EuroVis Workshop on Visual Analytics (2013)
M. Pohl and H. Schumann (Editors)

Modeling Incremental Visualizations

Marco Angelini, Giuseppe Santucci

University of Rome "La Sapienza", Rome, Italy

Abstract

An increasing number of applications call for the incremental/iterative drawing of a visualization. That is an obvious requirement when dealing with continuously changing data, like the emerging field of data streams or scientific visualizations that have the burden of rendering complex and evolving physical phenomena. This paper postulates that the same need is rising in the field of Visual Analytics and cloud based applications and, in order to provide a support for such processes, it presents a formal model for characterizing the iterative drawing of a visualization, describing the practical issues and outlining the main parameters that can be used to drive and evaluate the whole process. The proposed model is general enough to capture all of the above presented scenarios. Two examples are presented, showing the role that such a model can play in designing iterative visualizations.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms I.3.2 [Computer Graphics]: Graphics Systems—Distributed/network graphics I.3.8 [Computer Graphics]: Application—Visualization

1. Introduction

A number of applications call for the incremental drawing of a visualization. This is a clear requirement for scientific visualizations (see, e.g., [Ma09]) that very often deal with large amounts of continuously changing data, in order to produce up to date, accurate, and realistic visualizations. However, we foresee other scenarios that exhibit similar needs.

Data Streaming. This is an emerging field and one of the possible goals of an application is to produce a continuous visualization of the data. Typical constraints are that it is not feasible to store the whole stream and that, in some cases, it is even impossible to process all the data. That calls for using statistical indicators (e.g., kernel density estimation) that allow for representing in a compact way the processed information and for dealing with several kinds of approximations.

Cloud Streaming. In this case, while the data streamed across the cloud is finite, we assume that the transfer bit rate is several orders of magnitude less than the data size. If the goal of the cloud application is to produce a visualization of a remote large dataset it makes sense to foresee some mechanisms that allows for proceeding in an incremental way, using the initial part of the downloaded data to produce partial results. This is very common in the straightforward case of video streaming, in which the server organizes the data in self-contained chunks, i.e., pieces of data that can be used without other further information, and send the initial part of the movie as soon as possible, allowing the end user to start watching the video even if the whole file is still not available. However, the cloud usage is continuously increasing and the cloud data servers, that are fighting each others to conquer the market, are likely to provide additional services in order to mitigate the slowness of the connection. In particular we can foresee that specialized scientific data server will support a) random storage feature and b) data transfer in self-consistent chunks together with a minimal set of data aggregates (e.g., min, max, and number of items). That allows for using a partial download as a preliminary result, considering it as a random sample of the whole dataset.

Iterative VA algorithms. According to [Shn08] visual analysis is useless if the system does not allows for quick interaction (e.g., 10 frames per second updates) while exploring data. Among the several obstacles that prevent a tight interaction between the automated data analysis and visualization, the computation time of algorithms has a key role. Typical algorithms used in VA applications (e.g., clustering, dimension reduction, etc.) take a long time to produce the result and that slows down the interaction speed, making the system quite unusable. Also in this situation it makes sense to foresee some mechanisms that allow the user for interact-

DOI: 10.2312/PE.EuroVAST.EuroVA13.013-017

ing with incremental results, i.e., dealing with an incremental drawing of a visualization. In particular, we exploit the fact that several algorithms used in VA scenarios are iterative, producing a sequence of approximate results that converge to the final solution. The idea is to drive the visualization using such intermediate solutions, as soon as they are available, allowing for a faster interaction. In this case we foresee two kinds of approximations: a) errors coming from the visualization of partial results, obviously affecting the actual visualization and b) errors that rise from the *interaction* with a partial visualization, affecting further analysis activities.

This paper presents a formal model for characterizing the iterative drawing of a visualization, describing the practical issues and outlining the main parameters that can be used to drive and evaluate the whole process. The model is general enough to capture all of the above presented scenarios.

The paper is structured as follows. Section 2 presents related work, Section 3 introduces the proposed model, Section 4 presents two case studies, and Section 5 concludes the paper, outlining future work.

2. Related Work

To the best of the authors' knowledge, the idea of modeling different data streaming scenario with a unique formal model is a novel one. Different attempts have been made in order to apply visual analytics to data streams, but often they cope only with focused scenarios and are mainly tailored solution to a specific problem. In the following we report several related papers that inspired our work.

[CR98] and [CMS99] propose different models for infovis applications; [JKMG02] and [Twe97] deal with the characterization of interactive visualizations process. [CL08] and [CLW12] cope with infovis applications for dynamic data, focusing on the interpretability of the obtained visualizations. The proposal in [FDCD12] introduces a framework for coping with data streams, describing a series of required operations and tasks, without considering result approximations or different cases of data streams. Still in a data stream scenario [XWR10] discusses how to present significant visualization changes to the user, applying merge windows algorithms. [WFA*03] presents an adaptive visualization technique based on data stratification to ingest stream information adaptively when influx rate exceeds processing rate. Concerning big data processing, [PTMB09] proposes a framework that copes with computational time issues of both the main application and the visualization part, proposing methods for the manual skipping of meaningless iterations of the visualization process. [Ma09] offers an ample dissertation on the right use of visualization for large scale data application, and how to exploit approximation algorithms. Indeed it lacks a formal dissertation and it does not propose metrics or indicators for evaluating the obtained results. [FPDs12] copes with the concepts of incremental visualizations and approximation of final results based on samples, but it does not generalize these concepts. [WA12] studies and validates the accuracy of approximate, temporary visualization results, but limited to the field of clustering. In a similar way, [RZH12] discusses a visual approximation scenario for parallel coordinates. Concerning functions for result approximation and error estimation, [HCZD04] proposes a method for reducing the computational cost of approximating a density function, but tied to object tracking applications. [ZCWQch] and [Duo07] presents a similar work for data stream kernel density estimation.

3. Model

In this section we present the general principles underlying the proposed model, able to represent each application class as a stream of data with particular characteristics; moreover we introduce several *Quality indicators* that will be used to infer some properties about qualitative aspects of the process. Figure 1 presents the model time-flow.

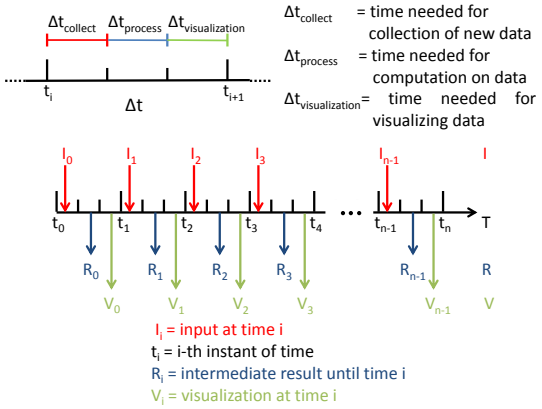

Figure 1: *Time-flow of the proposed model.*

The time flow is split in a series of quantum of time Δt: at each instant t_i, a set of new inputs I_i are produced and sent by the process; we need time for collecting and processing them, in order to produce the data that will drive the visualization, namely $R_i = \{A_i, S_i\}$, where A_i represent a series of aggregate indicators (e.g., mean, standard deviation, variance, elements count, etc.) and S_i constitutes the representation of the actual state of the system. According to the particular nature of the application and of the aggregation function, a statistical error $\varepsilon_{stat}(A_i)$ can be introduced during the computation of R_i (see, e.g., [ZCWQch]). The generic R_{i+1} is computed as follows:

$$\begin{cases} A_{i+1} = F(A_i, S_i, I_{i+1}) \\ S_{i+1} = G(S_i, I_{i+1}) \end{cases}$$

where F computes the aggregate A_{i+1} using the actual input and the previous state and aggregate and G is a transition

function that computes the state S_{i+1} using the actual input and the previous state.

At the end of $\Delta t_{process}$ these new data will be used for generating a new visualization V_{i+1}. We define the time needed for producing it as $\Delta t_{visualization}$.

So we can model the minumum quantum of time as:

$$Min\Delta t = \Delta t_{collect} + \Delta t_{process} + \Delta t_{visualization}$$

Figure 2 shows the general architectural schema used for modeling the applications discussed in Section 1.

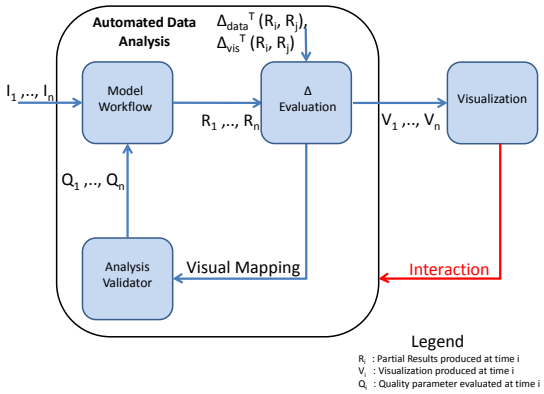

Figure 2: *Architectural schema of the application workflow.*

This architecture is composed of 2 main functional blocks, nominally the *Automated Data Analysis* block and the *Visualization* block. The Automated Data Anlysis block is in charge of supporting the abstraction of the problem as a data stream, and providing appropriate data to the visualization block; it is internally composed by 3 sub-blocks:

- **Model workflow**: this block takes the set of new inputs produced at instant t_i, I_i and the state of the system produced at instant t_{i-1}, S_{i-1} and produce a new set of intermediate results R_i.
- **Δ Evaluation**: the result produced by the previous block is directed to this block: Δ Evaluator is in charge of computing two process indicators, nominally $\Delta data^{actual}(R_i,R_j)$, representing the differences between the data produced in the iteration t_i and t_j, and $\Delta vis^{actual}(R_i,R_j)$, representing the *visual* differences between the visualizations associated with instant t_i and t_j. Such values are compared with suitable threshold values, $\Delta data^{T}(R_i,R_j)$ and $\Delta vis^{T}(R_i,R_j)$, in order to allow or not the rendering of a new visualization. It is worth noting that this comparison is not always made between the results of two consecutive time instants, but instead between R_i and the last t_j, with $j \leq i-1$, in which the visualization has changed. The goal is to reduce the time devoted to the visualization phase if there are not enough changes in data or visualization parameters that justify a new rendering.
- **Analysis Validator**: this block uses the data involved in the visualization process (computation results and visual mapping) to compute some quality indicators Q_i on the evolution of the analyzed process, in order to estimate both the error introduced in the evaluation of R_i and Δs and whether the automated analysis can be stopped or if it needs a further number of steps in order to produce more precise results. In particular, we define the following quality indicators:
  - $Q\Delta_{current} = R_N - R_i$: it represents the difference between the intermediate result R_i and the estimated final one at instant N.
  - $Q\Delta_{relative} = R_{i+1} - R_i$: it represents the difference between two consecutive results, and it is particularly useful for estimating how much data has changed in a single iteration of the process.
  - $Q\Delta_{absolute} = R_* - R_N$: it represent a fixed parameter of quality in order to understand how good is the estimated value R_N in respect to the ideal one R_*
  - $Q\varepsilon_{current} = R_N - R_i + \varepsilon_{stat}(A_i)$: it represents how much error is introduced during the execution of the process between the current iteration and the estimated final one.

4. Examples

This section presents two case studies, applying the formal model to a data streaming and a cloud streaming application.

Data Streaming We feed the visual analytics data streaming application with the NHTSA Fatality Analysis Reporting System (FARS) [NHT13] data of fatal car accidents in the USA, from year 1975 to year 1999. We assume that we are able to process all the incoming data storing only aggregate information (we simulate the situation in which it is not possible to store all the data). The goal of the application is to show the density distribution of accidents across the USA states and we assume that Δt will be much greater than the $Min\Delta t$: because the system is not a monitoring system but is intended just to show the accidents distribution we can imagine that the granularity of Δt is in term of hours: very likely the situation will change very slowly, even across days. That does not pose any severe constraint on the application timing and we can set $\Delta t_{collect} = \Delta_t - \Delta t_{process} - \Delta t_{visualization}$. Data are rendered using a choropleth map, split in N areas (states or counties) in which each accident density value at time i, $d_{i,k}, 1 \leq k \leq N$, is mapped on an ordered set of different shades of blue ($color(d_{i,k})$) clearly distinguishable by the user; we use this assumption in calculating Δvis^{actual}. On the basis of reference indicators, we will instantiate both $\Delta data^{actual}(R_i,R_j)$ and $\Delta vis^{actual}(R_i,R_j)$ as following:

$$\begin{cases} \Delta data^{actual}(R_i,R_j) = \Sigma_{k=1}^{N} \frac{|d_{j,k} - d_{i,k}|}{N}, \\ \Delta vis^{actual}(R_i,R_j) = \Sigma_{k=1}^{N} \frac{|color(d_{j,k}) - color(d_{i,k})|}{N}, \end{cases}$$

where, as mentioned in Section 3, $\Delta data^{actual}(R_i,R_j)$

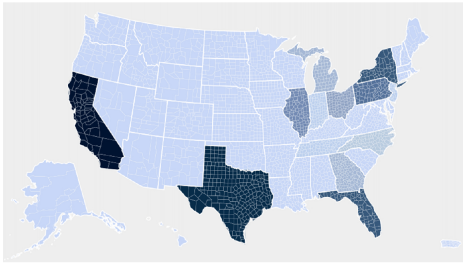

Figure 3: *last changed visualization at istant j*

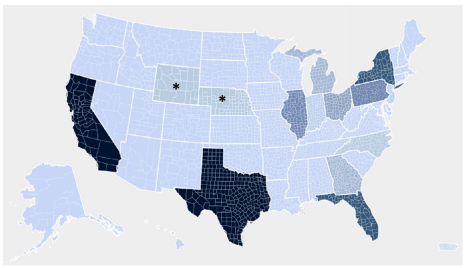

Figure 4: *changing state: $\Delta vis^{actual}(R_i,R_j)$ is above the threshold value and so a new rendering is alerted to the user (note the changes in Wyoming and Nebraska)*

will be used for cutting the visualization pipelines (evaluation+rendering) and passing directly at the following step, while $\Delta vis^{actual}(R_i,R_j)$ will be used for evaluating the visible differences in the visualization and alerting the user when a change occurs; all of this will be based on the comparison of actual values at interval i with corresponding threshold values $\Delta data^{T}(R_i,R_j)$ and $\Delta vis^{T}(R_i,R_j)$.

Figure 3 shows the last time interval j in which the visualization changed; each hour a new result R_i is produced, and the values of $\Delta data^{actual}(R_i,R_j)$ and $\Delta vis^{actual}(R_i,R_j)$ are computed: if $\Delta data^{actual}(R_i,R_j) < \Delta data^{T}(R_i,R_j)$ or $\Delta vis^{actual}(R_i,R_j) < \Delta vis^{T}(R_i,R_j)$ no new rendering is produced and the visualization stay the same; instead, if $\Delta vis^{actual}(R_i,R_j) > \Delta vis^{T}(R_i,R_j)$ a new visualization is produced, alerting the user that a noticeable change is available (see Figure 4).

Cloud Streaming The Cloud Streaming application is based on the idea of producing a visualization using a large, remote datafile, that is sent in a random fashion by the server and organized in self-consistent chunks. We simulated this situation splitting a million tuple file (the Vast 2011 mini challenge1 data [Vas11]) in N random chunks and feeding an application at precise time interval, simulating a slow download and visualization across the cloud.

As a class of problems that present slow but consistent amount of data loaded at each interval of time t_i, we can expect this time that the choice of the correct Δt will be driven by the $\Delta t_{collect}$ component, in order to not defer parts of data produced at time t_i to the next interval t_{i+1} and then risk to produce an amount of data not manageable in the followings time intervals. The process will terminate after N iterations producing, after a long time, a correct result R_N; however, it is very likely that the gain in fidelity does not justify a so long wait. In order to stop the process when the actual visualization is good enough we use the quality indicators $Q\Delta_{relative} = R_{i+1} - R_i$ and $\Delta vis^{actual}(R_i,R_j)$: when both of them are below the threshold we can stop the process and/or make the user aware that the visualization is worth to be used. Figure 5 shows the visualization driven by final result R_N while Figure 6 shows the visualization associated with the 40% of the download, in which both $Q\Delta_{relative} = R_{i+1} - R_i$ and $\Delta vis^{actual}(R_i,R_j)$ are below the threshold: the central data clusters are now quite evident.

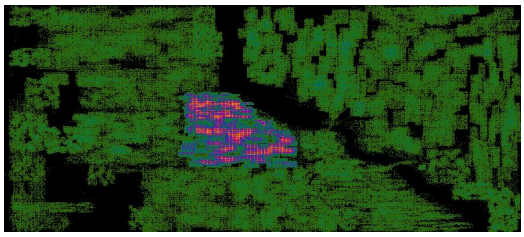

Figure 5: *Final state of the application*

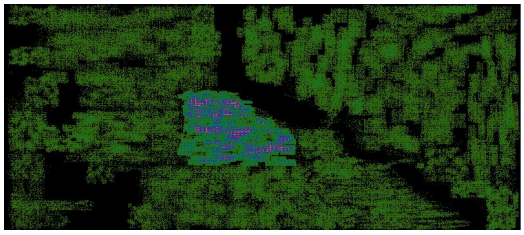

Figure 6: *The first intermediate valid state: quality indicators are below the threshold*

5. Conclusions

This paper addresses the problem of formal modeling a generic iterative drawing of a visualization, in terms of practical issues and parameters that can be used to drive and evaluate the whole process. Moreover, it provides an initial classification of the applications it is intended for; two examples provide a first understanding of the model features. We point out as future work further research in the field of estimation of final state for both iterative VA algorithms and cloud streaming scenarios, in order to provide practical applications to the model parts regarding the optimal final state R_* and the estimated final one R_N. Additionally, we plan to further expand the model capturing the error that rises from the visual interaction with partial results that is used to start new analytical activities.

References

[CL08] COTTAM J. A., LUMSDAINE A.: Stencil: A conceptual model for representation and interaction. In *IV* (2008), pp. 51–56. 2

[CLW12] COTTAM J. A., LUMSDAINE A., WEAVER C.: Watch this: A taxonomy for dynamic data visualization. In *IEEE VAST* (2012), pp. 193–202. 2

[CMS99] CARD S. K., MACKINLAY J. D., SHNEIDERMAN B. (Eds.): *Readings in information visualization: using vision to think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999. 2

[CR98] CHI E. H.-H., RIEDL J.: An operator interaction framework for visualization systems. In *Proceedings of the 1998 IEEE Symposium on Information Visualization* (Washington, DC, USA, 1998), INFOVIS '98, IEEE Computer Society, pp. 63–70. 2

[Duo07] DUONG T.: ks: Kernel density estimation and kernel discriminant analysis for multivariate data in r. *Journal of Statistical Software 21*, 7 (10 2007), 1–16. 2

[FDCD12] FISHER D., DELINE R., CZERWINSKI M., DRUCKER S.: Interactions with big data analytics. *interactions 19*, 3 (May 2012), 50–59. 2

[FPDs12] FISHER D., POPOV I., DRUCKER S., SCHRAEFEL M.: Trust me, i'm partially right: incremental visualization lets analysts explore large datasets faster. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2012), CHI '12, ACM, pp. 1673–1682. 2

[HCZD04] HAN B., COMANICIU D., ZHU Y., DAVIS L.: Incremental density approximation and kernel-based bayesian filtering for object tracking. In *in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Washington DC* (2004), pp. 638–644. 2

[JKMG02] JANKUN-KELLY T. J., MA K. L., GERTZ M.: A model for the visualization exploration process. In *Proceedings of the conference on Visualization '02* (Washington, DC, USA, 2002), VIS '02, IEEE Computer Society, pp. 323–330. 2

[Ma09] MA K.-L.: In situ visualization at extreme scale: Challenges and opportunities. *Computer Graphics and Applications, IEEE 29*, 6 (2009), 14–19. 1, 2

[NHT13] NHTSA: Fatality analysis reporting system (fars), 2013. 3

[PTMB09] PIRINGER H., TOMINSKI C., MUIGG P., BERGER W.: A multi-threading architecture to support interactive visual exploration. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1113–1120. 2

[RZH12] ROSENBAUM R., ZHI J., HAMANN B.: Progressive parallel coordinates. In *PacificVis* (2012), pp. 25–32. 2

[Shn08] SHNEIDERMAN B.: Extreme visualization: squeezing a billion records into a million pixels. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 2008), ACM, pp. 3–12. 1

[Twe97] TWEEDIE L.: Characterizing interactive externalizations. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems* (New York, NY, USA, 1997), CHI '97, ACM, pp. 375–382. 2

[Vas11] VAST2011: Geospatial and microblogging - characterization of an epidemic spread, 2011. 4

[WA12] WEAVER C., AHMED Z.: An adaptive parameter space-filling algorithm for highly interactive cluster exploration. *2012 IEEE Conference on Visual Analytics Science and Technology (VAST) 0* (2012), 13–22. 2

[WFA*03] WONG P. C., FOOTE H., ADAMS D., COWLEY W., THOMAS J.: Dynamic visualization of transient data streams. In *Proceedings of the Ninth annual IEEE conference on Information visualization* (Washington, DC, USA, 2003), INFOVIS'03, IEEE Computer Society, pp. 97–104. 2

[XWR10] XIE Z., WARD M. O., RUNDENSTEINER E. A.: Visual exploration of stream pattern changes using a data-driven framework. In *Proceedings of the 6th international conference on Advances in visual computing - Volume Part II* (Berlin, Heidelberg, 2010), ISVC'10, Springer-Verlag, pp. 522–532. 2

[ZCWQch] ZHOU A., CAI Z., WEI L., QIAN W.: M-kernel merging: towards density estimation over data streams. In *Database Systems for Advanced Applications, 2003. (DASFAA 2003). Proceedings. Eighth International Conference on* (March), pp. 285–292. 2