilities for future work. The financial analysts in our informal pilot study, who were used to traditional tools, argued that input speed should not be taken into account. In contrast, the considerably younger participants of our formal study mentioned a sense of achievement with temporally-based relaxation. Because of varying attitudes potentially resulting from demographic factors, we plan to incorporate a user profile in future versions to adapt to general input speed and account for a learning effect, and continue our work with financial analysts to obtain more feedback.

So far, relaxed selection techniques allow for limited scale invariance (i.e., only as defined by the tolerance regions). Future versions might allow for explicit scaling transformations and normalization as an approach to vertical stretching of patterns. Circular tolerance regions could be replaced by ellipses, which constrain potential matches further through directionality. In addition, a slider for constraining vertically relative matching could be introduced. When the range is set to a low value, the user query will match only subsequences of the time series within a small range above or below the vertical position of the initial selection.

Our user study showed that additional attention should be paid to query refinement. To adapt a query on the screen, pixel-precise input was necessary. Although we allow for tolerant input when adjusting circle sizes and point positions, users still found it hard, especially with very small circles. This could be addressed through multi-touch input; for example, a stretch gesture could be used for resizing, while a regular touch could be used to adjust point positions. Regarding visual feedback, users would like to know why a certain position is not a match. One solution would be to allow the user to request that the pattern be overlaid at any position of the displayed time-series graph.

ACKNOWLEDGMENTS

We thank Steven Henderson, Sean White, Sean Gustafson, Ohan Oda, and Lauren Wilcox for discussions.

REFERENCES

1. Buono, P., Aris, A., Plaisant, C., Khella, A., and Shneiderman, B. (2005). Interactive pattern search in time series. *Proc. VDA'05*, 175–186.
2. Buono, P., Simeone, A.L. (2008). Interactive shape specification for pattern search in time series. *Proc. AVI'08, 480–481.*
3. Chortaras, A. (2002). Efficient storage, retrieval and indexing of time series data. Master's thesis, Imperial College of Science, Technology and Medicine, University of London.
4. Hochheiser, H., Baehrecke, E., Mount, S., and Shneiderman, B. (2003). Dynamic querying for pattern identification in microarray and genomic data. *Proc. ICME'03*, 453–456.
5. Hochheiser, H. and Shneiderman, B. (2001). Interactive exploration of time series data. *Discovery Science*, 441–446.
6. Hochheiser, H. and Shneiderman, B. (2004). Dynamic query tools for time series data sets: Timebox widgets for interactive exploration. *Information Visualization*, 3(1):1–18.
7. Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S. (2001). Locally adaptive dimensionality reduction for indexing large time series databases. *SIGMOD Rec.*, 30(2):151–162.
8. Keogh, E., Hochheiser, H., and Shneiderman, B. (2002). An augmented visual query mechanism for finding patterns in time series data. *Proc. FQAS '02*, 240–250.
9. Keogh, E. and Pazzani, M. (1999). Relevance feedback retrieval of time series data. *Proc. SIGIR'99*, 183–190.
10. Kincaid, R. and Lam, H. (2006). Line graph explorer: Scalable display of line graphs using focus+context. *Proc. AVI'06*, 404–411.
11. Lin, J., Keogh, E., Lonardi, S., and Chiu, B. (2003). A symbolic representation of time series, with implications for streaming algorithms. *Proc. DMKD'03*, 2–11.
12. Lin, J., Keogh, E., Lonardi, S., Lankford, J., and Nystrom, D. (2004a). Visually mining and monitoring massive time series. *Proc. KDD'04*, 460–469.
13. Lin, J., Keogh, E., Lonardi, S., Lankford, J., and Nystrom, D. (2004b). Viztree: A tool for visually mining and monitoring massive time series databases. *Proc. VLDB '04*, 1269–1272.
14. Lin, J., Keogh, E., Lonardi, S., and Patel, P. (2002). Finding motifs in time series. *ACM SIGKDD Workshop on Temporal Data Mining,* 53–68.
15. Morrill, J. (1998). Distributed recognition of patterns in time series data. *Communic. ACM*, 41(5):45–51.
16. Pan, W. (2002). A comparative review of statistical methods for discovering differentially expressed genes in replicated microarray experiments. *Bioinformatics*, 18(4):546–554.
17. Perng, C.-S., Wang, H., Zhang, S., and Parker, D. (2000). Landmarks: A new model for similarity-based pattern querying in time series databases. *Proc. Data Eng. '00*, 0:33.
18. Ryall, K., Lesh, N., Lanning, T., Leigh, D., Miyashita, H., and Makino, S. (2005). Querylines: Approximate query for visual browsing. *Proc. CHI '05*, 1765–1768.
19. Shatkay, H. and Zdonik, S. (1996). Approximate queries and representations for large data sequences. *Proc. ICDE'96*, 536–545.
20. Silva, S. and Catarci, T. (2000). Visualization of linear time-oriented data: A survey. *Proc. WISE'00*, 310.
21. Wattenberg, M. (2001). Sketching a graph to query a time-series database. *Proc. CHI'01*, 381–382.
22. Yi, B.-K., Jagadish, H., and Faloutsos, C. (1998). Efficient retrieval of similar time sequences under time warping. *Proc. ICDE'98*, 201–208.