StreamSqueeze: A Dynamic Stream Visualization for Monitoring of Event Data

Florian Mansmann, Milos Krstajic, Fabian Fischer and Enrico Bertini

University of Konstanz, Germany

ABSTRACT

While in clear-cut situations automated analytical solution for data streams are already in place, only few visual approaches have been proposed in the literature for exploratory analysis tasks on dynamic information. However, due to the competitive or security-related advantages that real-time information gives in domains such as finance, business or networking, we are convinced that there is a need for exploratory visualization tools for data streams. Under the conditions that new events have higher relevance and that smooth transitions enable traceability of items, we propose a novel dynamic stream visualization called StreamSqueeze. In this technique the degree of interest of recent items is expressed through an increase in size and thus recent events can be shown with more details. The technique has two main benefits: First, the layout algorithm arranges items in several lists of various sizes and optimizes the positions within each list so that the transition of an item from one list to the other triggers least visual changes. Second, the animation scheme ensures that for 50 percent of the time an item has a static screen position where reading is most effective and then continuously shrinks and moves to the its next static position in the subsequent list. To demonstrate the capability of our technique, we apply it to large and high-frequency news and syslog streams and show how it maintains optimal stability of the layout under the conditions given above.

Keywords: Dynamic Visualization, Stream Analysis, Animated Visualization

1. INTRODUCTION

Today, many application fields deal with dynamic information streams for which a real-time analysis becomes a critical success factor. While purely analytical solutions are possible for well-defined problems, such as an alert in a security system or an automatic sale in a stock trading application, explorative analysis tasks, such as following hot topics in a news stream, could better be solved with visualization tools in which the user stays in the analysis loop.

Despite the fact that the problem of visualizing dynamic information streams has been identified as a major challenge for some years (e.g. in[1]), only little research has been conducted on the topic in the visualization community as we show in the related work section of this paper. In our opinion, this is mainly due to the two facts that 1) information streams are hard to deal with from a data management perspective and 2) dynamic information representation is a very challenging research issue from the visualization perspective. The first aspect is particularly challenging for visualization researchers, since the technical aspects of data streams involve recent issues, e.g. integrating dynamic data sources, novel database technology, still emerging querying languages, and high costs of the overall technical infrastructure.[2] While classical animation is very convincing for presentation purposes, it fails for analytics tasks in which several items need to be followed at once.[3] To that end, we believe that in order to solve this difficult problem we need to move to hybrid approaches using static item positions for better readability of text labels and smooth transitions of items to facilitate their traceability when making space for new and updated information.

In this paper we present a novel screen-filling visualization technique for analyzing dynamic information streams in or close to real-time. Our visualization concept builds upon two basic conditions:

- *C1*: New events in the data stream have higher relevance due to their more recent nature.
- *C2*: Smooth transitions enable traceability of items on the screen.

E-mails: Florian.Mansmann@uni-konstanz.de, Milos.Krstajic@uni-konstanz.de, Fabian.Fischer@uni-konstanz.de, Enrico.Bertini@uni-konstanz.de

By taking the first condition into account, the proposed layout algorithm arranges items in several lists of various size on the screen and optimizes the positions within each list so that the transition of an item from one list to the other triggers least visual changes. As a consequence of the second condition, our animation scheme ensures that during a screen update 50 percent of items in each list stay at static positions where reading is most effective while the remaining items continuously shrink and move to the adjacent screen position in the subsequent list.

In this paper, we show that our approach has several favorable properties: 1) recent items appear larger and can thus be shown with more details, 2) scalability due to the screen-filling nature of the approach and 3) local changes and continuous animation allow tracking of items. Besides showing these properties, we demonstrate the usage of our technique on large real-world streams in the domains of news and system log monitoring.

2. RELATED WORK

Although challenges in visualization of streaming data have been around for several years,[4] the community is still lacking systematic research in this area. The data streaming domain proves to be a difficult one, since the decisions that are made in visual analytics and information visualization design workflow are application- and data-dependent, more importantly, the individual data items arrive continuously in multiple, rapid, time-varying, possibly unpredictable and unbounded streams.[5] These dynamic, transient, and high-volume properties put strong constraints for creating effective visualizations.[6]

In the next few paragraphs, we briefly discuss related work in the fields of network security and text stream analysis since these closely link to the application section of our paper. In addition to that we also consider relevant information visualization research for time series analysis.

In[7] Best et al. described a system for real-time visual analytics of network activity. The visual analysis component of the system is based on LiveRac,[8] a tool for visual exploration of system management time-series data. Monitoring alerts on the computer network in a radial visualization layout is described in SpiralView.[9] MieLog[10] is another tool to explore system logs and to enhance the interpretation of such textual messages by using visualization.

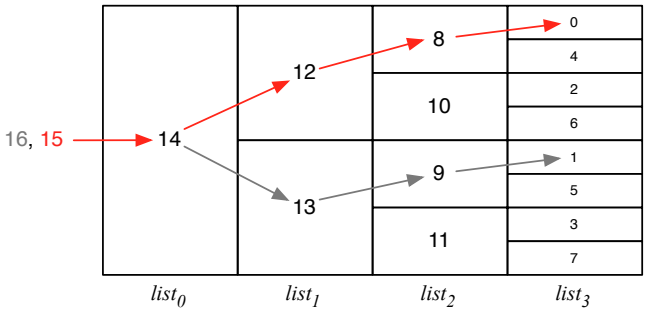
Online text streams represent a very challenging domain, because the unstructured nature of data requires expertise in different areas of research. In the domain of text streams and document collection visualization, the research has been mainly conducted with historic datasets, although some approaches that use online algorithms and incremental visualizations exist. However, the term *text streams* is used in publications in different contexts, i.e. to describe sequences of data items that are temporally ordered, or data streams that are processed online.

A well known approach for term trend visualization in a collection of documents is ThemeRiver.[11] The drawback of this technique is that the layout has to be recalculated every time new data is added in order to have an *optimal* visualization. Legibility of the visualization significantly decreases when the rate of change in each stream is high. The visual analytics system used for analysis of document collections TIARA[12] extends the stacked graph visualization metaphor used in ThemeRiver to show development of topics in collections of emails and patient records, which are represented by a sets of keywords distributed on a time axis. A visualization technique called Thread Arcs, used to gain better insight into threads from a collection of emails, is described in.[13] Newsmap* uses the treemap technique to represent current aggregated headlines retrieved from Google News†. This space-filling approach efficiently uses all available display space to display hierarchical news data, where the number of articles belonging to the same news cluster is mapped to the size of the rectangle, and the color is mapped to the news category. However, the technique is not incrementally updatable by default, the layout has to be recalculated every time new data comes in, and the user has no information about the temporal development of the aggregated news, which is an important aspect in tasks like breaking news monitoring.

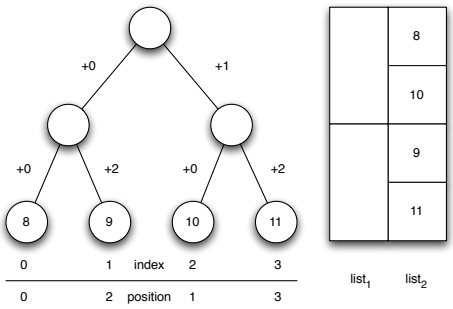
Narratives[14] uses line charts to visualize change of important terms over time, in the context of co-occurring terms in weblog archives. In,[15] the authors presented a visual analytics system for analysis of temporal development of news events and event groups retrieved from broadcast news data. The main visualization component uses bubble-like representation, which is well suited for news events that have rapidly changing dynamics. In[16] the authors present an algorithm for real-time news aggregation into article threads with a dynamic visualization technique, where *important* article threads are shown aligned on a time axis to give insight into their temporal development. The importance measure takes into account different parameters, such as recency, duration of a thread, and its size. The approach is used in a data-streaming scenario,

*http://marumushi.com/projects/newsmap

†http://news.google.com/



(a) Update schema of StreamSqueeze: When a new item is added to $list_0$, the oldest item with smallest index of this list is moved to the next list. Each of the moved items make space for the one from the previous list.



(b) Item positioning scheme shown for $list_2$ with a binary tree: Given the alternating updates of items in $list_1$, the changes in $list_2$ should occur in the same row under the condition that after four updates every item in $list_2$ has been updated.

Figure 1. Update and positioning scheme of the *StreamSqueeze* visualization.

but does not solve the problem of overlapping of large number of multiple time series. Hetzler et al.[17] visualize the incremental change in collection of documents which are projected in 2D plane by highlighting new (fresh) and old (stale) documents.

Time series visualization has been an extensively researched area in the visualization community. An overview of visual methods can be found in.[18] Other techniques include pixel visualization for time series in circle segments,[19] time series bitmaps used for working with large time-series databases,[20] multi-resolution techniques for large time series,[21] and importance-driven layouts for time series.[22] In,[3] the authors compare different techniques for visualization of multiple time series. They perform a user study in which they compare line graphs, small multiples, horizon graphs and their own approach using space management, space per series, identity, baseline and visual clutter as evaluation criteria. In,[23] Kincaid applied lens distortion to employ a focus-plus-context technique on very long electronic time series. Furthermore, the work of Xie et al.[24] focused on condensing time windows with small or no changes and thus to only visualize significant changes of the data stream. Animation is also commonly used to make transitions between static data graphics that represent different time slices. In these cases, the transitions are done between common statistical graphics, such as scatter plots, bar charts or pie charts. More details about possible transitions can be found in the taxonomy presented in.[25] A prominent example showing transitions is the Gapminder website[‡].

3. STREAMSQUEEZE

This section details our novel continuous space-filling visualization technique *StreamSqueeze*. To incorporate that recent events have higher relevance (condition C1) into our visualization scheme, we came up with the technique shown in Figure 1(a). In this case, the latest information item is displayed in large. Each subsequent list then contains twice as many items, which are scaled down accordingly.

Updating the visualization is a key aspect for analysis of events in dynamic information streams. For this particular visualization, this means that, as soon as a new item comes in, we remove the oldest one of the list, place it in the next column, and repeat this procedure over all columns shown on the screen as shown by the red arrows in Figure 1(a). The red arrows show one update sequence replacing $item_{14}$ with a new one, $item_{12}$ with $item_{14}$, $item_8$ with $item_{12}$ and $item_0$ with $item_8$. Note that the gray arrows show the update sequence for the next update.

3.1 Layout Function

Given the *StreamSqueeze* update schema, the horizontal positioning, or in other words the assignment of each item to a list, is trivial to compute based on the age order of the events, in which $a = 0$ marks the latest event and $a = 2^{|lists|} - 2$ the oldest displayed event:

$$list(a) = \lceil log_2(a+1) \rceil \tag{1}$$

[‡]http://www.gapminder.org/

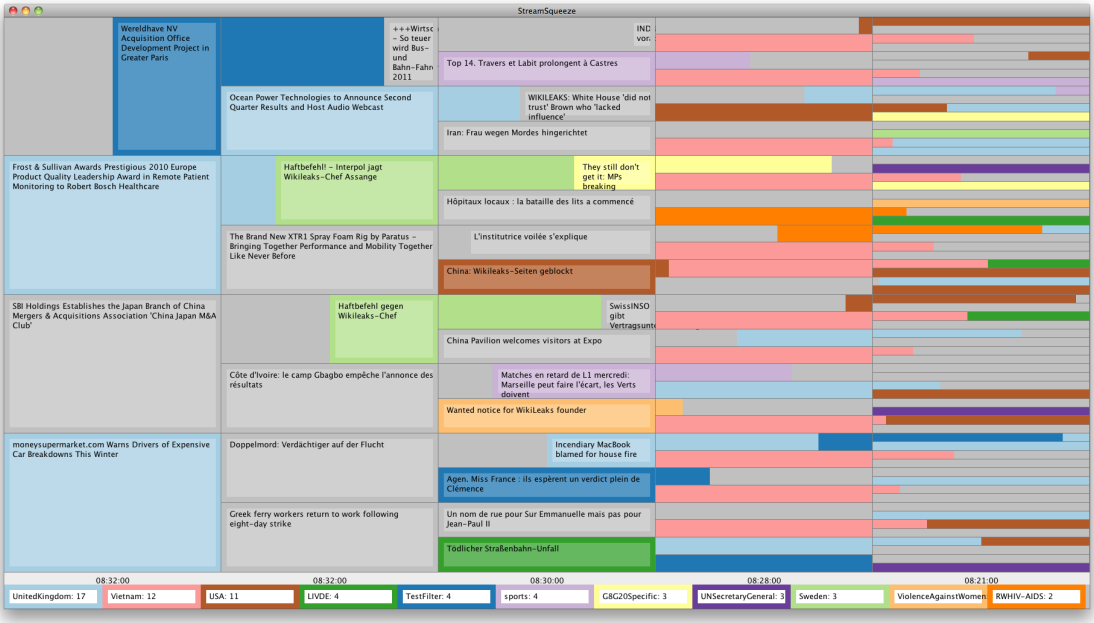


Figure 2. *StreamSqueeze* is a visualization technique for dynamic event data that 1) assigns more screen space to the latest events, 2) allows for better readability due to partially fixed text positions and 3) fosters traceability through animation and predictable positions (the events move from the leftmost column to right).

As an example, after item 12 is replaced by 14 as shown in Figure 1(a), its new age is $a = 3$ since there are three more recent events currently displayed and it is thus assigned to $list_2$.

Since we aim for smooth transitions rather than abrupt positional changes to enable traceability of items (C2) when moving an item from one list to the next one, we need to ensure that the item's positions before and after its update are adjacent. To achieve this, the vertical alignment function for placing the items within each list is based on a lookup of the index value modulo the size of the target list in a binary tree containing all values from 0 to 2^i, where i is the list id. Each time the binary tree is traversed to the right, we add 2^l, where l is the next level of the tree, as illustrated in Figure 1(b). The array (0,2,1,3) determines the positions of items 8 to 11 as shown in Figure 1(b). When the new item 12 arrives we need to remove item 8.

Given the above presented horizontal and vertical placement schemes, we can guarantee that after a new item is added, the position of an item either stays the same or it is moved to an adjacent screen position in the subsequent list. In the latter case, the moved item is scaled down by a factor of two. Since only the screen position of one item per list is changed, we can guarantee that this visualization technique triggers least visual changes.

Note that for using the *StreamSqueeze* technique with high-frequency streams in practice, the first few lists are often discarded as shown, for example, in Figure 2. The benefit of this is that the animated screen becomes more balanced in the sense that not every update completely changes the first list or half of the second one.

3.2 Animation Concept and Age Hints

As stated in condition C2 in the introduction, we assume that traceability can be achieved by smoothly moving items from one list to the next. However, in order to do this, we need to sacrifice the static positions of the items within a list at least to a certain degree.

In the first half of their lifetime within a column, the items' positions stay constant on the screen for better readability and easier interaction. The animation concept starts moving items out of the list once 50 percent of the list's items are added more recently than the considered item. The size of the item in the old list thus gives the user an indication of its forthcoming movement with the next update. Note that setting another percentage is possible, but low values result in fast movements that unnecessarily attract user attention and high values compromise the static text positioning.

Figure 2 illustrates how, for example the light green items in the center are smoothly moved out of the 2nd list into the third one. Note that the animation of the brown item in the center into the right 4th list starts earlier than its movement out of the 3rd list. This is due to the fact that items in this list move only half as fast as in the middle one. Since the last list contains more items than all previous columns together and since we only want to show one item at a continuous screen position, we refrain from animating all items of a column at once. Note that the items of one list contain several timestamps, there are several options of displaying the covered time range underneath each column, such as the first, last or median timestamp of the items. For the above referenced figure, we decided that the median is a good representation since it reduces the likelihood of identical timestamps next to each other when displaying labels of the full range. However, depending on the application scenario, the timestamp of the first or last item or the covered time could also reveal important real-time aspects of the data to the analyst.

Since many of the news headlines had identical timestamps, we added some time base jitter to make changes more traceable rather than adding many items at once. For demonstration purposes, we also artificially increased the speed of the news.

4. FINE-TUNING STREAMSQUEEZE

This section describes how we fine-tune the visualization technique. While the technique itself consists of a layout scheme and an animation concept, we introduce a number of possible variations of the layout scheme, several coloring schemes and an interaction concept to effectively support special-purpose analysis tasks.

4.1 Variations of StreamSqueeze

While we have shown that the proposed layout scheme introduces least visual changes under the given assumptions, this scheme completely blocks the visual position variables for other purposes. For many temporal visualizations, however, the x position is used to express the detailed temporal aspects of the shown items. While this can create overlap, it enables analysis tasks that are linked to interval-scaled aspects of time, which is not possible in our proposed layout. As an alternative layout, we therefore offer the option to sort certain lists of StreamSqueeze according to the temporal order of its contained items, which turns the temporal aspects into an ordered scale. As an example, this could be useful when monitoring stock trades since the temporal order of trades much defines future stock prices. A more complex option would be to introduce empty items into the screen to better express interval-scaled aspects of the sorted lists, which we will not investigate in this paper.

Analog to this temporal ordering, we also offer sorting based on metadata attributes, such as news categories or server names in the application examples in Section 5. This has the advantage, that it is possible to estimate the number of items in each category over the stream's history shown on the screen. Even with small item sizes, the combination of coloring and ordering can foster such insights.

In general, however, a major drawback of such sorted lists is that this destroys both the guarantee of least visual changes under the assumptions as well as the animation concept since it is likely to happen that an item from one list is moved to a non-adjacent screen position in the subsequent list. Note that the latter is even the case when two subsequent lists are sorted according to metadata categories since it is not guaranteed that the proportions of items of the lists stay constant over time. Since it is therefore not possible any more to animated items from one list into the other in the sorted layout, we use a light version of the item's color to give a hint of age and movement behavior to the users as shown in Figure 3.

4.2 Coloring Strategies

Coloring is a very powerful way to visually group items, which semantically belong together. In our concrete case, we color news items with identical categories or syslog messages from identical servers with the same color. In these cases, we restrict ourselves to 12 qualitative colors as suggested by the online application ColorBrewer.[26] Depending on the requirements of the interaction, red is reserved for marking moving items on the screen.

If the number of categories exceeds this threshold of twelve, we need to overcome a technical and perceptual issue. The simplest solution would be to reuse colors, which can result in misinterpretation. Note that the sorted layout variation is more suitable for such reassignments since sorting the categories for the layout reduces the likelihood that identically colored categories end up next to each other in the layout. This would only be the case if the eleven categories between them in the sorting order are missing in that particular list.

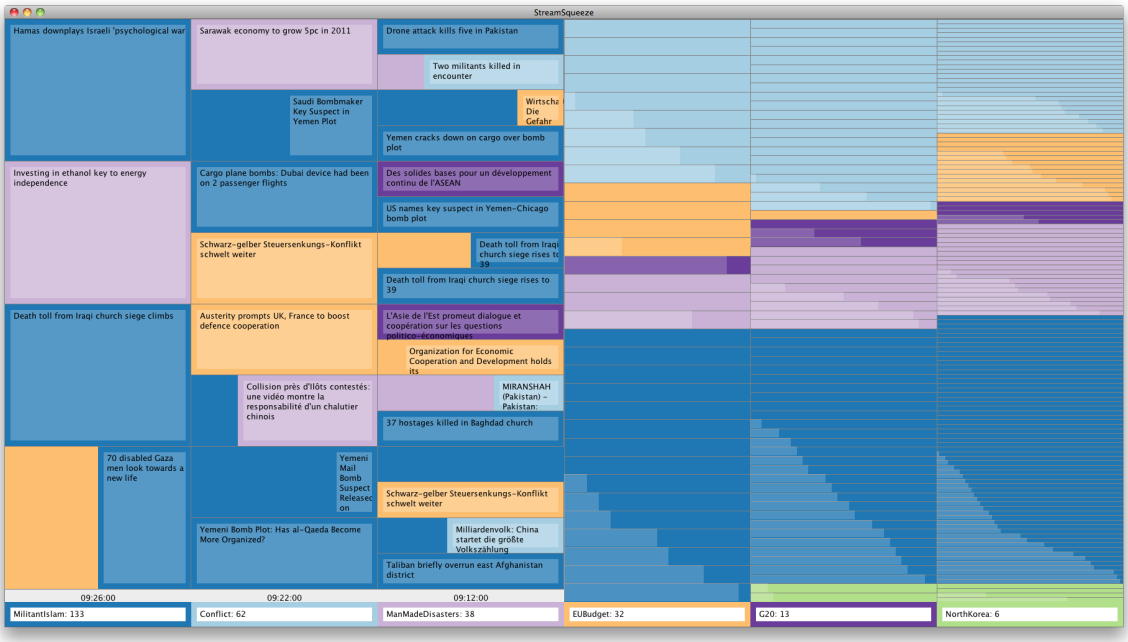


Figure 3. StreamSqueeze with sorting of the last three lists applied for crisis news monitoring. The six news categories are: North Korea, Militant Islam, G20, Man-made Disasters, Conflict, and EU Budget. The legend for color mapping is shown in the bottom row.

Our current solution overcomes the problem in a different manner by keeping counters for the number of items in each category and assigning a unique color to the twelve biggest categories. Items of all other categories will be drawn with gray background.

A drawback of this approach can be that there will be alternating assignments of gray and the least frequent color to two different categories resulting in a distracting unstable coloring scheme. However, this can be overcome by setting a threshold for reassigning the twelve colors, which is a value above the number of items in the least frequent colored class. For example, the least frequent colored category "politics" has 27 items, but will not be replaced by the still gray category "health" with 28 items since the threshold value is set to 33. Furthermore, we protocol the minimum age for each coloring class so that we can reduce the risk of alternating color assignments for the small classes.

4.3 Interaction Concept

StreamSqueeze is meant as an interactive visualization technique, in which the user can mark certain items in the screen, pause and resume the stream. While clicking moving objects is a challenge in itself, our proposed visualization technique has the favorable property that larger items in the earlier lists, which are essentially easier to click move faster than smaller items, which are harder to mark. This counterbalances the difficulties of user interaction with moving objects at least to some degree.

Another usage of the stream visualization is pausing and resuming of the streaming visualization in order to investigate certain items in detail. While pausing is easy to implement, resuming becomes a lot more difficult since the user often needs to understand what has happened in the meantime. Our current solution to this is that after resuming we speed up the paused items with a user defined constant until the real-time end of the stream is reached.

5. CASE STUDIES

In this section we demonstrate our technique in two different application domains: a news data stream and a syslog stream. The performance of the presented data streaming visualization technique in real-world environment depends on three important factors: First, the frequency of incoming events, second the number of monitored categories, and third the level of detail for each event.

There is an obvious trade-off that has to be made between these factors, which depends on the application and the task. For example, if the frequency of incoming event data is very high, the level of detail that can be comprehended by the user

in a monitoring scenario will be quite low. However, if it is necessary to show the content of the latest events in higher detail, the data stream might have to be filtered and/or throttled down to accommodate the user preferences. Additionally, the performance of the technique depends on external factors, such as available screen size and resolution. For the given two streams we removed the first columns of the visualization in order to cope with these high-frequency streams.

5.1 StreamSqueeze for News Monitoring

We have deployed the algorithm to work with an online news streaming system that was described in[27] in two different monitoring scenarios. This streaming system processes 80,000 to 100,000 articles per day from around 4,000 news sources in 43 languages. The data items contain various metadata that were extracted from the news reports, such as named entities, categories, url, country, and geo-location. In our first case, we have selected six news categories that would be considered important for the analyst in a news monitoring scenario: Man-made Disasters, North Korea, Conflict, EU Budget, Militant Islam, and G20. The user should be aware of any abrupt changes that could arise in real-time, and in this case the tool serves as an alerting system that can offer focus on the latest events, and provides additional details on demand. At the same time, these rapid changes that occur in the present have to be put into context of the development of events in this and all the other categories of interest in the past. Figure 3 shows a snapshot of the tool in which we are using the sorted variation of StreamSqueeze for the leftmost three lists. In this case, the items are sorted according to their news category, which is encoded by color.

The news data items are enriched with various meta-information that should be available to the user, either immediately or on demand. In order to allow the user to observe and interact with any of the items in the stream, the change of item's position should be kept minimal, which is in concordance with our algorithm's criterion of least visual changes. Additionally, the user can decide if the first columns, which can contain only 1, 2, or 4 events, should be kept or discarded from the visualization. Since the recent items appear larger than the old ones, they can be shown in more detail for quick inspection. Depending on the display size, the resolution, and the number of events in a single column, we can choose which metadata should be provided in the available item design space. We use the headline, but we could also use the tags that describe the article, or the named entities appearing in the news. An alternative would be to render the text in these lists significantly larger in relation to the available space. As an article shifts into the past, we limit the amount of information that can be presented in the display object, and we can use only the available named entities that are mentioned in the article, so that the user can get a brief idea about the individual news content.

Since the news aggregator gathers data from a large number of news websites, very often these different sources publish news articles that report on the same real-world event. The bigger the event, the more sources will report on it in a shorter period of time, and visual clustering occurs, where items of the same category next to each other are related (*locality* of events). The drawback of our layout sorting technique and the visual clustering it provides is that it strongly depends on the generality of the category.

In our second example (Figure 4) we do not filter the stream or restrict ourselves to the six pre-selected categories. The tool is set to monitor eleven categories that are most frequent in the stream. The advantage of this approach is that these most frequent categories will always stand out, and if the breaking event happens, we would immediately be able to see the context of the event within the category and its recent past behavior. Similarly to the first example, the category labels are placed in the bottom of the screen. As shown in the figure, the temporal sorting indicates if it is continuously reported on certain categories or if bursty events with a compact temporal extent occur such as the dark blue pattern in the last list indicating a story that was simultaneously reported on in the "Vietnam" category. Note that the imperfections of the sorting as shown by the step patterns in lighter colors relate to identical timestamps. In such cases, jitter is added for making the animation smoother, but the original timestamps are kept for sorting the data.

5.2 StreamSqueeze for SysLog

The ability of the StreamSqueeze algorithm to provide a visual overview of the most recent events in real-time streams makes it extremely useful for monitoring situations. One area where monitoring is often one of the core operational tasks is system administration. To be aware of emerging problems, security threats, anomalies, track the current usage or to investigate problems system administrators often need to take a look at log files in real-time. In most networks there is a central logging server in place which receives and collects all those generated system log events. In UNIX environments the so-called syslog protocol provides the ability to implement such remote logging capabilities.

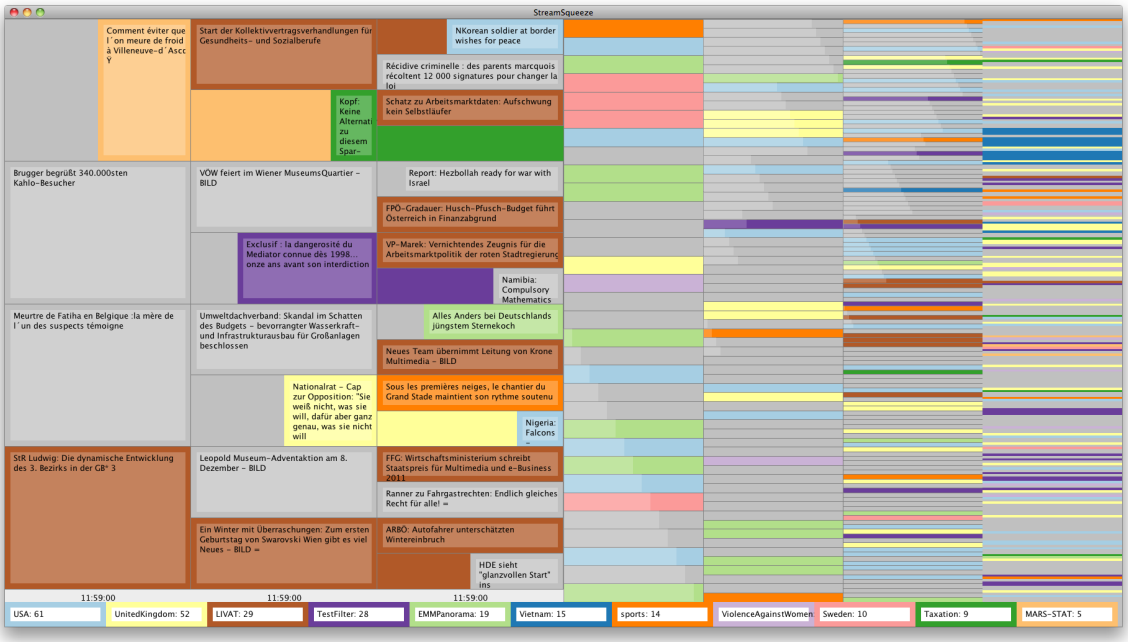


Figure 4. StreamSqueeze applied on top 11 news categories. Less frequent news categories were colored in gray. Temporal sorting of the last four lists reveals information about the temporal extent of news topics as seen on the compact dark blue topic in the "Vietnam" category.

We integrated the visualization into a visual analytics software for system log events, which is used by system administrators in our working group to access syslog data. In peak times the system receives around 100,000 events per hour. To be able to follow such high frequency streams, the application is able to filter the events in different ways to show only a limited set of servers or services, or even based on a scoring threshold to focus on very uncommon events. In the left area of the software (Figure 5) there are different filters applied to the current real-time stream. On the right (color window) the user is able to change the used colormap.

The traditional way on UNIX systems for investigating log files in real-time is to execute the "tail -f" command. This often used tool monitors textual log files and prints the most recent lines to the screen. From a user's perspective this works pretty well for low-frequency streams. If you have high frequency streams, it is pretty hard to follow individual events, because of the limited numbers of lines which can be displayed at once. When the system administrator is interested in messages from several servers each having a very different number of events, it is hard to spot those events from servers only producing few messages occasionally.

In the center of Figure 5 you can see an implementation of the StreamSqueeze algorithm utilizing a sorted layout for the four rightmost columns without text labels in exactly such a situation: Few servers are producing many events (orange, blue and purple colored events), while others (e.g, violet events) produce significantly less messages. The colors are mapped to the different servers to distinguish their events. As a result of our grouping schema the visibility of those low frequency servers is still maintained. The visualization is updated in real-time as soon as new events are collected by the logging server. Because of the concept that later columns have more rows, it is hard to pick and select those events for further exploration. But the concept of having more rows is important, because otherwise the user will not be able to interact and explore with the visualization due to rapid updates and changes. By increasing the number of rows an individual event will remain longer in the column and this gives the analyst more time to investigate this particular event.

6. DISCUSSION

6.1 Assessment of StreamSqueeze

The first and obvious advantage of the StreamSqueeze technique is that space is made for new information to be displayed in a dynamic usage scenarios. In particular, recent items always appear at the same column, in which the items can be depicted larger and with more details.

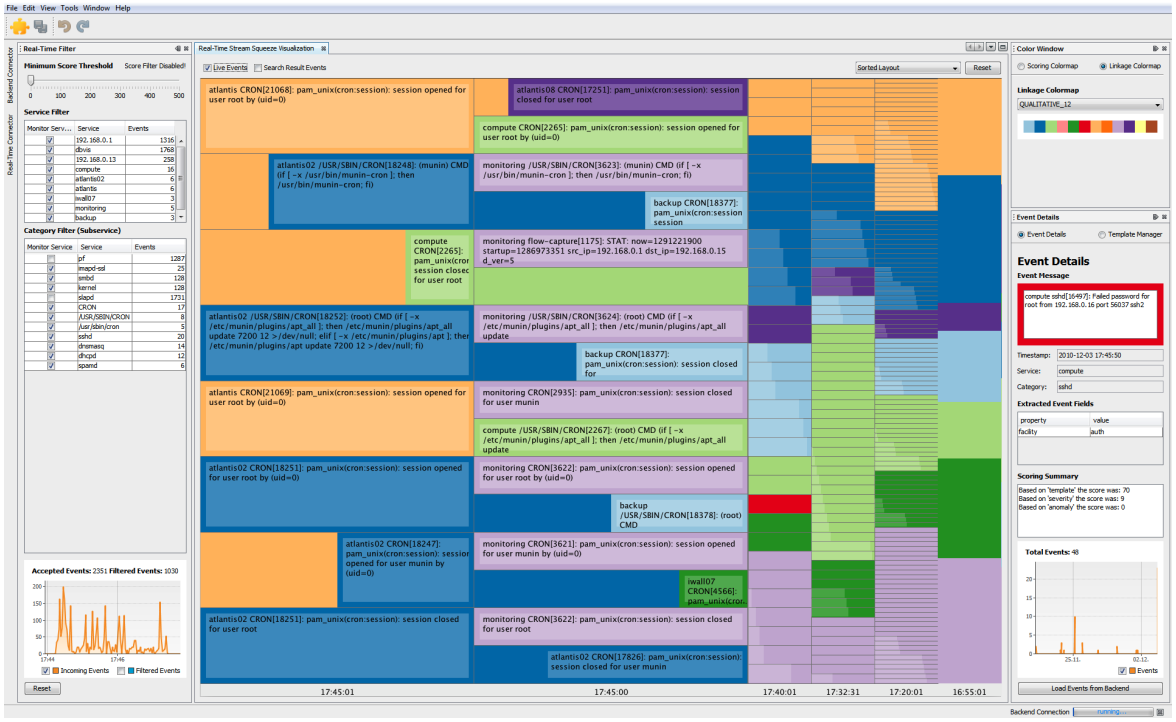


Figure 5. Event Visualizer using the StreamSqueeze visualization to enhance real-time monitoring for syslog messages. Distinct color are used to indicate which host created the respective log events. Details for the event marked in red are shown on the right.

Second, while other streaming techniques render items with either constant size or employ aggregation concepts, the screen-filling nature and continuous adaptation of the shown details according to the items' relative age make the technique scalable. In the example shown in this paper the number of displayed items ranges from 124 to 508. However, the visualization scheme can be extended in a straight forward way to show 2047 items (1024 items of 1 pixel height in the last column). Using an adapted version with several columns containing an equal number of items at the end can push this limit even further.

Third, local changes and continuous animation allow tracking of items on the screen across subsequent lists. Since the original layout of StreamSqueeze always moves an item into one of the two adjacent positions in the next list, we can guarantee that an update only triggers positional changes of one item per list and that these changes are all grouped in the horizontal row starting at the largest item in the first list and ranging to the last list. Therefore, we can claim that when discarding the animation concept, our visualization technique triggers least visual changes to the layout, which are horizontally grouped rather than scattered over the whole screen. Of course, this only holds under the assumption that we need to shift one item per list with each update.

While the animation concept of StreamSqueeze relaxes the condition of least visual changes to the screen, but give more support to enable traceability of items across the lists by continuously moving items.

One drawback of StreamSqueeze is that changing size, proportions and sometimes even removing text labels of the items when moving them from one list to the other do not allow for easy recognition of items. However, the screen position of each item canonically defines its previous positions in the lists with less items. Furthermore, interactive hints when hovering the mouse over an item without a text label shows more details and keeping a consistent color coding of the items supports recognition of an item to some degree.

In contrast to classical temporal visualization techniques, StreamSqueeze boldly substitutes the interval-scaled aspect of each item by a rough binning based on the logarithmically-scaled temporal order of the streamed events. For each such bin, represented by a list on the screen, only one timestamp is shown. However, visual hints such as partly moved items or items partly drawn with a white overlay compensate for this rigorous design decision.

6.2 Assessment of Variations, Coloring and Interaction

In Section 4, we presented variations of the previously proposed layout scheme, which is inherent to StreamSqueeze. Optimizing this layout scheme for special purposes therefore triggers many changes to the overall visualization concept. However, some usage scenarios significantly benefit from the sorting options, especially in the columns representing older events. Since the items of these columns are rendered so small that we cannot show text labels any more, we have less motivation to make them stay at constant screen position throughout their lifetime within that column. Major benefits of such a sorting on either the temporal aspects or the metadata assigned to these columns enables analysis tasks related to pattern detection in the frequency or the sequence of events through their colored representations.

However, a sorting destroys the original animation concept that draws part of the older items of a list in their adjacent future screen position within the next list. In such cases, an opaque white overlay indicates when the item is going to be moved out of the list.

Since coloring is an important aspect for recognition of items and for their visual grouping, we defined a constant color for each event category. Since the employed 12 color quantitative scale from ColorBrewer[26] is quite limiting, we decided to draw the remaining categories in gray. In many cases, the red color from this scale is reserved for interactive marking purposes.

In general, interaction with moving items on a high-frequency screen is quite challenging. Due to the fact that largely drawn recent items change their screen positions quicker than older ones, the increase in size makes these fast moving items easier to mark. In contrast, slowly moving items are drawn considerably smaller in size.

6.3 Comparison with other techniques

It is not easy to compare existing techniques to StreamSqueeze. This is due to the specific nature of our technique, and to the task for which it was designed for.

Most streaming techniques do not treat latest data differently than the data from the past, and this is the major difference between StreamSqueeze and the others. Only the Article Threads algorithm[16] treats recent data as important, keeps it displayed in the visualization. ThemeRiver[11] and EventRiver[15] are not very well suitable for maintaining layout stability, since the layout algorithms require that the visualization has to be recalculated once the new data comes in, and this would disrupt the positioning of the streams. With respect to locally restraining changes to certain screen areas, both ThemeRiver and EventRiver can handle similar streams much better than the other techniques.

When working with a data stream, the first challenge are the unbounded characteristics of the incoming data. On the one hand, the data is unbounded by the numerical values that the variables that are monitored might have, so usually some restrictions and assumptions have to be made during the development process. These assumptions depend on the task for which the visual analytics tool is designed, and on the mapping of data to visual variables. In line charts, for example, numerical values are mapped to the y-position on the corresponding time axis. If the numerical values become too high, the layout has to be rescaled, thus making *comparison of variables* very hard. In general, assigning the numerical value of a variable to position or size, which is in any other case the most desired mapping, can lead to overlapping or displacement, thus in both cases reducing *legibility*.

On the other hand, data can be unbounded by the number of variables, which is the case, for example, with dynamic text data streams, where the number of news topics cannot be known in advance. Designing a visualization that can accommodate both dynamic change of the number of variables and their numeric values still remains a challenge. Visualizing a fixed large *number of variables* is another criterion where existing techniques do not perform well, except for the Horizon Graph.[28] The *amount of information* reflects the number of dimensions that can be encoded for one data item.

The disadvantage of our technique is that, due to the sorting and optimization requirements and the constraint to show the fixed number of individual data items from the stream, some temporal information about the events gets lost. Other techniques, such as ThemeRiver, EventRiver, Horizon Graph, ArticleThreads and even simple line charts clearly outperform StreamSqueeze on this aspect. However, StreamSqueeze could be extended to use different time window aggregates in the columns, instead of a fixed number of events, and this is a part of our future work.

7. CONCLUSIONS

Since real-time information offers valuable advantages over competitors and invaluable security-related advantages over opponents, we believe that there is a need for visual support for exploratory analysis tasks on dynamic information streams in many application areas. In such monitoring scenarios, it is characteristic that more relevance is assigned to recent events and common analysis task relate to previously seen events.

In this paper we presented a novel continuous space-filling visualization technique called *StreamSqueeze* for visualizing information streams taking into account the higher relevance of recent events and the need for making individual items traceable. The main benefits our our technique are: 1) More screen space to the latest events due to their increased relevance for monitoring tasks. 2) Partially fixed text positions on the screen results in better readability of event labels. 3) Our technique triggers least visual changes in the event lists and only localized changes on the screen to foster traceability in an animated real-time monitor.

To demonstrate the applicability of our approach we applied it to high-frequency real-world data streams. The first case study detailed how the static sorted layout can be used to track topics in a large-scale online news stream. In this scenario, it was very important to keep fast updated items in the first few columns at static positions until they move to the columns on the right. As soon as labels became unreadable grouping of the prominent categories was more important than maintaining their static positions. The second case study showed StreamSqueeze for monitoring log messages from a number of different servers. Depending on the purpose and the configuration mode of the server, it might produce a lot or just a few log messages. Despite this fact, the grouping scheme of our visualization technique maintained the visibility of events from low-frequency servers while showing all recent server events at once.

In the future, we plan to continue our work on StreamSqueeze by extending it through an aggregation strategy to analyze longer time periods of high-frequency streams and by conducting a user study to assess the effects that our visualization technique has on readability and traceability in a dynamic monitoring scenario. Furthermore, we want to extend the visualization technique through link and brush interaction to ease referencing related events in the stream.

APPENDIX A. BIOGRAPHY OF THE PRINCIPAL AUTHOR

Florian Mansmann is a post-doctoral researcher at the University of Konstanz in Germany. He received a Bachelor degree in Information Engineering from the University of Konstanz in 2003 and a Masters degree in Business Information Management from the Vrije Universiteit Brussels in 2004. In 2008 he received a PhD from the University of Konstanz on the topic Visual Analysis of Network Traffic - Interactive Monitoring, Detection, and Interpretation of Security Threats. His research interests include information visualization, visual analytics, and network security. He currently is a program committee member IEEE InfoVis, VizSec, EuroVA and co-chaired Discovery Exhibition.

REFERENCES

[1] Thomas, J. and Cook, K., [*Illuminating the path: The research and development agenda for visual analytics*], IEEE Computer Society (2005).

[2] Keim, D. A., Kohlhammer, J., Ellis, G., and Mansmann, F., eds., [*Mastering The Information Age - Solving Problems with Visual Analytics*], Eurographics (2010).

[3] Javed, W., McDonnel, B., and Elmqvist, N., "Graphical Perception of Multiple Time Series," *Visualization and Computer Graphics, IEEE Transactions on* **16**(6), 927–934 (2010).

[4] Wong, P. C., Foote, H., Adams, D., Cowley, W., and Thomas, J., "Dynamic visualization of transient data streams," in [*IEEE Symposium on Information Visualization (INFOVIS)*], **0**, 13, IEEE Computer Society, Los Alamitos, CA, USA (2003).

[5] Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J., "Models and issues in data stream systems," in [*Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*], *PODS '02*, 1–16, ACM, New York, NY, USA (2002).

[6] Chin, G., Singhal, M., Nakamura, G., Gurumoorthi, V., and Freeman-Cadoret, N., "Visual analysis of dynamic data streams," *Information Visualization* **8**(3), 212–229 (2009).

[7] Best, D. M., Bohn, S., Love, D., Wynne, A., and Pike, W. A., "Real-time visualization of network behaviors for situational awareness," in [*Proceedings of the Seventh International Symposium on Visualization for Cyber Security*], *VizSec '10*, 79–90, ACM, New York, NY, USA (2010).

[8] McLachlan, P., Munzner, T., Koutsofios, E., and North, S., "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*], 1483–1492, ACM (2008).

[9] Bertini, E., Hertzog, P., and Lalanne, D., "SpiralView: towards security policies assessment through visual correlation of network resources with evolution of alarms," in [*IEEE Symposium on Visual Analytics Science and Technology, 2007. VAST 2007*], 139–146 (2007).

[10] Takada, T. and Koike, H., "Mielog: A highly interactive visual log browser using information visualization and statistical analysis," in [*In Proc. USENIX Conf. on System Administration*], 133–144 (2002).

[11] Havre, S., Hetzler, E., Whitney, P., and Nowell, L., "Themeriver: Visualizing thematic changes in large document collections," *IEEE Transactions on Visualization and Computer Graphics* **8**(1), 9–20 (2002).

[12] Wei, F., Liu, S., Song, Y., Pan, S., Zhou, M. X., Qian, W., Shi, L., Tan, L., and Zhang, Q., "Tiara: a visual exploratory text analytic system," in [*Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*], *KDD '10*, 153–162, ACM, New York, NY, USA (2010).

[13] Kerr, B., "THREAD ARCS: An Email Thread Visualization," *Information Visualization, IEEE Symposium on* **0**, 27–218 (2003).

[14] Fisher, D., Hoff, A., Robertson, G., and Hurst, M., "Narratives: A visualization to track narrative events as they develop," in [*Visual Analytics Science and Technology, 2008. VAST '08. IEEE Symposium on*], *Visual Analytics Science and Technology, 2008. VAST '08. IEEE Symposium on* , 115–122 (2008).

[15] Luo, D., Yang, J., Krstajic, M., Ribarsky, W., and Keim, D., "Eventriver: Visually exploring text collections with temporal references," *IEEE Transactions on Visualization and Computer Graphics* **99**(PrePrints) (2010).

[16] Krstajic, M., Bertini, E., Mansmann, F., and Keim, D. A., "Visual analysis of news streams with article threads," in [*StreamKDD '10: Proceedings of the First International Workshop on Novel Data Stream Pattern Mining Techniques*], 39–46, ACM, New York, NY, USA (2010).

[17] Hetzler, E. G., Crow, V. L., Payne, D. A., and Turner, A. E., "Turning the bucket of text into a pipe," in [*INFOVIS '05: Proceedings of the 2005 IEEE Symposium on Information Visualization*], 12, IEEE Computer Society (2005).

[18] Aigner, W., Miksch, S., Muller, W., Schumann, H., and Tominski, C., "Visual methods for analyzing time-oriented data," *IEEE Transactions on Visualization and Computer Graphics* **14**(1), 47–60 (2008).

[19] Ankerst, M., Keim, D. A., and Kriegel, H.-P., "Circle segments: A technique for visually exploring large multidimensional data sets," in [*Visualization '96, Hot Topic Session, San Francisco, CA*], (1996).

[20] Kumar, N., Lolla, N., Keogh, E., Lonardi, S., and Ratanamahatana, C. A., "Time-series bitmaps: a practical visualization tool for working with large time series databases," in [*SIAM 2005 Data Mining Conference*], 531–535, SIAM (2005).

[21] Hao, M., Keim, D., Dayal, U., and Schreck, T., "Multi-resolution techniques for visual exploration of large time-series data," in [*Eurographics/IEEE-VGTC Symposium on Visualization, 23 - 25 May 2007, Norrkoeping, Sweden*], (2007).

[22] Hao, M. C., Keim, D. A., Dayal, U., and Schreck, T., "Importance-driven visualization layouts for large time series data," in [*Proceedings of IEEE Symposium on Information Visualization (InfoVis '05)*], (2005).

[23] Kincaid, R., "Signallens: Focus+context applied to electronic time series," *IEEE Transactions on Visualization and Computer Graphics* **16**, 900–907 (2010).

[24] Xie, Z., Ward, M., and Rundensteiner, E., "Visual exploration of stream pattern changes using a data-driven framework," in [*Advances in Visual Computing*], Bebis, G., Boyle, R., Parvin, B., Koracin, D., Chung, R., Hammound, R., Hussain, M., Kar-Han, T., Crawfis, R., Thalmann, D., Kao, D., and Avila, L., eds., *Lecture Notes in Computer Science* **6454**, 522–532, Springer Berlin / Heidelberg (2010).

[25] Heer, J. and Robertson, G., "Animated transitions in statistical data graphics," *IEEE transactions on visualization and computer graphics* **13**(6), 1240–1247 (2007).

[26] Brewer, C. A., "ColorBrewer," (2011). http://www.ColorBrewer.org, accessed on March 28, 2011.

[27] Krstajic, M., Mansmann, F., Stoffel, A., Atkinson, M., and Keim, D., "Processing online news streams for large-scale semantic analysis," in [*1st International Workshop on Data Engineering meets the Semantic Web*], (2010).

[28] Saito, T., Miyamura, H. N., Yamamoto, M., Saito, H., Hoshiya, Y., and Kaseda, T., "Two-tone pseudo coloring: Compact visualization for one-dimensional data," in [*Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*], 23–, IEEE Computer Society (2005).