Towards a Visualization Process Model for Online Visualization

Marco Angelini* University of Rome "La Sapienza"
Giuseppe Santucci† University of Rome "La Sapienza"
Heidrun Schumann‡ University of Rostock
Hans-Jörg Schulz§ University of Rostock

ABSTRACT

With today's technical possibilities, a stable visualization scenario can no longer be assumed, as underlying data and operations are much more in flux than in traditional scenarios. We term such dynamic visualization scenarios *online visualization*. In contrast to traditional offline visualization that rely on monolithic visualization operators and a monolithic dataset, it permits the user to interact with, steer, and change the visualization at intermediate time points and not just after it has been completed. In this poster, we detail a possible extension of the Data State Reference Model (DSRM) [2] to capture not only offline, but also online visualization processes, as well as online/offline hybrids. We further showcase our extended model as a prospective user interface to communicate an online visualization's progress, as well as to interactively steer it.

1 INTRODUCTION

The concept of *online visualization* is based on the idea of allowing a user to view and interact with early "draft visualizations" [5] that may still miss parts of the data (e.g., showing just a few first samples) or that may still miss visualization stages (e.g., showing just data points but no labels yet) or both. To realize this idea, we assume that a dataset contains n *data items* and that it is given in i *data chunks*, where $i = 1$ denotes the traditional way of visualizing the whole dataset in one pass and $i = n$ denotes a stream of individual data items. Similarly, we assume that the visualization process consisting of m individual *processing operations* is given in j *processing steps*, where $j = 1$ denotes the traditional, offline way of generating the visualization in one monolithic procedure and $j = m$ steps denote the most fine-grained modularization into individual processing operations. Both subdivisions – on the data and on the process – can be used in combination to yield various different configurations. This gives us the flexibility to actually subsume a number of similar ideas that have been proposed under a variety of different names, such as fine-grain visualization [8], streaming data visualization [4, 10], per-iteration visualization [3], progressive visualization [7, 9], or incremental visualization [1]. So, while this idea has been stated and restated in various forms, a common terminology, let alone a general model for these different scenarios is so far missing. Hence, we introduce our approach for providing such a model for online visualization in the following section, and we exemplify its utility for capturing, monitoring, and steering a scenario of streaming data visualization in the section thereafter.

2 AN ONLINE VISUALIZATION MODEL

We extend the DSRM to model the subdivision of the process through enhanced operators, and the subdivision of the data by introducing enhanced connectors between them. We also explicitly model the sources (i.e., datasets) and the sinks (i.e., resulting views) in our model (Figure 1a), as it is common nowadays that multiple heterogeneous data sources are visualized in multiple views with some data sources being available in full, whereas others are made

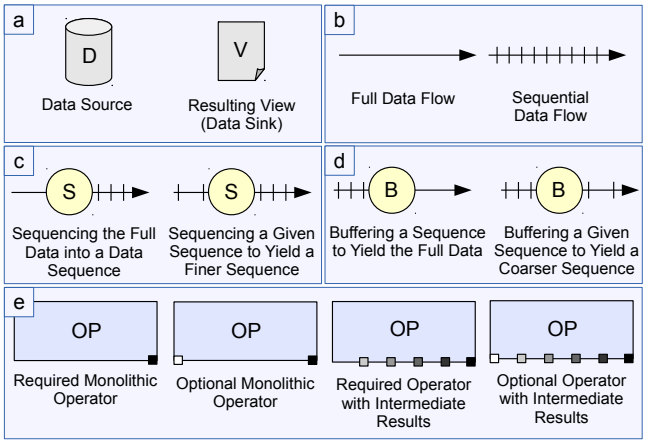

Figure 1: Our extended model of the visualization process adds explicit data sources and data views (a), sequential data flow (b), sequencing and buffering mechanisms (c+d), as well as operators that make available intermediate results (d).

available as a stream of data chunks. With this addition to the model, these cases can be clearly represented and distinguished.

Enhanced operators feature a number of white to black marks at their bottom, each of them representing an (intermediate) result: the black mark represents the complete result, the gray marks represent partial results, and the white mark represents the possibility to simply pass on the unchanged input as an output, effectively making the operator optional. The traditional monolithic operator has thus merely a black mark – or if it is optional, a white and a black mark. Online visualization operators, which have been subdivided to produce partial results, feature in addition a number of gray marks. The different kinds of operators according to how they are marked are listed in Figure 1 e). Executing such a subdivided operator means that it returns a series of results – first a result identical to the input if the operator is optional, then partial results of increasing quality, until finally the complete result is produced. Each produced result (identity, partial result, complete result) must be a valid input to all operators that follow immediately after. Moreover, enhanced operators feature *parameters* that influence the operator behavior and *metrics* that allow the user to monitor their progress.

Enhanced connectors feature a number of marks across the connector line, denoting a data stream, as opposed to passing the full dataset at once, which is denoted by an unmarked connector (Figure 1b). A stream of chunked data can originate natively from a data source that delivers its contents in this manner or from an online visualization operator that sends its output in the form of subsequent versions of increasing completeness. A full dataset can also be transformed into an artificial stream of data chunks by sequencing the data as it is passed from one operator to the next (Figure 1c). Similarly, it can also be buffered to yield the full dataset or to produce larger data chunks out of smaller ones (Figure 1d).

Note that our enhanced version of the DSRM can still be used to model offline visualization as a special case by only utilizing edges representing full data flow (no marks along the connectors) and nodes representing required monolithic operators (a single black mark denoting only the complete result).

*e-mail: angelini@dis.uniroma1.it
†e-mail: santucci@dis.uniroma1.it
‡e-mail: schumann@informatik.uni-rostock.de
§e-mail: hjschulz@informatik.uni-rostock.de

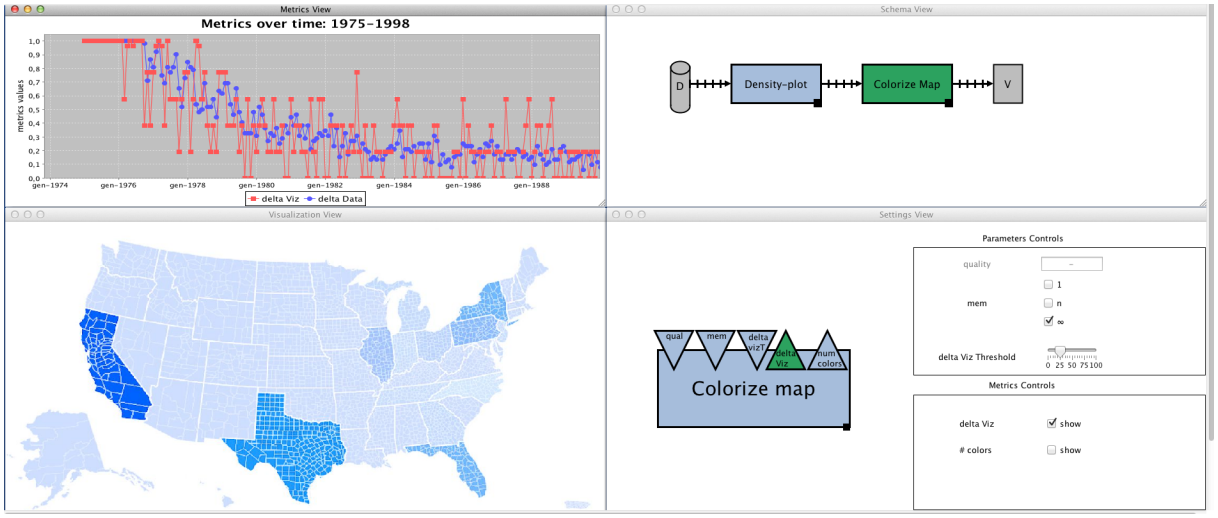

Figure 2: Detecting changes in the FARS 1975-1998 data streaming. The output visualization is shown in the bottom left window. The user is shown Δvis and $\Delta data$ in the top left window. He has selected the `Colorize Map` operator in the process overview in the upper right window and the bottom right window shows its details together with its parameters and metrics. The top left window has been zoomed in on the interval 1975-1989 and shows that both Δvis and $\Delta data$ are more stable after the first 5 years of data. However, some peaks on Δvis are much higher than corresponding peaks in $\Delta data$. This issue can be dealt with by tuning Δvis^T in the bottom right window.

3 Online Visualization for Subdivided Data

This section exemplifies an online visualization for natively chunked (i.e., streamed) data for detecting new trends in the data stream. So, the focus lies not on visualizing the incoming data but on highlighting changes that alter the overall data distribution.

We use the NHTSA Fatality Analysis Reporting System (FARS) [6] data of fatal car accidents in the US from year 1975 to year 1998 with 954,264 accidents as a data stream to which new car accidents are continuously added. The goal of the online visualization is to tune the system parameters in order to show the density distribution variations of accidents across the US. The data is collected at a constant rate (e.g., every hour, day, week, month) and data densities are rendered using a choropleth map, split in N areas (states or counties) in which each accident density value is mapped on an ordered set of different shades of blue.We use this assumption for calculating the delta metric $\Delta vis(i,i+1)$ that compares the last two visualizations (i.e., the visual difference between the last two intervals). Moreover, we define the metric $\Delta data(i,i+1)$, i.e., the underlying data changes between the last two intervals. Δvis is used to alert the user when a visual change occurs and $\Delta data$ is used for observing details and tuning system parameters.

Δvis is compared against a threshold value Δvis^T raising an alarm when it exceeds such a threshold. Parameters affecting the system are the data collection rate (hourly, daily, weekly, monthly), the choropleth color scale, and, obviously, the actual Δvis^T.

We model this online visualization with the process schema shown in Figure 3. The data source produces a data stream, where each chunk corresponds to the accidents occurred in a month. These chunks feed the monolithic operator `Density-Plot` that cumulates all past chunks and computes a density map, based on the proportion of all accidents across the states and counties of the US, depending on the parameter *Granularity* that can be set to either *state* or *county*. It further computes the metric $\Delta data$. The monthly updated density plots constitute the chunks that feed the operator `Colorize Map` that processes them independently associating each state (or county) with the corresponding color and computes the Δvis and the *NumColors* metrics, i.e., the number of distinct colors used in the visualization. This metrics are used to optimize the mapping between densities and colors. It accepts the Δvis^T as parameter and raises an alarm when needed. Such a schema is used in the user interface as a means for inspecting metrics and altering parameters as depicted in Figure 2 that shows the last interval i in which the visualization has changed. The figure further shows the metrics Δvis and $\Delta data$ that support the user in interacting with the `Colorize Map` operator, in particular in tuning the Δvis^T parameter observing the resulting differences between $\Delta data$ and Δvis.

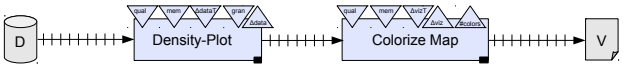

Figure 3: Online visualization process model of the streaming scenario described in Section 3.

References

[1] M. Angelini and G. Santucci. Modeling incremental visualizations. In *Proc. of EuroVA'13*, pages 13–17, 2013.

[2] E. H. Chi and J. T. Riedl. An operator interaction framework for visualization systems. In *Proc. of IEEE InfoVis'98*, pages 63–70, 1998.

[3] J. Choo, C. Lee, and H. Park. PIVE: A per-iteration visualization environment for supporting real-time interactions with computational methods. Technical report, Georgia Institute of Technology, 2013.

[4] J. A. Cottam. *Design and implementation of a stream-based visualization language*. PhD thesis, Indiana University, November 2011.

[5] D. Fisher, I. Popov, S. M. Drucker, and mc schraefel. Trust me, I'm partially right: Incremental visualization lets analysts explore large datasets faster. In *Proc. of ACM SIGCHI'12*, pages 1673–1682, 2012.

[6] NHTSA. Fatality Analysis Reporting System (FARS), 2013.

[7] R. Rosenbaum and H. Schumann. Progressive refinement: more than a means to overcome limited bandwidth. In *Proc. of VDA'09*, pages 72430I:1–12, 2009.

[8] D. Song and E. Golin. Fine-grain visualization algorithms in dataflow environments. In *Proc. of IEEE Vis'93*, pages 126–133, 1993.

[9] C. D. Stolper, A. Perer, and D. Gotz. Progressive visual analytics: User-driven visual exploration of in-progress analytics. *IEEE TVCG*, 20(12), 2014. to appear.

[10] P. C. Wong, H. Foote, D. Adams, W. Cowley, L. R. Leung, and J. Thomas. Visualizing data streams. In B. Kovalerchuk and J. Schwing, editors, *Visual and Spatial Analysis: Advances in Data Mining, Reasoning, and Problem Solving*, pages 265–291. Springer, 2004.