Multiple Visual Representation of Temporal Data

Chaouki Daassi[1+2], Marie-Christine Fauvet[2+3], and Laurence Nigay[1]

[1]Laboratoire CLIPS-IMAG BP 53-38041, Grenoble cedex 9, France
[2]Laboratoire LSR-IMAG BP 53-38041, Grenoble cedex 9, Grenoble, France
[3]The University of New South Wales, Sydney NSW 2052, Australia
[Chaouki.Daassi, Marie-Christine.Fauvet, Laurence.Nigay]@imag.fr

Abstract. Temporal data are abundantly present in many applications such as banking, financial, clinical, geographical applications and so on. For a long time, tools for data analysis have been only based on statistics. A more recent and complementary research avenue involves visual data analysis, which is dedicated to the extraction of valuable knowledge by exploiting human visual perception capabilities. Examples of visual data analysis tasks, while manipulating temporal data, include correlating data evolution and identifying patterns. In this paper we present an interactive visualisation tool named INVEST, dedicated to visual analysis of temporal data. INVEST includes different visualisation techniques in order to address the variety of users' tasks.
Keywords: Visualisation Techniques, Temporal Data, Data Analysis.

1 Introduction

Data can be analyzed using statistical tools: many algorithms are developed to automatically extract valuable information from a set of data. A promising and complementary research avenue involves visual data analysis, which relies on interactive visualisation techniques. Adopting this approach, many studies focus on the design of visualisation techniques for data analysis [5, 15], but few of them are dedicated to temporal data [20, 8]. Our work focuses on visualisation techniques for temporal data analysis.

A temporal data denotes the evolution of an object characteristic over a period of time. The value of a temporal data is called a history. For the sake of simplicity, we define a history as a collection of instant time-stamped or interval time-stamped data items, although there are many other ways of representing a history [11, 9]. Fig. 1, shows a history of numeric values, where each time-stamp denotes a month. The set of timestamps is the temporal domain of the observed data.

In this paper we present the framework called INVEST (INteractive Visualisation and Explorative System of Temporal data) that several complementary visualisation techniques that have been designed with regard to user's tasks while manipulating temporal data. The paper is organized as follows: in section 2, we present two novel visualisation techniques among five available in the INVEST. Section 3 presents the results of an experimental evaluation of the INVEST that confirm the offered. Finally,

section 4 describes the INVEST architecture and implementation while section 5 concludes..

2 Visualisation Techniques

Visualisation techniques could be classified into two categories depending on whether they rely on item-based techniques or pixel-based ones. Within the first category (item-based technique), data values are mapped onto graphical objects such as polygons, circles, etc., drawn in 2D [12, 14] or 3D data spaces [1, 5]. Data values are easily distinguishable from each other, so that the users can easily compare two elements. However, this representation mode reaches its limits when displaying a huge amount of data. Consequently, these techniques must either be augmented with sliders, or the data space must be deformed [16, 18]. The second category is made up of the pixel-oriented techniques [14], in which each data value is mapped to a pixel colored with some intensity: a huge amount of data can therefore be represented in a limited screen area. The disadvantage of such techniques is that the users cannot compare two elements, but rather have a global representation of the data space.
Our research focuses on the design of item-based and pixel-based visualisation techniques in a 2D space. In the following two paragraphs, we present two INVEST visualisation techniques, namely the concentric circles and the superposed histograms.

On the other hand, we have considered the design of visualisation techniques with regard to users' tasks. As pointed out by [3, 19, 7], the utility of any visualisation technique is a function of the task that the technique is being used to support. In the context of task-based design, we therefore empirically established a list of user 's tasks that manipulate temporal data. These tasks concern data correlation, pattern identification, identification of concentration point of particular values, etc. To do so, we conducted interviews with geographers about the tasks they perform during a data analysis process and about the interpretation they make based on graphical representations.

2.2 Concentric Circles Technique (CCT)

The Concentric Circles Technique is dedicated to the visualisation of one or two quantitative histories. As shown in Fig. 2, its interface is made of a set of concentric circles, each one denoting the evolution of the visualised history(ies) during a fixed-length period of time. Values are denoted as rectangles adjacent to the circles. The height of each rectangle and its color's intensity are proportional to the value that it denotes. Circles are arranged so as to reflect the time order associated with the data.

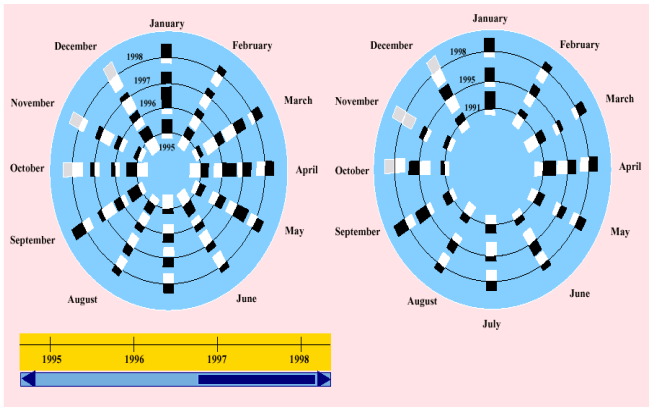

Fig. 2. The Concentric Circles Technique (CCT) .

We describe below the user's tasks addressed by the CCT. We compare our technique with the spiral representation of periodic data proposed in [4] (see Fig. 3).

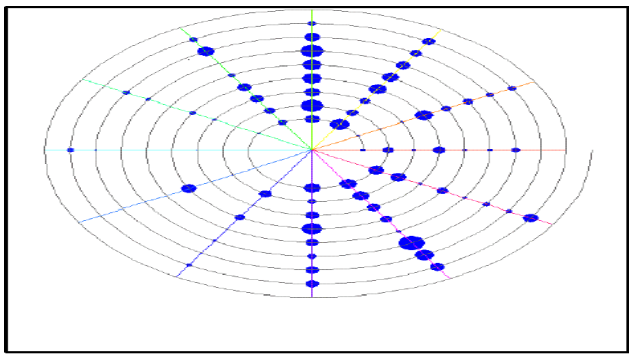

Fig. 3. Visualisation technique for serial periodic data [4].

The user's tasks addressed by the CCT are described below:

Navigate within a large data space. As shown in Fig. 3, in the spiral representation, the time axis is continuous and has a fixed origin. The size of the spiral is consequently proportional to the number of periods. It is clear that with a large number of periods (e.g. a hundred), the spiral representation cannot be applied, due to limitation of screen space. We face the same problem in the CCT, since the number of circles corresponds to the number of periods. To address this issue, the CCT's interface provides a time-slider, which allows the users to navigate through time. The displayed circles correspond to the periods falling within the time interval defined by the time-slider.

Correlate the evolution of two histories. The CCT supports the visualisation of two quantitative histories at a time. One of the histories is represented as explained above, while the other is represented with rectangles of a different color and oriented in an opposite direction. For instance, in Fig. 2, the rectangles denoting the values of one

of the histories are blue-colored (dark gray in the gray scaled version) and oriented away from the center, and the values of the second history are red-colored (light gray) and oriented towards the center.

Compare periods. To facilitate the comparison of two elements positioned far from each other on the screen, the CCT interface provides two visualisation areas. The first one, located on the left in Fig. 2 and named the *reference area*, contains circles ordered in time. The second one located on the right and named the *working area*, enables the user to place selected circles in an arbitrary order. Putting circles close to each other facilitates the comparison of data values belonging to different periods. The width of a rectangle is proportional to the number of rectangles per period and to the radius of the circle. Only the height of a rectangle is proportional to the value that the rectangle denotes. By placing two circles next to each other (radii are similar), rectangles have nearly the same width, and comparison of their heights can be performed more easily.

Identify general trends. With the spiral representation users must scour the data space along the spiral to observe data evolution. As opposed to the spiral representation, with the CCT, the user can directly observe the evolution of the data during a given period (e.g. a year or a month) as well as compare the evolution of the data from one period to the next one. For instance, it can be seen in Fig. 3 that the values reached in January tend to be higher than the rest of the year, and that every year, the values tend to be low during June and December.

2.2 Superposed Histograms Technique (SHT)

Fig. 4 shows the superposed histograms technique (SHT) that offers a different approach for visual analysis of serial temporal data.

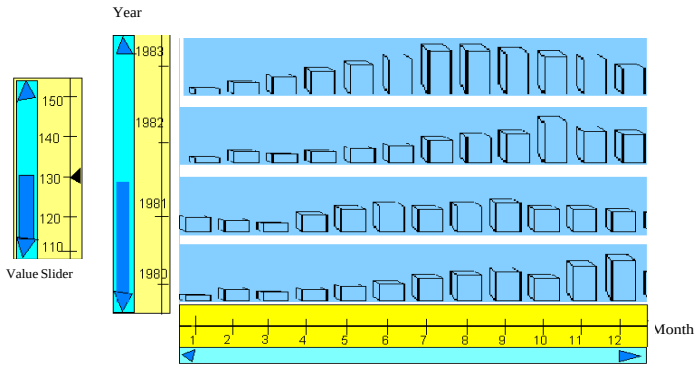

Fig. 4. Superposed Histograms Technique (SHT).

In SHT, data values have been mapped to 2.5 D objects (in the form of cubes). Time is represented along two dimensions. For this reason, the SHT includes two time-sliders, one vertical and one horizontal. Each data value is the result of a function ***f*** with two parameters that are two regular time-units. One time-unit *U1* is said to be regular with regard to another time-unit *U2*, only if each value of *U1* is composed of

a fixed number of *U2*. A concrete example of regular units is *year* and *month*. Each year contains twelve months. In the example of Fig. 4, years and months are mapped onto two axes: years along the vertical axis and months along the horizontal axis. Each data value corresponds to ***f*** (year, month). The SHT addresses three user's tasks:

Navigate in time. The SHT provides two tools to navigate in time. Users can navigate using the vertical time-slider or the horizontal one.

Study data evolution according to one reference value. The value-slider, represented on the left of Fig. 4, is used to fix a threshold of data values: Only values greater than this threshold are visualised on the screen. The length of a given graphical object (cube) is the ratio of the corresponding data value minus the threshold, and the maximum value minus the threshold.

Compare periods. Periods are represented in the form of superposed histograms. Such a representation is well suited for the comparison of periods as explained in [2]: if the user's task is to compare periods, it is recommended to superpose them or represent them in a circular form.

3. Usability Assessment

In this section we describe the protocol of the evaluation we have carried out. The goal of the experiment was to evaluate the usability of the INVEST with regard to the user's tasks. The usability of a given visualisation technique V_i with regard to a given user's task T_j is defined as the value of the function $F(V_i, T_j)$, which is in the range of 0 and 1. The experimental evaluation lets us empirically identify these values $F(V_i, T_j)$. To do so we asked six participants (Ph.D. and master students in Geography) to use INVEST while manipulating their real data: a pollution measurement of NO and NO2. All participants were familiar with temporal data manipulation and graphic interpretation. Before starting an evaluation session, each participant had a document that describes how to use the techniques and for which tasks the technique has been designed. While manipulating the data, the participant could ask for help of the experimenter if s/he was facing a problem. At the end of the session, the participant has been asked to fill up one form per technique.

First, the participant specified the results s/he has obtained for each task. For example, for the task “trend observation”, the participant has been asked to describe how the data evolve over time. Second, the participant has been asked to underline advantages and disadvantages of each technique with regard to a given task. In addition, s/he had to select a value between 0 and 20 to mark the usability of the technique according to a given task. The average of these values correspond to the values $F(V_i, T_j)$. Fig. 5 reports the results of the experiment according to five user's tasks and the two visualisation techniques CCT (light-grey) and SHT (dark-grey). Finally, the participant has been asked to propose other tasks it was easy to achieve using INVEST. By doing so, we could evaluate the completeness of our user's tasks list. For each participant, the session has lasted three hours and thirty minutes in average.

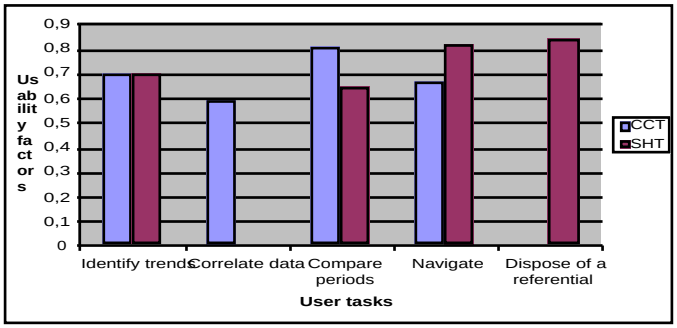

Fig. 5. Usability evaluation for both techniques (CCT and SHT).

The experimental evaluation leads to two conclusions:
First, we have empirically identified the $F(V_i, T_j)$ values. These values are embedded in the code of each technique. Indeed, in INVEST, each visualisation technique maintains knowledge about its usability factors according to each user's task. In the next section, we explain how we use these values at the implementation level.
Second, we experimentally prove that only one visualisation technique cannot fit all user's tasks. As shown in Fig. 5, for a given user's task Tj, each visualisation technique (V_i) does not have the same usability factor $F(V_i, T_j)$. For example, the SHT technique is not useful (usability factor equal to zero) when the user's task is to correlate two data evolutions, while it is very effective when the task is to observe data evolution according to a given referential. As a conclusion, visual data analysis systems should maintain a set of complementary visualisation techniques with regard to user's tasks. In the next section, we explain how we implement the multiple visualisation techniques within the software architecture model PAC-Amodeus.

4. Software architecture of the INVEST system

Software designers of interactive systems usually distinguish two parts in a system: the interface and the functional core. The former defines the perceptual part of the system that is manipulated by the user. The latter implements the internal components that are dependent on the application domain. We clearly establish this distinction within the code of INVEST by applying the software architecture model PAC-Amodeus [17]. It is an hybrid multi-agent software architecture model that represents the organization of the components of interactive software. It is a blend of the five software components advocated by the Arch model [21] and the PAC refining process expressed in terms of agents [6]. Fig. 6 shows the five components of the PAC-Amodeus model and the Dialogue Controller refined in terms of PAC agents.
The Functional Core (FC) implements domain specific concepts in a presentation independent way. The FC hosts the temporal data. The Functional Core Adapter (FCA) serves as a mediator between the Dialogue Controller (DC) and the domain-specific concepts implemented in the FC. It is designed to absorb the effects of changes in its direct neighbors. The FCA operates as a translator between the representation of the temporal data in the FC and the data structure used in the DC. Data exchanged with the DC are conceptual objects independent of the representation used

in the FC. This software design allows us to run INVEST using a temporal database or a set of temporal data files. Moreover, by only modifying the FCA, a relational database management system (new FC) can replace an object database management system (old FC).
At the other end of the spectrum, the Presentation Techniques Component (PTC) acts as a mediator between the DC and the Low Level Interaction Component (LLIC). The LLIC corresponds to the underlying platform, both hardware and software. In INVEST, this component corresponds to the AWT/Swing toolbox, INVEST being developed using JAVA. Because the PTC is platform independent, it is generally viewed as a logical LLIC. Presentation objects are translated in terms of interaction objects by the PTC. The distinction between presentation objects and interaction objects is subtle. A presentation object may correspond to an abstract interaction object or may correspond to a new interaction technique made of several interaction objects defined in the LLIC. Thus, the PTC defines a layer for portability, making the DC independent of the LLIC, as well as a layer for extending the LLIC services. For instance, each visualisation technique or part of it is defined by a JAVA bean, an extension of the toolbox. Such new interaction techniques belong to the PTC.

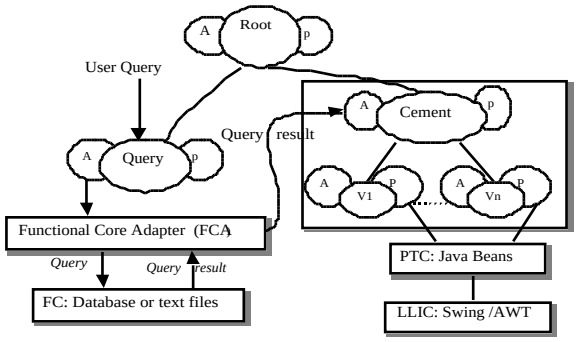

Fig. 6. INVEST architecture applying the PAC-Amodeus model.

The Dialogue Controller (DC) is the keystone of the model. It has the responsibility for task-level sequencing. The DC is decomposed into a set of cooperative PAC agents. The refinement of the DC in terms of PAC agents has multiple advantages including explicit support for concurrency (multithread dialogue, one dialogue per visualisation technique) and representation multiplicity (one representation per visualisation technique). A PAC agent is composed of three facets:
the Presentation implements the perceivable behavior of the agent. Within PAC-Amodeus, this facet is in direct contact with the Presentation Techniques Component (PTC); the Abstraction facet defines the competence of the agent. Within PAC-Amodeus, it is in direct contact with the Functional Core Adapter (FCA); the Control facet maintains the link between its two surrounding facets (i.e., Presentation and Abstraction) and the relationships with other agents.

Concretely, as shown in Fig. 6, the Dialogue Controller of INVEST is organized as a three-level hierarchy of PAC agents. One agent manages the query: its Abstraction facet communicates with the FCA to submit the query to the Temporal Database Management System. Once the query is executed, the FCA receives the data, trans-

lates them into a given data structure and submits them to the Abstraction of the Cement Agent. Each visualisation technique (V_i) is modeled as a PAC agent. The Presentation facet of a V_i agent relies on the corresponding JAVA bean in the PTC. Finally in order to establish communication between the Query agent and the Cement agent we add a Root agent. Within the hierarchy, the Cement Agent plays two roles:
First, it selects a subset of visualisation techniques that matches user's tasks. Let $N = \{N_1, N_2, .., N_k\}$ be the specified user's tasks. A given visualisation technique V_i is selected if $(\sum F(V_i, N_j) / card(N)) >= SelectivityFactor$, where $N_j \in N$, $j \in \{1…k\}$ and $SelectivityFactor \in [0\text{-}1]$. Both user's tasks and the SelectivityFactor are specified by the user, along with the query or the set of data.
Second, it maintains the visual consistency between the current selected visualisation techniques. According to the software principle which stresses “mutual ignorance” to enhance reusability, agents which implement visualisation techniques should not know each other. Any action with a visual side effect on a view is reported to the cement agent which broadcasts the update to the other siblings [17].

5. Conclusion and further work

The goal of the work presented here has been to gain understanding of the design of visualisation techniques for temporal data. We have shown that different visualisation techniques are necessary in order to address the variety of users' tasks. Adopting this approach, we have presented INVEST, a multiple visualisation techniques platform to visually analyze temporal data. Two INVEST visualisation techniques among five supported by INVEST are presented and their experimental evaluation underlines the fact that the two techniques are complementary because they do not address the same user's tasks. We finally explained how the multiple visualisation techniques are implemented by applying our PAC-Amodeus architecture model. The software design guarantees the portability of INVEST with regard to the Functional Core component. We are planning to extend INVEST so that it can communicate with several Functional Cores including text files, XML files, and object database management system. Finally we also plan to carry out further usability experiments with more participants (colleagues of Geography Laboratory) in order to assess the usability of the visualisation techniques with regard to the user's tasks.

References

1. http://www.cs.auc.dk/3DVDM.
2. Bertin J., Graphics and Graphic Information Processing, Walter de Gruyter & Co, Berlin, 1981.
3. Casner S.. A task-Analytic Approach to the Automated Design of Graphic Presentations. ACM Transactions on Graphics, vol. 10, N° 2, April 1991, Pages 111-151.
4. Carlis J.V. and Konston J.A. Interactive Visualisation of Serial Periodic Data. In Proc. of the ACM Conference on User Interface Software and Technologie UIST´98, San Francisco, Ca, 1998.
5. Cook D., Virtual Reality: Real Ponderings, http://www.public.iastate.edu/~di5/Chance/paper.ps.gz.

6. Coutaz J., PAC-ing the Architecture of Your User Interface, DSV-IS´97, 4th Eurographics Workshop on Design, Specification and Verification of Interactive Systems, Springer Verlag Publ., P: 15-32.
7. Chuah Mei C., Roth Steven F., Mattis J. and Kolojejchick J. SDM: Selective Dynamic Manipulation of Visualisations. In proceedings of UIST'95 Conference, Pittsburgh PA USA, Pages 61-70.
8. Daassi C., Dumas M., Fauvet M-C, Nigay L. and Scholl P-C. Visual Exploration of Temporal Object Databases, In proc. of BDA00 Conference, 24-27 October 2000, Blois, France, P: 159-178.
9. Dumas M, Tempos: une plate-forme pour le développement d´application temporelles au dessus de SGBD à objets. Thèse de Doctorat de l´Université Joseph-Fourier, Grenoble, France, 2000.
10. Etzion E., Jajodia S. and Sripada S. M. (Ed.). Temporal Databases: Research and Practice. Springer Verlag Pub. LNCS 1399, 1998.
11. Fauvet M-C. Dumas M., and Scholl P-C. A representation independent temporal extension of ODMG´s Object Query Language. In proc. of BDA99 Conference, Bordeaux, France, October1999.
12. Fredrikson A., North C., Plaisant C, and Shneiderman B.: Temporal, Geographical, and Categorial Aggregations Viewed through Coordinated Displays: A Case Study with Highway Incident Data. Workshop on New Paradigms in Information Visualisation and Manipulation 1999: 26-34.
13. Gram C. and Cockton G. Design Principles for Interactive Software. St Edmundsbury Press, 1996.
14. Keim D. Pixel-oriented Database Visualisation, SIGMOD Record, Special Issue on Information Visualisation, December. 1996.
15. Keim D. Visual Data Mining, Conf. On Very Large Databases (VLDB´97), Athens, Greece, 1997.
16. Mackinlay J.D., Robertson G., and Card S.K. The Perspective wall: Detail and Context smoothly integrated, Human factors in computing systems conference proceedings on Reaching through technology, 1991, Pages 173 – 176.
17. Nigay, L. and Coutaz, J. Software Architecture Modelling: Bridging Two Worlds Using Ergonomics and Software Properties, pp 49-73 in Formal Methods in Human-Computer Interaction, Palanque and Paterno (eds). Springer-Verlag, Berlin, 1997.
18. Nigay L. and Vernier F. Design Method of Interaction Techniques for Large Information Spaces, AVI´98, May 24-27, 1998, p: 37-46.
19. Roth Steven F., Kolojejchick J., Mattis J. and Goldstein J. Interactive Graphic Design Using Automatic Presentation Knowledge. In Proc. of CHI'94 conference, Boston USA, Pages 112-118.
20. Shahar Y. and Cheng C. Intelligent Visualisation and Exploration of Time Oriented Clinical Data. Technical Report TR SMI-98-0732, Stanford University, 1998.
21. The UIMS Workshop Tool Developers : A Metamodel for the Runtime Architecture of an Interactive System, SIGCHI Bulletin, 1992.