Exploratory Analysis of Time-Series with ChronoLenses

Jian Zhao, Fanny Chevalier, Emmanuel Pietriga, and Ravin Balakrishnan

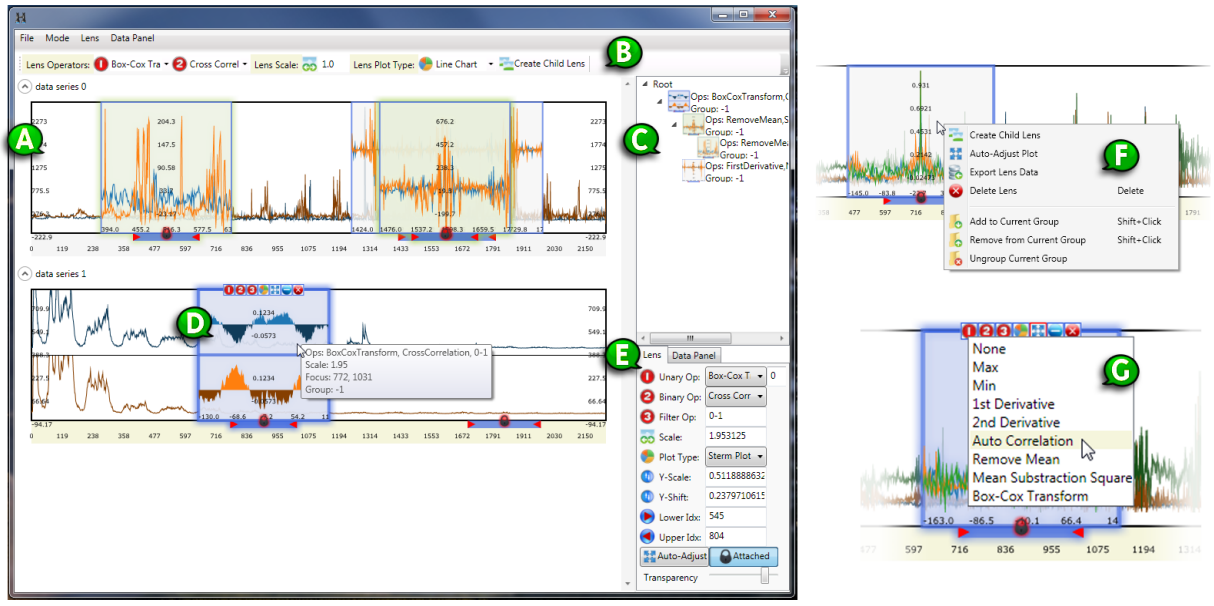

Fig. 1. The ChronoLenses interface includes (A) a charts panel showing the time-series; (B) a lens creation toolbar; (C) a lens analysis pipeline view of (D) the currently selected lens; (E) a property panel showing details of the currently selected lens. A context menu (F) can be invoked to perform lens-based operations, and (G) the lens toolbar allows quick access to a lens' parameters.

Abstract—Visual representations of time-series are useful for tasks such as identifying trends, patterns and anomalies in the data. Many techniques have been devised to make these visual representations more scalable, enabling the simultaneous display of multiple variables, as well as the multi-scale display of time-series of very high resolution or that span long time periods. There has been comparatively little research on how to support the more elaborate tasks associated with the exploratory visual analysis of time-series, e.g., visualizing derived values, identifying correlations, or discovering anomalies beyond obvious outliers. Such tasks typically require deriving new time-series from the original data, trying different functions and parameters in an iterative manner. We introduce a novel visualization technique called ChronoLenses, aimed at supporting users in such exploratory tasks. ChronoLenses perform on-the-fly transformation of the data points in their focus area, tightly integrating visual analysis with user actions, and enabling the progressive construction of advanced visual analysis pipelines.

Index Terms—Time-series Data, Exploratory Visualization, Focus+Context, Lens, Interaction Techniques.

1 Introduction

Time-series data are found in almost every domain, ranging from finance to many engineering and scientific disciplines. Time has inherently unique characteristics: for most purposes outside the theory of relativity, things evolve over time, but time does not depend on other variables. Time is considered uniform and absolute. It is thus often treated as a special variable, in terms of both how the data is structured and how it is presented to users. Tasks associated with the analysis of temporal datasets typically focus on the evolution of other (dependent) variables with respect to time: identifying trends and recurring patterns, establishing correlations and possibly predicting the future based on past and current behavior.

Visual representations of time-series take advantage of people's innate perceptual abilities to process information and detect structure [32], making it significantly easier for users to discover trends and patterns at different scales, but also to identify anomalies in the data [10, 31]. However, basic time-series visualizations using line plots do not scale well; and as time-series data are often very large, featuring multiple, possibly heterogeneous, dependent variables measured for long periods of time and/or at high sampling rates, visualization of real-world time-series data poses significant challenges and has been an active area of research for many years. Many interactive visualization tools have been developed to address this scalability problem, offering innovative alternatives to the common line plot visualizations or enhancing the visualization with advanced interactive features.

There has been comparatively little research on how to support the more elaborate tasks typically associated with the exploratory visual analysis of time-series, e.g., visualizing derived values, identifying correlations, or identifying anomalies beyond obvious outliers. Such tasks typically require deriving new time-series from the data, visualizing those time-series and relating them to the original data plots.

- *Jian Zhao is with DGP, University of Toronto, Canada. E-mail: jianzhao@dgp.toronto.edu.*
- *Fanny Chevalier is with OCAD University, Canada. E-mail: fchevalier@ocad.ca.*
- *Emmanuel Pietriga is with INRIA, France. E-mail: emmanuel.pietriga@inria.fr.*
- *Ravin Balakrishnan is with DGP, University of Toronto, Canada. E-mail: ravin@dgp.toronto.edu.*

Manuscript received 31 March 2011; accepted 1 August 2011; posted online 23 October 2011; mailed on 14 October 2011.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

Visual exploration techniques take advantage of human abilities to drive the data exploration process and are especially useful for undirected searches, when users know little about the data or have only vague exploration goals [23, 35]. As emphasized by Keim's visual analytics mantra – *"Analyze first, Show the important, Zoom, Filter and analyze further, Details on demand"* [24] – this process is iterative. For it to be efficient, visual representations, that support human judgment, and interactions, that re-parameterize the visual representation, should be tightly integrated, enabling users to quickly choose and refine parameter values that best suit the task at hand [2]. New plots derived from the original data should be put in context and made easy to relate both to the original data and to other plots that have been derived as part of the exploratory process.

We introduce a novel, domain-independent time-series visualization technique called ChronoLenses, aimed at supporting users in exploratory visual analysis tasks. ChronoLenses, as many other types of lenses, delimit a region of interest in the data to be put in focus through magnification [12, 33], visual filtering [15] or other arbitrary transformations of the underlying content [8]. Based on the metaphor of direct manipulation, ChronoLenses perform on-the-fly transformation of the data points in their focus area, tightly integrating visual analysis with interaction. Users can build pipelines composed of lenses performing various transformations on the data (e.g., *remove mean, compute 1st derivative, auto-correlation*), effectively creating flexible and reusable time-series visual analysis interfaces. At any moment, users can change the parameters of already created lenses, with the modifications instantaneously propagating down through the pipeline, providing immediate visual feedback that supports the iterative exploration process. Figure 1 gives an overview of the technique.

After a review of related work, we introduce tasks and requirements associated with the exploratory analysis of time-series, gathered from expert users in varied domains including operations monitoring and control for a radio-observatory, environmental research related to weather forecast, as well as financial and network streaming data analysis. We then describe the general concept of ChronoLenses, followed by its implementation in a research prototype application that provides rich interaction and analytical support for time-series. Example analysis pipelines follow with usage scenarios based on our work with network analysts, and astronomers of the ALMA radio-observatory, demonstrating how ChronoLenses can be used for the monitoring and offline analysis of complex time-series data. We conclude the paper with a discussion of the current implementation's limitations.

2 Related Work

2.1 Time-series Visualization

Visual representations of time-series date back to the 18th century with seminal work using line charts by Playfair [37]. Many new static and dynamic techniques have been proposed since then. See [1, 32, 36] for relevant surveys. Some innovative techniques focus on helping users identify periodic patterns in the data. For instance, van Wijk [20] uses a calendar-based display to visualize time-oriented series aggregated on a daily, weekly or monthly basis via similarity clustering. Other visualizations, such as SpiralGraph [40] and SpiralView [7], lay out the data on a spiral-shaped time axis to reveal cycles. These techniques can yield meaningful representations of data that feature periodicity, making recurring patterns easy to spot. VizTree [28] takes a different approach, computing a symbolic representation of time-series and then representing the sequence of symbols using a suffix tree. The resulting visualization is radically different from other temporal visualizations and can be disconcerting at first, but represents a potentially powerful alternative to other techniques for identifying patterns in large datasets.

Most of the work on time-series visualization has focused on the scalability of the more conventional line plot and bar chart representations, either by proposing variations on the original plotting techniques or by enhancing them with advanced interactive visual filtering techniques. Variations on conventional line plots include Small multiples and sparklines [37], Horizon graphs [18], and Braided graphs [22]. These techniques focus on the issue of optimal space management, enabling the display of an increased number of time-series compared to regular line plots. Lopez-Hernandez *et al.* [29] address the same problem for a specific type of time-series – univariate oscilloscope digital signals – by wrapping the signal in time, representing the different traces on layers that can be brushed by users.

2.2 Multi-scale Representations

Multi-scale visualization techniques adapt the representation depending on available screen real-estate. Time-series can be represented at several levels of detail and abstraction using different qualitative scales, or hybrids combining qualitative and quantitative information [6], gradually revealing more information as more space gets allocated to the chart, or as the user zooms in a region of interest [27]. Line Graph Explorer [25] provides an overview+detail interface for the exploration of large collections of time-series. It displays a compact overview of the entire collection by encoding the y-dimension of individual line graphs using color instead of space and viewing selected graphs in detail as standard line graphs. The technique is also interesting as the overview visualization lends itself well to sorting and clustering of the graphs using various similarity distances.

Hao *et al.* [17] present a space-filling, multi-resolution matrix representation of time-series where the color of a cell encodes the magnitude of the corresponding value in the data. The same authors propose another space-filling visualization capable of displaying multiple time-series, where the various charts are organized based on their importance in a treemap-like structure [16]. Stack zooming [21] uses a relatively similar technique, taking advantage of the one-dimensional nature of time-series to present them as hierarchies of strips, where each strip contains magnified versions of one or more region(s) of interest delimited in the above strip. Stack zooming can also be seen as a specialization of the DragMag [39] that supports recursive magnification and automatic arrangement of magnification windows. The technique enables efficient exploration of time-series at multiple levels of detail. LiveRAC [31] offers another variation on this approach, organizing charts in a reorderable spreadsheet-like matrix that allows side-by-side visual comparisons at multiple levels of detail. The technique supports semantic zooming [12], providing visual representations adapted (cell-wise) to the allocated screen space.

2.3 Lens-based Interaction for Time-series Visualizations

Timeboxes [19] take a different approach to the scalability problem. Timeboxes act like visual filters that specify constraints on what time-series to display, only showing plots that intersect the one or more boxes that form conjunctive queries, enabling the dynamic exploration of large data sets by direct manipulation of the rectangular boxes. In that sense, timeboxes can be seen as data filtering lenses, that have an impact on the visualization beyond the region of interest.

SignalLenses [26] provide another type of lens, conceptually closer to the usual lens-based focus+context visualization techniques [12]. SignalLenses are used for the visual analysis of large electronic time-series and help perform tasks such as anomaly detection and motif discovery. They provide in-place magnification of the signal, achieving a smooth transition between the focus and context regions through 1D distortion [11, 33]. Though they can provide additional measurement tracks to assist the exploration of time-series data by computing time-aligned properties, SignalLenses are limited to the magnification of plots in the region of interest. The MagicAnalytics lenses of KronoMiner [41] go beyond simple magnification, building on the concept of Magic Lenses [8] originally designed for arbitrary 2D graphics. MagicAnalytics lenses compute and display the result of a function involving two time-series. They represent one of the basic building blocks of ChronoLenses, though limited to single step transformations.

ChronoLenses go beyond these simple transformations, offering multi-step transformations of one or more time-series. These can be purely visual transformations, or transformations that change the value of the data (1st derivative, point-wise maximum, etc.). ChronoLenses enable users to construct elaborate visual analysis pipelines, with modifications automatically propagating downstream, providing immediate visual feedback that supports the iterative exploration process.

3 Tasks and Design Requirements

To better understand the needs for interactive visualization when exploring time-series data, we conducted informal interviews with expert users from various domains, including: astronomers, experts doing research in network streaming data analysis, finance, and weather forecasting. Based on their feedback and on the literature [4], we identified low-level tasks that users typically perform to carry out time-series analyses. From these tasks, we derived design requirements for the ChronoLenses technique, as introduced in this section.

The analysis of time-series datasets typically implies feature extraction and data comparison by transforming one or more time-series into another. Such computations usually require performing one, or a combination of, the following low-level tasks:

T1 Single-data stream transformation. Transform each data point of a series by applying an operator. Examples are data alteration, e.g., Fourier transforms and Box-Cox transforms [9]; bias reduction, e.g., remove means and remove trends; repeated pattern identification, e.g., auto-correlation; and operations related to delays and time lags, e.g., differencing and seasonal differencing.

T2 Cross-data stream analysis. Compute a new time-series from two or more input time-series. Examples are data comparison, e.g., subtraction; similarity examination, e.g., inner product; relationship discovery, e.g., cross-correlation; and series aggregation, e.g., point-wise maximum value.

However, exploring time-series often implies going through a more elaborate analysis pipeline that combines various tasks [24]. Practically speaking, this implies that users should be able to make compound queries on the data by iteratively performing sequences of low-level tasks **T1** and **T2**, combining and modifying the transformation parameters as part of the visual exploration process. From these observations we derive the following requirements:

R1 Dynamic transformations. Low-level tasks **T1** and **T2** are the core components of the visual analysis process. The transformations that support them should be easy to perform through operators applied to the input time-series data. Visual representations and interactions that re-parameterize these transformations should be tightly integrated so as to facilitate exploratory analysis.

- (a) *Dynamic selection of input region of interest:* users should be able to dynamically select and modify what timespan(s) in the input data are to be processed through the operators.
- (b) *Dynamic transformation parameters:* users should be able to easily configure and edit transformation parameters.
- (c) *Immediate visual feedback:* the system should provide instant visual update to reflect these changes.

R2 Dynamic analysis pipeline. Enabling the easy combination of operators makes it possible to progressively build and refine the analysis pipeline, thereby helping formulate complex queries [24].

- (a) *Dynamic composition:* users should be able to build the analysis pipeline iteratively, combining basic transformations through the incremental composition of operators that take as input arbitrary combinations of time-series from the original dataset and time-series that result from earlier transformation steps upstream in the pipeline.
- (b) *Reuse of intermediate results and easy backtracking:* The above requirement entails that all intermediate time-series transformation steps in the pipeline should be reusable as input to downstream operators, enabling users to branch out and explore, and possibly compare, alternatives at any point while sharing the data transformation steps that come earlier in the analysis process.
- (c) *Visual representation of the pipeline:* the system should provide an overview of the analysis pipeline, helping users maintain a mental map of the transformation steps and keep track of the exploration history.

We also rely on general principles for visual exploration as listed in [41], including: direct manipulation; overview first, zoom & filter, details on demand; and support for dynamic multi-focus exploration.

4 ChronoLenses Framework

ChronoLenses is an interactive visualization technique for the exploratory analysis of time-series data, whose design was driven by the above requirements. It computes on-the-fly transformations of data points and displays the result of those transformations in place, using the metaphor of lenses. The MagicAnalyticsLens technique introduced in [41] relies on the Magic Lens concept [8] and forms one of the basic building blocks for ChronoLenses. While MagicAnalytics lenses enabled users to apply four single-step basic transformations to exactly two input time-series, ChronoLenses extends the concept, allowing for an arbitrary number of input time-series and introducing several additional operators. But most importantly ChronoLenses enable multi-step transformations to be specified, where the result of time-series transformed through a given lens can serve as input to another lens, and where multiple pipelines can be branched out to explore multiple alternative visualizations simultaneously, easing the formulation of complex queries.

4.1 Lens Parameters

We define a lens $\mathscr{L}$ as the transformation of an input time-series in the focus region of that lens into a resulting time-series. Time-series can be seen as streams of data that get transformed through the lenses that constitute an analysis pipeline structured as a dataflow. Transformations are computed on-the-fly according to a set of parameters that can be dynamically adjusted[1]. To support tasks **T1** and **T2**, $\mathscr{L}$ can either transform a single data stream, or perform cross-data stream operations. Data streams can be univariate or multivariate. Each lens $\mathscr{L}$ is defined by four transformations, all optional:

Unary Operator: $\mathscr{L}_{unary}(\cdot)$ defines the transformation that applies to a single data stream in the focus region of the lens (**T1**);

Binary Operator: $\mathscr{L}_{binary}(\cdot,\cdot)$ defines the transformation that takes the data in the focus region of the lens (processed by $\mathscr{L}_{unary}$ if set to anything else than the identity transform) as the first operand, and the output data stream resulting from the parent lens in the hierarchy (detailed later), if any, as the second[2] operand (**T2**);

Filter: $\mathscr{L}_{filter}(\cdot,\theta)$ defines visual filters that hide time-series specified by parameter θ (applies to multivariate data streams only);

Scaling: $\mathscr{L}_{scale}(\cdot,s)$ determines the magnification factor s applied to the data rendered in the lens' focus.

Operators $\mathscr{L}_{unary}$ and $\mathscr{L}_{binary}$ perform actual computations on the input data and can take some data points outside the lens' time span, such as when computing the 1st derivative. On the contrary, $\mathscr{L}_{filter}$ and $\mathscr{L}_{scale}$ only affect the visual representation of the processed data within that time span. They do not need to access data outside it.

4.2 Pipelines of Lenses

As mentioned earlier, the output of a lens, i.e., the time-series resulting from the transformation of an input time-series, can be fed to one or more lenses. In other words, lenses can be piped, effectively creating a lens-based data flow pipeline that can be used to progressively build elaborate visual analysis interfaces. The hierarchical combination of lenses relies both on a layering system and on cross-data stream parent-child relationships between lenses. The tree structure not only helps users keep track of the sequence of exploration steps, but also makes it possible to *backtrack*, that is, adjust the intermediate processing stages iteratively, the system providing immediate visual feedback

[1] ChronoLenses support the representation of stacked multivariate time-series. In that case, each individual series is processed separately using the $\mathscr{L}$ transformation, and is then stacked in the lens' frame.

[2] Most of the operations that consider more than two input data streams can usually be simulated by cascading a series of binary operations.

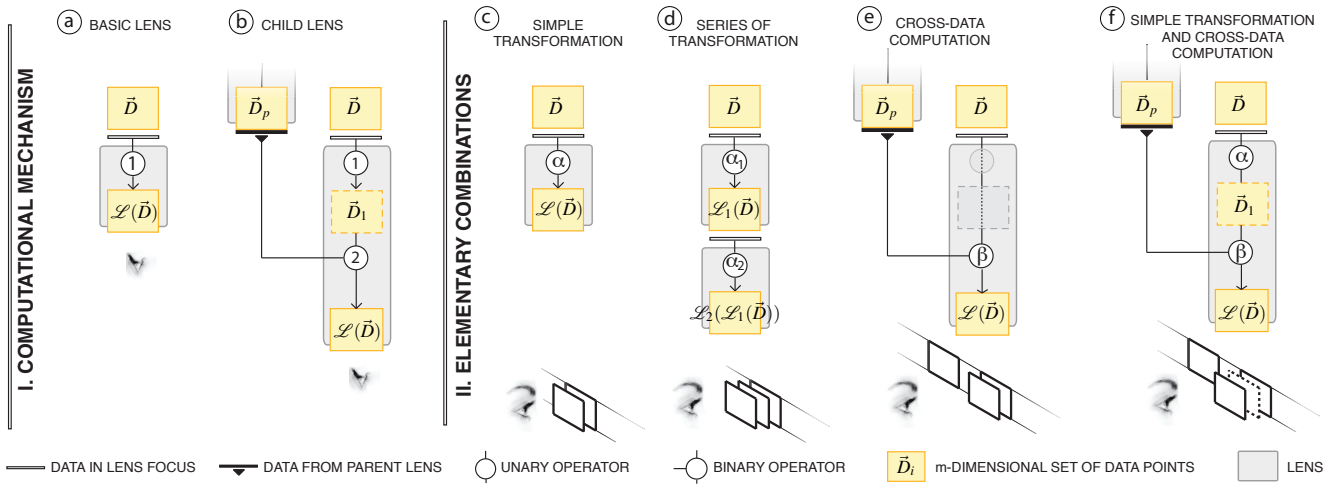

Fig. 3. Schematic representation of the ChronoLenses combination mechanism (I), and all possible elementary combinations (II).

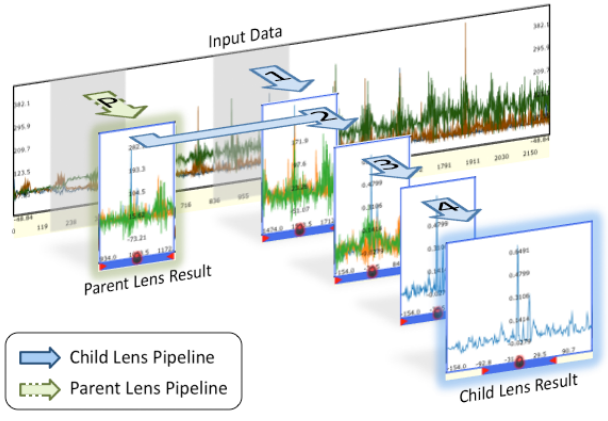

Fig. 2. ChronoLenses rendering pipeline for lens $\mathscr{L}$. (P) processing pipeline in parent lens; (1) applying *Unary Operator* $\mathscr{L}_{unary}$; (2) applying *Binary Operator* $\mathscr{L}_{binary}$; (3) applying *Filter Operator* $\mathscr{L}_{filter}$; (4) applying *Scaling Operator* $\mathscr{L}_{scale}$.

of the consequences of these parameter adjustments by propagating them downwards in the hierarchy. In the following, we describe the steps involved in rendering data seen through a lens, and then describe how lenses can be combined to build a full analysis pipeline.

4.2.1 Rendering Pipeline

Figure 2 illustrates the steps of the rendering pipeline applied to some multivariate input data, eventually leading to the visualization of the resulting time-series through lens $\mathscr{L}$, that gets part of its input from a parent lens.

More formally, the rendering pipeline can be described as follows. Let $\vec{D} = (d_1, d_2, \ldots, d_m)^T$ be the m-dimensional set of data points defined by the focus region of lens $\mathscr{L}$. We note $\mathscr{L}_{op}(\vec{D}_i)$ the result of applying one of the transformations steps to the data, where $op \in \{unary, binary, filter, scale\}$. The final result $\mathscr{L}(\vec{D})$ is obtained by performing the following four computational steps:

C1 Apply unary operator $\mathscr{L}_{unary}$, such that $\vec{D}_1 = \mathscr{L}_{unary}(\vec{D})$ (Figure 2-1);

C2 Apply binary operator $\mathscr{L}_{binary}$, such that $\vec{D}_2 = \mathscr{L}_{binary}(\vec{D}_1, \vec{D}_p)$ (Figure 2-2), where $\vec{D}_p$ is the set of data points resulting from applying the parent lens to the same or to another set of input data points (Figure 2-P);

C3 Apply visual filter operator $\mathscr{L}_{filter}$ such that $\vec{D}_3 = \mathscr{L}_{filter}(\vec{D}_2, \theta)$, where θ defines the subset of time-dependent variables to be rendered in case of multivariate input (Figure 2-3);

C4 Apply scaling operator $\mathscr{L}_{scale}$ such that $\vec{D}_4 = \mathscr{L}_{scale}(\vec{D}_3, s)$, where s defines the magnification factor (Figure 2-4). We note $\mathscr{L}(\vec{D}) = \vec{D}_4$ the final result.

Any of the four operators can be set to *null*, in which case the associated step is equivalent to an identity transform. As mentioned earlier, operators $\mathscr{L}_{unary}$ and $\mathscr{L}_{binary}$ perform actual computations on the input data while $\mathscr{L}_{filter}$ and $\mathscr{L}_{scale}$ only affect the visual representation of the data. Thus, a simple magnifying lens can be obtained by only setting $\mathscr{L}_{scale}$. And a lens showing only one time-series in a multivariate plot can be obtained by only setting $\mathscr{L}_{filter}$ with θ limited to that particular series. The following section focuses on the more complex computational steps **C1** and **C2**, and describes complete analysis pipelines involving multiple lenses.

4.2.2 Analysis pipeline

Users can specify elaborate transformations by combining multiple lenses. Two lenses $\mathscr{L}_1$ and $\mathscr{L}_2$ can be combined either by overlaying them in the chart, in which case the operators defining the lenses get applied sequentially to the input time-series data points (first those of $\mathscr{L}_1$, then those of $\mathscr{L}_2$); or by declaring $\mathscr{L}_1$ as the *parent* lens of $\mathscr{L}_2$, the transformed output of $\mathscr{L}_1$ being an operand of $\mathscr{L}_2$'s binary operator. Figure 3.I gives a schematic representation of these two combination mechanisms. Basic transformations that apply to a single data stream only require a lens with a unary operator (Figure 3-a) as defined in **C1**. Cross-data-stream computations require the definition of a child lens (Figure 3-b), that takes as input a data stream resulting from a parent lens and the data in the child lens' focus. The two streams are processed through the child lens' binary operator (**C2**), with the unary operator (**C1**) pre-processing the data points in the child lens' focus, if set. Figure 3.II illustrates all possible elementary lens combinations:

E1 *Simple transformation* (Figure 3-c): the most basic lens, where $\mathscr{L}_{unary}$ is set and $\mathscr{L}_{binary}$ is *null*. It is conceptually similar to a Magic Lens [8], with the unary operator (noted α) transforming the data in the lens' focus region;

E2 *Series of transformations* (Figure 3-d): two lenses that are piled up (they observe the same timespan) on separate layers as in traditional graphics editors. The first lens processes the input time-series data points that fall into its focus through unary operator α_1. The resulting data is fed to the second lens and gets processed by unary operator α_2. It is possible to pile up an arbitrary number of lenses, as with Fishkin and Stone's Movable Filters [14].

E3 *Cross-data computation* (Figure 3-e): cross-data computations are achieved by creating a parent-child relationship between two lenses. Output data from the parent lens and time-series data points in the child lens' focus region are the operands given to binary operator β. The unary operator of the child lens is set to *null* (identity transform).

E4 *Simple transformation and cross-data computation* (Figure 3-f): both the unary operator α and the binary operator β are set. The lens first applies transformation α to the time-series data points in its focus. The result of this transformation, $\vec{D}_1$, is then fed to the binary operator β along with the output of the parent lens' transformation, $\vec{D}_P$.

More elaborate analysis pipelines can be built by creating hierarchies consisting of the above elementary combinations (see Section 6).

5 ChronoLenses Interface

We developed a proof-of-concept, full-featured interactive visualization tool implementing the ChronoLenses technique for the visual exploration of multivariate time-series. The interface (Figure 1) consists of five main components: (A) the *chart panel* displaying the different time-series in a classical stacked view; (B) a *lens creation toolbar*; (C) a *lens analysis pipeline view* displaying all ancestors of the currently selected lens (D); and (E) a *property panel* showing the latter lens' settings. Following the design requirements listed in Section 3, the interactive visualization was designed to allow rapid exploration of time-series and construction of analytical pipelines. This section describes the interactive visual interface of our prototype. Concrete examples of use are introduced in the following section.

Figure 4-l shows the user interface of a lens $\mathscr{L}$. The region of interest (time span $\vec{D}$ in the lens focus) is visually represented as a blue *focus bar* on the time axis (Figure 4-c). The time-series $\mathscr{L}(\vec{D})$ is displayed inside the lens' frame (Figure 4-b). A toolbar located on the top border of the lens provides quick access to the lens' main parameters (Figure 4-a). This toolbar is visible only when the cursor is hovering over the lens, so as to minimize visual clutter. Detailed information of a lens is also provided in a tooltip when hovering.

The user can modify the visual representation of plots, offering different perspectives on the data. Our prototype supports classical plots such as line charts and dot plots, but also statistical plots including histograms and Q-Q normal plots for comparing data distributions.

5.1 Creating an Analysis Pipeline

Advanced analysis pipelines are typically built by creating and progressively combining multiple lenses corresponding to the desired elementary operations **E1-E4** defined in Section 4.2.2.

To create a basic lens, the user first specifies its parameters through the creation toolbar, for instance choosing a *unary operator* such as *remove means*, and setting the *binary operator* to *null*. She then selects the data to be processed through the lens by initiating a rubber-band selection on any time-series loaded in the charts panel. This selection, achieved thanks to a simple mouse drag (Figure 4-e), defines the time span of interest, that will become the lens' focus.

Lenses that enable cross-data stream analyses obviously accept two input streams[3]. One stream is defined by the lens' focus, set as explained above. The second stream is provided as the output of another lens. This lens is considered as the *parent lens*; it feeds data to the *child lens*, that performs the cross-data stream analysis. The parent lens must exist in order to create a child lens.

The user can derive a child lens from the currently selected lens, by clicking on the *Create Child Lens* button in the creation toolbar or on the equivalent contextual menu item (Figure 1-f), after having set the desired parameters for the *unary operator* (if relevant) and *binary operator* (e.g. *cross-correlation*). The time span of a child lens must match that of the parent lens. This is to guarantee that input streams contain the same number of data points for binary operations, and thus make sure that cross-data stream operations are consistent. However, it would be possible to support unequal time spans between parent and child lenses, using time-series transformations that would make the number of data points between both streams consistent using, e.g., linear interpolation methods for time-series.

The created lenses are z-ordered using layers and can be piled up according to this ordering. Piling up lenses entails piping their operators as described in Section 4.2.2. In the current implementation, two lenses get actually piped when the upper lens' time span is fully contained within that of the lens underneath: the result of the latter then becomes the input of the former, as when composing Movable Filters by overlapping them [14].

To help the user keep awareness of the analysis pipeline, a pipeline view is provided, depicting the currently selected lens' ancestors (Figure 1-C). The tree structure displays dynamic miniatures of the different lens branches, alongside labels detailing the operators associated with each lens in the pipeline. The current lens is visually emphasized with a thick blue border, and always corresponds to the top-left element in the pipeline view. By hovering over a lens either in the pipeline view or in the main chart view, the user can get an overall preview of all its ancestors. All lenses upstream are visually identified using an outer-glowing effect varying from green (closest ancestor) to yellow (furthest ancestor) in both views. The hovered lens is also emphasized with a blue glow effect (Figure 1-D).

5.2 Visual Exploration through Direct Manipulation

Visual exploration, the process of interactively browsing through different regions of a dataset to gain a better understanding of it, is not only useful as a quick and effective technique for hypothesis confirmation. It is also, and perhaps more importantly, essential for discovering the unexpected and raising new questions [38]. Direct manipulation has been successfully applied to time-series interactive visualizations as a means to facilitate visual exploration in conjunction with immediate visual feedback; see, e.g., PatternFinder [10] and KronoMiner [41]. ChronoLenses, by virtue of their progressive analysis pipeline building process, are meant to facilitate step-by-step exploration through direct manipulation, while giving users freedom to edit intermediate steps. Changes to a lens in the hierarchy are automatically propagated to its descendants, and all affected plots are visually updated accordingly.

5.2.1 Modifying Lens Parameters

To facilitate opportunistic discovery of regions of interest, we make it possible to redefine the focus region of a lens by simply dragging the lens frame (Figure 4-g) or the associated focus bar (Figure 4-h) left or right. The immediate redisplay of the newly computed result in both the chart view and the pipeline view enables quick access to different regions of the data stream, and thereby effectively supports visual exploration. The red arrow controllers placed on the focus bar (Figure 4-f) make it possible to adjust the lens' time span. The time span of all lenses downstream is adjusted accordingly so as to keep the number of data points consistent, as explained earlier. The user can also decide to move a lens to a different chart to further her analysis on another dataset, as illustrated in Figure 4-k.

The lens frame's width can be increased using the mouse wheel. The time span considered does not change, which means that resizing affects the $\mathscr{L}_{scale}$ parameter: the lens behaves as a magnifying glass[4]. Resizing on the vertical axis works similarly, except that the lens frame's height never changes and is always equal to the underlying chart's bounding box height. The y-axis origin can be adjusted by dragging the plot up and down inside the frame. An auto-adjust function is also provided, that automatically adjusts the y-axis' origin and scale so as to make the best use of available screen real-estate within the lens' frame as the user drags it.

[3] As explained in Section 4.2.1, these streams can contain multivariate time-series. The binary nature of the transformation does not mean that input time-series are restricted to two variables. See Section 5.3 for more information about handling multivariate time-series.

[4] The above-mentioned blue bar still shows the original region of interest.

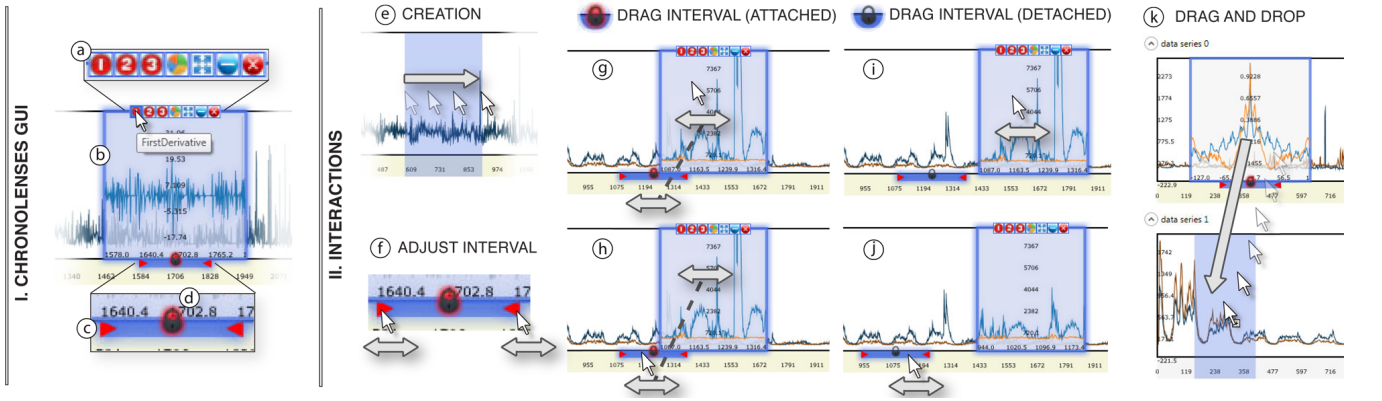

Fig. 4. (I) UI of a ChronoLens: (a) toolbar enabling adjustment of the lens' main parameters, (b) tooltip (pops-up when hovering), (c) focus bar with resizing handles, (d) lock icon for detaching/reattaching the lens' frame from/to the focus bar; (II) associated interactions.

5.2.2 Groups of Lenses

As mentioned earlier, the z-ordering of lenses that belong to the same analysis pipeline implicitly defines in what order transformation operations are applied to the data. The z-ordering and relative position of lenses that make a given analysis pipeline is important and should be kept constant during visual exploration via direct manipulation (when brushing the time-series plots with the lenses). Having to move each lens would be extremely cumbersome and would greatly impede the interactive visual exploration process. To avoid this, lenses can be grouped. All lenses that belong to the same group move synchronously as the user drags any one of them. The user can therefore create a compound lens set in which lenses are piled up in a specific order, thus creating a reusable analysis pipeline.

Grouping lenses can also be useful when analyzing multiple charts at a time, or when looking for seasonality. Such tasks require the interval between lenses (time lag) in the data stream to remain constant. By defining a group consisting of the different lenses to be moved synchronously, the user can keep the time lag between these constant.

A lens can be added to the currently selected group through the contextual menu (Figure 1-F). All lenses belonging to the same group as the currently selected one are outlined with a green stroke, giving feedback about the time-lag constraints that exist between those lenses. Group membership is also reflected in the tree view, where the group number is explicitly specified alongside the lens miniatures.

5.2.3 Dealing with Visual Clutter

The instantiation of many lenses can quickly lead to visual clutter. Lenses that are part of the same analysis pipeline will often overlay each other, either partially or fully occluding one another. This can be a problem when the user wants to visualize intermediate computational steps of the pipeline. Occlusion problems can also arise when magnification lenses (any lens with $\mathscr{L}_{scale} > 1$) look at time spans that are disjoint but close to one another.

To address these issues, we introduce a mechanism that enables the user to detach the lens frame from its associated focus bar, thereby giving her control over where to display the lens' output, independently of the actual input region of interest (focus bar location). Figures 4-g to 4-j illustrate this mechanism. To detach a lens from its focus bar, the user simply clicks on the lock icon (Figure 4-d) to unlock the lens from the focus bar. In that mode, dragging the lens frame does not affect the focus interval (Figure 4-i), making it possible to space out lenses or, on the contrary, move them closer to facilitate comparison. In the same manner, dragging the focus bar only affects the input time span, the lens frame remaining in place (Figure 4-j). Wherever a lens and its focus bar are positioned, locking them again will keep the position offset constant (Figure 4-g-h). When hovering over a lens frame or a focus bar, both are emphasized with a glow effect making it easier to identify what focus bar is associated with what lens, and conversely.

Other mechanisms that help manage visual clutter include *delete* and *minimize* buttons in the toolbar (Figure 4-a). The former deletes the lens, the latter hides the lens' frame, but not its focus bar. The bar remains visible so as to keep the user aware of its existence. At all times, double-clicking it makes the lens' frame visible again.

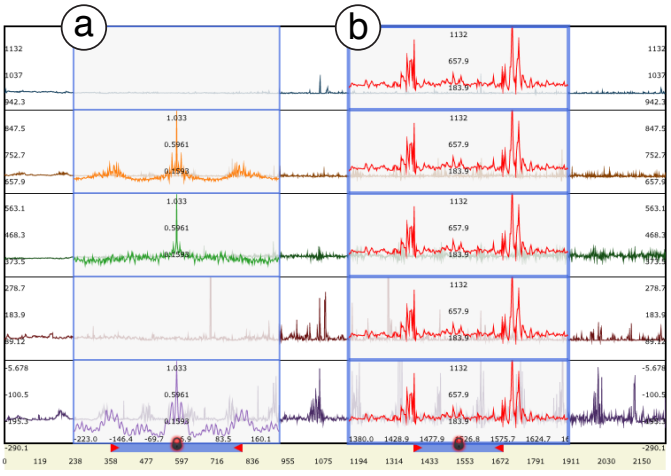

Fig. 5. Lenses applied to multivariate data: (a) auto-correlation for three series out of five; (b) point-wise aggregated maximum showing the resulting plot (common to all series) overlaid on top of each variable's plot.

5.2.4 Multi-foci Exploration

As many other types of datasets, time-series are amenable to multi-scale navigation. For a given series, the user might be interested in behaviors and event patterns that occur at various time scales, from months or years down to fractions of a second.

Magnification lenses support the process of interactively drilling down into the data, and enable the comparison of different time spans (a common practice when analyzing time-series data [13, 21, 41]) simply by instantiating multiple lenses. Magnification lenses can easily be instantiated in ChronoLenses: any lens with $\mathscr{L}_{scale} > 1$ is a magnification lens that enables drilling down into the data. However, magnification lenses pose some problems from an interaction perspective [5, 33]. In-place magnification of the focus region means that the immediate surroundings of the region of interest will be occluded if the lens is simply overlaid on the original data, hiding potentially valuable information and hindering navigation. Smooth transition between the magnified focus and surrounding context can be achieved using spatial distortion [26], but the introduced deformation has a cost in terms of legibility and interpretation of the visualization.

An alternative drill-down method called Stack Zooming was recently proposed by Javed *et al.* [21]. The technique consists in stacking multiple views as the user drills down into the data. The con-

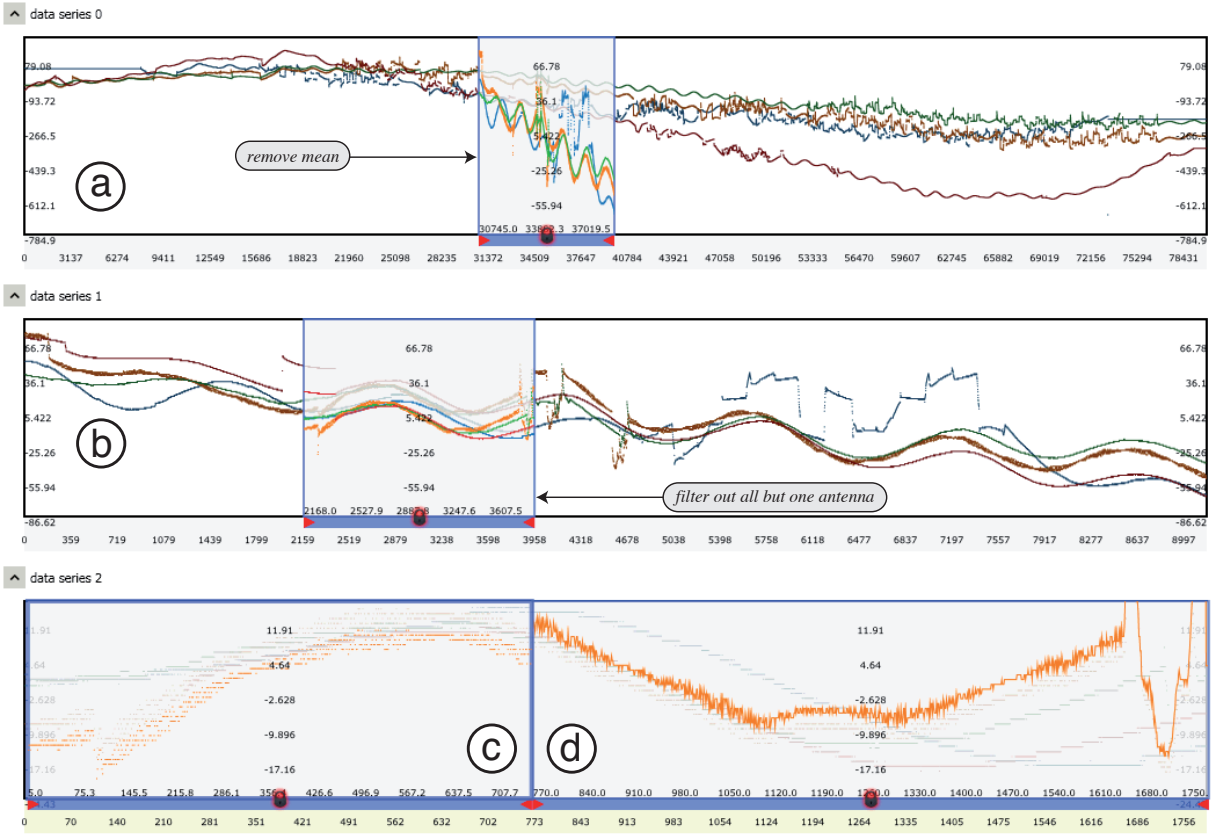

Fig. 6. Exploring ALMA Line Length Correction Stretcher Voltage plots for four antennas: (a) two-day overview at a sampling rate of one second; (b) magnification of the 5 hours seen through the *remove mean* lens applied in (a); magnification of the 50 minutes for a single antenna seen through a filtering lens, rendered in scatterplot mode (c) and line-plot mode (d).

tent of a given view is a magnification of a region in the view immediately above, delimited by a lens-like viewfinder similar to a (one-dimensional) DragMag [39]. The technique can seamlessly be integrated with ChronoLenses and is implemented as follows.

The user creates a new empty chart panel in the main view. Then, dragging-and-dropping an existing lens into this empty panel duplicates the data within the span of the lens and displays it in this new panel. The visual representation is stretched horizontally to fill the panel, thus providing a magnified view of the lens' content that is dynamically updated whenever that lens moves. Figure 6 illustrates this technique, with chart (b) providing a magnified version of the region seen through the lens in chart (a). Lenses can be instantiated on this new panel, enabling users to drill-down recursively and build a truly multi-scale view hierarchy in which lenses at any level can be freely adjusted. Lenses that form this hierarchy are of course not limited to simple magnification and can apply to the data any of the elaborate transformations enabled by the ChronoLenses operators.

5.3 Multivariate Time-Series

In ChronoLenses, multivariate datasets can be visualized either overplotted in the same chart (Figure 1, top chart) or separately, stacked on top of one another (Figure 1, bottom chart). When applying a lens to a multivariate data stream, each series gets processed independently, except for some operators that aggregate the data, such as *point-wise minimum* or *maximum*. In that case, the output is a univariate stream, that can be duplicated and overlaid in all charts. We distinguish aggregated operators from other operators by color-coding in red all the replicas of the unique plot resulting from data aggregation (Figure 5-b) as opposed to the one-to-one color mapping used when the different variables are processed separately (Figure 5-a).

To focus on a subset of variables, the user can filter out series that she does not want to be considered in the computational pipeline by setting the filter operator $\mathscr{L}_{filter}$ accordingly. To do so, she can either select all the series to be taken into account in the corresponding menu accessible from the lens toolbar, or specify the list in the property panel (Figure 1-E), by entering the corresponding textual expression using a very simple syntax. When filtered out, streams are neither computed nor rendered, as the first and fourth streams in Figure 5-a. Note that when a lens defines an aggregated operator, filtering out a stream has an impact on the result, as the stream is no longer considered as an operand. Again, we offer the user full control over what data gets processed, making the analysis process highly flexible and customizable.

6 Use Case Scenarios

In this section, we illustrate how ChronoLenses can be used with two use cases involving real datasets.

6.1 ALMA Observatory Usage Scenario

The Atacama Large Millimeter/submillimeter Array (ALMA, [3, 34]) is a single telescope (under construction in the Chilean Andes) that will eventually be composed of 66 high-precision antennas. Observations are based on the principle of interferometry: a source in the sky is observed by at least two antennas; the signals (radio waves) captured by each antenna are then combined by a central computer called the correlator to form images suitable for performing scientific analysis. Time-series visualization are used by both operators and astronomers for a variety of tasks, ranging from checking some of the thousands of monitor points in the system to performing scientific data quality assurance during observations. In the following, we focus on one example where ChronoLenses can help users in their daily task.

When combining the signals coming from the different antennas taking part in a given observation, the correlator must know the length of the path traveled by the signal through the fiber optics cables with an accuracy of hundredths of a millimeter. A round-trip laser signal gets sent to all antennas in order to continuously monitor the length of the optical fibers, as the latter can expand and contract due to temperature

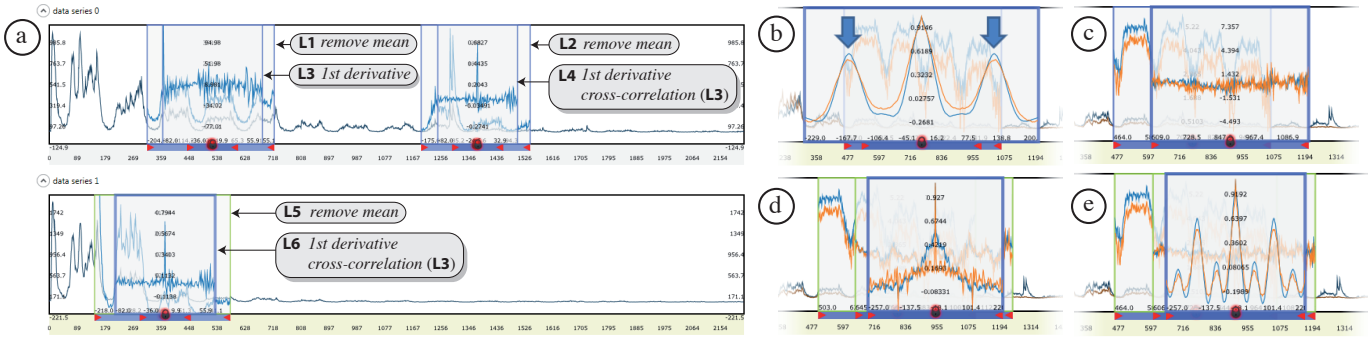

Fig. 7. Analysis of a month of the visiting population of two P2P video-on-demand channels: a) analysis of the evolution of the correlation within and between the channels (see Section 6.2.1); b) looking for seasonal pattern lag value, the blue arrows show a strong correlation at a delay of 1440s (=24h) and c-d) comparing the delayed time spans (see Section 6.2.2).

variations. These changes in path length must be compensated in real-time. This is achieved by the Line Length Correction (LLC) system, which ensures that path lengths are all stabilized to about 1 micron. Operators are interested in monitoring the LLC stretcher voltage for each antenna, and observing potential deviation of an antenna's LLC stretcher voltage compared to that of the other antennas.

From a time-series visualization perspective, different things can be happening at different time scales: from several days to a few minutes. Efficient multi-scale and multi-focus visualization, as enabled by the $\mathscr{L}_{scale}$ operator and the variation on stack zooming [21] implemented in ChronoLenses, is thus an essential feature. Figure 6-a plots voltage against time for 2 days at a sampling rate of 1 second.

Starting from this 2-day overview, the operator is first interested in finding out whether all antennas are behaving normally or if one or more are somehow deviating. All antennas are expected to behave more or less similarly. Direct visual comparison between the plots, overlaid or stacked, is sufficient to see the deviation of one antenna during the second day (Figure 6-a).

To better see the smaller and shorter fluctuations, the operator creates a lens spanning a 5-hour period, with $\mathscr{L}_{unary}$ operator *remove mean* (Figure 6-a). The lens is dragged and dropped to create panel 6-b. The content of that panel is a stretched version of what is seen through 6-a's lens. The operator can clearly see that the fluctuations are in phase, which implies that they all come from a single source. She could then plot various monitoring points simultaneously based on system or environmental components likely to cause these fluctuations and create lenses defining a cross-correlation $\mathscr{L}_{binary}$ operator (not shown here), eventually tracing the source to temperature fluctuations in the local oscillator room.

Finally, zooming in further to display just 50 minutes (Figure 6-c), the operator can see some odd features in the signal for one of the antennas. She filters out antennas that behave normally using a lens defining the appropriate $\mathscr{L}_{filter}$. She also switches from a scatterplot to a line-plot rendering inside the lens focus (Figure 6-d), revealing fast oscillations at a much lower scale that occur on top of the slow ones observed earlier for this particular antenna. This antenna-specific issue eventually gets traced to a device repeatedly turning on and off in the antenna.

6.2 Peer-to-peer Network Fluctuation Analysis

We consider a network system manager who is trying to optimize bandwidth usage on a centralized peer-to-peer system. She uses line graphs of the visiting population of two different video-on-demand channels. The relative evolution through time of the two channels, temporal distances, and the activity load before and after peak events are important clues that the analyst can rely upon, assisting her in the decision making process. For instance, identifying regular activity patterns helps better predict future fluctuations and therefore better pinpoint the needs for server pool optimization. Being able to spot anomalies such as overloads in their context also helps identify the potential causes, and plan for technical solutions.

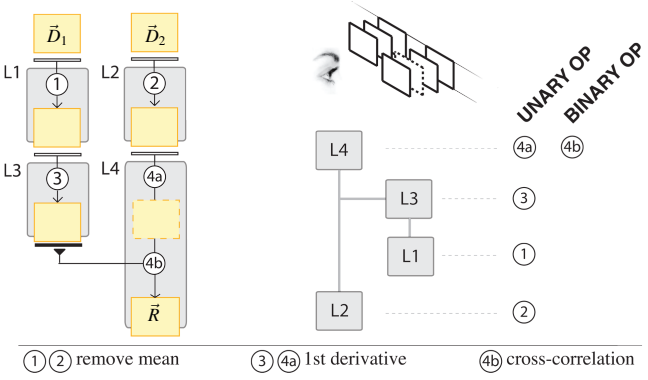

Fig. 8. Comparing the evolution of two time-series using ChronoLenses. Left: the analysis pipeline, right: the corresponding lens hierarchy.

6.2.1 Analyzing the Evolution of Correlations

Our dataset consists of the visiting population of two different channels over a month, sampled from the server every 10 minutes (Figure 7). The overall yearly or monthly trends of the channels might be different. For instance, one might be rising much faster than the other, or one might fall while the other is rising a bit. In this first scenario, the user is rather interested in the smaller scale fluctuations, comparing evolution and identifying potential correlations at this scale.

The user first focuses on a single channel, looking for repeating patterns. To do so, she compares differentiated data at different times by performing the following steps: 1) remove mean, to make the data more stationary and thus more amenable to comparison for our purposes; 2) compute the first derivative (or differencing) of those; and 3) perform the cross-correlation between the two for comparison. The pipeline that supports this type of analysis, described below, is composed of four lenses, as shown in Figure 8.

The user creates lenses L1 and L2, with the $\mathscr{L}_{unary}$ operator set to *remove mean* (step 1). She then creates L3 ($\mathscr{L}_{unary}$ operator set to *1st derivative*) and places it on top of L1 (step 2). Its child lens L4 gets positioned on top of L2 with operators $\mathscr{L}_{unary}$ and $\mathscr{L}_{binary}$ set to *1st derivative and cross-correlation* respectively (steps 2-3). In the end, L4 outputs the correlation of the data below it (1st derivative of L2) and its parent's output (L3). The discovery of recurring patterns usually requires performing computations on data with parameterized-yet-fixed lags in time. Here, we use lens groups – that make all member lenses move synchronously – to build dynamically-parameterized operators enabling efficient exploration of the effect of different time lags and time spans via direct manipulation.

Grouping L1 and L3 on the one hand (L1-3), and L2 and L4 on the other hand (L2-4) makes it possible to drag one or the other focused intervals along time and look for strong correlation by observing the

result of L4. When such a correlation is identified, as in Figure 7-a, the user can group L1-3 and L2-4 altogether and drag them. If L4 exhibits a stable plot while dragging, then a recurrent trend is identified. If on the contrary the plot varies, direct manipulation with immediate feedback helps better understand the correlation variation before and after peaks of interest, whereas it might be more difficult to apprehend when computing automatic pattern detection.

When the analyst has located a pattern of interest, she can focus on the second channel to look for correlations. She creates pipeline L5-6 similar to group L2-4 using the data from the second channel as input: L5 (remove mean) and L6 (L3's child placed on top of L5 with 1st derivative and cross-correlation operators), as depicted in Figure 7-a.

Different analysis tasks can easily be performed with the current set of lenses: (1) drag the group L5-6 until the plot is like, or the opposite of, L2-4 to identify correlations between the two series (see Figure 7-a); (2) group L2-4 and L5-6 so that they preserve their relative distance while dragging one or the other in order to compare the correlation fluctuations by simultaneously looking at the results of lenses L4 and L6. If those behave the same, then a similar evolution is identified and the relative lag is known. If they always behave the opposite, there is an inverse causality; and (3) by extending the group to L1-3 and dragging the whole, the analyst can also explore if a pattern is preserved in time. The user can also change the lenses' operators at any time during exploration, as for example changing L3, L4 and L6's $\mathscr{L}_{unary}$ operator to *2nd derivative* and looking for insights when more lag is involved (20 minutes in our example).

6.2.2 Comparing different Time Spans in the same Series

Discovering seasonality is another important task in time-series data analysis. It helps better predict future fluctuations. In this second part of the scenario, our user is interested in finding seasonal differencing on the two channels simultaneously, in order to gain a better understanding of what characterizes the typical data streaming cycle, if such a cycle exists. Gaining this knowledge will eventually help develop strategies to optimize bandwidth usage, but also assist in spotting abnormal behavior when it occurs.

Box-Cox Transforms [9] can be used to perform a variance stabilizing transformation. Seasonal differencing parameterized with a time lag can then be applied to find out if the non-stationarity of the original time-series is removed [30], but this requires to know the lag value in advance. The identification of an appropriate lag value to perform seasonal differencing can be made easier by building the following pipeline: an *auto-correlation* lens (L1) on top of a *Box-Cox transform* lens (L2). A strong correlation is observed at a delay of approximately 1440 minutes = 24 hours (see Figure 7-b).

The analyst deletes L2 and furthers her exploration as she creates L2's child lens L3 with *Box-Cox transform* and *subtraction*, synchronized so that L3 is always 24 hours ahead of its parent (Figure 7-c). An *auto-correlation* lens is then applied on top (after the differencing operation) to evaluate stability of the series. At this position in the original time-series (Figure 7-d), there does not seem to be any significant seasonal pattern, as high correlation values exist only at zero-delay. But moving the lens group along the timeline, one pattern is eventually discovered (Figure 7-e), calling for further exploration.

7 Discussion

ChronoLenses aim at facilitating the creation of customized analysis pipelines for easy exploration and navigation through time-series datasets. Our initial prototype, although already offering a large set of functionalities, still has limitations. It could be improved in several ways, as discussed in this section.

7.1 Layering and Treeview Limitations

In its current implementation, the ChronoLenses interface does not support effective layer management for the z-ordering of lenses that pile up. Although the grouping of lenses keeps the z-ordering constant, when not grouped, selecting a lens puts it on the topmost layer. This might be annoying when dealing with complex pipelines as clicking on a lens only to get information about it or even by mistake can alter the pipeline. Making layer management more stable and more easily configurable through a dedicated synchronized view would reduce the need for advanced planning when building the pipeline and would aid keep an accurate mental map of the data flow.

Similarly, the lens hierarchy treeview could be enhanced. As is, the hierarchical view dynamically changes as the user selects a different lens. This might be distracting and difficult to interpret. Moreover it is not interactive. There is an opportunity for improvement here, as an additional and complementary representation of the analysis pipeline could be used as an alternative means of performing exploration tasks if enriched with interactive capabilities. In particular, allowing for the creation of lenses, duplication of analysis branches, or other changes to the analysis pipeline from this alternate view would make it easier to build up and manage complex pipelines.

7.2 Integration of SigmaLens Focus+Context Techniques

Another limitation of the current implementation is related to the positioning of lenses. When displaying an overview of large time-series, ChronoLenses only draws a subset of all points (subsampling). Magnifying the data through a lens enables the display of more detail in context. However, as the lens' magnification factor increases, small movements of the lens translate to larger jumps in terms of time span observed in the lens' focus. For instance, given a magnification factor of 10x, moving the lens by 10 pixels will move the time span observed through the lens by 100 equivalent pixels at that zoom factor (see [5] for a detailed description of this problem of quantization). The problem also exists when controlling the time span seen in a plot through a lens observing a lower scale version of that plot à la Stack zooming. Adapting one of the high-precision magnification lens techniques introduced in [5] would solve this issue, enabling both fast repositioning and precise selection within the lens focus.

A related issue is that our lenses occlude the area immediately surrounding the region of interest, as basic magnifying lenses do. SignalLenses [26] address this issue by distorting the representation in a bounded transition region between the lens focus and the context. We chose not to rely on spatial distortion because of the issues it raises in terms of making accurate analytical comparison and interpretation. However, there are other solutions to this problem, such as speed-coupled Sigma Lens focus targeting techniques [33]. For instance, the Speed-coupled Blending Lens technique consists in coupling the lens focus' opacity to its speed, smoothly fading out the lens as speed increases and smoothly fading it back in when it comes to a stop, having reached its target position. The technique addresses the problems of visual occlusion and would be relatively straightforward to implement in ChronoLenses as we already support translucency in lens renderings. However, potential issues related to visual interference between the layers during lens repositioning would have to be investigated.

8 Conclusions and Future Work

We have presented ChronoLenses, a novel domain-independent visualization technique for the visual exploration of time-series data. ChronoLenses relies on the metaphor of lenses, that compute on-the-fly data transformations in place. ChronoLenses allow users to progressively build elaborate analysis pipelines by interactively compounding elementary operations, thus supporting complex user-defined exploratory analysis tasks.

The concept of ChronoLenses can be extended to other types of data than time-series, e.g., image analysis and image processing techniques for exploring temporal media such as video, where content-aware operators (feature detection algorithms, sharpening and color correction filters) would complement more generic operators (magnification, track filtering).

In addition to exploring new applications of the framework, we plan to improve our implementation by addressing the limitations discussed in the previous section, and by exploiting analytical mechanisms to guide exploration. For instance, lenses could be made to snap to local optima in the visualized time-series as the user drags them, using measures such as, e.g., the strongest local cross-correlation.

ACKNOWLEDGMENTS

We wish to thank all expert users that participated in our interviews and gave us feedback about the ChronoLenses tool, including: Denis Barkats from the Joint ALMA Office, National Radio Astronomy Observatory; Lindsey Davis from the National Radio Astronomy Observatory, Norbert Driedger and Brian Greaves from Environment Canada; Yashar Ganjali, from the Computer Systems and Networks Group at University of Toronto; Igor Jurisica from the Ontario Cancer Institute; Di Niu, from the Electrical and Computer Engineering department at University of Toronto; and Joseph Schwarz from the European Southern Observatory. We also thank Theophanis Tsandilas for helpful comments about early drafts of this paper.

REFERENCES

[1] W. Aigner, S. Miksch, W. Muller, H. Schumann, and C. Tominski. Visualizing Time-oriented Data - a Systematic View. *Computers & Graphics*, 31(3):401–409, June 2007.

[2] W. Aigner, S. Miksch, W. Müller, H. Schumann, and C. Tominski. Visual Methods for Analyzing Time-Oriented Data. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):47–60, 2008.

[3] ALMA. Atacama large millimeter/submillimeter array, Last accessed: June 2011. `http://www.almaobservatory.org`.

[4] N. Andrienko and G. Andrienko. *Exploratory Analysis of Spatial and Temporal Data*. Springer, 2006.

[5] C. Appert, O. Chapuis, and E. Pietriga. High-precision Magnification Lenses. In *Proceedings of the 28th international conference on Human factors in computing systems (CHI)*, pages 273–282. ACM, 2010.

[6] R. Bade, S. Schlechtweg, and S. Miksch. Connecting Time-oriented Data and Information to a Coherent Interactive Visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI)*, pages 105–112. ACM, 2004.

[7] E. Bertini, P. Hertzog, and D. Lalanne. SpiralView: Towards Security Policies Assessment through Visual Correlation of Network Resources with Evolution of Alarms. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 139–146, 2007.

[8] E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. D. DeRose. Toolglass and Magic Lenses: the See-through Interface. In *Proceedings of the Conference on Computer graphics and interactive techniques (SIGGRAPH)*, pages 73–80. ACM, 1993.

[9] G. E. P. Box and D. R. Cox. An Analysis of Transformations. *Journal of Royal Statistical Society*, B26:211–246, 1964.

[10] P. Buono, A. Aris, C. Plaisant, A. Khella, and B. Shneiderman. Interactive Pattern Search in Time Series. In *Proceedings of Visualization and Data Analysis*, pages 175–186, 2005.

[11] M. S. T. Carpendale and C. Montagnese. A Framework for Unifying Presentation Space. In *Proceedings of the 14th annual ACM symposium on User interface software and technology (UIST)*, pages 61–70. ACM, 2001.

[12] A. Cockburn, A. Karlson, and B. B. Bederson. A Review of Overview+detail, Zooming, and Focus+context Interfaces. *ACM Computing Surveys (CSUR)*, 41:1–31, 2009.

[13] N. Elmqvist, P. Dragicevic, and J. Fekete. Color Lens: Adaptive Color Scale Optimization for Visual Exploration. *IEEE Transactions on Visualization and Computer Graphics*, 2010. In press, Rapid post.

[14] K. Fishkin and M. C. Stone. Enhanced dynamic queries via movable filters. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI)*, pages 415–420. ACM/Addison-Wesley Publishing Co., 1995.

[15] G. W. Furnas. Generalized Fisheye Views. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI)*, pages 16–23. ACM, 1986.

[16] M. C. Hao, U. Dayal, D. A. Keim, and T. Schreck. Importance-Driven Visualization Layouts for Large Time Series Data. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)*, pages 203–210. IEEE Computer Society, 2005.

[17] M. C. Hao, U. Dayal, D. A. Keim, and T. Schreck. Multi-Resolution Techniques for Visual Exploration of Large Time-Series Data. In *EuroVis*, pages 27–34, 2007.

[18] J. Heer, N. Kong, and M. Agrawala. Sizing the Horizon: the Effects of Chart Size and Layering on the Graphical Perception of Time Series Visualizations. In *Proceedings of the 27th international conference on Human factors in computing systems (CHI)*, pages 1303–1312. ACM, 2009.

[19] H. Hochheiser and B. Shneiderman. Dynamic Query Tools for Time Series Data sets: Timebox Widgets for Interactive Exploration. *Information Visualization*, 3(1):1–18, 2004.

[20] J. J. Van Wijk. Cluster and Calendar based Visualization of Time Series Data. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)*, pages 4–9, 1999.

[21] W. Javed and N. Elmqvist. Stack Zooming for Multi-focus Interaction in Time-series Data Visualization. In *IEEE Pacific Visualization Symposium (PacificVis)*, pages 33–40, 2010.

[22] W. Javed, B. McDonnel, and N. Elmqvist. Graphical Perception of Multiple Time Series. *IEEE Transactions on Visualization and Computer Graphics*, 16:927–934, 2010.

[23] D. Keim. Information Visualization and Visual Data Mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8, January 2002.

[24] D. Keim, F. Mansmann, J. Schneidewind, and H. Ziegler. Challenges in Visual Data Analysis. In *Proceedings of the 10th International Conference on Information Visualization (IV)*, pages 9 –16. IEEE, 2006.

[25] R. Kincaid. Line Graph Explorer: Scalable Display of Line Graphs using Focus+context. In *In Working Conference on Advanced Visual interfaces (AVI)*, pages 404–411. ACM, 2006.

[26] R. Kincaid. SignalLens: Focus+Context Applied to Electronic Time Series. *IEEE Transactions on Visualization and Computer Graphics*, 16:900–907, 2010.

[27] H. Lam, T. Munzner, and R. Kincaid. Overview Use in Multiple Visual Information Resolution Interfaces. *IEEE Transactions on Visualization and Computer Graphics*, 13:1278–1285, 2007.

[28] J. Lin, E. Keogh, S. Lonardi, J. P. Lankford, and D. M. Nystrom. Visually mining and monitoring massive time series. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 460–469. ACM, 2004.

[29] R. Lopez-Hernandez, D. Guilmaine, M. McGuffin, and L. Barford. A Layer-oriented Interface for Visualizing Time-series Data from Oscilloscopes. In *IEEE Pacific Visualization Symposium (PacificVis)*, pages 41–48, 2010.

[30] H. Madsen. *Time Series Analysis*. Chapman & Hall/CRC, Nov. 2007.

[31] P. McLachlan, T. Munzner, E. Koutsofios, and S. North. LiveRAC: Interactive Visual Exploration of System Management Time-series Data. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems (CHI)*, pages 1483–1492. ACM, 2008.

[32] W. Müller and H. Schumann. Visualization methods for time-dependant data-an overview. In *Proceedings of the 2003 Winter Simulation Conference*, pages 737–745, 2003.

[33] E. Pietriga, O. Bau, and C. Appert. Representation-Independent In-Place Magnification with Sigma Lenses. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):455–467, 2010.

[34] J. Schwarz, E. Pietriga, M. Schilling, and P. Grosbol. Goodbye to WIMPs: A Scalable Interface for ALMA Operations. In I. N. Evans, A. Accomazzi, D. J. Mink, and A. H. Rots, editors, *Astronomical Data Analysis Software and Systems (ADASS)*, ASP Conference Series. ASP, 2010.

[35] B. Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. *IEEE Symposium on Visual Languages (VL)*, pages 336–343, 1996.

[36] S. F. Silva and T. Catarci. Visualization of Linear Time-Oriented Data: A Survey. In *Proceedings of the International Conference on Web Information Systems Engineering (WISE)*, pages 310–319. IEEE, 2000.

[37] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, May 2001. 197 pages.

[38] J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[39] C. Ware and M. Lewis. The DragMag image magnifier. In *Conference companion on Human factors in computing systems (CHI)*, pages 407–408. ACM, 1995.

[40] M. Weber, M. Alexa, and W. Müller. Visualizing Time-Series on Spirals. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)*, pages 7–13, 2001.

[41] J. Zhao, F. Chevalier, and R. Balakrishnan. KronoMiner: Using Multi-Foci Navigation for the Visual Exploration of Time-Series Data. In *Proceedings of the 29th international conference on Human factors in computing systems (CHI)*, pages 1737–1746. ACM, 2011.