

A Graph-based Model for Navigating Visualisation

Rajehndra Nagappan
Department of Computer Science
The Australian National University
CANBERRA ACT 0200, Australia
Raj.Nagappan@cs.anu.edu.au

Abstract

This paper presents a model for describing visualisation by considering individual parts of an instance and relating them to each other. The resulting part-relationship structure forms a mental graph that can be used to navigate throughout the visualisation as a whole. This model is effective for exploring contemporary, complex visualisation instances which are difficult to analyse using traditional mechanisms.

Keywords: *information visualisation, data visualisation, visualisation model, visualisation algorithm.*

1. Introduction

Visualisation theory has traditionally focused on a single static graphic as the outcome of the visualisation process. Recently though attention has shifted towards more dynamic, interactive visualisation environments. This is due to a number of reasons: improving technology and methods; increasing size and complexity of data; and an emerging role of visualisation as an exploration and analysis tool rather than just as a presentation tool.

However, while there are many software tools and case-specific algorithms available, there is little consensus on the reasons why these systems are successful (or not). What is required is a model that describes the changing nature of contemporary visualisation practice — one in which data is large and complex, tasks are high-level (rather than elementary) and visualisation instances evolve to reflect ongoing and changing inquiries by users.

This paper presents a model of visualisation that relates the various parts of a visualisation instance to each other. This forms a map of concepts that can be used to navigate through a complex visualisation. The model is related to existed theories of visualisation. In this paper we focus exclusively on relationships between parts of a visualisation instance; Nagappan [4] describes other aspects.

2. Background

The model for navigating through visualisation complements three existing pieces of visualisation theory. In this section those theories are given a brief overview. Additionally the type of data dealt with is described.

2.1. The visualisation Pipeline

The *visualisation pipeline* [3] is a model that describes visualisation as a unidirectional pipe from a set of raw data through to a visualisation instance. This is shown in Figure 1. Intermediate stages of the pipeline filter and transform the data from its native state to a visual representation.

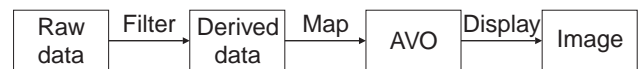


Figure 1. Visualisation pipeline.

Data is first filtered, producing derived data. The derived data is then mapped to an Abstract Visualisation Object (AVO). The AVO is displayed to produce the final image. From [3].

Roberts extended the pipeline metaphor to allow multiple instances of visualisations to be derived from a single source of data [7, 8]. This arrangement can be thought of as ‘forking’ the pipe to allow a single early stage to provide data to multiple later stages. This is illustrated in Figure 2. The benefit of this approach is that the same raw data can be used to drive many similar visualisations, where each visualisation focuses on a different set of properties from the underlying data.

2.2. Information Gain

Information foraging theory suggests that information systems evolve to maximise gains of information and

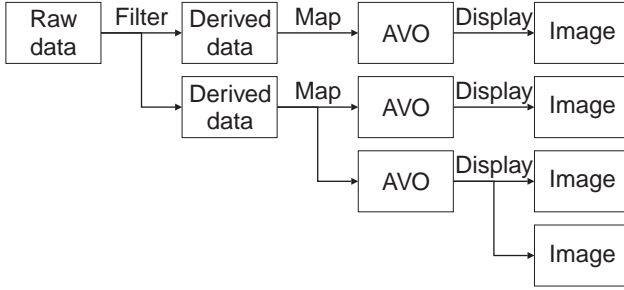


Figure 2. Multiple view and multiform pipeline.

Forking may occur at any of the filter, map or display processes. The further downstream (i.e. closer to the final image) that forking occurs the closer the resultant images are likely to be. From [7].

insight per unit of effort expended [6]. Successful visualisations are those which provide a great deal of information whilst requiring the user to expend very little effort.

Let a given visualisation consist of multiple components or patches, in which each patch is a sector of the visualisation that provides some congruent, meaningful and potentially useful piece of information. For instance, in Roberts' multiple view pipeline in Figure 2 each visualisation image is a patch. Let a patch be denoted as i , and the amount of information gained from a patch be $g(i)$. Let the time or effort required to explore i be $t(i)$. Finally, let the time or effort spent between patches, i.e. shifting the user's attention, be the set $\{t_b\}^\ddagger$. The rate that the user gains information, R , is thus given by:

$$R = \frac{\sum_i g(i)}{\sum_i t(i) + \sum_{\{t_b\}} t_b}$$

2.3. Navigation of Information Spaces

The *direct walk* is where a user navigates through an information space from a starting point to a target point by a series of direct manipulation and interaction 'steps' [2]. The goal is to provide software tools structured so that the user is required to perform as few steps as possible to reach the target point. This metaphor suggests that users are primarily interested in point-to-point movements within an information space. Software tools that require fewer movement steps result in increased rates of information gain, R , since the time to move between patches is reduced.

2.4. Relational Data

This model is designed to be used primarily with relational data (the definition used is by Ullman [9]). A domain

[†]. Movements between patches are arbitrary, just because two patches exist does not mean that a user must move between them.

d is a set of values. A relation R is a subset of the Cartesian product of a number of domains, $R \subseteq d_1 \times d_2 \times \dots \times d_n$, where n is the degree. A datum is a tuple t that is an element of R . Frequently this is represented as a table; domains represent columns while tuples represent rows. The name of a column is an attribute of the relation.

The Cartesian product of relations R and S of degree m and n respectively forms a new relation T of degree $m + n$ such that the first m coefficients of each tuple in T are a tuple in R and the last n coefficients are a tuple in S . The natural join between R and S , $join(R, S)$ operates as follows:

1. Compute $R \times S$.
2. For each attribute a occurring in both R and S select those tuples whose values agree for both $R.a$ and $S.a$, where $R.a$ and $S.a$ are the attributes in $R \times S$ corresponding to attribute a in relations R and S respectively. Discard all other tuples.
3. For each attribute a form a projection which removes $S.a$ but preserves all other attributes. Call the remaining attribute $R.a$ by the common name a .

The natural join will combine relations such that if there are any common attributes then they will be used to constrain tuples such that no tuple can have different values for the same attribute.

3. A Graph-based Model for Navigating Visualisation

A major shortcoming of the visualisation pipeline and the direct walk is that they are both unidirectional, point-to-point metaphors for describing visualisation. Contemporary uses of visualisation deal with complex data and many iterations of queries and exploration. What is required is a model that allows arbitrary movement and inquiry within an evolving and adapting visualisation instance.

We achieve this by applying a graph-based model to visualisation. Visualisations may be subdivided into a set of coupled visualisation modules or patches. Individual modules are not isolated from each other. As described in Section 2.1, visualisation modules are often related; these relationships may include common data, common filtering or common display attributes. Figure 3 shows a number of visualisation modules and the relationships between them. Relationships between modules have measurable strength, the strength indicates the degree of commonality between them. Strong links indicate that the modules possess many common characteristics whilst weak links indicate that the modules possess few characteristics in common. The nature of relationships are described in the following subsection

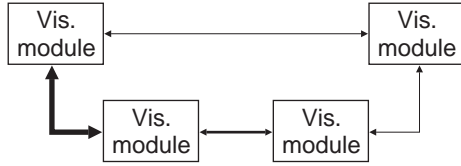


Figure 3. Graph-based model of visualisation. Each visualisation module bears a relationship with its neighbours. The strength of the relationship is given by the number of common attributes, and is shown here by the thickness of the link.

3.1. Mechanics of the Graph

Each visualisation module is a relation between visual attributes. Tuples are visualised by mapping data attributes to visual attributes. For brevity in this paper we will refer to the data attribute that a visual attribute encodes rather than the visual attribute itself.

The relationship between a pair of visualisation modules is their set of common attributes. More common attributes leads to a stronger relationship between them. To compose two or more visualisation modules together (for comparison, collaboration, movement between them, etc.) requires a natural join operation between them. If two modules have no common attributes then there is no way of telling which tuples in one module correlate with which tuples in the other. If there is one common attribute then many tuples will be able to be correlated, however if multiple tuples share the same value in this particular attribute then they still cannot be differentiated. As the number of attributes in common rises the correlations between individual tuples become stronger and clearer.

Conversely, Bertin noted that there is a limit to the number of attributes that a visualisation module can successfully encode [1]. With a fixed amount of attributes available there is a trade off between strength of commonality and variety of information. More common attributes will increase the relationship strength but will result in fewer attributes being displayed overall. Fewer common attributes will allow a greater breadth of information to be displayed but will reduce the strength of relationships between modules. The deciding factor is task: tasks requiring high correlations will use the former arrangement while tasks requiring overall surveys will use the latter.

3.2. Navigation Through the Graph

This view of visualisation allows point-to-point direct walking such as that given in Section 2.3. However, the graph based view also facilitates meandering or browsing through the visualisation. In this case the user wishes to try one or more movements through the visualisation, or

inquiries, but is not quite sure where they are headed or what they expect. Similar to movement through a physical space, the user can move arbitrarily through the visualisation and note points of interest, returning to them later if desired. The relationships between modules are the edges that the user must traverse; stronger relationships are equivalent to edges with lower cost whilst weaker relationships are equivalent to edges with higher cost. Navigation through a graph is commonly seen in we browsing applications.

3.3. Rate of Information Gain

The weighting of relationships between visualisation modules allows assessment of the degree of effort required for any given movement between two modules. Stronger relationships reduce the cognitive load required to understand a new visualisation module, thus reducing the traversal time t_b between them. Reduction in traversal time results in an increase in R . Additionally the relationship strength is a forward indicator of the likely profitability of a new module. If the set of common attributes contains interesting information then there is a good chance that the new module will provide more interesting information. This is known as *information scent* [6].

3.4. The Visualisation Pipeline Revisited

Each visualisation module is the end product of a visualisation pipeline. Figure 4 shows the multiform visualisation pipeline attached to the graph model. The links between visualisation modules represent movement of the user's attention back up the pipeline to the point of forking and then down to the new visualisation module. Longer movements that require returning to earlier stages of the visualisation pipeline result in weaker links between the visualisation modules and higher cognitive loads when traversing between them. This is described in more detail in [4].

The multiple-view pipeline only supports fanning out of information flow. Conversely, the graph view supports both fanning out and fanning in of information flow. Typical uses of fanning in are where multiple independent mapping modules feed the same display module, or where a common set of mapping modules feed into different display modules that each receive additional input from elsewhere. Both of these situations are shown in Figure 5. Fanning in occurs when there are common attributes between modules.

Further, fanning in of information flow allows multiple visualisation modules to be considered as parts of a larger whole. This encapsulation of modules within modules

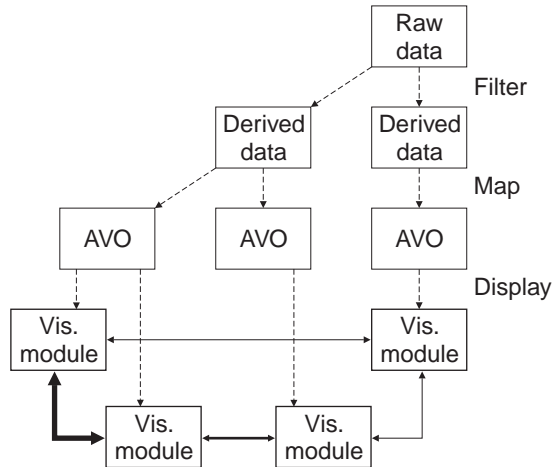


Figure 4. The visualisation pipeline and the graph model of visualisation.

The graph model is the culmination of other visualisation process models, the multi-view visualisation pipeline is shown here.

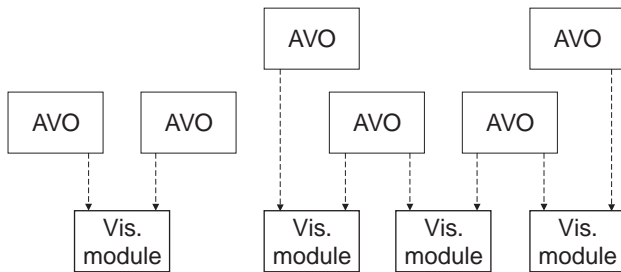


Figure 5. Fanning in of information flow.

The graph model supports visualisation process models in which multiple sources of information can flow into the same target module.

allows for a hierarchical nesting of visualisation modules. Larger modules encapsulate collections of complimentary smaller visualisation modules that possess strong relationship links. The resulting module thus possesses an increased information gain, g , since it represents the conglomeration of gains from each smaller module and the gains from collaboration between the smaller modules.

3.5. Evolving Visualisation

Frequently a user will use a given visualisation instance to form new queries and thus create new visualisation instances. The graph model of visualisation supports evolutionary development of visualisation instances through: addition of new visualisation modules into the graph; deletion of old visualisation modules from the graph; and change of existing visualisation modules within the graph. When a module changes the relationship links dynamically

change to reflect the new relationships between visualisation modules.

4. Applying the Model

The graph model of visualisation is realised in the Multidimensional Data Orb (mdOrb) [5], shown in Figure 6. This is a visualisation system for multidimensional data that is based on partitioning a Virtual Environment (VE) into smaller visualisation regions. The technique first distributes a set of partitioning axes in a radial arrangement from a single common origin, with one axis for each dimension in the data. The axes are used to partition the space into a set of pyramids defined by triads of adjacent axes. Each axis triad forms a skewed rectangular prism.

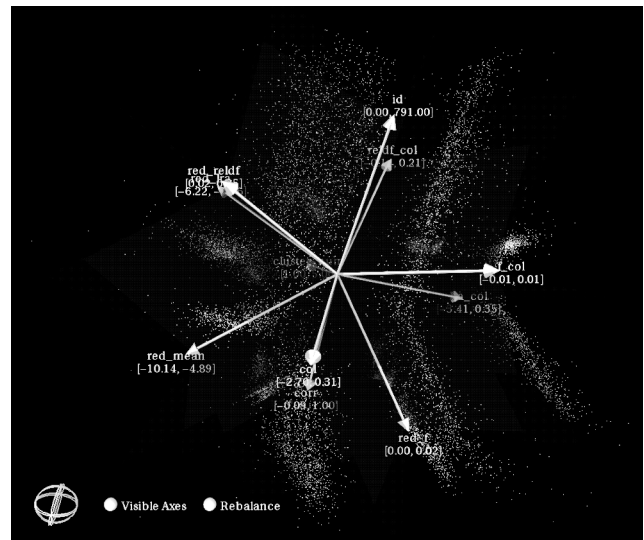


Figure 6. Multidimensional data orb.

The orb is formed by a tessellation of skewed Cartesian three-spaces into a convex triangular mesh. The axes radiate from the centre, three-spaces are defined by triangles that span adjacent triads of axes. This visualisation shows eleven dimensional star data.

Each rectangular prism acts as a skewed Cartesian three-space for a 3D scatterplot of points. This visualisation is of an eleven dimensional feature set describing a collection of approximately 750 variable stars. The points in each three-space are given by the values of each point from the n dimensional data space in the dimensions specified by the bounding axes. A single data point is represented by a mark in every three-space, where each mark is a point in the 3D scatterplot.

The mdOrb does not display every possible combination of dimensions concurrently. Rather the only combinations shown are those in close proximity to each other as determined by the current axis triads. However, each axis can be

moved around the orb interactively. As the user moves an axis old triads are destroyed and new ones are formed. This allows the dynamic formation of new data correlations between dimensions.

Each skewed three-space is a visualisation module that is a projection of the data space into a skewed 3D scatterplot defined by its axes. Any two adjacent three-spaces share a common pair of axes, and a single axis defines a boundary for every three-space surrounding it (usually several). Clearly each visualisation module is related to its neighbours by one or two common axes. These common axes are the foundations for common properties in this visualisation instance. As a user moves around the mdOrb the correlations that they make between three-spaces are due to axes in common. The graph for the mdOrb consists of visualisation nodes which are the individual three-space projections, and edges which are the set of common axes between pairs of neighbouring projections. The dynamic nature of the mdOrb shows that the graph model withstands change and evolution of the visualisation.

An analysis of mdOrb using the graph model is shown in Figure 7. Each dimension or axis is an AVO; it is an abstract representation of data in that dimension but is not yet rendered. Each scatterplot is a visualisation module. A scatterplot combines three dimensions, hence it accepts three sources of AVO inputs. Each dimension is used in all the scatterplots that surround its axis, so it provides data to multiple visualisation modules. The dimensions which are common between scatterplots are the common properties used to relate the scatterplots to each other, shown by the solid edges at the bottom of the figure. The predecessor stages of filtering data into each dimension individually is not shown.

5. Future Work

The graph representation of visualisation is just one part of a larger model that describes the structure and composition of complex data visualisation tools. Ongoing work includes: task analysis; measurement of the nature and complexity of visualisation modules and their relationships; application of the model to a broader range of visualisation paradigms; and evaluation of the model.

6. Conclusion

Successful visualisation systems create better, shorter paths of navigation through an information space. The graph model of visualisation presented in this paper provides a concrete description of movement through a visual-

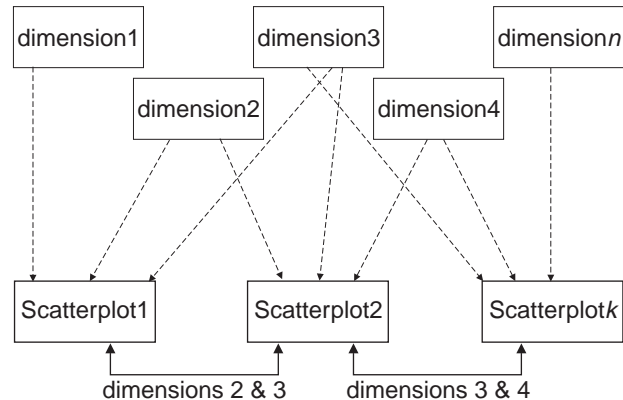


Figure 7. Analysis of mdOrb using the graph model.

Each dimension of data is an AVO. Each scatterplot is a visualisation module. The AVOs are combined in different scatterplots to show different combinations of dimensions. The dimensions which are common between scatterplots are used to relate the scatterplots to each other.

isation instance. It supports point-to-point movement, meandering and browsing, and iterative and evolutionary visualisation instances. The graph model provides strong cues as to why complex visualisations are (or are not) understandable and interpretable.

7. Acknowledgements

Thanks to my supervisor Mark Grundy for his comments and contributions, and to Bill Clarke for the star data. Thanks to the Cooperative Research Centre for Advanced Computational Systems (ACSys) under which this work is funded.

References

- [1] Bertin J. *Semiology of Graphics: Diagrams, Networks, Maps*. Translated by Berg W. University of Wisconsin Press. 1983.
- [2] Card S.K., Pirolli P., Mackinlay J.D. "The Cost-of-Knowledge Characteristic Function: Display Evaluation for Direct-Walk Dynamic Information Visualisations." *Proceedings of ACM Human Factors in Computing Systems*. pp 238-244. 1994.
- [3] Haber R.B., McNabb D.A. "Visualisation Idioms: A Conceptual Model for Scientific Visualisation Systems." *Visualisation in Scientific Computing*. Nielson G.M., Shriver B. eds. IEEE Computer Society Press. 1990.
- [4] Nagappan R. "A Compositional Model for Multidimensional Data Visualisation." *Visual Data Exploration and Analysis VIII*, in *Proceedings of SPIE*, vol. 4302. January 2001.

- [5] Nagappan R. "Visualising Multidimensional Non-Geometric Data Sets." Visual Data Exploration and Analysis VII, in *Proceedings of SPIE*, vol. 3960. 2000.
- [6] Pirolli P., Card S.K. "Information Foraging." *Psychological Review* 106(4). pp 643-675. 1999.
- [7] Roberts J.C. "Multiple-View and Multiform Visualisation." Visual Data Exploration and Analysis VII, in *Proceedings of SPIE*, vol. 3960. 2000.
- [8] Roberts J.C. "On Encouraging Multiple Views for visualisation." *Proceedings of IEEE Information visualisation IV'98*. pp 8-14. 1998.
- [9] Ullman J.D. *Principles of Database and Knowledge-Base Systems, Volume 1*. Computer Science Press. 1988.